Numerical Experiments on Percolation Clusters and Random Walks

Tiffany Ferreira  Senior  Computational Mathematics & Physics

David Prairie  Junior  Computational Mathematics & Computer Engineering

University of Massachusetts Dartmouth

Special Thanks To: NSF, CSUMS, Professor Yanlai, Professor Gary Davis, Professor Alfa Heryudono

17 August, 2012
Abstract

The random walk theory is able to assist in many real world simulations; in this case it was used to find behavior of particles in an open field. From the simulation runs an understanding of how particles behave given a set probability to walk in a particular direction. In addition, percolation theory was used to look for a correlation in clusters for these points. This report will depict the methods used behind these simulations and also the results obtained.
Table of Contents

I Introduction.......................................................................................................................... 4
II Motivation............................................................................................................................ 4
III Methodology....................................................................................................................... 5
IV Numerical Results................................................................................................................ 9
V Future Work.......................................................................................................................... 14
VI Reflections.......................................................................................................................... 14
VII Conclusions....................................................................................................................... 15
I Introduction

To understand the behavior of particles in a given medium two simplified models can be developed to replicate the random position and motion of particles. These two models are based on two mathematical concepts known as percolation theory and random walk theory. Both of these have relevant applications in the physical world.

Percolation theory describes the behavior of points connected by bonds in a given area. The points are typically called sites or vertices, and the bonds may be called edges or connections. A cluster is a connected group of points that lie within the radii of other points. The critical radius is known as the minimum radius needed to have a complete cluster from one side of the model to the other. This study focuses on the change in the critical radii as the number of points increases.

Random walks have been a topic of interest in many fields of study. The simplest two-dimensional case considers a point allowed to travel either to the left, right, up or down. As a general rule the behavior of an unconstrained random walk is relatively well known. This experiment is similar to the unconstrained walk with a few subtle differences. The revised idea begins with a particle that has the ability to travel in eight directions on a square two-dimensional lattice. Altering the probability that the particle will travel in a given direction will produce some effects on the number of steps the particle takes to travel from the starting point on the left hand side to the collector, which is located on the right hand side of the grid.

Using programs such as C++, Mathematica, and Excel will allow accurate results to be produced and stored. The data recorded by these two models will then be analyzed to identify trends and other useful information. Some results may be compared to previous works, while others will lead to new discoveries and questions about the behaviors of these two theories.

II Motivation

Throughout the universe many applications of percolation theory and random walks are displayed in various phenomena. Applications in material science, chemistry, and physics involve the behavior of connected clusters described by the laws governing percolation theory. For a two-dimensional lattice it may appear simple, but this theory can be extended for a lattice of any dimension. Random walks appear in the studies of many fields such as economics, physics, population dynamics, neuroscience, and computer science. Understanding the laws dictating these processes may allow us to predict outcomes for cases with the same structure.

One topic with increasing importance is the ability to create and transport electricity. This may be related to either percolation theory or modeled by a random walk. In photovoltaic cells the study of material science becomes relevant to percolation theory. When searching for a material which provides optimal conductivity the porosity of the material becomes a variable. The relation to percolation theory arises when the porosity of the material is...
determined by how many atoms can fit in a given area of space. The more sites in a given area will change the critical radius each point will need to have to have a percolation network. These differences may affect the efficiency of the photovoltaic cells, and therefore is of interest.

Random walks have been a topic of interest since the early 20th century. Their ability to model systems that involve stochastic activity has proven to be useful. Models involving random walks include estimating the price of shares, Brownian motion, and finding the ideal chain in polymer physics. The behavior of random walks with no constraints has been studied thoroughly and has produced similar results in various tests. Altering the probabilities of which direction the particle will proceed to will provide new results on the minimum, maximum, and average number of steps the particle will take to make it from the left side of the graph to the right side of the graph. Using percolation theory and random walks the attempt to mimic the random motion of electron transport will be completed by computational methods developed in the computer programs Mathematica and C++.

III Methodology

To replicate the movement of an electron in disorganized material as percolation on a random two-dimensional lattice, a program in Mathematica was developed to simulate the connectivity between a varying number of sites and the minimum percolation radius between the sites. The user has the ability to change: the number of particles being tested, the size of the field the particles are distributed over, the size the particle is displayed as, and how many times you would like the program to iterate with a given number of particles. After selecting these choices the program will be able to calculate the shortest path from the leftmost point to the rightmost point on the graph. It will also be able to tell you the minimum radius needed for each point to be connected. Appendix A will display the modified Mathematica code.

This program allows for the calculations of some useful information of the data that is collected. After running the program multiple times the user is able to calculate data such as the mean critical radius, the standard deviation, the run time of the simulation, the distribution of the critical radii, as well as the kurtosis and skewness of the distribution. Another advantage this program has is the ability to use a feature called Parallelize[], which allows for quicker computations by utilizing both kernels in the computer.

The program initializes by generating a table consisting of a given number of random points located between \( x = [-1,1] \) and \( y = [-1,1] \). A visual plot of these random points can be displayed if desired. After the random points are generated the simulation then begins to sort through the coordinates in order to find the leftmost and rightmost point on the grid. It then begins creating the optimized path between the random points. From this path the program determines the smallest radius needed for the shortest path to exist.

The function minimumPercolationNetwork[] is initialized by a list of random points and an upper bound on the critical radius. For these results the upper bound value will be 0.5 due to the range of the number of particles as well as the designated area. Executing this function will return a minimum critical radius for which a percolation network will exist.
The simulation includes a function called isPercolationNetwork[]. This function requires a list of random points, a critical radius, and it includes a Boolean statement, enabling the graph to be displayed or not, and returns a visual of the random points with the percolation network highlighted if one exists. Combining these two functions allows the simulation to generate the critical radius for a given number of points many times, from which the data can be analyzed numerically.

These are three images generated using the Mathematica program

Using Microsoft Excel to organize and utilize the data obtained, crucial information will be revealed. Evaluating the variance in the critical radius with respect to the variance in the number of particles used will be able to produce a trend line for the data. The distribution of the critical radius data will be analyzed as well. Utilizing many of the components in Excel and Mathematica the numerical results produced can be evaluated and made visual for percolation on a random two-dimensional lattice.

In order to develop results in this project a program was written to simulate the random walk of the electron particle. This simulator was written in C++, a high level language, which contains many key features. These features include the ability to run a random walk or a self-avoiding random walk, choose the number of particles, choose the number of blockers within the map, the amount of times you want to run the simulation, change the probability for the particle to move from right, and the outputting of data for analyzing. In addition, it would automatically analyze some of the data and output results that are more easily readable.

The outputs of the simulator are key for allowing the user to obtain results needed for modeling. These outputs consist of the max, min, and average number of steps to go from the starting point to the collector. In addition the coordinates of the blockers are printed, the number of steps for a particle to become trapped, the coordinates of the particles as they move across the field, and the amount of time each simulation took to run.

From this data it is possible to plot the paths of the particles without slowing down the simulation. You also have the ability to use the max, min, and average steps for comparing the amount of steps needed to move from the start to the collector (end) with the probability of the particle changing. These outputs allow the user to develop viable results for modeling.

Beyond the surface of the user interface lies 779 lines of integrate systems of loops, random cases, and valid verifiers. A key point to take note of would be the utilization of s predefined
structure called POINT. This structure allowed the program to take a more organized structure with the ability to call upon one variable that held all the data and information of its point instead of dealing with multiple 2D vectors. Instead it was written with a vector of Points, thus making it much more manageable. This vector is created towards the beginning of the program and is passed along to each function as needed and returned when needed. This allowed a smoother flowing program.

The flow of the program contains four sections; a main function, initialization functions, operation functions (where the simulation is ran), and analyzing functions. The main function allows the user to change all the variables such as number of points or number of blockers. This function will also take you to the other three sections of the program.

The initialization function will set up the vector of points, position all the blockers within the field, and it will also set up the output files. The reasoning the output files need to be setup prior to the simulation running is because these files are accessed multiple times throughout the program and during each of these times new data is appended to the end of the file.

Following the initialization functions is the operation functions, which contains five functions within it. Three of these are for the random walk and the other two are for the self-avoiding random walk. This report will focus on the random walk functions, which consist of the random walk function, the valid function, and a random number generator function.

The random walk function will call upon the random number generator to obtain a number between one and eight (the eight directions the particle can travel). Once this number is obtained it will go to a case statement and determine if the particle is able to move in that direction. Some reasons why is might not be able to move in that direction could be that moving there will take it outside of the field or there could already be a particle there. If it is not able to make the move it will then obtain a new random number, otherwise it will move the particle and move on to updating the checking functions.

The checking functions will keep track of where every particle is and will set a flag once the particle reaches the collector, thus stopping it from being moved. This also prints out some of the data such as the coordinates of the particles throughout the simulation.

Lastly to note how the random number is obtained. This is obtained by selecting a random number between one and ten thousand. It will use the probability to go to the right in order to select the probabilities for each other directions. The equation used for this is as follows:

\[ x \] is the probability to go to the left and \[ y \] is the probability to go to the right. The following table will show how the probability is calculated for the other directions.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>Predefined</td>
</tr>
<tr>
<td>North East and South East</td>
<td>5 ( \times x )</td>
</tr>
<tr>
<td>North and South</td>
<td>3 ( \times x )</td>
</tr>
<tr>
<td>North West and South West</td>
<td>2 ( \times x )</td>
</tr>
<tr>
<td>West</td>
<td>( x )</td>
</tr>
</tbody>
</table>
These numbers are then used to determine which direction the particle will move in. This set up allows the user to select a probability between 1 and 100 percent chance of moving to the right.

With all this a lot of data is created, which is checked within the analyze functions. Here it will read threw all the simulations ran and determine what the max, min, and average number of steps required to reach the end. In addition it will print out the coordinates of the points and the coordinates of the blockers. This data is printed out to their own files for ease of reading.

Lastly the data that was printed out was taken and put into tables within Microsoft Excel and used to plot the average number of steps needed to go from the left side of the field to the right side of the field. This graph was then used to determine an equation of the line that models how a particle behaves with various probabilities.

IV Numerical Result

After allowing the simulation in Mathematica to run one thousand times for twenty-five different numbers of particles the data collected could be analyzed. The information collected was the mean critical radius, the standard deviation, and the distribution of the critical radii. After organizing the information collected other data such as trend lines and normalizing factors could be calculated. Attached in appendix A are the exact numerical results obtained during these experiments.

The distribution of the critical radii appears to be skewed to the right for each of the twenty-five simulations. To be sure this distribution would hold for a larger number of iterations, the program was run ten thousand times for two hundred particles and the distribution produced was similar. Below are a few histograms to visualize the skewness of the distribution of critical radii for three different numbers of particles. The first is an image of two hundred particles and the second is an image of four hundred and twenty-five particle. Both of these simulations were run one thousand times. Lastly, in order to make sure that the skewness of the histogram did not arise from having a small number of samples the third image is two hundred particles run ten thousand times.
Figure 1-3: These three histograms show the minimum critical radius as the horizontal axis and the number of times that critical radius occurred.
One useful piece of data that appears in previous works of two-dimensional percolation is the normalized critical radius. In order to calculate the bonding radius the critical radius found by the simulation must be normalized by some factor $R_s$. The way to derive $R_s$ is as follows:

$$N_s \pi R_s^2 = 4$$

$$R_s = 2(N_s \pi)^{-1/2}$$

Notice that $N_s$ is the number of points in the given area, it is being multiplied by the area of a circle with radius $R_s$, and four is the area of the grid size that was determined by the programmer.

In comparison to previous results obtained it appears that as the number of points is increased the normalized critical radius asymptotically approaches a limit near 1.3. The graph that follows depicts this relation.

Results similar to graph above have been created at an earlier time by researchers interested in probabilistic bond percolation. Their results involve relating how much of the sample is percolation versus the normalized critical radius. When considering the probabilistic bond probability as 0.5 their results range from 1.22 to 1.34. These results are similar to the ones obtained during this experiment.

After calculating the mean critical radius for each of the twenty-five different numbers of sites, it is useful to plot the mean radius versus the number of sites in the simulation. From this graph Excel is able to plot a trend line for the mean critical radius ($R_c$) as well as the
normalizing factor ($R_s$). It seems as if the data collected follow a power law relation. The two trend lines appear to have a power law relation with an exponent near $-0.5$ with some scalar quantity multiplied by the variable. Physically this implies that as the number of particles grows the average critical radius needed for percolation to occur decreases. The following graph portrays this relationship. The dotted lines are the trend lines that most accurately fit the data points collected by the simulation.

The set up for the random walk simulation was one particle being randomly placed on the left boundary and having it randomly walk around until it reaches the right boundary. This set up was run two hundred thousand times with the number of steps needed to reach the end being recorded. In addition, the minimum, maximum, and average number of steps needed was calculated over the entire set of runs. This data was then taken and recorded based on the probability to go to the right. As said earlier, it was started as a 5% chance to the right and increased by 5% until 100%. This gave twenty data sets of minimums, maximums, and averages. It was this data that was analyzed.

When looking at these results you will see that the flow of the data goes down to the right. In addition, the absolute minimum number of steps is one hundred steps; this is due to the fact that the field is one hundred by one hundred. The average line is fairly smooth with little to no bumps. However, the maximum and minimum lines are not so smooth with a
couple bumps during their progression. When plotting a trend line over the average, we were able to have a closely fitting line that would work within the bounds of greater than zero percent up to one hundred percent. The reason why it will not work at zero percent is because with a zero percent chance to go to the right it would be impossible for the point to move from the left to the right.

The following is the equation of the trend line:

\[
y = -2E^{-10}x^6 + 3E^{-08}x^5 + 7E^{-06}x^4 - 0.0016x^3 + 0.1346x^2 - 7.1706x + 321.76
\]

This brought some curiosity as to the distribution to the number of steps within each probability. The charts in appendix B depict the distribution. These charts show how the distribution is that of an increase to a peak then decrease back down. As the probabilities increase these humps become closer to the left and narrow. This would make sense because the minimum number of steps is one hundred, so as the probability increases the average number of steps will become closer to one hundred, thus moving the peak to the left.

Overall the resulting data was what was previously estimated. Although, it was thought that the distribution would have been smoother, but there were some points beyond the general curve. With the increasing of the probability to the right the results were also as expected; which was that as you increase the probability to the right, the average number of steps to reach the right would decrease significantly.
VI Future Work

After this program concludes there are a few ways the percolation aspect of this project could be extended to produce more results in the future. One aspect of this project that has the potential to be continued is research on the distribution of the critical radii. It would be interesting to do more research specializing in the skewness of the data. From preliminary research it appears to follow a gamma distribution. Unfortunately time constraints will prohibit any further progress towards understanding the physical meaning behind this distribution. It would also be interesting to examine if there are any relations to material science from the data that has been collected thus far.

VII Reflections

Tiffany Ferreira

Over the course of the summer the CSUMS program has allowed me to build many valuable skills that will be used throughout the rest of my career as a research assistant. The importance of a peer network has allowed me to engage in real world collaboration techniques in order to complete an interesting project. When I first came into this program I had little knowledge of any computer programming. However, with the help of a few professors and my partner I was able to complete a few working programs both in Mathematica and Python.

Although the topic I choose initially proved to be too difficult for the level of my physics courses, I was able to redirect the goal of my project to something more realistic. Another way that I have improved my skills is in the presentation aspect of this program. I no longer feel shy when presenting my findings in a small group of peers and advisors.

Another positive aspect of this program is attending the annual SIAM meeting, which took place in Minneapolis, MN. Going to this meeting taught me many different presentation techniques that have the ability to make what you are talking about more interesting and understandable to a wide audience. Being able to communicate your ideas is a very difficult task, but after this summer I feel more confident in my abilities and will continue to keep what I have learned in mind as I continue on the way to my career.

David Prairie

During the CSUMs program this summer I have sharpened some skills and learned new skills. Coming into this summer I was fairly confident with my C++ background and my ability to design and develop programs within this language. I was unsure as to what I wanted to research this summer, however a few days into the program, it was suggested that I try looking to the random walk theory. After getting some parameters I began developing a program that would simulate a random walk.
Through writing this simulator I have strengthened my knowledge and abilities within the C++ language. I am now much more confident when approaching developing a program within C++. In addition, I have learned how to analysis data more effectively. I was able to take the data from my simulation and compare it upon different runs. I learned how to make sure the data collected was viable data. Overall I feel as if I have gained a lot through this program. I am now more confident with my abilities in programming, analyzing, and researching.

VII Conclusions

In conclusion, the ability to model phenomena involving stochastic behavior can be replicated by computer simulations created by a variety of different programs. The percolation program developed in Mathematica provides visuals as well as relative information on percolation theory. The distribution of the critical radii does not follow a normal distribution, and is skewed to the right. This could potentially be another topic to spend more time researching. The random walk program developed in C++ provides us with interesting information on the path of the random walk after altering the probabilities of each direction it may travel. If a particle has a greater probability to travel to the right, then the number of steps it takes to make it to the collector decreases. With the use of teamwork and the guidance of helpful professors many useful techniques were acquired over the course of this summer program, and these techniques will remain relevant in future research opportunities.
Appendix A

The modified code:

```mathematica
Percolation Program

Options[isPercolationNetwork] = {ShowGraph -> False}
linked[mat_?MatrixQ, r_?Positive] := Map[Boolean[0 < # < r] &, mat, {2}]
isPercolationNetwork[points : (_?NumericQ, _?NumericQ) .., r_?Positive, opts : OptionsPattern[]] :=
Module[{ed = Outer[EuclideanDistance, points, points, 1],
  leftmost = Position[points, {Min[points[[All, 1]]], _}][[1, 1]],
  rightmost = Position[points, {Max[points[[All, 1]]], _}][[1, 1]],
  With[{gg = AdjacencyGraph[linked[ed, r], VertexCoordinates -> points]},
    If[OptionValue[ShowGraph], HighlightGraph[gg, PathGraph[FindShortestPath[gg, leftmost, rightmost]]],
      Length[FindShortestPath[gg, leftmost, rightmost]] > 1]]}

{ShowGraph -> False}

minimumPercolationNetwork[points : (_?NumericQ, _?NumericQ) .., r0_?Positive] :=
Module[{x = r0}, While[isPercolationNetwork[points, x], x = x - 0.01];
  Print[x + 0.01];
  isPercolationNetwork[points, x + 0.01, ShowGraph -> True]]

minimumPercolationNetwork[points : (_?NumericQ, _?NumericQ) .., r0_?Positive] :=
Module[{x = r0}, While[isPercolationNetwork[points, x], x = x - 0.01];
  Print[x + 0.01];
  isPercolationNetwork[points, x + 0.01];
  r + 0.01]

randPts = Table[RandomReal[{-1, 1}, 2], {200}];
randPlot = ListPlot[randPts, PlotStyle -> {PointSize[0.01]}, PlotRange -> {{-1, 1}, {-1, 1}}, AspectRatio -> 1, Frame -> True]

Timing[
  A = Parallelize[Table[minimumPercolationNetwork[Table[RandomReal[{-1, 1}, 2], {350}], 0.5], {i, 1, 1000, 1}]]
  Save["percolation350", A]
]

Exact Numerical Data:

<table>
<thead>
<tr>
<th># of sites(Ns)</th>
<th>Mean(Rc)</th>
<th>Normalizing Factor(Rs)</th>
<th>Rc/2*Rs</th>
<th>Stand. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.48915</td>
<td>0.2256758</td>
<td>1.083744912</td>
<td>0.0360621</td>
</tr>
<tr>
<td>50</td>
<td>0.39543</td>
<td>0.159576912</td>
<td>1.238995025</td>
<td>0.060434</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>0.32592</td>
<td>0.130294003</td>
<td>1.250709903</td>
<td>0.0486716</td>
</tr>
<tr>
<td>100</td>
<td>0.28484</td>
<td>0.112837917</td>
<td>1.262164387</td>
<td>0.0437549</td>
</tr>
<tr>
<td>125</td>
<td>0.25508</td>
<td>0.100925301</td>
<td>1.263706909</td>
<td>0.0379741</td>
</tr>
<tr>
<td>150</td>
<td>0.23397</td>
<td>0.092131773</td>
<td>1.269757394</td>
<td>0.0348016</td>
</tr>
<tr>
<td>175</td>
<td>0.21639</td>
<td>0.085297447</td>
<td>1.268443585</td>
<td>0.0330175</td>
</tr>
<tr>
<td>200</td>
<td>0.20264</td>
<td>0.079788456</td>
<td>1.269857885</td>
<td>0.0290323</td>
</tr>
<tr>
<td>225</td>
<td>0.19009</td>
<td>0.075225278</td>
<td>1.263471569</td>
<td>0.0260302</td>
</tr>
<tr>
<td>250</td>
<td>0.18237</td>
<td>0.071364965</td>
<td>1.27772781</td>
<td>0.0253836</td>
</tr>
<tr>
<td>275</td>
<td>0.1727</td>
<td>0.068043824</td>
<td>1.269035085</td>
<td>0.0235428</td>
</tr>
<tr>
<td>300</td>
<td>0.16501</td>
<td>0.065147002</td>
<td>1.26644355</td>
<td>0.0216996</td>
</tr>
<tr>
<td>325</td>
<td>0.15869</td>
<td>0.062591215</td>
<td>1.267669921</td>
<td>0.0208667</td>
</tr>
<tr>
<td>350</td>
<td>0.15518</td>
<td>0.060314404</td>
<td>1.286425721</td>
<td>0.0219465</td>
</tr>
<tr>
<td>375</td>
<td>0.14935</td>
<td>0.05826925</td>
<td>1.28155074</td>
<td>0.0209599</td>
</tr>
<tr>
<td>400</td>
<td>0.14497</td>
<td>0.056418958</td>
<td>1.284763174</td>
<td>0.0203522</td>
</tr>
<tr>
<td>425</td>
<td>0.13917</td>
<td>0.054734429</td>
<td>1.271320463</td>
<td>0.0181533</td>
</tr>
<tr>
<td>450</td>
<td>0.13619</td>
<td>0.053192304</td>
<td>1.280166393</td>
<td>0.0184153</td>
</tr>
<tr>
<td>475</td>
<td>0.13214</td>
<td>0.051773587</td>
<td>1.276133332</td>
<td>0.0174678</td>
</tr>
<tr>
<td>500</td>
<td>0.12923</td>
<td>0.05046265</td>
<td>1.280451967</td>
<td>0.0162102</td>
</tr>
<tr>
<td>525</td>
<td>0.12568</td>
<td>0.049246504</td>
<td>1.276029659</td>
<td>0.0168826</td>
</tr>
<tr>
<td>550</td>
<td>0.12287</td>
<td>0.048114249</td>
<td>1.276856666</td>
<td>0.0159348</td>
</tr>
<tr>
<td>575</td>
<td>0.12051</td>
<td>0.047056664</td>
<td>1.250709903</td>
<td>0.0152076</td>
</tr>
<tr>
<td>600</td>
<td>0.11891</td>
<td>0.046065887</td>
<td>1.290651378</td>
<td>0.0147489</td>
</tr>
<tr>
<td>625</td>
<td>0.11635</td>
<td>0.045135167</td>
<td>1.288906285</td>
<td>0.0146353</td>
</tr>
</tbody>
</table>
Appendix B

Distribution of steps for walking from left to right and the varying probabilities (5% to 95%).
Numerical Experiments on Percolation Clusters and Random Walks

25 Percent

30 Percent
Numerical Experiments on Percolation Clusters and Random Walks

65 Percent

70 Percent
95 Percent
References


This code can be found at: