

Approximate Alignments in Rectangular Arrays

Eric H. Kuo*

March 14, 2003

Abstract

An *approximate alignment* within a rectangular array of squares is a subset of marked squares that obey the rule that each marked square has at least one marked square to the right or above it, and at least one marked square to the left or below it. The motivation for approximate alignments comes from the graph structure that arises in the dynamic programming algorithm for the string matching problem. If we consider only approximate alignments that are horizontally and vertically convex, then a bijection exists with plane partitions. Generating functions are derived to enumerate approximate alignments in an $m \times n$ array with k squares, and it is shown that for $m = 3$, the generating function is unimodal. Finally, a conjecture is made about the order of growth of the numbers of nonconvex approximate alignments.

1 Introduction

1.1 Motivation from String Matching Problem

The structure of approximate alignments arise from the string matching problem. Given an alphabet Σ and two strings $s_1, s_2 \in \Sigma^*$, the goal is to determine a transformation between s_1 and s_2 that minimizes mutations, insertions, and deletions. A transformation is performed by inserting gaps into s_1 and s_2 to form new strings s'_1, s'_2 . These two new strings are then compared, linking one character or gap in s'_1 with another in s'_2 such that adjacent characters in s'_1 are linked to adjacent characters in s'_2 . If two linked characters between s'_1 and s'_2 are the same, we have a *match*. If two linked characters are different, we have a *mismatch*. A link between a gap in s'_1 and a character in s'_2 is an *insertion*, while a link between a character in s'_1 and a gap in s'_2 is a *deletion*. The *edit distance* between s'_1 and s'_2 is a formula $-m + \alpha x + \beta y$ where m, x, y are the numbers of matches, mismatches, and indels (insertions and deletions), and $\alpha, \beta \geq 0$ are penalties assessed for each mismatch and indel, respectively. This formula is devised to reward matches and penalize mismatches and indels. The string matching problem is to find the appropriate places to place the gaps in s_1 and s_2 so that the edit distance is minimized.

The algorithm for computing the optimal alignment is to create a $|s_1| \times |s_2|$ grid, each dimension indexed by the letters in s_1 and s_2 . Directed edges are placed horizontally and vertically between adjacent points; thus a horizontal edge represents an insertion in s_1 but a gap in s_2 , while a vertical edge represents an insertion in s_2 but a gap in s_1 . Diagonal edges are placed between points that are one step horizontal and one step vertical from each other. These diagonal edges represent matches or mismatches between s_1 and s_2 . The edges are weighted by the scores of matches, mismatches, insertions, or deletions. The path between the corners of the grid with the maximum weight can be computed in $O(|s_1| \cdot |s_2|)$ time; the dynamic programming algorithm

*Computer Science Division, Soda Hall #1776, University of California, Berkeley, CA, 94720, USA. E-mail: ekuo@cs.berkeley.edu. This research was supported by NSF Graduate Fellowship.

is known as the Needleman-Wunsch [3] algorithm. The total number of paths between the corners of the grid is recursively given by

$$D(n, m) = D(n - 1, m) + D(n - 1, m - 1) + D(n, m - 1),$$

where $D(0, n) = 1$ for all n . It turns out that $D(n, n)$ are the Delannoy numbers. The maximal path might not be unique; there may be several paths with the maximum weight.

In typical computational biology applications, s_1 and s_2 are large DNA sequences (on the order of hundreds of kilobases or megabases) and the Needleman-Wunsch algorithm is not feasible. Certain methods for speeding up the alignments rely on restricting the dynamic programming problem to a subgraph. These are known in the literature as approximate alignments [4] or envelopes [2]. In this paper we initiate a study of the structure of approximate alignments, and begin with their enumeration. In order to simplify the calculations and presentation, We restrict ourselves (for the most part) to grids without diagonal edges. Most of the results generalize (with suitable modifications) to that case as well.

1.2 Definition

Consider an $m \times n$ array of squares where m and n are the numbers of columns and rows, respectively. In this paper, we will label the squares such that (x, y) represents the square in the x th column from the left and the y th row from the bottom. Each square will be marked or unmarked. We can impose the following rules about how the squares in the array are marked:

Rule 1.1 *At least one square in the array must be marked.*

Rule 1.2 *If a square (x, y) is marked and not on the upper right corner of the array, then of squares $(x+1, y)$ and $(x, y+1)$, at least one of them is marked. If one of those squares does not exist (when (x, y) is on the boundary), the other must be marked. Similarly, if (x, y) is marked and not on the lower left corner, then of squares $(x-1, y)$ and $(x, y-1)$, at least one of them is marked. If one of those squares does not exist, then the other is marked.*

Definition 1 *An approximate alignment in an $m \times n$ array is a set of marked squares that obey Rules 1.1 and 1.2.*

Figure 1 shows one such approximate alignment in a 9×6 array.

One consequence of these rules is that between any marked square S and the lower left corner, we can find a path within the region of marked squares in which we go left or down a square in each step. Similarly, we can find a path from S to the upper right corner only using marked squares. Since at least one square in the array is marked, the lower left and upper right corners always must be marked.

2 Horizontally and Vertically Convex Approximate Alignments

Let us consider the case in which the approximate alignment is both horizontally and vertically convex (HVC). That is, if we draw a horizontal or vertical segment between two points within the approximate alignment, then the segment lies completely in the approximate alignment.

Lemma 2.1 *Let $x_1 < x_2$ and $y_1 > y_2$. If squares (x_1, y_1) and (x_2, y_2) are in a HVC approximate alignment, then squares (x_1, y_2) and (x_2, y_1) must also be marked.*

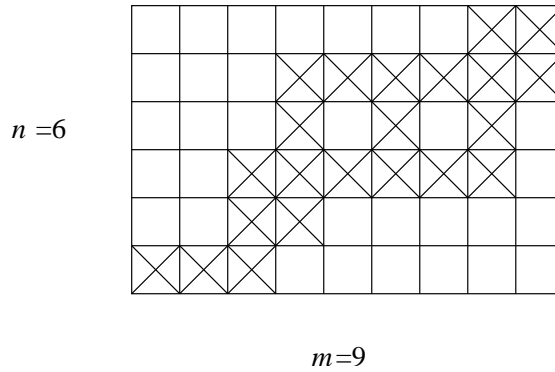


Figure 1: Sample approximate alignment in 9×6 array.

Proof: Suppose that (x_1, y_2) is not marked. Since a path of marked squares must exist from (x_2, y_2) to the lower left corner $(1, 1)$, some square (x_1, y_3) in column x_1 below row y_2 must be marked. But then there is an unmarked square (x_1, y_2) between the marked squares (x_1, y_1) and (x_1, y_3) , which violates the condition of vertical convexity. Thus (x_1, y_2) must be marked. Similarly, from the existence of a path from (x_1, y_1) to the upper right corner, we conclude that (x_2, y_1) must be marked too. ■

A corollary of this lemma would be that since (x_1, y_2) and (x_2, y_1) are marked in addition to (x_1, y_1) and (x_2, y_2) , then (x, y) is marked when $x_1 \leq x \leq x_2$ and $y_1 \geq y \geq y_2$. This follows from horizontal and vertical convexity. So if the upper left and lower right corner squares of a rectangular block are marked, then all the squares in that block must be marked. Note that the lemma generally does not hold when $x_1 < x_2$ and $y_1 < y_2$.

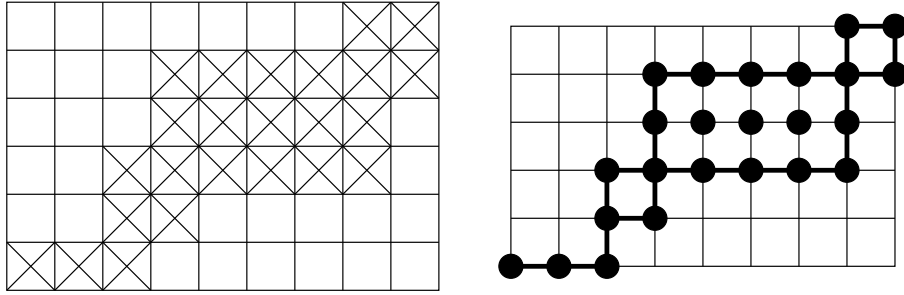
One natural question to ask is how many different HVC approximate alignments are there for an $m \times n$ array. The answer lies in the following theorem:

Theorem 2.2 *The number of HVC approximate alignments for an $m \times n$ array is equal to the number of plane partitions that fit in a $2 \times (m - 1) \times (n - 1)$ box.*

Proof: Take the dual of the array, i.e. substituting a point for each square and connecting the points in a lattice. The size of the dual lattice is $(m - 1) \times (n - 1)$, i.e. the lengths of the boundaries of the lattice. Then we mark each point that corresponds to a marked square. The marked points lie on or between two lattice paths from the lower left to the upper right corner points. The first path is traced by going right if the point to the right is marked, and going up otherwise. The second path is traced by going up if the point above is marked, and going right otherwise. Note that each point on the second path is either on the first path, or it is to the left of one point on the first path, and above another point on the first path. Let us call the first and second paths the *lower* and *upper* paths.

We assign a height function to the $(m - 1)(n - 1)$ squares in the dual lattice as follows: for each square that lies below and to the right of the lower path, assign it height 0. For each square between the two paths, assign it height 1. Finally, for each remaining squares lying above and to the left of the upper path, assign it height 2. The result is a plane partition in a $2 \times (m - 1) \times (n - 1)$ box. Figure 2 shows a sample bijection.

We can reverse the procedure to convert a plane partition in a $2 \times (m - 1) \times (n - 1)$ box into a HVC approximate alignment in a $m \times n$ array, thus establishing the bijection between the two sets of objects. ■



2	2	2	2	2	2	2	1
2	2	2	1	1	1	1	0
2	2	2	1	1	1	1	0
2	2	1	0	0	0	0	0
2	2	0	0	0	0	0	0

Figure 2: Top left: HVC approximate alignment. Top right: Corresponding dual lattice. Bottom: Corresponding plane partition.

The number of plane partitions in an $r \times s \times t$ box was first proved by Percy MacMahon to be

$$\prod_{i=1}^r \prod_{j=1}^s \frac{i+j+t-1}{i+j-1}.$$

One proof can be found in [1]. So the number of possible HVC approximate alignments in an $m \times n$ array is

$$\prod_{i=1}^{m-1} \prod_{j=1}^{n-1} \frac{i+j+1}{i+j-1}$$

or alternatively,

$$\prod_{i=1}^{n-1} \frac{(i+m-1)(i+m)}{i(i+1)}.$$

Now what if we ask, “Of all the HVC approximate alignments of an $m \times n$ array, how many of them have exactly k marked squares?” To answer this question, we apply the transfer matrix method, as described in section 4.7 in Stanley’s *Enumerative Combinatorics* [5].

Note that we can partition the array into diagonals of the form $x+y=p$, for $2 \leq p \leq m+n$. These diagonals run downwards to the right. From a corollary of Lemma 2.1, if squares (x_1, y_1) and (x_2, y_2) are on the same diagonal and marked, then every square between (x_1, y_1) and (x_2, y_2) must also be marked. Thus for diagonal $x+y=p$, we can represent the marked squares by an interval $[x_1, x_2]$, where x_1 and x_2 are the minimum and maximum x-coordinates of the marked squares on the diagonal. For the HVC approximate alignment of Figure 2, the marked intervals on the diagonals are $[1,1]$, $[2,2]$, $[3,3]$, $[3,3]$, $[3,4]$, $[4,4]$, $[4,5]$, $[4,6]$, $[5,7]$, $[6,8]$, $[7,8]$, $[8,8]$, $[8,9]$, and $[9,9]$.

Suppose $[x_1, x_2]$ is marked on diagonal $x + y = p$. What are the possible intervals of marked squares on diagonal $x + y = p + 1$? Since $(x_1, p - x_1)$ is marked, squares $(x_1, p - x_1 + 1)$ or $(x_1 + 1, p - x_1)$ must also be marked. However, the squares with x-coordinate less than x_1 on diagonal $x + y = p + 1$ cannot be marked, for none of the squares directly below or left on the previous diagonal are marked. Thus the left endpoint (or rather, *endsquare*,) on diagonal $x + y = p + 1$ must be either x_1 or $x_1 + 1$ (unless when $x_1 = m$, for which the left endsquare on $x + y = p + 1$ is definitely m). Similarly, the right endsquare on diagonal $x + y = p + 1$ is either x_2 or $x_2 + 1$. Thus there are four possible intervals of marked squares unless when $x_1 = x_2$, in which $[x_1 + 1, x_2]$ is not a possible interval.

We now define the generating function

$$F(p, [x_1, x_2]; q) = \sum_{k \geq 1} f(p, [x_1, x_2], k) q^k$$

where $f(p, [x_1, x_2], k)$ is the number of ways k squares between (and including) $(1,1)$ and the diagonal $x + y = p$ can be marked such that of the squares on $x + y = p$, only the squares on the interval $[x_1, x_2]$ are marked. In addition, the marked squares must form an induced subset of a HVC approximate alignment on some rectangular array.

For instance, $F(2, [1, 1]; q) = q$. We also define $F(p, [x_1, x_2]; q) = 0$ when $x_1 \leq 0$, $x_2 \leq 0$, $x_1 \geq p$, $x_2 \geq p$, or $x_1 > x_2$.

We can express $F(p, [x_1, x_2]; q)$ in terms of of the generating functions based on the previous diagonal:

$$\begin{aligned} F(p, [x_1, x_2]; q) &= q^{x_2 - x_1 + 1} (F(p - 1, [x_1 - 1, x_2 - 1]; q) + F(p - 1, [x_1 - 1, x_2]; q) + \\ &F(p - 1, [x_1, x_2 - 1]; q) + F(p - 1, [x_1, x_2]; q)) \end{aligned}$$

This equation is based on the possible intervals of marked squares on the diagonal $x + y = p - 1$. By adding squares $[x_1, x_2]$ on diagonal $x + y = p$, we are adding an additional $x_2 - x_1 + 1$ squares to the previous alignment. Thus we multiply by a factor of $q^{x_2 - x_1 + 1}$.

Note that when $p = m + n$ and $x_1 = x_2 = m$, $F(m + n, [m, m]; q)$ is the generating function in which $f(m + n, [m, m], k)$ is the number of HVC approximate alignments of k marked squares in an $m \times n$ array.

We now use the transfer matrix method to compute the generating function of the alignments for which $[x_1, x_2]$ is the interval of marked squares on diagonal $x + y = p$. To use the transfer matrix method, we apply bounds $x_1 \leq m$ and $x_2 \leq m$. We index the matrix in the order $[1,1], [2,2], [1,2], [3,3], [2,3], [1,3]$, etc. That way, given the matrix for which x_1 and x_2 are bounded by m , we can extend the matrix for the bound $m + 1$ by adding $m + 1$ rows and columns, indexed by $[m + 1, m + 1], [m, m + 1], \dots, [2, m + 1], [1, m + 1]$.

Let M_m stand for the transfer matrix for arrays of width m . If $m = 1$, the matrix is simply $M_1 = (q)$. The matrices for $m=2$ and $m=3$ are

$$M_2 = \begin{pmatrix} q & 0 & 0 \\ q & q & q \\ q^2 & 0 & q^2 \end{pmatrix} \text{ and } M_3 = \begin{pmatrix} q & 0 & 0 & 0 & 0 & 0 \\ q & q & q & 0 & 0 & 0 \\ q^2 & 0 & q^2 & 0 & 0 & 0 \\ 0 & q & 0 & q & q & 0 \\ 0 & q^2 & q^2 & 0 & q^2 & q^2 \\ 0 & 0 & q^3 & 0 & 0 & q^3 \end{pmatrix}.$$

Let e_m be a column vector of length $\frac{m(m+1)}{2}$ in which the first entry is q , and all the others are 0. The vector e_m represents the values of $F(2, [x_1, x_2]; q)$. By successively multiplying M_m on the left of e_m , we get the generating functions $F(p, [x_1, x_2]; q)$ as p increases. Thus $M_m^{p-2} e_m$ is the vector of generating functions $F(p, [x_1, x_2]; q)$. So if we wish to find the generating function for HVC approximate alignments of an $m \times n$

array, i.e. $F(m+n, [m, m]; q)$, we need to look at the $[m, m]$ -th (i.e., $\frac{m(m-1)}{2} + 1$ -th) element in the vector $M_m^{m+n-2}e_m$. Note that this is simply q times the entry in row $[m, m]$, column 1 of the matrix M_m^{m+n-2} . Using Theorem 4.7.2 in Stanley [5], we can compute a “meta-generating function”

$$\sum_n F(m+n, [m, m]; q)\lambda^n = \frac{\lambda^{2-m}q(-1)^{m(m-1)/2} \det(I - \lambda M_m : 1, [m, m])}{\det(I - \lambda M_m)}$$

where $(A : j, i)$ denotes the resulting matrix after deleting row j and column i from the matrix A . For instance, when $m = 3$, the resulting generating function is

$$\sum_n F(n+3, [3, 3]; q)\lambda^n = \frac{\lambda q^3 - \lambda^3 q^7}{(1 - \lambda q)^3 (1 - \lambda q^2)^2 (1 - \lambda q^3)}.$$

Note that when we extend M_m into matrix M_{m+1} , all entries to the right of M_m are 0. The lower right $(m+1) \times (m+1)$ submatrix is an upper triangular matrix. Thus it is easy to evaluate $\det(I - \lambda M_{m+1})$: simply multiply $\det(I - \lambda M_m)$ by $\prod_{i=1}^{m+1} (1 - \lambda x^i)$. Thus we can easily show by induction on m that

$$\det(I - \lambda M_m) = \prod_{i=1}^m (1 - \lambda q^i)^{m-i+1}.$$

When we set $q = 1$ in $F(m+n, [m, m]; q)$, we get the number of approximate alignments in an $m \times n$ array. As we noted in Theorem 2.2, $F(m+n, [m, m]; 1)$ is also the number of plane partitions in a $2 \times (m-1) \times (n-1)$ box. In the appendix we prove that

$$\sum_{m=0}^{\infty} p(m, n)x^m = \frac{1}{(1-x)^{2n+1}} \sum_{k=0}^{n-1} p(k, n-1-k)x^k,$$

where $p(m, n)$ is the number of plane partitions that fit in a $2 \times m \times n$ box. Therefore

$$\begin{aligned} \det(I - \lambda M_m : 1, [m, m])|_{q=1} &= \lambda^{m-2} (1 - \lambda)^{m(m+1)/2} \sum_{n>0} p(m-1, n-1)\lambda^n \\ &= \lambda^{m-2} (1 - \lambda)^{m(m+1)/2} \frac{\lambda}{(1 - \lambda)^{2m-1}} \sum_{k=0}^{m-2} p(k, m-2-k)x^k \\ &= \lambda^{m-1} (1 - \lambda)^{(m-1)(m-2)/2} \sum_{k=0}^{m-2} p(k, m-2-k)x^k. \end{aligned}$$

2.1 Unimodality for the Case $m = 3$

Let us now further examine the generating functions $F(m+n, [m, m]; q)$ for $m = 3$. We will show that for every n , $F(3+n, [3, 3]; q)$ is unimodal. Recall that $f(m+n, [m, m], k)$ is the number of HVC approximate alignments with k marked squares in an $m \times n$ array. Let us define

$$h(m, n, k) = f(m+n, [m, m], k).$$

We prove some facts about $h(3, n, k)$:

Proposition 2.3 *Let n be a positive integer.*

1. $h(3, n, n + 2) = \frac{1}{2}n(n + 1)$.
2. $h(3, n, n + 3) = n(n - 1)$.
3. If $n + 4 \leq k \leq 2n + 1$, then $h(3, n, k) = h(3, n - 1, k - 3) + (2n + 3 - k)(2n + 2 - k)$.
4. If $k = 3n - 2k'$, where $0 \leq 2k' \leq n - 1$, and k' is an integer, then $h(3, n, k) = \frac{(k'+1)(k'+2)(2k'+3)}{6}$.
5. If $k = 3n - (2k' + 1)$, where $0 \leq 2k' + 1 \leq n - 1$, and k' is an integer, then $h(3, n, k) = \frac{(k'+1)(k'+2)(k'+3)}{3}$.

Proof:

1. In a $3 \times n$ array, there must be at least $n + 2$ marked squares, forming a path between two opposite corners. The path goes 2 steps to the right, and $n - 1$ steps up. Thus $h(3, n, n + 2) = \binom{n+1}{2} = \frac{1}{2}n(n + 1)$.
2. In this case, $n + 2$ squares form a path between the corners, and one more square needs to be marked. When this final square is marked, it completes a marked 2×2 subarray of squares. This happens since the final square needs to be supported by two neighboring squares. These neighboring squares are in turn already connected by a fourth square. There are $n - 1$ choices for which pair of rows this 2×2 block could occupy. Let's say they occupy rows r_1 and $r_1 + 1$ and columns c and $c + 1$. Then in another row r_2 (which may be r_1 or $r_1 + 1$), another pair of squares are side by side, occupying the other pair of adjacent columns. There are n rows for this pair of squares to occupy. For each row, there is only one approximate alignment for this pair and the 2×2 block to coexist. If $r_2 \leq r_1$, then the 2×2 block is in columns 2 and 3; otherwise, if $r_2 \leq r_1 + 1$, then the 2×2 block is in columns 1 and 2. So there are $h(3, n, n + 3) = n(n - 1)$.
3. We can divide these approximate alignments into two categories. In the first category, we could unmark the highest marked square in each column and still have an approximate alignment for a $3 \times (n - 1)$ array. This is the source for the term $h(3, n - 1, k - 3)$ on the right hand side. To get the other term, we must analyze the other category—the one in which Rule 1.2 is broken when the three highest squares are unmarked. That means the highest marked square in one column is in the same row as the lowest marked square of the next column. Since there are k marked squares and $n + 2$ of them form a path from corner to corner, that leaves $k - n - 2$ extra squares to be distributed between at most two adjacent columns. Thus there is a $2 \times (k - n - 1)$ rectangular block of marked squares. There are $n + 1 - (k - n - 1) = 2n + 2 - k$ choices for which group of adjacent rows this rectangular block may occupy. The other single-row pair could be in one of $2n + 3 - k$ rows (they cannot share a row with the interior of the rectangular block). For each choice, we get one possible HVC alignment. This is the source for the other term, $(2n + 3 - k)(2n + 2 - k)$, so $h(3, n, k) = h(3, n - 1, k - 3) + (2n + 3 - k)(2n + 2 - k)$. This analysis holds only for $n + 4 \leq k \leq 2n + 1$, for when $k \geq 2n + 2$, at least two rows has all 3 squares marked.
4. We can interpret $2k'$ as the number of unmarked squares in the array. Thus we can imagine starting with an array with all its squares marked and then choosing $2k'$ squares to unmark. These $2k'$ squares must be clustered around the upper left and lower right corners, and each cluster can be at most 2 columns wide. We can imagine dividing $2k'$ into two parts k'_1 and k'_2 . The k'_1 squares can be arranged in one of $\frac{1}{2}k'_1 + 1$ ways if k'_1 is even, or $\frac{1}{2}(k'_1 + 3)$ if k'_1 is odd. We can arrange this as a formal power series:

$$1 + x + 2x^2 + 2x^3 + 3x^4 + 3x^5 + \dots = \frac{1 + x}{(1 - x^2)^2}.$$

So the number of ways to split $2k'$ squares into two clusters is simply the coefficient of $x^{2k'}$ in

$$(1 + x + 2x^2 + 2x^3 + 3x^4 + 3x^5 + \dots)^2.$$

This coefficient is

$$\sum_{i=0}^{k'} (i+1)(k'+1-i) + \sum_{i=0}^{k'-1} (i+1)(k'-i) = \frac{(k'+1)(k'+2)(2k'+3)}{6}$$

which can be verified easily by induction.

5. The analysis here is essentially the same as in the previous case, except that we examine the coefficient of $x^{2k'+1}$ in $(1+x+2x^2+2x^3+\dots)^2$. This coefficient is

$$2 \sum_{i=0}^{k'} (i+1)(k'+1-i) = \frac{(k'+1)(k'+2)(k'+3)}{3}.$$

■

We now find an explicit formula for the case when $k \leq 2n+1$. Let us define $t = k - n - 2$, which is the number of marked squares greater than the minimum number of squares necessary to have an approximate alignment. We can express $h(3, n, n+2+t)$ as follows:

Proposition 2.4 *Define*

$$\begin{aligned} h_1(n, t) &= \frac{(1+t)n^2}{2} + \frac{(2-2t-3t^2)n}{4} + \frac{-10t+3t^2+7t^3}{24} \\ h_2(n, t) &= \frac{(1+t)n^2}{2} + \frac{(1-2t-3t^2)n}{4} + \frac{-3-7t+3t^2+7t^3}{24}. \end{aligned}$$

Let $0 \leq t \leq n-1$. Then $h(3, n, n+2+t) = h_1(n, t)$ if t is even, and $h(3, n, n+2+t) = h_2(n, t)$ if t is odd.

Proof: We verify the base cases. For $t = 0$ and $t = 1$,

$$\begin{aligned} \frac{(1+0)n^2}{2} + \frac{(2-2 \cdot 0-3 \cdot 0^2)n}{4} + \frac{-10 \cdot 0+3 \cdot 0^2+7 \cdot 0^3}{24} &= \frac{n^2+n}{2} \\ \frac{(1+1)n^2}{2} + \frac{(1-2-3)n}{4} + \frac{-3-7+3+7}{24} &= n^2-n = n(n-1). \end{aligned}$$

For the inductive step, we need to show that

$$\begin{aligned} h_1(n, t) - h_1(n-1, t-2) &= (n+t+1)(n+t) = (2n+3-k)(2n+2-k) \\ &= h(3, n, n+2+t) - h(3, n-1, (n-1)+2+(t-2)) \\ h_2(n, t) - h_2(n-1, t-2) &= (n+t+1)(n+t) = (2n+3-k)(2n+2-k) \\ &= h(3, n, n+2+t) - h(3, n-1, (n-1)+2+(t-2)). \end{aligned}$$

This can be verified as an exercise. ■

The next step in proving unimodality lies in the analysis of the expressions $h_1(n, t+1) - h_2(n, t)$ and $h_2(n, t+1) - h_1(n, t)$. The solutions to $h_1(n, t+1) - h_2(n, t) = 0$ are

$$t = \frac{1}{7}(-4 + 6n \pm \sqrt{9 + 8n + 8n^2}),$$

and the solutions to $h_2(n, t+1) - h_1(n, t) = 0$ are

$$t = \frac{1}{7}(-5 + 6n \pm \sqrt{25 + 24n + 8n^2}).$$

In each case, the expressions are negative when t lies between the solutions, and positive when t lies outside that interval. Moreover, the larger solutions exceed $\frac{1}{7}(6 + \sqrt{8})n > n$, which exceeds the maximum value for t . (Remember, $t \leq n - 1$.) We also note that

$$0 \leq \frac{1}{7}(-4 + 6n - \sqrt{9 + 8n + 8n^2}) - \frac{1}{7}(-5 + 6n - \sqrt{25 + 24n + 8n^2}) \leq \frac{1}{7}(1 + 2\sqrt{8}) < 1.$$

So as soon as $h_1(n, t + 1) - h_2(n, t)$ or $h_2(n, t + 1) - h_1(n, t)$ is negative for some value $t = t_0$, then the value of the other function must be negative at $t = t_0 + 1$.

For the final step, we observe that $\frac{(k'+1)(k'+2)(2k'+3)}{6}$ and $\frac{(k'+1)(k'+2)(k'+3)}{3}$ are increasing functions for k' , so $h(3, n, n + 2 + t)$ continues to decrease when $t > n - 1$. That completes the proof for the following theorem:

Theorem 2.5 *For a $3 \times n$ array, the sequence $\{h(3, n, k)\}_{k=n+2}^{3n}$ of the number of HVC approximate alignments with k marked squares is unimodal.*

From examining the values of $h(4, n, k)$, the sequence $\{h(4, n, k)\}_{k=n+3}^{4n}$ appears to be unimodal for each n . We conclude this section with a conjecture about unimodality:

Conjecture 2.6 *The sequence $\{h(m, n, k)\}_{k=m+n-1}^{mn}$ of the number of HVC approximate alignments with k marked squares in an $m \times n$ array is unimodal.*

3 Non-convex Approximate Alignments

In this section, we remove the restriction that an approximate alignment must be convex. Thus we will consider approximate alignments that may have holes in them. The smallest alignment with a hole is a 3×3 array in which every square is marked except for the one in the middle.

Once again, we will use the transfer matrix method to compute the generating function in which the coefficients represent the number of approximate alignments of k squares in a $m \times n$ array. First, we need to analyze how squares are marked on each diagonal. This time, the marked squares on a diagonal do not have to form an unbroken interval. In fact, any nonempty subset of squares on a diagonal could be marked. Thus for a diagonal $x + y = p$, we will let S_p be the set of x -coordinates of the marked squares on the diagonal. For instance, $S_9 = \{4, 6\}$ and $S_{11} = \{6, 8\}$ in Figure 1. (Sometimes we may also use S_p to refer to the set of marked squares on $x + y = p$.) There are $2^{p-1} - 1$ possible sets that could be S_p .

Now we prove some relations between the set of marked squares on two adjacent diagonals.

Proposition 3.1 *If $x \in S_p$, then either x or $x + 1 \in S_{p+1}$. If $x \in S_{p+1}$, then*

1. $x \in S_p$ if $x = 1$;
2. $x - 1 \in S_p$ if $x = p$;
3. either x or $x - 1 \in S_p$ if $1 < x < p$.

The proof of this proposition easily follows from Rule 1.2.

We now define the generating function

$$G(p, S_p; q) = \sum_{k \geq 1} g(p, S_p, k) q^k$$

such that $g(p, S_p, k)$ is the number of ways k squares between (and including) $(1,1)$ and the diagonal $x+y = p$ can be marked such that of the squares on $x+y = p$, only the squares in S_p are marked. In addition, the marked squares must form an induced subset of an approximate alignment on some rectangular array.

As an example, $G(4, \{1, 3\}; q) = q^5$, for the only possible alignment would include squares $(1,1)$, $(1,2)$, $(1,3)$, $(2,1)$, and $(3,1)$. Once again note that $g(p, \{m\}, k)$ is the number of approximate alignments of k marked squares in an $m \times n$ array.

Now we construct the transition matrix that shows the possible elements of S_{p+1} given the set S_p . This time, we index the matrix as follows: \emptyset , $\{1\}$, $\{2\}$, $\{1, 2\}$, $\{3\}$, $\{1, 3\}$, $\{2, 3\}$, $\{1, 2, 3\}$, etc. The rule is that given the ordering for all the subsets of the first m integers, we create a new list by first adding $m+1$ to each subset in the ordering, and then append this new list to the end of the old list. Another way to view this indexing scheme is to write the integers in binary form, and include i in the corresponding set if and only if the i th digit to the right is a 1. Thus $25 = 11001_2$ corresponds to $\{1, 4, 5\}$. Although S_p cannot be the empty set in our rules, we include it here for convenience.

We let N_m be the $2^m \times 2^m$ matrix for which each S_p is bounded above by m . Then

$$N_1 = \begin{pmatrix} 1 & 0 \\ 0 & q \end{pmatrix}.$$

We show how to create N_{m+1} given N_m .

Proposition 3.2 *If we divide N_m into four equal-sized blocks A, B, C, D arranged such that*

$$N_m = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

then B is simply the zero matrix, and

$$N_{m+1} = \begin{pmatrix} A & 0 & 0 & 0 \\ C & D & 0 & 0 \\ 0 & qA & qA & qA \\ 0 & qD & qC & qD \end{pmatrix}.$$

Proof: First, we show that B is the zero matrix. This follows since if S_{p+1} does not contain element m , then m cannot be in S_p . This fact is reflected in the upper right quadrant of N_m . For the same reason, the blocks in the upper right quadrant of N_{m+1} are zero.

To construct the rest of N_{m+1} , we note that the upper left quadrant is the same as N_m since the transitions are the same when neither S_p nor S_{p+1} contain $m+1$. The third row of N_{m+1} corresponds to the case when S_{p+1} contains $m+1$ but not m . The fourth row of N_{m+1} describes the case when S_{p+1} contains both m and $m+1$.

If S_p contains neither m nor $m+1$, then it is impossible for S_{p+1} to contain m ; thus blocks (3,1) and (4,1) in N_{m+1} are 0.

Let us call a pair of adjacent diagonals (S_p, S_{p+1}) *legal* if those two diagonals can appear in the same approximate alignment.

Consider the case that $m \in S_p$ and $m+1 \in S_{p+1}$. If $m \notin S_{p+1}$, then (S_p, S_{p+1}) is legal if and only if $(S_p - \{m, m+1\}, S_{p+1} - \{m+1\})$ is legal. Thus blocks (3,2) and (3,4) in N_{m+1} are qA since A represents the transition matrix for when neither S_p nor S_{p+1} contain m or $m+1$, and the factor of q represents the extra element $m+1$ in S_{p+1} . If $m \in S_{p+1}$, then (S_p, S_{p+1}) is legal if and only if $(S_p - \{m+1\}, S_{p+1} - \{m+1\})$ is legal. The transition matrix for this case is D , so blocks (4,2) and (4,4) in N_{m+1} are qD .

	1	2	3	4	5
1	1	1	1	1	1
2	1	3	6	10	15
3	1	6	21	57	134
4	1	10	57	250	954
5	1	15	134	954	6040
6	1	21	289	3375	35723
7	1	28	592	11458	202280
8	1	36	1178	38017	1112435
9	1	45	2311	124411	5999391
10	1	55	4512	403373	31948906

Table 1: Number of approximate alignments for $m \times n$ array, for $1 \leq m \leq 5$, $1 \leq n \leq 10$.

Finally, we consider the case when $m + 1 \in S_p$ but $m \notin S_p$, and $m + 1 \in S_{p+1}$. Then (S_p, S_{p+1}) is legal iff $(S_p - \{m + 1\}, S_{p+1} - \{m + 1\})$ is legal. Thus blocks (3,3) and (4,3) are qA and qC . ■

Let e_m be a column vector of length 2^m where the second entry is q , and all other entries are 0. This vector represents the initial values of $G(2, S; q)$, with S ranging over the subsets of $[m]$. Successively multiplying by N_m on the left of e_m produces the generating functions $G(p, S; q)$. The generating function for approximate alignments of an $m \times n$ array is $G(m + n, \{m\}; q)$. This generating function appears in the $(2^{m-1} + 1)$ th entry of the vector $N_m^{m+n-2}e_m$, and it is q times the entry in the $(2^{m-1} + 1)$ th row and 2nd column of N_m^{m+n-2} . The meta-generating function is

$$\sum_n G(m + n, \{m\}; q)\lambda^n = -\frac{\lambda^{2-m}q \det(I - \lambda N_m : 2, 2^{m-1} + 1)}{\det(I - \lambda N_m)}.$$

Note that by setting $q = 1$, the coefficient of λ^n becomes the number of approximate alignments in an $m \times n$ array. Table 1 shows how many approximate alignments an $m \times n$ array has.

For each value of m , we could come up with an explicit formula for approximate alignments in an $m \times n$ array in terms of n . Let us define $G(m, n) = G(m + n, \{m\}, 1)$. That is, $G(m, n)$ is the number of approximate alignments in an $m \times n$ array. Clearly, $G(1, n) = 1$ and $G(2, n) = \frac{1}{2}n(n + 1)$ for all n . In general, if we want to find an explicit formula for $G(m, n)$ given a fixed m , then we need to compute the eigenvalues of the matrix $I - \lambda N_m$, in which we set $q = 1$. For instance, the eigenvalues of $I - \lambda N_3$ are 2, 1 with multiplicity 6, and 0. Thus the formula for $G(3, n)$ will be of the form $k_1 \cdot 2^n + p(n)$, for some polynomial of degree of at most 5. And indeed,

$$G(3, n) = 4 \cdot 2^n + \frac{1}{24}(n^4 + 2n^3 - 13n^2 - 62n - 96).$$

However, when we get to $m = 4$ or higher, the exact formulas are not so nice and easy. Indeed, for $I - \lambda N_4$, the eigenvalues will include the roots of the cubic $x^3 - 4x^2 + 3x - 1$, which is irreducible in the ring of rational polynomials.

We could get an idea of how fast $G(m, n)$ grows for fixed m as n increases. With the help of Mathematica, we find the eigenvalue with the largest magnitude. Fortunately, in all cases up to $m = 10$, these eigenvalues have been real and of multiplicity 1. Table 2 shows the largest eigenvalue for m between 1 and 10. The order of growth for $G(m, n)$ is $O(\lambda_m^n)$, where λ_m is the largest (real) eigenvalue of $I - \lambda N_m$.

If we compute the ratios λ_{m+1}/λ_m , we get the sequence

$$1, 2, 1.57395, 1.6664, 1.64290, 1.6491, 1.64762, 1.64801, 1.64792, \dots$$

The sequence seems to converge to 1.6479+. We formulate this as a conjecture:

m	largest eigenvalue
1	1
2	1
3	2
4	3.14790
5	5.24566
6	8.61812
7	14.2126
8	23.4169
9	38.5914
10	63.5955

Table 2: Largest eigenvalues for N_m .

Conjecture 3.3 For integers, m, n ,

$$\lim_{m \rightarrow \infty} \lim_{n \rightarrow \infty} \frac{G(m+1, n+1)G(m, n)}{G(m+1, n)G(m, n+1)} = 1.6479 + .$$

4 Appendix: Plane Partitions Generating Function Identity

Let $p(m, n)$ be the number of plane partitions that fit into a $2 \times m \times n$ box. An explicit formula for $p(m, n)$ is

$$p(m, n) = \frac{(m+n+1)!(m+n)!}{(m+1)!m!(n+1)n!}.$$

Lemma 4.1 For integers $m \geq 0, n \geq 1$,

$$p(m, n) = \sum_{k=0}^{n-1} \binom{2n+m-k}{2n} p(k, n-1-k).$$

Proof: The ratio between the $k+1$ st and k th summands is

$$\frac{(n-k)(n-k-1)(m-k)}{(1+k)(2+k)(2n+m-k)} = \frac{(-n+k)(-n+1+k)(-m+k)}{(1+k)(2+k)(-2n-m+k)}.$$

Therefore the summation in question is a hypergeometric series:

$$\sum_{k=0}^{n-1} \binom{2n+m-k}{2n} p(k, n-1-k) = \frac{(2n+m)!}{(2n)!m!} {}_3F_2 \left[\begin{matrix} -n, -n+1, -m \\ 2, -2n-m \end{matrix} ; 1 \right]$$

Recall a theorem of Saalschütz about hypergeometric series:

$${}_3F_2 \left[\begin{matrix} a, b, c \\ d, e \end{matrix} ; 1 \right] = \frac{(d-a)_{|c|} (d-b)_{|c|}}{(d)_{|c|} (d-a-b)_{|c|}}$$

where $d+e = a+b+c+1$ and c is a nonpositive integer, and $(a)_n$ is the rising factorial

$$(a)_n = \prod_{i=0}^{n-1} (a+i).$$

Note that if a is a positive integer, then $(a)_n = \frac{(a+n-1)!}{(a-1)!}$.

By plugging in appropriate values to Saalschütz's Theorem, we get

$${}_3F_2 \left[\begin{matrix} -n, -n+1, -m \\ 2, -2n-m \end{matrix} ; 1 \right] = \frac{(n+2)_m (n+1)_m}{(2)_m (2n+1)_m}.$$

Thus,

$$\begin{aligned} \sum_{k=0}^{n-1} \binom{2n+m-k}{2n} p(k, n-1-k) &= \frac{(2n+m)! (n+2)_m (n+1)_m}{(2)_m (2n+1)_m} \\ &= \frac{(m+n+1)! (m+n)!}{(m+1)! m! (n+1)! n!} = p(m, n). \end{aligned}$$

■

From this lemma, we derive the following corollary:

Corollary 4.2 *For any integer $n \geq 1$,*

$$\sum_{m=0}^{\infty} p(m, n) x^m = \frac{1}{(1-x)^{2n+1}} \sum_{k=0}^{n-1} p(k, n-1-k) x^k.$$

Proof: This corollary follows since

$$\frac{1}{(1-x)^{2n+1}} = \sum_{j=0}^{\infty} \binom{2n+j}{2n} x^j,$$

and so the coefficient of x^m in $\frac{1}{(1-x)^{2n+1}} \sum_{k=0}^{n-1} p(k, n-1-k) x^k$ is

$$\sum_{k=0}^{n-1} \binom{2n+m-k}{2n} p(k, n-1-k) = p(m, n).$$

■

References

- [1] Gessel, I., and G. Viennot. Binomial determinants, paths, and hook length formulae. *Advances in Mathematics* (1985) 13:295-308.
- [2] Holmes, I. *Studies in Probabilistic Sequence Alignment and Evolution* (1998), Ph.D. thesis, The Sanger Centre, Hinxton, Cambridgeshire CB10 1SA, UK.
- [3] Needleman, S.B., Wunsch, C.D. A General Method Applicable to the Search for Similarities in Amino Acid Sequence of Two Proteins. *J. Mol. Biol.* (1970) 48:443-453.
- [4] Pachter, L., F. Lam, M. Alexandersson. Picking Alignments from (Steiner) Trees. *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB 2002)*, 2002.
- [5] Stanley, R. *Enumerative Combinatorics, Vol. 1*. Cambridge University Press, Cambridge, UK (1997).