Size Matters and Layout Matters: Layout Algorithms

January 9, 2026

1 Introduction

This document describes the layout algorithms implemented in Holes.py, which was used to run simulations reported in the articles "Size Matters: Optimal Test Pit Size for Dispersed Archaeological Test Excavations" and "Layout Matters: Determining Optimal Test Pit Layout for Dispersed Archaeological Test Excavation Programs" by Oakes and McLaren. See those articles for background. We simulate archaeological digs to explore success of different test pit sizes, numbers of test pits, and layout strategies, with different sizes and types of archaeological sites. Our simulation takes place on a square field of dimension $n \times n$ ($100m \times 100m$, 150×150, and $200m \times 200m$ in the simulations used in the articles). The test pit sizes for simulations used in the "Size Matters" article are $0.5m \times 0.5m$ (small pits) and $1m \times 1m$ (large pits). Only small $0.5m \times 0.5m$ pits are used in "Layout Matters". An archaeological site is placed randomly on the field, and pits dug according to a layout algorithm to see if they find the site (or find individual artefacts in some experiments conducted with real world data in "Size Matters").

2 Hexagonal-Like Layout Algorithm

This is the only algorithm used for the simulations reported in "Size Matters". In "Layout Matters" we explore other layout algorithms (described below) as well as this one. We define a staggered grid as one consisting of several straight and parallel rows of test pits, with the distance betwen adjacent pits in a row being equal, the distance between rows being equal, and with each second row indented (staggered) by half the distance between adjacent pits in a row. In addition we enforce borders around the edges of the field where no pits are dug. The reason for fixing borders is to avoid anomalous results when a pit is close to the edge of the field, noting that the archaeological site is placed wholly within the field. The left and right borders are half the distance between pits in a row, and the top and bottom borders are half the distance between rows. Such a grid is shown in figure 1. For a given total number of pits h we need to decide how

many pits will be dug in our field horizontally (h_x), and how many vertically (h_y).

First some definitions:

- h is the total number of pits,
- h_x and h_y are the number of pits along the x axis and the y axis of the field respectively,
- n_x and n_y are the width and height of the field respectively (we only consider square fields in the experiments),
- a, b, and c are distances between pits as shown in figure 1,
- $pitSize$ is the length (and width) of the square test pits.

A staggered layout is hexagonal when a and b shown in figure 1 are equal. A hexagonal layout is considered optimal in certain circumstances where the archaelogical sites are circular. In the hexagonal-like algorithm we choose values for h_x and h_y that make the layout almost hexagonal (while keeping the borders around the edges of the field as described above). In section 3 we vary the borders to create a truly hexagonal layout. In experiments the hexagonal-like and hexagonal algorithms had very similar success rates.

In this section, except where otherwise apparent, we treat pits as points, ignoring $pitSize$. We assume $pitSize$ is much less than the size of the field.

We now derive some equations for h_x and h_y, starting with some equations evident from figure 1:

$$h = h_x \cdot h_y \tag{1}$$

$$n_y = c \cdot h_y \tag{2}$$

$$n_x = a(h_x - 1) + a + \frac{a}{2} \tag{3}$$

$$= a(h_x + \frac{1}{2}) \tag{4}$$

$$a^2 = \left(\frac{a}{2}\right)^2 + c^2 \qquad \text{Pythagorean Theorem} \tag{5}$$

Now we write a formula for a in terms of n_x and h_x:

$$a = \frac{n_x}{h_x + \frac{1}{2}} \qquad \text{from 4} \tag{6}$$

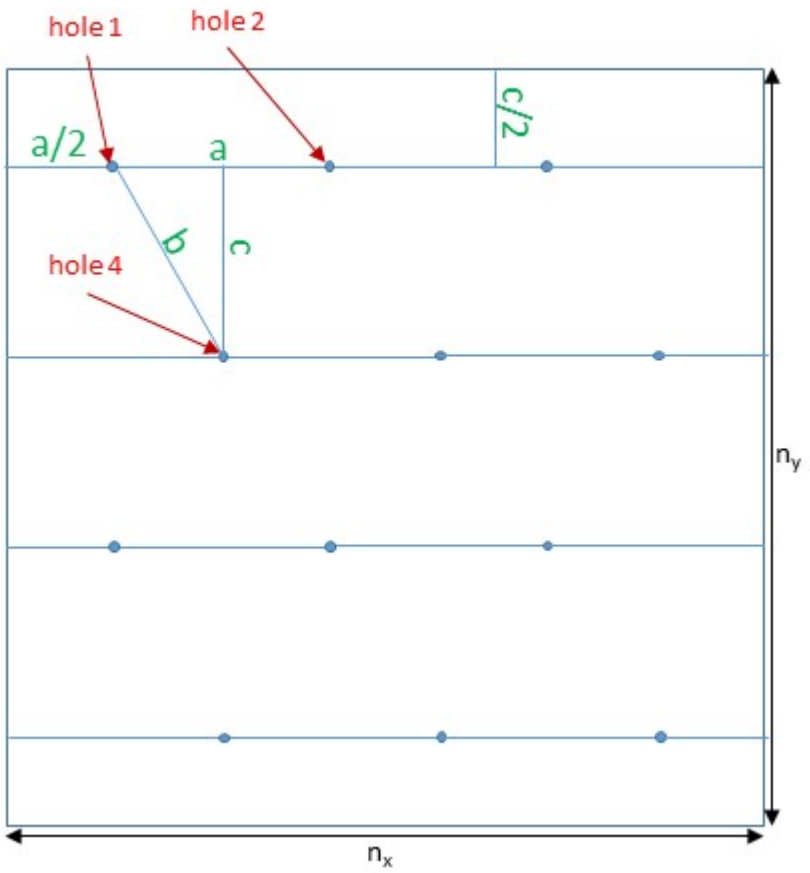

Figure 1: Staggered grid for $h = 12$ test pits. A staggered layout is hexagonal when $a = b$.

Now we write a formula for a in terms of n_y and h_y:

$$\begin{aligned}
a^2 &= \frac{a^2}{4} + c^2 \qquad \text{from 5} && (7)\\
\frac{3}{4}a^2 &= c^2 && (8)\\
a &= \frac{2c}{\sqrt{3}} && (9)\\
&= \frac{2\left(\frac{n_y}{h_y}\right)}{\sqrt{3}} && (10)\\
&= \frac{2n_y}{\sqrt{3}h_y} && (11)
\end{aligned}$$

We equate the two formulae for a and find a quadratic equation for h_x:

$$\begin{aligned}
\frac{n_x}{h_x + \frac{1}{2}} &= \frac{2n_y}{\sqrt{3}h_y} \qquad \text{from 6 and 11} && (12)\\
\sqrt{3}h_y n_x &= 2n_y\left(h_x + \frac{1}{2}\right) && (13)\\
\sqrt{3}\frac{h}{h_x}\, n_x &= 2n_y\left(h_x + \frac{1}{2}\right) \qquad \text{from 1} && (14)\\
\sqrt{3}h n_x &= 2h_x n_y\left(h_x + \frac{1}{2}\right) && (15)\\
&= 2n_y h_x^2 + n_y h_x && (16)\\
0 &= 2n_y h_x^2 + n_y h_x - \sqrt{3}h n_x && (17)
\end{aligned}$$

We then solve for h_x using the quadratic formula:

$$\begin{aligned}
h_x &= \frac{-n_y + \sqrt{{n_y}^2 - 4\cdot 2n_y \cdot \left(-\sqrt{3}h n_x\right)}}{2\cdot 2n_y} && (18)\\
h_x &= \frac{-n_y + \sqrt{{n_y}^2 + 8\sqrt{3}n_x n_y h}}{4n_y} && (19)\\
\text{and} \qquad h_y &= \frac{h}{h_x} && (20)
\end{aligned}$$

We round the values to get integer values for h_x and h_y (this, while necessary, is where the algorithm diverges from the truly hexagonal layout captured in the

equations) and recalculate a and c to reflect the rounded values:

$$c' = \frac{n_y}{round(h_y)} \quad (21)$$

$$a' = \frac{n_x}{round(h_x) + \frac{1}{2}} \quad (22)$$

$$border_x = \frac{a'}{2} \quad (23)$$

$$border_y = \frac{c'}{2} \quad (24)$$

Then we layout the pits with the given borders, so that $round(h_x)$ pits in a row are a' apart, $round(h_y)$ rows are c' apart, and every second row is indented (staggered) by $a'/2$.

If $border_x$ or $border_y$ is less than $pitSize/2$ the layout algorithm fails (note a pit location is specified by the coordinates of its centre).

3 Hexagonal Layout Algorithm

This section describes one way to implement a hexagonal layout algorithm. We do not claim that it is the best way. We start with the equations for a hexagonal-like layout with a horizontal border of $a/2$ and vertical border of $c/2$, calculating h_x and h_y as in equations 19 and 20. Then we vary the borders so that it is exactly hexagonal. First fix $border_y$ and c to their original values and change $border_x$ and a:

$$c' = \frac{n_y}{round(h_y)} \quad (25)$$

$$a' = \frac{2c'}{\sqrt{3}} \qquad \text{from 9} \quad (26)$$

$$border_x = \frac{n_x - a'(round(h_x) - \frac{1}{2})}{2} \quad (27)$$

$$border_y = \frac{c'}{2} \quad (28)$$

It can be shown that for a square field, when h_x is similar to h_y, this increases the value of a' compared to the value in equation 22, shrinking $border_x$ (it appears to be true more generally). If $border_x < pitSize/2$, the pits won't fit (pits are specified by the coordinates of their centre). In that case we set $border_x$ it to its minimum value of $pitSize/2$ and vary the other values as follows:

$$a'' = \frac{n_x - pitSize}{round(h_x) - \frac{1}{2}} \quad (29)$$

$$c'' = \frac{\sqrt{3}a''}{2} \qquad \text{from 9} \quad (30)$$

$$border_x = \frac{pitSize}{2} \quad (31)$$

$$border_y = \frac{n_y - c''(round(h_y) - 1)}{2} \quad (32)$$

Our value for a may now be a' or a'', so we just refer to it as a. Likewise for c. If $a < pitSize$, $border_x < pitSize/2$, $c < pitSize$, or $border_y < pitSize/2$, we are unable to create a hexagonal layout using this method. Otherwise we layout the pits with the given borders, so that $round(h_x)$ pits in a row are a apart, $round(h_y)$ rows are c apart, and every second row is indented (staggered) by $a/2$. This algorithm was not used in the articles except to test that the hexagonal-like and hexagonal algorithms are similar in performance.

4 StaggerXY Layout Algorithm

The hexagonal-like and hexagonal layout algorithms result in a particular type of staggered grid. This staggering is in the horizontal direction, with every second row indented. We can also stagger in the vertical direction, so that every second pit in each row is shifted down. This improves on the hexagonal-like layout in some cases when the archaeological site becomes elongated rather than being circular. Adapting the hexagonal-like layout algorithm to do this requires a slight difference to the maths used to calculate h_x and h_y. The difference begins with a difference to equation 2, namely

$$n_y = c(h_y + \frac{1}{2}) \quad (33)$$

Then similar maths to section 2 gives

$$4n_y h_x^2 + (2n_y - \sqrt{3}n_x)h_x - 2\sqrt{3}n_x h = 0 \quad (34)$$

$$h_y = \frac{h}{h_x} \quad (35)$$

$$c' = \frac{n_y}{round(h_y) + \frac{1}{2}} \quad (36)$$

$$a' = \frac{n_x}{round(h_x) + \frac{1}{2}} \quad (37)$$

$$border_x = \frac{a'}{2} \quad (38)$$

$$border_y = \frac{c'}{2} \quad (39)$$

Equation 34 can be solved for h_x with the quadratic formula.

Then we layout the pits with the given borders, so that $round(h_x)$ pits in a row are a' apart, $round(h_y)$ rows are c' apart, every second row is indented (staggered) by $a'/2$, and every second pit in each row is shifted down by $c'/2$. Note that due to the vertical staggering the layout is no longer hexagonal (or hexagonal-like).

This algorithm was used the the "Layout Matters" article.

5 Random Layout Algorithm

The performance of the systematic algorithms above declines rapidly as site elongation ratios increase (in either the vertical or horizontal direction), as the elongated site can sit between the regularly spaced test pits. To overcome this, we consider the random and halton layout algorithms which perform comparatively well at larger site elongation ratios. The random layout algorithm chooses pit locations according to a pseudo-random number generator (of course conventional computers cannot produce truly random results!). The pits do not overlap, and margins of half the distance between pits (and rows) if the grid were square (i.e. $\frac{n_x}{2\sqrt{h}}$). This algorithm was used in the "Layout Matters" article.

6 Halton Layout Algorithm

The Halton layout algorithm lays out the pits according to a 2-dimensional scrambled Halton sequence, which appears random but does not cluster pits close to each other (which may happen in the random layout). See [Owen, A.B. (2017). A randomized Halton algorithm in R. arXiv:1706.02808] for the details of a scrambled halton sequence, which algorithm is implemented in the SciPy python library [Virtanen P. et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17(3), 261-272. DOI: 10.1038/s41592-019-0686-2]. The halton layout algorithm was used in simulations for the "Layout Matters" article.

7 Randomised-StaggerXY Layout Algorithm

While the random and halton layout algorithms perform better than the hexagonal-like and staggerXY algorithms for larger site elongation ratios, they do not perform as well as those algorithms for smaller site elongation ratios. So we consider a hybrid algorithm. We start with the staggerXY algorithm and then randomise the positions of the pits in both x and y directions. The position of each pit (first the x position and then the y position) is randomised according to a normal distribution with the original position as mean, so it is more likely to land close to its original position, with the probability declining as distance from the original position increases. The standard deviation in the x direction is one eighth the horizontal distance between pits, and in the y direction is

one eighth the vertical distance between rows. The amount by which it is randomised is capped at half the distance between pits in a row (for adjustment in the x direction) and half the distance between rows (for adjustment in the y direction). In addition the borders of the staggerXY algorithm are maintained for consistency, so pits on the left and right borders are not randomised in the x direction and pits on the top and bottom borders are not randomised in the y direction. Other standard deviations are possible, but this one worked well in experiments, providing a balance between performance at small and larger site elongation ratios. This algorithm was used in the "Layout Matters" article.