Counting Abelian Squares More Efficiently

Ryan Bennink

Quantum Computational Science Group, Oak Ridge National Laboratory

I present a recursive formula for calculating the number of abelian squares of length $n+n$ over an alphabet of size d. The presented formula is similar to a previously known formula but has substantially lower complexity when $d \gg n$.

INTRODUCTION

An abelian square is a word whose first half is an anagram of its second half, for example `intestines = intes · tines` or `bonbon = bon · bon`. Abelian squares are fundamentally interesting combinatoric structures [1–7] that can arise in a variety of contexts in applied mathematics. The work reported here was motivated by a problem in quantum computing. As discussed in a related manuscript [8], the expressiveness of a certain class of variational quantum circuits can be related to the number of abelian squares over a certain alphabet. However, due to fact that the alphabet in this case is exponentially large, existing formulas for evaluating the number of abelian squares were found to be impractical.

In this letter I present a recursive formula for calculating the number of abelian squares of length $n+n$ over an alphabet of size d that is efficient even when d is very large. I first review the problem of counting abelian squares and an existing recursive formula [9] whose cost is $O(n^2 d)$. Then I derive a new recursive formula whose cost is only $O(n^2 \min(n,d))$, a substantial improvement when $d \gg n$. I furthermore give a constructive interpretation of the formula.

BACKGROUND

Let $f_d(n)$ denote the number of abelian squares of length $n + n$ over an alphabet of d symbols. Trivially, $f_1(n) = 1$ for all n and $f_d(0) = 1$ for all d. It is also not difficult to see that $f_d(1) = d$. To determine $f_d(n)$ for arbitrary d and n, we define the *signature* (sometimes called the Parikh vector) of a word $w \in \{a_1, \ldots, a_d\}^*$ as $(m_1, \ldots, m_d)$ where m_i is the number of times the symbol a_i appears in w. Note that two words are anagrams if and only if they have the same signature. Thus the number of abelian squares is the number of pairs (x, y) such that x and y have the same signature.

$d\backslash n$	0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1	1
2	1	2	6	20	70	252	924	3432
3	1	3	15	93	639	4653	35169	272835
4	1	4	28	256	2716	31504	387136	4951552
5	1	5	45	545	7885	127905	2241225	41467725
6	1	6	66	996	18306	384156	8848236	218040696

Table I. Number of abelian squares of length $n + n$ over an alphabet of size d [9].

The number of words with a particular signature $(m_1, \ldots, m_d)$ is given by the multinomial coefficient

$$\binom{m_1 + \cdots + m_d}{m_1, \ldots, m_d} = \frac{(m_1 + \cdots + m_d)!}{m_1! \cdots m_d!}. \tag{1}$$

The number of ways to choose a pair of words, each with signature $(m_1, \ldots, m_d)$, is just the square of this quantity. Therefore the number of abelian squares of length $n + n$ is

$$f_d(n) = \sum_{0 \leq m_1 + \cdots + m_d \leq n} \binom{n}{m_1, \ldots, m_d}^2. \tag{2}$$

The first few values of $f_d(n)$ are shown in Table I.

Eq. (2) is not easy to evaluate when n is large, as the number of signatures grows combinatorially in d and n. Richmond and Shallit [9] derived a recursive formula using a simple constructive argument: To create size (n, n) abelian word pair (x, y) over alphabet $\{a_1, \ldots, a_d\}$, one can first choose the number $i \in \{0, \ldots, n\}$ of occurrences of a_d in each word. There are $\binom{n}{i}$ ways to distribute these occurrences in each word. Then there are $f_{d-1}(n - i)$ ways to create an abelian pair over $\{a_1, \ldots, a_{d-1}\}$ for the remaining $n - i$ symbols in each word. Setting $k = n - i$ and summing over the choice of k yields

$$f_d(n) = \sum_{k=0}^{n} \binom{n}{k}^2 f_{d-1}(k). \tag{3}$$

Using this formula, $f_d(n)$ can be obtained by starting with $f_1(0) = \cdots = f_1(n) = 1$ and computing $f_i(0), \ldots, f_i(n)$ in turn for $i = 2, \ldots, d$ (Fig. 1 left). The cost of computing the values of f_i given the previously computed values of f_{i-1} is $O(1 + 2 + \cdots + n) = O(n^2)$. Thus the complexity of evaluating $f_d(n)$ using (3) is $O(n^2 d)$, a huge improvement over (2) when n and d are both small. In contexts where d is very large, however, (2) is impractical.

AN ALTERNATIVE RECURSIVE FORMULA

In this section I derive an alternative to (3) whose cost of evaluation is only $O(n^2 \min(n,d))$. Let A_d denote an alphabet of d symbols. The number of abelian squares $(x,y) \in A_d^n \times A_d^n$ can be expressed as the sum of the number of anagrams of each word x:

$$f_d(n) = \sum_{x \in A_d^n} \binom{n}{m_1, \ldots, m_d}. \tag{4}$$

Here m implicitly denotes the signature of $x = (x_1, \ldots, x_n)$. We split off the sum over x_n:

$$f_d(n) = \sum_{x' \in A_d^{n-1}} \sum_{x_n \in A_d} \binom{n}{m_1', \ldots, m_{x_n}' + 1, \ldots m_d'} \tag{5}$$

where m' is the signature of $x' \equiv (x_1, \ldots, x_{n-1})$. We have

$$\binom{n}{m_1', \ldots, m_{x_n}' + 1, \ldots m_d'} = \frac{n}{m_{x_n}' + 1} \binom{n-1}{m_1', \ldots, m_d'}. \tag{6}$$

Then

$$f_d(n) = \sum_{x' \in A_d^{n-1}} \sum_{x_n \in A_d} \frac{n}{m_{x_n}' + 1} \binom{n-1}{m_1', \ldots, m_d'}. \tag{7}$$

By symmetry x_n can be replaced by any value; choosing d yields

$$f_d(n) = d \sum_{x' \in A_d^{n-1}} \frac{n}{m_d' + 1} \binom{n-1}{m_1', \ldots, m_d'}. \tag{8}$$

Now, each x' with a given signature contributes the same value to the sum. We may thus replace the sum over x' by a sum over the signatures of x', weighted by the number of occurrences of each signature:

$$f_d(n) = d \sum_{m_1' + \cdots + m_d' = n-1} \frac{n}{m_d' + 1} \binom{n-1}{m_1', \ldots, m_d'}^2. \tag{9}$$

We henceforth suppress the primes on m. The goal now is to move the dependence on m_d out of the sum, leaving something which has the form of (2). We have

$$\binom{n-1}{m_1, \ldots, m_d} = \binom{n-1}{m_d} \binom{n-1-m_d}{m_1, \ldots, m_{d-1}}. \tag{10}$$

This yields

$$f_d(n) = d \sum_{m_1 + \cdots + m_d = n-1} \frac{n}{m_d + 1} \binom{n-1}{m_d}^2 \binom{n-1-m_d}{m_1, \ldots, m_{d-1}}^2. \tag{11}$$

$$= d \sum_{m_d=0}^{n-1} \frac{n}{m_d + 1} \binom{n-1}{m_d}^2 \sum_{m_1 + \cdots + m_{d-1} = n-1-m_d} \binom{n-1-m_d}{m_1, \ldots, m_d}^2. \tag{12}$$

In terms of $k \equiv n - 1 - m_d$,

$$f_d(n) = d \sum_{k=0}^{n-1} \frac{n}{n-k} \binom{n-1}{n-1-k}^2 \sum_{m_1 + \cdots + m_{d-1} = k} \binom{k}{m_1, \ldots, m_d}^2. \tag{13}$$

Comparison of the latter sum to (2) reveals that it is none other than $f_{d-1}(k)$. The remaining quantities can be simplified as follows:

$$\binom{n-1}{n-1-k} = \binom{n-1}{k}, \tag{14}$$

$$\frac{n}{n-k} \binom{n-1}{n-1-k} = \binom{n}{k}. \tag{15}$$

Making these substitutions yields the main result:

$$f_d(n) = d \sum_{k=0}^{n-1} \binom{n}{k} \binom{n-1}{k} f_{d-1}(k). \tag{16}$$

Note the close similarity between (16) and (3). The crucial difference is that in (16) the sum goes up to only $n - 1$; that is, each level of recursion decreases *both* n and d (Fig. 1 right). Thus only $\min(n, d)$ levels of recursion are needed. The cost of this algorithm is $O(n^2 \min(n, d))$.

Eq. (16) can be interpreted in terms of the following approach approach to constructing an abelian pair: There are d choices for the first symbol a of x. Let $k \in \{0, \ldots, n-1\}$ be the number of occurrences in each word of symbols from A_d/a. There are $\binom{n-1}{k}$ choices to place those other symbols in x and $\binom{n}{k}$ places to place those other symbols in y. Then, one creates an abelian pair of size (k, k) over A_d/a, which is an alphabet of size $d - 1$.

Fig. 2 shows $f_d(n)$ as a function of n for exponentially increasing values of d. (The lines for $d \geq 64$ are truncated due to some of the results being outside the range of double-precision arithmetic.) The entire plot, comprising 1000 computed points, took less than two seconds to compute in MATLAB on a standard computer.
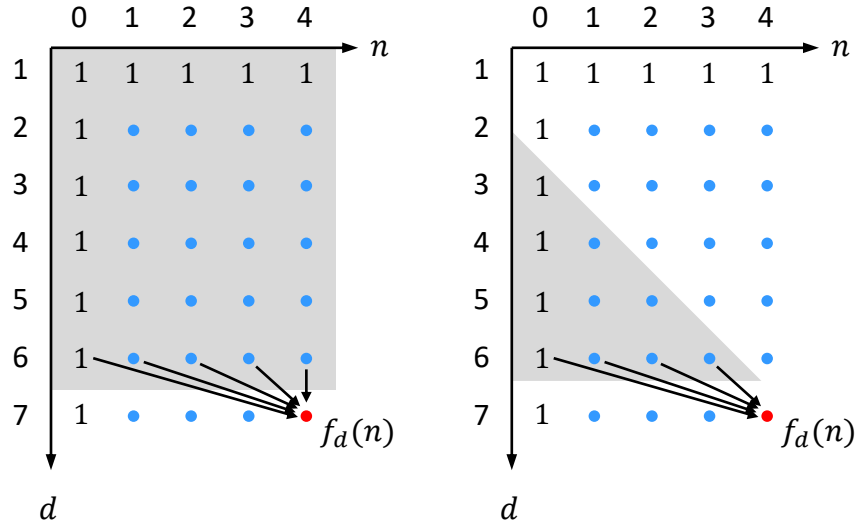
Figure 1. Computational dependencies for two different recursive formulas for $f_d(n)$, the number of abelian squares. (left) Dependency graph for eq. (3), obtained from [9]. (right) Dependency graph for eq. (16). In each case, the desired quantity $f_d(n)$ is shown as a red dot, arrows show the direct dependencies, and the gray shaded region covers all the quantities that must be calculated to determine $f_d(n)$. The pattern on the left leads to a cost of $O(n^2 d)$, while the pattern on the right leads to a cost of $O(n^2 \min(n, d))$.
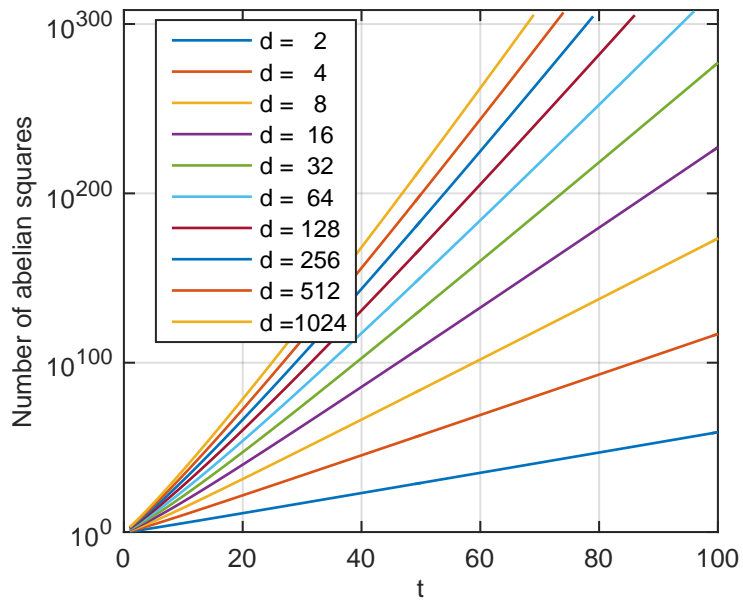

Figure 2. Number $f_d(n)$ of abelian squares of length $n + n$ over an alphabet of size d.

———

[1] Paul Erdős. Some unsolved problems. *Michigan Math. J.*, 4(3):291–300, 1957. ISSN 0026-2285, 1945-2365. doi:10.1307/mmj/1028997963.

[2] Veikko Keränen. Abelian squares are avoidable on 4 letters. In W. Kuich, editor, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 41–52, Berlin, Heidelberg, 1992. Springer. ISBN 978-3-540-47278-0. doi:10.1007/3-540-55719-9-62.

[3] Costas S. Iliopoulos, Dennis Moore, and W. F. Smyth. A characterization of the squares in a Fibonacci string. *Theoretical Computer Science*, 172(1):281–291, February 1997. ISSN 0304-3975. doi: 10.1016/S0304-3975(96)00141-7.

[4] Arturo Carpi. On the number of Abelian square-free words on four letters. *Discrete Applied Mathematics*, 81(1):155–167, January 1998. ISSN 0166-218X. doi:10.1016/S0166-218X(97)88002-X.

[5] Julien Cassaigne, Gwénaël Richomme, Kalle Saari, and Luca Q. Zamboni. Avoiding abelian powers in binary words with bounded abelian complexity. *Int. J. Found. Comput. Sci.*, 22(04):905–920, June 2011. ISSN 0129-0541. doi:10.1142/S0129054111008489.

[6] Mari Huova, Juhani Karhumäki, and Aleksi Saarela. Problems in between words and abelian words: K-abelian avoidability. *Theoretical Computer Science*, 454:172–177, October 2012. ISSN 0304-3975. doi:10.1016/j.tcs.2012.03.010.

[7] M. Crochemore, C. S. Iliopoulos, T. Kociumaka, M. Kubica, J. Pachocki, J. Radoszewski, W. Rytter, W. Tyczyński, and T. Waleń. A note on efficient computation of all Abelian periods in a string. *Information Processing Letters*, 113(3):74–77, February 2013. ISSN 0020-0190. doi:10.1016/j.ipl.2012.11.001.

[8] Ryan Bennink. Counting abelian squares for a problem in quantum computing. *Journal of Combinatorics*, (submitted).

[9] L. B. Richmond and Jeffrey Shallit. Counting Abelian Squares. *Electron. J. Combin.*, 16(1):R72, June 2009. ISSN 1077-8926. doi:10.37236/161.