Honeycomb Lattices with Defects

Meryl A. Spencer  
Department of Physics,  
University of Michigan, Ann Arbor MI 48104, USA

Robert M. Ziff  
Center for the Study of Complex Systems and Department of Chemical Engineering,  
University of Michigan, Ann Arbor MI 48109-2136, USA  
(Dated: December 9, 2015)

In this paper we introduce a variant of the honeycomb lattice in which we create defects by randomly exchanging adjacent bonds, producing a random tiling with a distribution of polygon edges. We study the percolation properties on these lattices as a function of the number of exchanged bonds using a novel computational method. We find the site and bond percolation thresholds are consistent with other three-coordinated lattices with the same standard deviation in the degree distribution of the dual; here we can produce a continuum of lattices with a range of standard deviations in the distribution. These lattices should be useful for modeling other properties of random systems as well as percolation.

I. INTRODUCTION

Honeycomb lattices are extremely widespread, forming the structure of many natural and artificial objects. Honeycombs in graphene and other crystals are arranged in a honeycomb pattern, as are the bubbles that form in thin foams. Consequently condensed matter and materials science researchers have a great interest in the properties of this lattice. Biologists study cells in epithelial tissues, which arrange themselves into honeycomb lattices, not to mention the lattice’s namesake, the beehive. The lattice is common in art and architecture too; you might even be sitting in a room with a honeycomb-tiled floor at this very moment!

The ideal honeycomb lattice used in art, architecture and math has been well studied (i.e., [3-5]), but naturally occurring systems are rarely perfect. Physical systems will almost always contain some defects which cause the systems to vary from the ideal lattice to some degree. In this paper we investigate how the severity of these defects affects the percolation properties of the lattice. We present a new lattice network which we call the “Defective Honeycomb lattice” (DHC) formed by systematically swapping bonds in the honeycomb lattice, as shown in figure 1. We will describe how the basic properties of the lattice change as more bonds are flipped, as well as giving an algorithm for computationally creating such lattices.

The percolation threshold describes the connectivity of a lattice. If sites or bonds are occupied with some probability $p$, the percolation threshold is the minimum value of $p$ at which there is a connected path from one side of the lattice to the other, in the limit of an infinite system. The percolation thresholds are known for many lattices, including the honeycomb lattice for bond percolation. In this paper we will compare the percolation thresholds of several known two-dimensional three-coordinated lattices with our DHC lattices, which are also three coordinated. We will show that there is a strong relationship between the percolation threshold and the variance in the number of faces in the polygons, equivalent to the degree of vertices on the lattice duals.

II. DEFECTIVE HONEYCOMB (DHC) LATTICE

Many naturally occurring biological systems, including cells in epithelial tissues, are approximate honeycomb lattices, but there is little understanding of the formal properties of these lattices. Understanding how defects in these lattices affects their properties will give us a more accurate picture of how naturally occurring honeycomb networks behave.

A. Definition

The inspiration for the DHC lattice is the T1 topological process in foams [1]. In the T1 process one edge sep-
arating two bubbles shrinks down to a point and then re-
grows in a perpendicular direction. This has the effect of
swapping which bubbles are neighbors, and the number of
neighbors of the four bubbles involved. When viewed
as a lattice this process is equivalent to rearranging the
five bonds between six neighboring vertices connected in
an H shape as shown in figure 2.

The DHC lattice is a honeycomb lattice in which a
certain number of the lattice bonds have been flipped.
The DHC lattice is defined by two parameters: $n$, the
number of hexagonal tiles per row in the original $n \times n$
hexagonal lattice as arranged in figure 2 (top left), and $F$,
the number of bonds which are flipped. To generate the
lattice $F$ bonds are chosen uniformly at random; the
same bond may be chosen more than once. The order
in which the bonds are flipped matters, as flipping bond
$a-b$ and then $b-c$ is not equivalent to flipping bond $b-c$
and then $a-b$.

In order to consistently describe lattices of different
size we will specify $f = F/(3n^2)$, the fraction of edges
flipped, instead of $F$, to characterize the number of
flipped bonds in the lattice.

### B. Example

Figure 3 shows the generation of a $4 \times 4$ lattice with 10
flips, so $f = 10/48 \approx 0.21$. In the first panel none of the
dges have been flipped and the system is the original
honeycomb configuration. In the second panel the first
bonds chosen, bond 9–10 shown in red, has been flipped.
This has the effect of creating two pentagonal tiles and
two heptagonal tiles in the otherwise unperturbed hon-
eycomb lattice, increasing the standard deviation in the
number of edges per tile, $\sigma$. As the edges are flipped the
lattice becomes more and more deformed and $\sigma$ increases
to 1.50. The quantity $\sigma$, which is the same as the stan-
dard deviation in the coordination number of sites on the
dual lattice, is important since it gives a quantitative
measure of the extent of the difference between the new
lattice and the original honeycomb. It is defined as

$$\sigma = \sqrt{\frac{1}{n^2} \sum_i (z_i - 6)^2} \quad (1)$$

where $z_i$ is the number of edges of the $i$-th polygon
face, equal to the number of neighbors of the $i$-th ver-
tex in the dual lattice. The number 6 represents the
mean value of $z_i$, which follows from Euler’s formula,
vertices−edges+faces = 2, assuming $n$ is large, and also
from the fact that with each flip, the total number of
bonds, tiles, and vertices is unchanged from the original
honeycomb values. We also consider the variance $v = \sigma^2$.

### C. Properties

There are several important properties of the DHC lat-
tice. The lattice is two dimensional, so the percolation
critical exponents of the system are known [6]. When a
bond is flipped, four tiles are affected. Two of them (tiles
$A$ and $B$ in figure 2) lose a bond and go from $n$-gons to
$(n-1)$-gons, while the other two (tiles $C$ and $D$ in figure
2) go from $n$-gons to $(n+1)$-gons. Therefore the average
number of edges per tile (equivalently the average degree
on the dual lattice) remains constant at 6. As bonds are
flipped the standard deviation in the number of edges
per tile changes. The flipping of a bond changes the con-
nection between sites, but each site continues to have ex-
actly 3 neighbors. Equivalently the dual lattice remains
fully triangulated. We will use these properties to define
a class of planar lattices, the honeycomb variant class,
where each lattice is two-dimensional, three-coordinated
and has an average dual degree of six. We will compare
the percolation thresholds of the DHC to the rest of the
honeycomb variant class of lattices in section V.

III. GENERATION OF LATTICES

In the remainder of this paper we will determine addi-
tional properties of the DHC lattices through computa-
tional simulation. The process of generating the lattices
is mostly straightforward, however there are a few non-
trivial components of the algorithm which we will address
here.
For physical reasons, we would like our lattice to remain planar. Furthermore, for percolation, the critical exponents have different values in different dimensions, and may change if the lattice is non-planar. As we flip bonds we need a way to guarantee that our lattice stays planar. The specific coordinates of the sites are not relevant, but for an acceptable bond flip there must exist some arrangement of the sites on the plane for which none of the bonds cross. It turns out that the need for a planar representation of the lattice specifies a unique exchange of bonds for any flip trial. Figure 3 shows the two topologically distinct ways to flip bond 1–b. In both resulting lattices tiles A and B lose bond a–b and tiles C and D gain bond a–b; however they differ in the reconnection of the bonds between sites 1, 2, 3 and 4, and the flipped sites a and b. In the top picture bond 2–a is broken and replaced by bond 2–b and bond 4–b is broken and replaced by bond 4–a. In the bottom picture reconnections are correct because the lattice has a planar representation which is shown. In the bottom case there is no planar representation on the lattice—no matter where sites a and b are located there will be some bonds that cross each other—therefore we need to be careful when reconnecting the bonds after a flip to preserve the planar nature of the lattice.

Lattices are often stored on computers as simple neighbor lists, however here a neighbor list would not be able to determine which of the two distinct ways to flip bond a–b would result in a planar representation. In each case sites a and b exchange one neighbor with each other, but without knowledge of which neighbors belong to which tiles it is impossible to determine if sites 2 and 4 should be exchanged or sites 1 and 3. In order to solve this problem we stored our lattices as a list of all of the sites that make up each tile. The sites were stored in clockwise order to further simplify the flipping algorithm. For example tilelist = [1, 5, 2, 18, 7] denotes that the tile with ID of 3 is a pentagon made of the sites 1, 2, 5, 7, 18, and bonds 1–5, 5–2, 2–18, 18–7 and 7–5. Storing the lattice in this way makes the algorithm given in algorithm I for flipping bond a–b simple. It is also easy to translate into the standard neighbor list as the three neighbors of site 2 are all of the IDs immediately following 2 in the entire tile list.

In order to use algorithm I to flip bonds in the lattice we must start with the lattice as lists of clockwise sites in every cell. We used the labeling method shown in figure 3 to determine tile and site IDs. Tiles are initially labeled 0 to n^2 – 1 in order by row and then column. The top site of each hexagon initially has an ID of ID_tile = 2(ID_tile) and the next clockwise site has ID_tile = 2(ID_tile) + 1. This produces a label for every site and tile in the lattice. The lattice is then initialized according to the algorithm given in appendix A.

IV. RELATIONSHIP BETWEEN FLIPS AND DEFECTS

The DHC lattice is characterized by the number of bonds flipped, and we would like to know how the lattice...
Many-sided polygons.

Average distribution of polygons over ten independent 128 × 128 lattices at different flipping fractions. As more bonds are flipped a greater percentage of the tiles are triangles and a few many-sided polygons. The reason for this long-time behavior is that every time a bond is flipped two tiles lose an edge and two tiles gain an edge, so as more bonds are flipped the number of bonds per tile is forced away from the mean of six.

There is an ambiguity in the definition of the lattice when $f$ gets large. If the next bond to be flipped is part of a triangle it cannot be flipped or a tile with only two sites would be created and the lattice would become a multi-graph (where two sites are connected by more than one edge) which is not the behavior we were looking for. Bonds that are part of triangular tiles should not be flipped, but it is up to us to define whether choosing an unflippable bond counts as a flip or not. We in fact chose to count those bonds towards the flip trial count of the lattice.

We were not able to fully determine the long-time behavior of the infinite DHC system due to a specific finite-size effect. We used finite latices with periodic boundary conditions to approximate the infinite system. As the bonds in the system are flipped the tiles which neighbor each other change, and eventually one tile will gain enough bonds that it will wrap around the finite system and neighbor itself, dividing the lattice into two. When this occurred we stopped the simulation as the lattice was no longer properly defined.

V. Determination of $p_c$

We found the percolation thresholds of the DHC computationally using a method described fully in Appendix B. At the percolation threshold the ratio

$$y(s, p) = \frac{sP_{\geq s}}{\langle s' \leq s \rangle}$$

reaches a constant value of $\delta - 1$, for large $s$, where $s$ is the size of a cluster, $\langle s' \leq s \rangle$ is the expectation value of $s$ for
TABLE I: Honeycomb variant (three-coordinated) lattices

| Lattice   | Common Name | $\sigma$ | $p_{c_{\text{site}}}$ | $p_{c_{\text{bond}}}$ |
|-----------|-------------|----------|------------------------|------------------------|
| $(6^3)$   | honeycomb   | 0.000    | 0.697040 [3]           | 0.652704 [9]           |
| -         | Poisson-Voronoi | 1.314 [10] | 0.71410 [11]           | 0.66693 [11]           |
| $(4, 8^2)$| bathroom tile | 2.061    | 0.729723 [12]          | 0.676803 [12]          |
| $(4, 6, 12)$| cross     | 2.828    | 0.747801 [12]          | 0.693731 [12]          |
| $\frac{6}{2}(3^2) + \frac{1}{2}(9^2)$ | martini | 3.000    | 0.764826 [13]          | 0.707107 [14]          |
| $(3, 12^2)$| three-twelve | 4.235    | 0.807901 [15]          | 0.740421 [12]          |

Variance in Degree of Dual Lattice

Percolation Threshold (Bond or Site)

0.64
0.66
0.68
0.70
0.72
0.74
0.76
0.78
0.80
0.82
0.84

Site Threshold Honeycomb Variant Lattices
Bond Threshold Honeycomb Variant Lattices
Site Threshold Defective Honeycomb
Bond Threshold Defective Honeycomb

To find the thresholds, we carried out 100,000 simulations on lattices of size $n = 1500$, for values of $\sigma$ equal to 0.00, 0.24, 0.55, 0.77, 1.09, 1.28, 1.52, 1.81, and 2.10 for both site and bond percolation. The details are shown in figure 5.

The final results are shown in Figure 5, which gives the relationship between the $p_c$ and $\sigma$ of the lattices. We also include in that plot the known thresholds of the honeycomb variant lattices (other lattices with coordination number 3). As $\sigma$ increases both the site and bond percolation thresholds increase in a fairly predictable manner. The honeycomb variant lattices we used, which were all the three-coordinated ones where thresholds are known, are summarized in table I.

clusters up to size $s$, $P_{2s}$ is the probability that a vertex is in a cluster of at least size $s$, and $\delta = 1/(\tau - 2) = 91/5$ (in two dimensions) is a critical exponent of percolation. By graphing $y(s, p)$ for different values of the occupation probability $p$ and finding when it reached 17.2 for large $s$, we were able to determine $p_c$ with high precision.

FIG. 5: Graph of the relationship between the percolation threshold and dual-lattice variance $v = \sigma^2$. The bond (lower) and site (upper) thresholds for the DHC lattice and the previously studied honeycomb variant lattices listed in table I are shown. The linear regression for the bond thresholds is $p_c = 0.00649 v + 0.6533$ and for the site threshold is $p_c = 0.00783 v + 0.6978$.

TABLE II: Site and bond thresholds for the DHC lattices from simulations on 1500 x 1500 systems

| $\sigma$ | $p_{c_{\text{site}}}$ | $p_{c_{\text{bond}}}$ |
|----------|------------------------|------------------------|
| 0.24     | 0.69770 ± 0.00025      | 0.65320 ± 0.00025      |
| 0.55     | 0.69995 ± 0.00015      | 0.65497 ± 0.00025      |
| 0.77     | 0.70200 ± 0.00025      | 0.65715 ± 0.00025      |
| 1.09     | 0.7078 ± 0.00004       | 0.66130 ± 0.000025     |
| 1.28     | 0.71075 ± 0.000025     | 0.66440 ± 0.000025     |
| 1.52     | 0.71659 ± 0.00004      | 0.66860 ± 0.000025     |
| 1.81     | 0.72380 ± 0.000025     | 0.67470 ± 0.000025     |
| 2.10     | 0.73147 ± 0.000012     | 0.68140 ± 0.000025     |
VI. DISCUSSION

We introduced the concept of the DHC lattice as a way to better model real-world lattices which are not perfect honeycombs. The lattices are generated probabilistically by choosing edges uniformly at random from a honeycomb lattice to ‘flip’. This process creates a new lattice with increasing number of defects as the bonds are flipped. We can quantify the severity of the defects by measuring \( \sigma \), the standard deviation, or \( \sigma = \sigma^2 \), the variance in the number of edges per tile. The variance is linearly proportional to the fraction of edges flipped \( f \) for small \( f \). As more edges are flipped the distribution of polygons in the lattice changes such that most tiles are triangles or have a large number of edges.

There is a series of well-studied three-coordinated lattices which have many of the same basic properties of the DHC lattices. We compared the percolation threshold of these lattices and found that the percolation threshold for all of the lattices increases linearly in \( v \). This means that the variance is a useful quantity to use to determine the percolation threshold of this class of lattice. The increase of thresholds as \( v \) increases means that ideal honeycomb lattice is more robust to connecting paths through the network than its defective counterparts. Patterning in biological systems takes energy and so we would expect that patterns like the honeycomb should provide some advantage over disordered lattices in order to be evolutionarily favorable. The fact that the honeycomb pattern is more robust against site or bond failures than similar networks with defects may be part of the reason it is so common in the natural world.

Appendix A: Initializing the Lattice

In order to use algorithm \( \square \) the lattice must be stored as a list of clockwise-ordered vertices around each cell. There are many different solutions which give unique labels to every vertex, and working out the proper boundary conditions and determining the vertices in each tile can be time consuming. We have reproduced our algorithm here for interested parties wishing to produce their own DHC lattices. We used the labeling method shown in figure to determine tile and site IDs. Tiles are labeled 0 to \( n^2 - 1 \) in order by row and then column. The top site of each hexagon has id \( \text{ID}_{\text{site}} = 2(\text{ID}_{\text{tile}}) \) and the next clockwise site has \( \text{ID}_{\text{site}} = 2(\text{ID}_{\text{tile}}) + 1 \). The following pseudocode produces a two-dimensional array of \( n \times n \) hexagons. The \( i \)th row of the array gives the clockwise ID’s of vertices of
Algorithm 2 Initialize Lattice

1: for \( z=0 \) to \( n^2-1 \) do
2: \( \alpha = \frac{z-(z \mod n)}{n} \) (mod 2)
3: \( z' = 2(z+n \mod n) \)
4: \( \text{tiles}[z][0] = 2z \)
5: \( \text{tiles}[z][1] = 2z+1 \)
6: if \( \alpha = 1 \) then
7: \( \text{tiles}[z][2] = z' \)
8: if \( z \mod n = 0 \) then
9: \( \text{tiles}[z][3] = 2(z+2n-1 \mod n^2) + 1 \)
10: \( \text{tiles}[z][4] = 2(z+2n-1 \mod n^2) \)
11: else
12: \( \text{tiles}[z][3] = z' - 1 \)
13: \( \text{tiles}[z][4] = z' - 2 \)
14: end if
15: end if
16: if \( z \mod n = n-1 \) then
17: \( \text{tiles}[z][2] = 2z + 2 \)
18: \( \text{tiles}[z][2] = z' + 2 \)
19: else
20: \( \text{tiles}[z][2] = z' + 1 \)
21: \( \text{tiles}[z][2] = z' \)
22: end if
23: if \( z \mod n \neq 0 \) then
24: \( \text{tiles}[z][5] = 2z - 1 \)
25: else
26: \( \text{tiles}[z][5] = 2(z+n-1) + 1 \)
27: end if
28: end for

Appendix B: Determining PC

In section \( \text{V} \) we presented data showing the relationship between the percolation threshold of some DHC lattices with the honeycomb variant lattices. The site and bond percolation thresholds for the honeycomb variant lattices are known (see figure \( \text{I} \)). There are many established ways to estimate the percolation threshold of a lattice using simulations \([6, 16]\), however we found that a novel method based on a cluster growing algorithm was more practical for our problem. In this section we will describe the theory behind our method and outline the computational model we used.

The theory behind the method is straightforward. We found a ratio of properties which we will call \( y \) (defined below) that is a constant function of \( s \) (for large \( s \)) at the percolation threshold. We can then use a normal cluster growing algorithm to determine this ratio at different occupation probabilities, and narrow in on the value of \( p_c \).

The following is a short theoretical derivation proving that our ratio \( y \) should be constant for \( s \) at the percolation threshold. Let \( n_s \) be the number of clusters of size \( s \), and \( P_s \) be the probability that any site is in a cluster of size \( s \). It is well known \([6]\) that at \( p_c \) the number of clusters of size \( s \), is \( A s^{-\tau} \), where \( A \) is some constant and \( \tau \) is the Fisher scaling exponent. From this relationship it follows that the probability that a site on the lattice belongs to a cluster of size \( s \) is given by

\[
P_s = s n_s = A s^{1-\tau}. \tag{B1}\]

which implies that the probability a site is in a cluster of at least size \( s \) is given by

\[
P_{\geq s} = \int_s^{\infty} P_{s'} ds' = \int_s^{\infty} A(s')^{1-\tau} ds', \tag{B2}\]

at \( p_c \). The value of \( \tau \) is between 2 and 2.5 in every dimension so the integral in (B2) converges and evaluates to

\[
P_{\geq s} = \frac{A s^{2-\tau}}{\tau - 2}. \tag{B3}\]
FIG. 6: Example of finding the percolation threshold using our method. We grew 10000 clusters on a honeycomb lattice at various occupation probabilities $p$. Statistics on the average $P_{\geq s}$ and $(s' \leq s)$ were gathered and the ratio $y = sP_{\geq s}/\langle s' \leq s \rangle$ was plotted versus the cluster size $s$. At the percolation threshold the resulting curve should be flat. On the left we get the approximate value of $p_c$ for site percolation on the honeycomb lattice. On the right we narrow in on the exact value. Using this method we determined that the site percolation threshold was $0.697000 \pm 0.00005$ which agrees with the known value of $0.697040 [3]$.

We want our ratio of lattice properties $y$ to be constant at $p_c$, so we incorporated the average value of the size of all clusters up to size $s$:

$$\langle s' \leq s \rangle = \int_1^s s' P_{s'} = \int_1^s A(s')^{2-\tau} ds' = \frac{As^{3-\tau} - 1}{3-\tau}. \tag{B4}$$

It follows that the ratio $y$, defined as

$$y = \frac{sP_{\geq s}}{\langle s' \leq s \rangle} = \frac{3 - \tau}{\tau - 2} = \frac{86}{5}.$$

is constant for large $s$.

In order to use the ratio $y$ to find the value of $p_c$ we ran a standard epidemic cluster growing algorithm with occupation probability $p$ and recorded the number of sites in each cluster until appropriate statistics had been collected. Clusters were grown until a preset maximum cutoff $s_{\text{max}}$ was hit. Data were binned by $\log_2 s$, that is, bins for $s = 1, 2-3, 4-7, 8-15, \ldots$ and those that hit $s_{\text{max}}$, with bins both for the number in each bin, and the total $s$ of all the clusters in each bin. Summing these bins respectively above $s$ and below $s$ allowed us to calculate $y$ for $s = 2, 4, 8, \ldots, s_{\text{max}}$.

When the occupation threshold is above $p_c$ the curve for $f$ is above $17.2$, and when the occupation threshold is below $p_c$ the curve is below $17.2$. Therefore, we can narrow in on the percolation threshold by testing various values of $p$ and graphing them as shown in figure [6]. First we scan through a range of $p$ values using a large step size and short simulations to determine the approximate $p_c$. We then run extensive simulations on just a few values of $p$ near the threshold to get an accurate value of $p_c$. This method is easy to program and efficient when one is trying to find just the threshold, and not the behavior of the system for all values of $p$, in which case other methods [17, 18] are more efficacious.

For this method to work well, the lattice has to be big enough so that the maximum cluster size $s_{\text{max}}$ could be reached before the boundaries are hit. Then there are no finite-size effects due to the boundaries. However, there are still finite-size (lattice) effects for small clusters; these are generally unimportant for $s > 1000$. 