

# Using functional equations to enumerate 1324-avoiding permutations

Fredrik Johansson\* and Brian Nakamura†

## Abstract

We consider the problem of enumerating permutations with exactly  $r$  occurrences of the pattern 1324 and derive functional equations for this general case as well as for the pattern avoidance ( $r = 0$ ) case. The functional equations lead to a new algorithm for enumerating length  $n$  permutations that avoid 1324. This approach is used to enumerate the 1324-avoiders up to  $n = 31$ . We also extend those functional equations to account for the number of inversions and derive analogous algorithms.

## 1 Introduction

Let  $a_1 \dots a_k$  be a sequence of  $k$  distinct positive integers. We define the *reduction* of this sequence, denoted by  $\text{red}(a_1 \dots a_k)$ , to be the length  $k$  permutation  $\tau = \tau_1 \dots \tau_k$  that is order-isomorphic to  $a_1 \dots a_k$  (i.e.,  $a_i < a_j$  if and only if  $\tau_i < \tau_j$  for every  $i$  and  $j$ ). Given a (permutation) pattern  $\tau \in \mathcal{S}_k$ , we say that a permutation  $\pi = \pi_1 \dots \pi_n$  *contains* the pattern  $\tau$  if there exists  $1 \leq i_1 < i_2 < \dots < i_k \leq n$  such that  $\text{red}(\pi_{i_1} \pi_{i_2} \dots \pi_{i_k}) = \tau$ , in which case we call  $\pi_{i_1} \pi_{i_2} \dots \pi_{i_k}$  an *occurrence* of  $\tau$ . We also define  $N_\tau(\pi)$  to be the number of occurrences of pattern  $\tau$  in the permutation  $\pi$ . For example, if the pattern  $\tau = 123$ , the permutation 53412 avoids the pattern  $\tau$  (so  $N_{123}(53412) = 0$ ), whereas the permutation 52134 contains two occurrences of  $\tau$  (so  $N_{123}(52134) = 2$ ).

For a pattern  $\tau$  and non-negative integer  $r \geq 0$ , we define the set

$$\mathcal{S}_n(\tau, r) := \{\pi \in \mathcal{S}_n : \pi \text{ has exactly } r \text{ occurrences of the pattern } \tau\}$$

and also define  $s_n(\tau, r) := |\mathcal{S}_n(\tau, r)|$ . For the  $r = 0$  case, we say that the two patterns  $\sigma$  and  $\tau$  are *Wilf-equivalent* if  $s_n(\sigma, 0) = s_n(\tau, 0)$  for all  $n$ . Additionally, two patterns that are Wilf-equivalent are said to belong to the same *Wilf-equivalence class*. Note that the  $r = 0$  case corresponds with “classical” pattern avoidance, which has been well-studied. Work on the more general problem ( $r \geq 0$ ) has usually been restricted to patterns of length 3 and small  $r$ . Some recent work include [6, 5, 7, 8, 11, 14, 19, 20].

A little more is known for the pattern avoidance problem, but the problem quickly gets difficult. For each  $\tau \in \mathcal{S}_3$ , it is well known that  $s_n(\tau, 0) = \frac{1}{n+1} \binom{2n}{n}$  (the Catalan numbers) [13]. For the length 4 patterns, there are three cases (Wilf-equivalence classes) to consider: 1234, 1342, and 1324. The enumeration for the 1234-avoiding permutations was solved by Gessel in [12]. Later, Bóna solved the case for the 1342-avoiding permutations in [4]. The pattern 1324, however, has been notoriously difficult to enumerate.

There is currently no non-recursive formula known for computing  $s_n(1324, 0)$ , and precise asymptotics are not known either. Marinov and Radoičić developed an approach in [16] using

\*RISC, Johannes Kepler University, 4040 Linz, Austria. [fredrik.johansson@risc.jku.at]

†CCICADA, Rutgers University-New Brunswick, Piscataway, NJ, USA. [bnaka@dimacs.rutgers.edu]

generating trees and computed  $s_n(1324, 0)$  for  $n \leq 20$ . Another approach (using insertion encoding) was used by Albert et al. in [1] to compute  $s_n(1324, 0)$  for  $n \leq 25$  (five more terms). Given the difficulty of this pattern, Zeilberger has even conjectured that “Not even God knows  $s_{1000}(1324, 0)$ ” [10].

Given the difficulty of exact enumeration, work has also been done on studying how the sequence  $s_n(\tau, 0)$  grows for various patterns. We define the *Stanley-Wilf limit* of a pattern  $\tau$  to be:

$$L(\tau) := \lim_{n \rightarrow \infty} (s_n(\tau, 0))^{1/n}. \quad (1)$$

Thanks to results by Arratia [2] and Marcus and Tardos [15], we know that for each pattern  $\tau$ , the limit  $L(\tau)$  exists and is finite. For patterns of length three, the Stanley-Wilf limit is known to be 4. Additionally, Regev [21] showed that  $L(1234) = 9$ , while Bóna’s result in [4] gives us  $L(1342) = 8$ . The exact limit for the pattern 1324, however, is still unknown. The best known lower bound is by Albert et al. [1], who showed that  $L(1324) \geq 9.47$ . The best known upper bound has seen some improvements in recent years. The recent “best” upper bound was improved by Claesson, Jelínek, and Steingrímsson in [9] to  $L(1324) \leq 16$ . That approach was then refined by Bóna in [3] to show that  $L(1324) < (7 + 4\sqrt{3}) \approx 13.93$ .

Additionally, Claesson, Jelínek, and Steingrímsson conjectured that the number of length  $n$  permutations avoiding 1324 with exactly  $k$  inversions was non-decreasing in  $n$  (for each fixed  $k$ ). They show that if this conjecture holds, then  $L(1324) \leq e^{\pi\sqrt{2/3}} \approx 13.002$ . Neither the current lower bound nor this potential new upper bound appear to be “close” to the exact value of  $L(1324)$ . For example, Steingrímsson’s survey paper [23] considers empirical data and suggests that it may be close to 11. Some data we consider in this paper suggests a similar story.

The paper is organized in the following manner. In Section 2, we derive functional equations for computing  $s_n(1324, r)$ . Furthermore, the approach is specialized to the avoidance case ( $r = 0$ ) to derive an algorithm for enumerating the 1324-avoiding permutations. In Section 3, we describe technical details on the algorithm and use it to compute  $s_n(1324, 0)$  up to  $n = 31$ , giving us 6 new terms. We use this new data to make some empirical observations. In Section 4, we extend the functional equations to keep track of the number of inversions. We conclude with a few final remarks in Section 5.

## 2 Functional equations for the pattern 1324

We begin by extending the functional equations approach in [17] to the pattern 1324. We will first derive functional equations that can be used to compute  $s_n(1324, r)$ . The approach will be presented in full detail so that this article is self-contained. We then specialize this approach to the  $r = 0$  case and develop a new algorithm for enumerating the 1324-avoiding permutations.

### 2.1 A general approach to $s_n(1324, r)$

Given a non-negative integer  $n$ , we define the polynomial (in the variable  $t$ )

$$f_n(t) := \sum_{\pi \in \mathcal{S}_n} t^{N_{1324}(\pi)}. \quad (2)$$

Observe that the coefficient of  $t^r$  in  $f_n(t)$  is exactly equal to  $s_n(1324, r)$ .

In addition to the variable  $t$ , we introduce  $n(n+1)/2$  catalytic variables  $x_{i,j}$  with  $1 \leq i \leq j \leq n$  and  $n(n+1)/2$  catalytic variables  $y_{i,j}$  with  $1 \leq j \leq i \leq n$ . Note that the subscripts of the two

sets of catalytic variables range over different quantities. We define the weight of a permutation  $\pi = \pi_1 \dots \pi_n$  to be

$$\text{weight}(\pi) := t^{N_{1324}(\pi)} \prod_{1 \leq i \leq j \leq n} x_{i,j}^{\#\{(a,b) : \pi_a < \pi_b, \pi_a = i, \pi_b > j\}} \cdot \prod_{1 \leq j \leq i \leq n} y_{i,j}^{\#\{(a,b,c) : \pi_b < \pi_a < \pi_c, \pi_a = i, \pi_b \geq j\}}$$

where it is always assumed that  $1 \leq a < b < c \leq n$ . For example,

$$\begin{aligned} \text{weight}(213) &= x_{1,1} x_{1,2} x_{2,1} x_{2,2} \cdot y_{2,1} \\ \text{weight}(41325) &= t \cdot x_{1,1}^3 x_{1,2}^2 x_{1,3} x_{1,4} x_{2,2} x_{2,3} x_{2,4} x_{3,3} x_{3,4} x_{4,4} \cdot y_{3,1} y_{3,2} y_{4,1}^3 y_{4,2}^2 y_{4,3} \end{aligned}$$

In essence, the weight stores information about 1324 patterns, 213 patterns (which may become the “324” of a 1324 occurrence), and 12 patterns (which may become the “13” of a 213 occurrence) through the exponents of the variable  $t$ , the variables  $y_{i,j}$ , and the variables  $x_{i,j}$ , respectively.

We will define a multi-variate polynomial  $P_n$  on all the previously defined variables. For notational convenience, we first write the  $x_{i,j}$  variables and the  $y_{i,j}$  variables as matrices of variables:

$$X_n := \begin{bmatrix} x_{1,1} & & \cdots & & x_{1,n} \\ & \ddots & & & \\ \vdots & & x_{i,i} & & \vdots \\ & & & \ddots & \\ x_{n,1} & & \cdots & & x_{n,n} \end{bmatrix}, \quad Y_n := \begin{bmatrix} y_{1,1} & & \cdots & & y_{1,n} \\ & \ddots & & & \\ \vdots & & y_{i,i} & & \vdots \\ & & & \ddots & \\ y_{n,1} & & \cdots & & y_{n,n} \end{bmatrix} \quad (3)$$

where we will disregard the entries below the diagonal in  $X_n$  and the entries above the diagonal in  $Y_n$ .

For each  $n$ , we now define the multi-variate polynomial:

$$P_n(t; X_n, Y_n) := \sum_{\pi \in \mathcal{S}_n} \text{weight}(\pi)$$

Observe that  $P_n(t; \mathbf{1}_n, \mathbf{1}_n) = f_n(t)$  is our desired polynomial, where  $\mathbf{1}_n$  is the  $n \times n$  matrix of all 1's. For a fixed  $r \geq 0$ , our goal is to quickly compute the coefficient of  $t^r$  in  $P_n(t; \mathbf{1}_n, \mathbf{1}_n)$ , which is exactly  $s_n(1324, r)$ . We will do this by deriving a functional equation for  $P_n$ . This follows readily from the following result:

**Lemma 1.** *Let  $\pi = \pi_1 \dots \pi_n$  and suppose that  $\pi_1 = i$ . If  $\pi' := \text{red}(\pi_2 \dots \pi_n)$ , then*

$$\text{weight}(\pi) = x_{i,i}^{n-i} x_{i,i+1}^{n-i-1} \dots x_{i,n-1}^1 \cdot \text{weight}(\pi')|_{A'} \quad ,$$

where  $A'$  is the set of substitutions given by

$$A' := \begin{cases} x_{b,c} \rightarrow x_{b,c+1} & b < i, c \geq i \\ x_{b,c} \rightarrow x_{b+1,c+1} & b \geq i, c \geq i \\ x_{b,c} \rightarrow y_{i,1} y_{i,2} \dots y_{i,b} \cdot x_{b,c} \cdot x_{b,c+1} & b < i, c = i - 1 \\ y_{b,c} \rightarrow y_{b+1,c} & b \geq i, c < i \\ y_{b,c} \rightarrow y_{b+1,c+1} & b \geq i, c > i \\ y_{b,c} \rightarrow t y_{b+1,c} \cdot y_{b+1,c+1} & b \geq i, c = i \end{cases} .$$

*Proof.* We assume  $i$  to be a fixed value. Observe that  $N_{1324}(\pi)$  is equal to the number of occurrences of 1324 in  $\pi_2 \dots \pi_n$  **plus** the number of occurrences of 213 in  $\pi_2 \dots \pi_n$ , where the term corresponding to the “1” is greater than  $i$ .

If we re-insert  $i$  at the beginning of  $\pi'$ , we would shift all the terms  $i, i+1, \dots, n-1$  up by 1. This gives us the substitutions:

$$\begin{array}{ll} x_{b,c} \rightarrow x_{b,c+1} & b < i, c \geq i \\ x_{b,c} \rightarrow x_{b+1,c+1} & b \geq i, c \geq i \\ y_{b,c} \rightarrow y_{b+1,c} & b \geq i, c < i \\ y_{b,c} \rightarrow y_{b+1,c+1} & b \geq i, c > i. \end{array}$$

We make a few more observations. First, in  $\text{weight}(\pi)$ , the exponents of  $x_{k,i-1}$  and  $x_{k,i}$  are equal and the exponents of  $y_{k,i}$  and  $y_{k,i+1}$  are equal for each  $k$  (since  $\pi_1 = i$ ). This gives the substitutions  $x_{b,i-1} \rightarrow x_{b,i-1} \cdot x_{b,i}$  if  $b < i$  and  $y_{b,i} \rightarrow y_{b+1,i} \cdot y_{b+1,i+1}$  if  $b \geq i$ .

Second, the number of 1324 patterns that include the first term  $\pi_1 = i$  is the sum of the exponents of  $y_{j,i+1}$  for  $i+1 \leq j \leq n$ . The substitution  $y_{b,i} \rightarrow y_{b+1,i} \cdot y_{b+1,i+1}$  changes to  $y_{b,i} \rightarrow t y_{b+1,i} \cdot y_{b+1,i+1}$  (for  $b \geq i$ ).

Third, the number of 213 patterns that include the first term  $\pi_1 = i$  (i.e., the “2” term is equal to  $i$ ) and whose “1” term is at least  $k$  is equal to the sum of the exponent of  $x_{j,i}$  for  $k \leq j \leq i-1$ . The substitution  $x_{b,i-1} \rightarrow x_{b,i-1} \cdot x_{b,i}$  changes to  $x_{b,i-1} \rightarrow y_{i,1} y_{i,2} \dots y_{i,b} \cdot x_{b,i-1} \cdot x_{b,i}$  (for  $b < i$ ).

This gives us our substitution set  $A'$ . Finally, the new “ $i$ ” would create new 12 patterns and would require an extra factor of  $x_{i,i}^{n-i} x_{i,i+1}^{n-i-1} \dots x_{i,n-1}^1$  for the weight.  $\square$

Now, we define the operator  $R_1$  on an arbitrary  $n \times n$  square matrix  $Y_n$  and  $i < n$  to be:

$$R_1(Y_n, i) := \begin{bmatrix} y_{1,1} & \cdots & y_{1,i-1} & t y_{1,i} y_{1,i+1} & y_{1,i+2} & \cdots & y_{1,n} \\ \vdots & \ddots & & \vdots & & & \vdots \\ y_{i-1,1} & & y_{i-1,i-1} & & \cdots & & y_{i-1,n} \\ y_{i+1,1} & \cdots & y_{i+1,i-1} & t y_{i+1,i} y_{i+1,i+1} & y_{i+1,i+2} & \cdots & y_{i+1,n} \\ \vdots & & \vdots & \vdots & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & & \ddots & \vdots \\ y_{n,1} & \cdots & y_{n,i-1} & t y_{n,i} y_{n,i+1} & y_{n,i+2} & \cdots & y_{n,n} \end{bmatrix}. \quad (4)$$

If  $i = n$ , then  $R_1(Y_n, i)$  is defined to be the  $(n-1) \times (n-1)$  matrix obtained by deleting the  $n$ -th row and  $n$ -th column from  $Y_n$ . In essence, the  $R_1$  operator deletes the  $i$ -th row, merges the  $i$ -th and  $(i+1)$ -th columns (via term-by-term multiplication), and multiplies this new column by a factor of  $t$ . It is important to note that while this operator is defined on any  $n \times n$  matrix, it will only be applied to our “matrix of variables”  $Y_n$  to get a smaller  $(n-1) \times (n-1)$  matrix.

In addition, we define another operator  $R_2$  on two  $n \times n$  square matrices  $X_n$  and  $Y_n$  and  $1 < i \leq n$  to be:

$$R_2(X_n, Y_n, i) := \begin{bmatrix} x_{1,1} & \cdots & x_{1,i-2} & w_1 & x_{1,i+1} & \cdots & x_{1,n} \\ \vdots & \ddots & & \vdots & & & \vdots \\ x_{i-2,1} & \cdots & x_{i-2,i-2} & w_{i-2} & x_{i-2,i+1} & \cdots & x_{i-2,n} \\ x_{i-1,1} & \cdots & x_{i-1,i-2} & w_{i-1} & x_{i-1,i+1} & \cdots & x_{i-1,n} \\ x_{i+1,1} & \cdots & x_{i+1,i-2} & w_{i+1} & x_{i+1,i+1} & \cdots & x_{i+1,n} \\ \vdots & & \vdots & \vdots & & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,i-2} & w_n & x_{n,i+1} & \cdots & x_{n,n} \end{bmatrix} \quad (5)$$

where

$$w_k := \begin{cases} y_{i,1}y_{i,2} \cdots y_{i,k} \cdot x_{k,i-1} \cdot x_{k,i} & k \leq i-1 \\ 0 & k > i-1 \end{cases} .$$

If  $i = 1$ , then  $R_2(X_n, Y_n, i)$  is defined to be the  $(n-1) \times (n-1)$  matrix obtained by deleting the 1-st row and 1-st column from  $X_n$ . In essence, the  $R_2$  operator modifies  $X_n$  by deleting the  $i$ -th row, merging the  $(i-1)$ -th column with the  $i$ -th column (via term-by-term multiplication), and scaling that new column by products of terms from  $Y_n$ .

Lemma 1 now directly leads to the following:

**Theorem 1.** *For the pattern  $\tau = 1324$ ,*

$$P_n(t; X_n, Y_n) = \sum_{i=1}^n x_{i,i}^{n-i} x_{i,i+1}^{n-i-1} \cdots x_{i,n-1}^1 \cdot P_{n-1}(t; R_2(X_n, Y_n, i), R_1(Y_n, i)). \quad (\text{FE1324})$$

Although all the entries in the matrices are changed for consistency and notational convenience, we will continue to disregard the entries below the diagonal in subsequent matrices  $X_k$  and the entries above the diagonal in subsequent matrices  $Y_k$ . We can apply the same computational tricks shown in [18, 17]. For example, it is not necessary to compute  $P_n(t; X_n, Y_n)$  completely symbolically and substitute  $x_{i,j} = 1$  and  $y_{i,j} = 1$  at the end. Instead, we can apply functional equation (FE1324) directly to  $P_n(t; \mathbf{1}_n, \mathbf{1}_n)$  and subsequent  $P_k$  terms. We may also use the following lemma from [17], which is obvious from the definition of the operator  $R_1$ :

**Lemma 2.** *Let  $A$  be a square matrix where every row is identical (i.e., the  $i$ -th row and the  $j$ -th row are equal for every  $i, j$ ). Then,  $R_1(A, i)$  will also be a square matrix with identical rows.*

By Lemma 2, repeated applications of the  $R_1$  operator to the all ones matrix  $\mathbf{1}_n$  will still result in a matrix with identical rows. It is therefore sufficient to keep track of only a single row. It is also helpful to note that repeated applications of  $R_1$  to the matrix  $\mathbf{1}_n$  will result in a matrix whose entries are powers of  $t$ .

While the lemma does not apply to the  $R_2$  operator, this still allows us to simplify the polynomial  $P_n$  and its functional equation by reducing the number of catalytic variables from  $n(n+1)$  variables to  $n(n+1)/2 + n$  variables. Let  $Q_n(t; C; d_1, \dots, d_n)$  denote the polynomial  $P_n(t; C, D)$  where every entry of the  $n \times n$  matrices  $C$  and  $D$  are powers of  $t$  and every row in  $D$  is  $[d_1, \dots, d_n]$ . We get the analogous functional equation:

$$Q_n(t; C; d_1, \dots, d_n) = \sum_{i=1}^n c_{i,i}^{n-i} c_{i,i+1}^{n-i-1} \cdots c_{i,n-1}^1 \cdot Q_{n-1}(t; R_2(C, D, i); d_1, \dots, d_{i-1}, t d_i d_{i+1}, d_{i+2}, \dots, d_n). \quad (\text{FE1324c})$$

Repeatedly applying this recurrence to  $Q_n(t; \mathbf{1}_n; 1 [n \text{ times}])$  allows us to compute our desired polynomial since  $Q_n(t; \mathbf{1}_n; 1 [n \text{ times}]) = P_n(t; \mathbf{1}_n, \mathbf{1}_n) = f_n(t)$ . Extracting the coefficient of  $t^r$  from this polynomial gives us  $s_n(1324, r)$ .

Additionally, for a fixed  $r$ , the sequence  $s_n(1324, r)$  can be computed more quickly by discarding higher powers of  $t$  (just as in [18, 17]). Although this approach is too memory intensive for larger  $r$ , for small  $r$ , this method is still much faster than naive methods that construct the set  $\mathcal{S}_n(\tau, r)$ . The approach has been implemented in the procedure `F1324rN(r, N)` in the accompanying Maple package `F1324`.

For example, the Maple call `F1324rN(0, 19)`; for the first 19 terms of  $s_n(1324, 0)$  produces the sequence:

$$1, 2, 6, 23, 103, 513, 2762, 15793, 94776, 591950, 3824112, 25431452, 173453058, \\ 1209639642, 8604450011, 62300851632, 458374397312, 3421888118907, 25887131596018$$

and the Maple call `F1324rN(1,17)`; for the first 17 terms of  $s_n(1324, 1)$  produces the sequence:

$$0, 0, 0, 1, 10, 75, 522, 3579, 24670, 172198, 1219974, 8776255, \\ 64082132, 474605417, 3562460562, 27079243352, 208281537572.$$

## 2.2 Specializing to $r = 0$

Unfortunately, the previous algorithm developed for the pattern 1324 is very memory intensive, even for  $r = 0$ . In this subsection, we outline how to extract a simpler recurrence specifically for the pattern avoidance case.

We will specialize for the  $r = 0$  case beginning at functional equation (FE1324c). Recall that  $Q_n(t; C; d_1, \dots, d_n)$  is the polynomial given by  $P_n(t; C, D)$  where every entry of the  $n \times n$  matrices  $C$  and  $D$  are powers of  $t$  and every row in  $D$  is  $[d_1, \dots, d_n]$ . We had the functional equation

$$Q_n(t; C; d_1, \dots, d_n) = \\ \sum_{i=1}^n c_{i,i}^{n-i} c_{i,i+1}^{n-i-1} \dots c_{i,n-1}^1 \cdot Q_{n-1}(t; R_2(C, D, i); d_1, \dots, d_{i-1}, td_i d_{i+1}, d_{i+2}, \dots, d_n)$$

and wanted to compute  $Q_n(t; \mathbf{1}_n; 1 [n \text{ times}]) = f_n(t)$  and extract the coefficient of  $t^r$ , which is exactly  $s_n(1324, r)$ .<sup>1</sup>

Since all the variables  $c_{k,l}$  (from matrix  $C$ ) and  $d_k$  represent powers of  $t$ , it is sufficient to keep track of powers of  $t$  through most of the algorithm. This allows us to consider the analogous function  $H_n(t; U; v_1, \dots, v_n)$ , where  $U$  is an  $n \times n$  matrix of non-negative integers and each  $v_i$  is a non-negative integer. More precisely,  $H_n(t; U; v_1, \dots, v_n)$  is the polynomial  $P_n(t; C, D)$ , where  $C$  and  $D$  are  $n \times n$  matrices,  $c_{i,j} = t^{u_{i,j}}$  for every  $1 \leq i, j \leq n$ , and every row of  $D$  is  $[t^{v_1}, \dots, t^{v_n}]$ .

In addition, we define the analogous operator  $R'_2$  on an  $n \times n$  square matrix  $U_n$  (of non-negative integers), a length  $n$  vector of non-negative integers  $[v_1, \dots, v_n]$ , and  $1 < i \leq n$ :

$$R'_2(U_n, [v_1, \dots, v_n], i) := \begin{bmatrix} u_{1,1} & \cdots & u_{1,i-2} & w'_1 & u_{1,i+1} & \cdots & u_{1,n} \\ \vdots & \ddots & & \vdots & & & \vdots \\ u_{i-2,1} & \cdots & u_{i-2,i-2} & w'_{i-2} & u_{i-2,i+1} & \cdots & u_{i-2,n} \\ u_{i-1,1} & \cdots & u_{i-1,i-2} & w'_{i-1} & u_{i-1,i+1} & \cdots & u_{i-1,n} \\ u_{i+1,1} & \cdots & u_{i+1,i-2} & w'_{i+1} & u_{i+1,i+1} & \cdots & u_{i+1,n} \\ \vdots & & \vdots & \vdots & & \ddots & \vdots \\ u_{n,1} & \cdots & u_{n,i-2} & w'_n & u_{n,i+1} & \cdots & u_{n,n} \end{bmatrix} \quad (6)$$

where

$$w'_k := \begin{cases} (v_1 + v_2 + \dots + v_k) + u_{k,i-1} + u_{k,i} & k \leq i-1 \\ 0 & k > i-1 \end{cases} \quad (7)$$

If  $i = 1$ , then  $R'_2(U_n, [v_1, \dots, v_n], i)$  is defined to be the  $(n-1) \times (n-1)$  matrix obtained by deleting the 1-st row and 1-st column from  $U_n$ . In essence, the  $R'_2$  operator modifies  $U_n$  by deleting the  $i$ -th row, merging the  $(i-1)$ -th column with the  $i$ -th column (via term-by-term addition), and adding partial sums of  $[v_1, \dots, v_n]$  into the new column.

<sup>1</sup>Recall that  $\mathbf{1}_n$  is the  $n \times n$  matrix where every entry is 1.

We now have the functional equation (analogous to Eq. (FE1324c)):

$$H_n(t; U; v_1, \dots, v_n) = \sum_{i=1}^n t^{e_i} \cdot H_{n-1}(t; R'_2(U, [v_1, \dots, v_n], i); v_1, \dots, v_{i-1}, (v_i + v_{i+1} + 1), v_{i+2}, \dots, v_n) \quad (\text{FE1324e})$$

where  $e_i = (n-i)u_{i,i} + (n-i-1)u_{i,i+1} + \dots + (1)u_{i,n-1}$ . Observe that  $H_n(t; \mathbf{0}_n; 0 [n \text{ times}])$  is now our desired polynomial  $f_n(t)$ .<sup>2</sup>

Since we are specifically considering the  $r = 0$  case, we can make additional observations and simplifications. First, we are only interested in the constant term of  $f_n(t)$ . As in [18, 17], we only need to keep track of polynomials of the form  $a_0 + a_1 t$  in intermediate computations, where  $a_1$  represents permutations with *at least one* occurrence of the pattern we are tracking. Because of this, we may consider all matrices/vectors used in  $H_n$  to be 0-1 matrices/vectors. After every addition (for example, in the  $w'_k$  term in  $R'_2$ ), we can take the minimum of the resulting sum and 1.

Next, observe that  $v_1, \dots, v_n$  only appear in the  $R'_2$  operator, and in particular, in the partial sums for  $w'_k$  in Eq. 7. Suppose that some of the  $v_1, \dots, v_n$  are equal to 1, and let  $j$  be the smallest number such that  $v_j = 1$ . Then,

$$H_n(t; U; v_1, \dots, v_n)|_{t=0} = H_n(t; U; 0 [j-1 \text{ times}], 1 [n-j+1 \text{ times}])|_{t=0}.$$

In particular, the variables  $v_1, \dots, v_n$  are unnecessary, and it is sufficient to keep track of how many 0's there are. We can then consider this slightly simpler function

$$\tilde{H}_n(t; U; k) := H_n(t; U; 0 [k \text{ times}], 1 [n-k \text{ times}])$$

where  $0 \leq k \leq n$ .

Finally, observe that whenever  $e_i > 0$  in (FE1324e), we can discard the entire term since we are only interested in the constant term of the final polynomial. Observe that  $e_i > 0$  if and only if  $u_{i,j} > 0$  for some  $j \geq i$ . This observation (combined with how the  $R'_2$  operator “modifies” the matrix  $U_n$ ) implies that we only need to keep track of the left-most 1 within each row of  $U_n$ . If there are multiple 1's on a row, the left-most 1 is sufficient to force  $e_i > 0$  as long as it is not in the  $n$ -th column. Therefore, we can consider a function of the form

$$H_n^0(t; b_1, \dots, b_n; k) := \tilde{H}_n(t; B_n; k) = H_n(t; B_n; 0 [k \text{ times}], 1 [n-k \text{ times}])$$

where  $0 \leq k \leq n$  and  $1 \leq b_j \leq n+1$  for each  $j$  and  $B_n$  is the  $n \times n$  matrix where the  $j$ -th row is  $[0 [n \text{ times}]]$  if  $b_j = n+1$  and otherwise is  $[0 [b_j-1 \text{ times}], 1 [n-b_j+1 \text{ times}]]$ .

This approach is implemented in the Maple package `F1324` in the procedure `AV1324(n)`. An improved implementation in C++ is provided in the program `av1324.cpp` and is discussed in greater detail in the next section.

### 3 Computational details and results

#### 3.1 Algorithmic details for enumerating 1324-avoiders

For notational convenience, let  $a_n \equiv s_n(1324, 0)$  denote the number of 1324-avoiding permutations of length  $n$ . The values  $a_n$  up to  $n = 25$  were previously computed in [1] and subsequently listed in A061552 of OEIS ([22]). To extend this list, we have written a C++ implementation of the

<sup>2</sup>We denote the  $n \times n$  matrix consisting of all zeros by  $\mathbf{0}_n$ .

algorithm described in the previous section. The implementation is in the program `av1324.cpp`, which is available from the authors' websites.

The main part of the program is a recursive function  $G(n, k, b) \equiv H_n^0(t; b_1, \dots, b_n; k)$  which takes as input an integer  $n \leq 1$ , an integer  $0 \leq k \leq n$ , and a vector of integers  $b = [\tilde{b}_0, \dots, \tilde{b}_{n-1}]$  satisfying  $0 \leq \tilde{b}_j \leq n$  (these correspond to the  $b_j$  in  $H_n^0$  as  $\tilde{b}_j = b_{j+1} - 1$ , being zero-aligned since this is more natural in C++). We represent  $b$  as an array of bytes, zero-padded to a fixed maximum length of 32 (sufficient to compute  $a_n$  up to  $n = 32$ ). The output of  $G$  is an integer, which we represent as a 128-bit unsigned integer (sufficient to compute  $a_n$  up to at least  $n = 34$ , since  $a_{34} < 34! < 2^{128}$  and all recursive calls to  $G$  must produce values that are no larger than the output).

We use full memorization to reduce the number of recursive calls that have to be made. The `std::map` type in the C++ standard library is used to associate vectors  $b$  with output values  $G(n, k, b)$ , using one such map for each pair  $n, k$ . The `std::map` type implements a self-balancing binary tree with  $O(\log N)$  insertion and lookup time where  $N$  is the size of the cache. The keys  $b$  are compared lexicographically by casting to 64-bit integers and processing eight bytes at a time.

We compiled the program with GCC 4.3.4 and ran it on the MACH computer at the Johannes Kepler University of Linz, using a 2.66 GHz Intel Xeon E7-8837 CPU with 1024 GiB of memory allocated to the process. The memory limit allowed computing  $a_n$  up to  $n = 31$ . The 6 new terms are:

$$\begin{aligned} a_{26} &= 49339914891701589053 \\ a_{27} &= 402890652358573525928 \\ a_{28} &= 3313004165660965754922 \\ a_{29} &= 27424185239545986820514 \\ a_{30} &= 228437994561962363104048 \\ a_{31} &= 1914189093351633702834757 \end{aligned}$$

We computed all  $a_n$  consecutively in one run to benefit from already cached function values (computing a single  $a_n$  in isolation would not give any significant memory savings). With  $a_1, \dots, a_{30}$  already computed, the evaluation of  $a_{31}$  took 33 hours and used 920 GiB of memory, and the computation as a whole took 50 hours.

Detailed results from the computation are presented in Table 4 (in the Appendix). Besides the running time and total memory usage, we record the number of cache hits (calls to  $H$  replaced by cache lookups) and the number of cache misses (calls to  $G$  that require evaluation). Up to  $n = 15$ , we also show the number of calls to  $G$  used by the algorithm with caching disabled, measured in a separate run of the program.

All quantities appear to grow slightly faster than exponentially. Over the measured range, the number of function calls (as well as the size of the cache in memory) is roughly proportional to  $2.2^n$ . If the memory overhead of the implementation were reduced by half (or if we had a computer with twice as much memory), we could thus compute roughly one more entry. With caching disabled, the number of recursive function calls grows much faster, making this version impractical (for  $n \leq 15$ , the number of calls is roughly  $2.4a_n$ ).

At present, we do not know if the algorithm can be modified to significantly reduce the memory consumption required by full memorization, without significantly increasing the running time. Such a modification might allow computing several more entries in the sequence  $\{a_n\}$ , particularly if the job could be parallelized.



### 3.2 Observations on the asymptotics

Since the enumeration problem is solved for the patterns 1234 and 1342, it is not hard to derive asymptotic information on the sequences enumerating their respective pattern avoiders. For the pattern 1234, we have

$$s_n(1234, 0) \sim 9^n n^{-4}$$

and for the pattern 1342, we have

$$s_n(1342, 0) \sim 8^n n^{-5/2}.$$

The convergence occurs fairly quickly, even when observing the first 31 terms. On the other hand, it is not even known if the asymptotics for  $s_n(1324, 0)$  look like  $\mu^n n^\theta$  (for constants  $\mu$  and  $\theta$ ). Shalosh B. Ekhad was kind enough to compute some numerical data for the authors.

Using the first 29 terms, the first 30 terms, and the first 31 terms, the approximate value for  $\theta$  and  $\mu$  for the patterns 1234 and 1342 are:

n	$\theta$	$\mu$
29	-3.990065278	8.978066441
30	-3.990767318	8.979528508
31	-3.991374852	8.980845382

Table 1: Approximate values for  $\theta$  and  $\mu$  using the first  $n = 29, 30, 31$  terms of  $s_n(1234, 0)$ .

n	$\theta$	$\mu$
29	-2.507234370	7.987629199
30	-2.506672206	7.988446482
31	-2.506140202	7.989181549

Table 2: Approximate values for  $\theta$  and  $\mu$  using the first  $n = 29, 30, 31$  terms of  $s_n(1342, 0)$ .

However, when the same guessing is done for the pattern 1324, the convergence is much slower. The values are:

n	$\theta$	$\mu$
29	-8.365614110	10.40595402
30	-8.506078382	10.42830233
31	-8.643316748	10.44936383

Table 3: “Approximate” values for  $\theta$  and  $\mu$  using the first  $n = 29, 30, 31$  terms of  $s_n(1324, 0)$ .

It should be re-emphasized that it is not known whether this sequence fits the asymptotic form of  $\mu^n n^\theta$ . The empirical asymptotics suggests that either the convergence is much slower than the other length 4 patterns or that the sequence does not have that asymptotic form to begin with (unlike the other patterns). In addition, the  $\mu$  values would suggest that the Stanley-Wilf limit  $L(1324)$  is at least 10.45. More detailed numerical data on the asymptotics (from Shalosh B. Ekhad) can be found on the authors’ websites.

## 4 Extending to inversions

In this section, we show how the previous functional equations can be adapted to refine the values by the number of inversions. The number of inversions in a permutation is one of the most commonly studied permutation statistic and in essence, quantifies how “unsorted” a permutation is. Given a permutation  $\pi = \pi_1 \dots \pi_n$ , the *inversion number* of  $\pi$ , denoted by  $\text{inv}(\pi)$ , is the number of pairs

$(i, j)$  such that  $1 \leq i < j \leq n$  and  $\pi_i > \pi_j$ . An equivalent definition is that  $\text{inv}(\pi) = N_{21}(\pi)$ , the number of 21 patterns in  $\pi$ .

We again consider the pattern 1324. For each  $n$ , we define the bivariate polynomial

$$g_n(t, q) := \sum_{\pi \in \mathcal{S}_n} q^{\text{inv}(\pi)} t^{N_{1324}(\pi)}.$$

Observe that  $g_n(t, 1)$  is exactly  $f_n(t)$  from Section 2.

For the pattern  $\tau = 1324$ , the polynomial  $P_n$  can now be “generalized” as

$$P_n(t, q; X_n, Y_n) := \sum_{\pi \in \mathcal{S}_n} q^{\text{inv}(\pi)} \text{weight}(\pi).$$

Note that  $P_n(t, 1; X_n, Y_n)$  is exactly  $P_n(t; X_n, Y_n)$  from Section 2.

We now make an important observation. Given a permutation  $\pi = \pi_1 \dots \pi_n$ , suppose that  $\pi_1 = i$ . Then,  $\text{inv}(\pi)$  is equal to the number of inversions in  $\pi_2 \dots \pi_n$  **plus** the number of terms in  $\pi_2 \dots \pi_n$  that are less than  $i$  (which is exactly  $i - 1$ ). For any previously define functional equation, it is enough to insert a factor of  $q^{i-1}$  into the summation.

We can now quickly derive the modified functional equations for the pattern. The functional equation (FE1324) now becomes:

**Corollary 1.** *For the pattern  $\tau = 1324$ ,*

$$P_n(t, q; X_n, Y_n) = \sum_{i=1}^n q^{i-1} x_{i,i}^{n-i} x_{i,i+1}^{n-i-1} \dots x_{i,n-1}^1 \cdot P_{n-1}(t, q; R_2(X_n, Y_n, i), R_1(Y_n, i)). \quad (\text{qFE1324})$$

Similarly, the functional equation (FE1324c) for  $Q_n(t; C; d_1, \dots, d_n)$  now becomes the analogous:

**Corollary 2.** *For the pattern  $\tau = 1324$ ,*

$$Q_n(t, q; C; d_1, \dots, d_n) = \sum_{i=1}^n q^{i-1} c_{i,i}^{n-i} c_{i,i+1}^{n-i-1} \dots c_{i,n-1}^1 \cdot Q_{n-1}(t, q; R_2(C, D, i); d_1, \dots, d_{i-1}, t d_i d_{i+1}, d_{i+2}, \dots, d_n). \quad (\text{qFE1324c})$$

The functional equation (FE1324e) for  $H_n(t; U; v_1, \dots, v_n)$ , which corresponds specifically to the pattern-avoidance case, also becomes the analogous:

**Corollary 3.** *For the pattern  $\tau = 1324$ ,*

$$H_n(t, q; U; v_1, \dots, v_n) = \sum_{i=1}^n q^{i-1} t^{e_i} \cdot H_{n-1}(t, q; R'_2(U, [v_1, \dots, v_n], i); v_1, \dots, v_{i-1}, (v_i + v_{i+1} + 1), v_{i+2}, \dots, v_n) \quad (\text{qFE1324e})$$

The enumeration algorithm derived from the functional equation (qFE1324e) has been implemented in the procedure `qAv1324r(n, r, q)` in the Maple package `F1324`. It is also worth noting that the same extension can be done for tracking non-inversions by inserting  $q^{n-i}$  into the summations (instead of  $q^{i-1}$ ). This has also been implemented in the Maple package `F1324` in the procedure `pAv1324r(n, r, p)`.

In [9], Claesson, Jelínek, and Steingrímsson consider refining the number of 1324-avoiding permutations by the inversion number to prove a new upper bound on the Stanley-Wilf limit. They conjectured that for each fixed  $k$ , the number of 1324-avoiders with exactly  $k$  inversions is non-decreasing in  $n$ . If this conjecture is proven true, their result would improve the upper bound of the growth rate. Using our algorithm, we are able to empirically confirm this conjecture for  $n, k \leq 23$ , and the explicit values can be found on the authors' websites. We suspect that a more careful analysis of the functional equations and the algorithm may lead to new insights regarding this conjecture.

## 5 Conclusion

In this paper, we extended the functional equations approach of [18, 17] to derive functional equations as well as an enumeration algorithm for computing  $s_n(1324, r)$ . We were able to specialize this approach to  $r = 0$  to develop a new enumeration algorithm for computing the number of 1324-avoiding permutations. This new approach was used to compute 6 new terms for the sequence as well as make some empirical observations on the asymptotics. The functional equations were also extended to refine  $s_n(1324, r)$  by the number of inversions.

While some of the key contributions of this paper are algorithms, it is important to note that all the intermediate steps were rigorous functional equations. In particular, the specialized functional equations for enumerating 1324-avoiders can be viewed as recursively defined functions on 0-1 matrices. We suspect that a more careful analysis of these functions may yield insight into the sequence enumerating the 1324-avoiders and perhaps the Stanley-Wilf limit  $L(1324)$ . In addition, the functional equations for tracking inversions were used to confirm the conjecture by Claesson, Jelínek, and Steingrímsson for up to  $n, k \leq 23$ . We suspect that a more careful analysis of these functional equations may also provide insight toward resolving their conjecture, which would then lower the upper bound on  $L(1324)$  to  $e^{\pi\sqrt{2/3}} \approx 13.002$ .

**Acknowledgments:** We would like to thank Doron Zeilberger and Manuel Kauers for their very helpful comments and suggestions. We would also like to thank Shalosh B. Ekhad for computing numerical data on the asymptotics for us.

## References

- [1] M. H. Albert, M. Elder, A. Rechnitzer, P. Westcott, and M. Zabrocki. On the Stanley-Wilf limit of 4231-avoiding permutations and a conjecture of Arratia. *Adv. in Appl. Math.*, 36(2):96–105, 2006.
- [2] Richard Arratia. On the Stanley-Wilf conjecture for the number of permutations avoiding a given pattern. *Electron. J. Combin.*, 6:Note, N1, 4 pp. (electronic), 1999.
- [3] Miklós Bóna. A new upper bound for 1324-avoiding permutations. arxiv:1207.2379 [math.co], 2012.
- [4] Miklós Bóna. Exact enumeration of 1342-avoiding permutations: a close link with labeled trees and planar maps. *J. Combin. Theory Ser. A*, 80(2):257–272, 1997.
- [5] Miklós Bóna. The number of permutations with exactly  $r$  132-subsequences is  $P$ -recursive in the size! *Adv. in Appl. Math.*, 18(4):510–522, 1997.

- [6] Miklós Bóna. Permutations with one or two 132-subsequences. *Discrete Math.*, 181(1-3):267–274, 1998.
- [7] Alexander Burstein. A short proof for the number of permutations containing pattern 321 exactly once. *Electron. J. Combin.*, 18(2):Paper 21, 3, 2011.
- [8] David Callan. A recursive bijective approach to counting permutations containing 3-letter patterns. arxiv:math/0211380 [math.co], 2002.
- [9] Anders Claesson, Vít Jelínek, and Einar Steingrímsson. Upper bounds for the Stanley-Wilf limit of 1324 and other layered patterns. *J. Combin. Theory Ser. A*, 119(8):1680–1691, 2012.
- [10] Murray Elder and Vince Vatter. Problems and conjectures presented at the third international conference on permutation patterns. arxiv:math/0505504 [math.co], 2005.
- [11] Markus Fulmek. Enumeration of permutations containing a prescribed number of occurrences of a pattern of length three. *Adv. in Appl. Math.*, 30(4):607–632, 2003.
- [12] Ira Gessel. Symmetric functions and P-recursiveness. *J. Combin. Theory Ser. A*, 53(2):257–285, 1990.
- [13] Donald E. Knuth. *The art of computer programming*. Vol. 1: Fundamental algorithms. Addison Wesley, Reading, Massachusetts, 1973.
- [14] Toufik Mansour and Alek Vainshtein. Counting occurrences of 132 in a permutation. *Adv. in Appl. Math.*, 28(2):185–195, 2002.
- [15] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Combin. Theory Ser. A*, 107(1):153–160, 2004.
- [16] Darko Marinov and Radoš Radoičić. Counting 1324-avoiding permutations. *Electron. J. Combin.*, 9(2):Research paper 13, 9 pp. (electronic), 2002/03. Permutation patterns (Otago, 2003).
- [17] Brian Nakamura. Approaches for enumerating permutations with a prescribed number of occurrences of patterns. *Pure Math. Appl. (P.U.M.A.)*, to appear.
- [18] Brian Nakamura and Doron Zeilberger. Using Noonan-Zeilberger functional equations to enumerate (in polynomial time!) generalized Wilf classes. *Adv. in Appl. Math.*, 50(3):356–366, 2013.
- [19] John Noonan. The number of permutations containing exactly one increasing subsequence of length three. *Discrete Math.*, 152(1-3):307–313, 1996.
- [20] John Noonan and Doron Zeilberger. The enumeration of permutations with a prescribed number of “forbidden” patterns. *Adv. in Appl. Math.*, 17(4):381–407, 1996.
- [21] Amitai Regev. Asymptotic values for degrees associated with strips of Young diagrams. *Adv. in Math.*, 41(2):115–136, 1981.
- [22] Neil Sloane. The On-Line Encyclopedia of Integer Sequences, <http://oeis.org/>, 2013.
- [23] Einar Steingrímsson. Some open problems on permutation patterns. *London Mathematical Society Lecture Note Series*, to appear.

## Appendix

$n$	$a_n = s_n(1324, 0)$	Cache hits	Cache misses	Calls, cache disabled	
1	1	0	1	1	
2	2	0	4	4	
3	6	4	10	14	
4	23	17	21	54	
5	103	49	41	239	
6	513	121	79	1187	
7	2762	280	153	6417	
8	15793	628	300	36936	
9	94776	1386	595	223190	
10	591950	3032	1194	1402845	
11	3824112	6607	2422	9113389	
12	25431452	14383	4963	60903142	
13	173453058	31328	10260	417167046	
14	1209639642	68314	21375	2920322177	
15	8604450011	149166	44828	20843563430	
16	62300851632	326163	94562		
17	458374397312	714178	200491		
18	3421888118907	1565935	427006	Time, ms	Mem, KiB
19	25887131596018	3438097	913101	$3.4 \cdot 10^2$	$5.2 \cdot 10^4$
20	198244731603623	7558183	1959618	$9.1 \cdot 10^2$	$9.7 \cdot 10^4$
21	1535346218316422	16636000	4219286	$2.5 \cdot 10^3$	$2.0 \cdot 10^5$
22	12015325816028313	36659838	9111542	$6.4 \cdot 10^3$	$4.1 \cdot 10^5$
23	94944352095728825	80876277	19729578	$1.6 \cdot 10^4$	$8.7 \cdot 10^5$
24	757046484552152932	178616038	42827166	$4.1 \cdot 10^4$	$1.9 \cdot 10^6$
25	6087537591051072864	394883523	93177487	$1.0 \cdot 10^5$	$4.0 \cdot 10^6$
26	49339914891701589053	873872819	203150306	$2.6 \cdot 10^5$	$8.7 \cdot 10^6$
27	402890652358573525928	1935710217	443784326	$6.5 \cdot 10^5$	$1.9 \cdot 10^7$
28	3313004165660965754922	4291690537	971213858	$1.6 \cdot 10^6$	$4.2 \cdot 10^7$
29	27424185239545986820514	9523492671	2129084186	$4.3 \cdot 10^6$	$9.1 \cdot 10^7$
30	228437994561962363104048	21150884205	4674743970	$1.4 \cdot 10^7$	$2.0 \cdot 10^8$
31	1914189093351633702834757	47012202538	10279369333	$4.2 \cdot 10^7$	$4.4 \cdot 10^8$
				$1.2 \cdot 10^8$	$9.6 \cdot 10^8$

Table 4: Results of the computation of the number of 1324-avoiding permutations of length  $n$  using our C++ implementation. The table includes the number of calls to the function  $G$  that result in a cache hit and the number of calls that result in a cache miss. For small  $n$ , the number of evaluations  $G$  that would be required without memorization are presented in the rightmost column. For large  $n$ , the elapsed CPU time and memory usage of the memorized version are shown.