Math IA

Expected Value of an Optimal Path

Joshua Beard

February 2023

Introduction and Rationale

In computer science, finding an optimal path given some heuristic and graph is a common and important problem. This year in computer science class, my teacher presented the class with a simple problem: Given a 5x5 grid of houses, each with an associated amount of candy they supply, and a max amount of houses you can visit, write code that finds the best path (assuming you can only move to adjacent houses) in the grid. This problem fascinated me, and I spent a significant amount of time creating code to find the optimal path in any grid, and for any length. This brought up the question though, for some arbitrary grid and arbitrary path, what is the expected value of the optimal path? I first approached this question through a computer science lens, by adapting my code to run Monte Carlo simulation. The results from this were good, but as a consequence of the method, the results were imprecise. It also took a large amount of time to find these values given the speed of the algorithm (especially when applied many times). This pushed me to look for a mathematical solution, that hopefully would supply an exact solution. This would be useful for several reasons: I could check the accuracy of my code, or I could use it for optimisation by looking for patterns in the problem that could be reapplied to the code.

Aim and Approach

The aim of this paper will be finding a mathematical model to solve for the expected value of the optimal path in a $5x5$ grid of nodes given a path length. Each node will be assigned a random integer value on the interval $[1, 10]$. The heuristic we will be using to determine an optimal path is the average of all the nodes included in the path. Whichever path within the graph has the highest average is defined as the optimal path.

To approach this, first we will look at the current system for finding the expected values using a Monte Carlo simulation. This approach is useful because it gives a dataset that we can analyze to see what form of function might match the data. Intuitively, I suspect that the function will be of the form $\frac{1}{f(x)}$. This is because when the path length equals 1, the expected value of that path will be very close to 10 since it is the essentially the max of 25 random numbers in the range $[1, 10]$. Alternatively, when the path length is 25, the expected value of the optimal path is 5.5. This is because no max is being taken, since there is only one valid path of length 25. If no max is being taken, then the expected value is the same as the expected value of a single node, which is 5.5. Given that the desired function's range is on the interval $[5.5, 10)$ when the function's domain is on the interval $[1, 25]$, and there are diminishing returns as you near both sides, it seems likely that this function is

the multiplicative inverse of some another function. It is possible that the desired function could be a sigmoid.

Despite this, I do not plan on solving for a best fit curve. A best fit would not give a prediction with the accuracy that I desire, so instead I will be solving for the desired function through a different method. I noticed that this problem is similar to finding the max of multiple dice rolls, but with some complications. When the path length is equal to one, the problem is identical to finding the expected value of the max of 25 10-sided dice, but as the path length increases, a problem emerges. Since each path has more nodes, the probability of having certain values increases. In other terms, the probability distribution goes from flat (all "rolls" have the same odds), to curved. This makes it more challenging to find the expected value of the max path, since it is no longer as simple as a maximum between flat distributions. To solve this problem, we will construct a contingency table that describes the frequency of each value for each path length, then we will look at the possible maximums using a "max" contingency table. This process will be explained in much greater detail later. Through this process, we can exactly calculate the expected value for any path length, but there is another problem. These large contingency tables is become too large to compute, so creating a closed form formula is critical if this problem will be realistically computable.

Data Collection and Results

To collect data, I created a program that finds the optimal path for a given length in a random $5x5$ grid of nodes. For each path length, (between 1 and 25), the program runs 20 trials and finds the value of the optimal path. The average of these 20 trials is then evaluated and output. The following table is the resulting data.

Path Length	Average Value (20 trials)
1	9.95
2	9.4
3	8.8166
4	8.7125
5	8.33
6	8.1666
7	7.8357
8	7.4875
9	7.3833
10	7.385
11	6.8
12	7.0083
13	6.6923
14	6.5607
15	6.5433
16	6.3875
17	6.1764
18	6.1416
19	6.1736
20	5.9475
21	5.9880
22	5.5522
23	5.5391
24	5.5229
25	5.574

Figure 1: Data Table

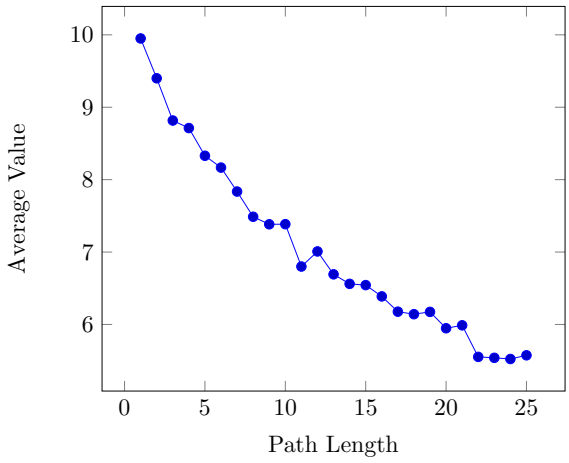

Figure 2: Line Plot of Path Length vs. Average Value

This shows how the function for path length vs expected value could be $\frac{1}{f(x)}$. As shown in Figure 1.b, the function appears to be demonstrating asymtotic behaivior as path length increases. As path length increases, the average value approaches 5.5.

Mathematical Analysis

First, imagine instead of a $5x5$ grid of nodes, we have a $2x1$ grid.

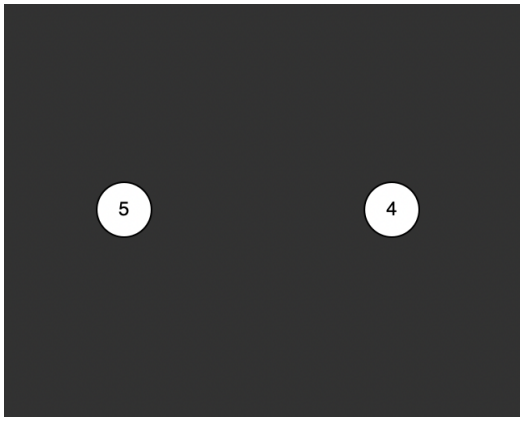

Figure 3: A hypothetical 2x1 grid

Solving this for a path length of one is the same as picking the maximum between the two nodes, which given its simplicity feels like a good place to start. Since these two nodes each range from 1 to 10, we can make a table where each side is the possible values of the nodes, and each cell is the max of each corresponding value.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	3	4	5	6	7	8	9	10
2	2	2	3	4	5	6	7	8	9	10
3	3	3	3	4	5	6	7	8	9	10
4	4	4	4	4	5	6	7	8	9	10
5	5	5	5	5	5	6	7	8	9	10
6	6	6	6	6	6	6	7	8	9	10
7	7	7	7	7	7	7	7	8	9	10
8	8	8	8	8	8	8	8	8	9	10
9	9	9	9	9	9	9	9	9	9	10
10	10	10	10	10	10	10	10	10	10	10

Figure 4: Contingency Table for simple case

The numbers inside this table are the possible values for the maximum of the two nodes, and the frequency of each number in the table is the frequency that the num-

ber should appear when choosing the max of the two random values. To calculate the expected value, we simply average the numbers inside the table to get the expected value of the max of the two nodes, but instead of just summing the numbers and dividing by 100, lets take this opportunity to understand the underlying patterns in this simple case to try to create a formula.

First, notice that each number seems to form a "layer" inside the table. These layers look like they form squares of the same side length as the number they contain. Additionally, these squares seem to be empty inside. This means that we can calculate the amount of each number inside a square with the following formula:.

$$n^2 - (n-1)^2$$

which simplifies to:

$$2n - 1$$

This expression tells us how many times a number, n, will appear inside this contingency table. This allows us to construct a formula that describes this case. Since $2n-1$ is the amount of times a certain number appears within the table, then $n(2n-1)$ is the sum of a given number n, within the table. Summing this expression from 1 to 10, then dividing by 100, would give us the expected value of this case. This can be written as the following:

$$\frac{\Sigma_{n=1}^{10} n(2n-1)}{100} => 7.15$$

This means that the expected value of the optimal path with length one on a $2x1$ grid of nodes is 7.15. Now we must generalize so we can find the expected value for a path length of one on the $5x5$ grid of nodes. First, notice that for this case, the reason that $n^2 - (n-1)^2$ works is because we are calculating the "shell" of numbers formed when the max operation is tabulized. This generalizes well to larger amounts of nodes, because the table (from now on refered to as a matrix) becomes higher dimentional. Specifically, when taking the maximum of n nodes, a n dimentional matrix is required to calculate the expected value of the maximum node. We will still be subtracting "shells", but they will simply be in higher dimentions. Fundimentally, the math is no different. Instead of using the equation that used a difference of squares, we can now use a difference of higher powers:

let d = number of nodes (matrix dimention)

$$n^2 - (n-1)^2 \rightarrow n^d - (n-1)^d$$

Substitute this into the previous equation

$$\frac{\Sigma_{n=1}^{10} n(n^d - (n-1)^d)}{10^d}$$

This equation can be used to find the expected value of a path with length one on the $5x5$ grid by letting d=25. It is $\frac{79394362625910020199166 7}{80000000000000000000000}$, or about 9.92429. Since the experimental data gave a value of 9.95 for this case, this is really good - about 0.2 percent error. This error is caused by random error in by the data collection.

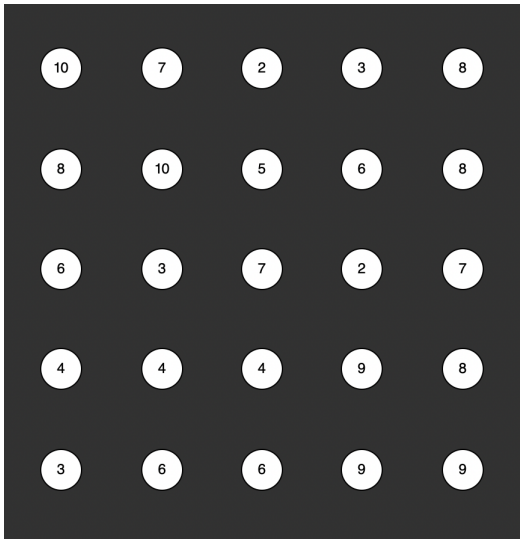

Figure 5: A hypothetical 5x5 grid

To generalize to a $5x5$ grid with different path lengths, this method will need to be significantly adapted. The challenge for greater path length is that the distribution of possible values for a path changes. While a path of length one has a flat distribution, a path with a length greater than one has a non-constant distribution, as shown below.

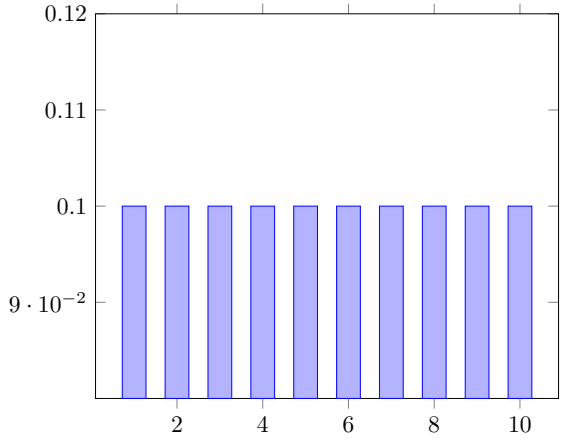

Figure 6: Distribution of possible values for path length one

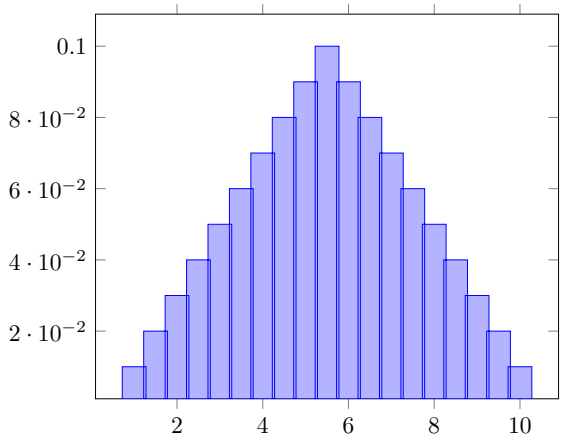

Figure 7: Distribution of possible values for path length two

Given that this distribution is not constant, we must account for the frequency of each value when finding the expected value of the maximum between multiple of these distributions. We can do through some rearangement of the math. First, construct the table that represents this distribution. For now, we will only look at the case where the path length is two, since anything more becomes challenging to visualize, but the math actually stays the same. We also need to divide the numbers on the edges by two since we are look at the average value of the nodes on the path, not the

sum of all. Since we are currently investigating a path with length two, we divide by two. Later, when looking at paths with different lengths, we divide the edges by that length. This is not important to understand to follow the math though, since it is simply to help our data line up with the arbitrary choices I made for how this data is being represented. The math would be the same if we did not divide by two, but the data collected by the computer would also need to scaled accordingly. Here is the table.

	0.5	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**
0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6
1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5
2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7
2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5
3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8
3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5
4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9
4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Figure 8: Table of all possible values for a path of length two

How that we have a table to describes all the values a path of length two can have, we flatten it (move all values into a 1D matrix). We then use this 1D matrix as the edge of the contingency table that gives us all possible maximum values, and apply the same method as before. Note that the table is truncated since the full table would be $100x100$.

	1	**1.5**	**1.5**	**2**	**2**	**2**	**2.5**	**2.5**	**2.5**	**2.5**	···
1	1	1.5	1.5	2	2	2	2.5	2.5	2.5	2.5	···
1.5	1.5	1.5	1.5	2	2	2	2.5	2.5	2.5	2.5	···
1.5	1.5	1.5	1.5	2	2	2	2.5	2.5	2.5	2.5	···
2	2	2	2	2	2	2	2.5	2.5	2.5	2.5	···
2	2	2	2	2	2	2	2.5	2.5	2.5	2.5	···
2	2	2	2	2	2	2	2.5	2.5	2.5	2.5	···
2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	···
2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	···
2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	···
2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	···
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 9: Contingency table for path length of two on a 1x3 grid

Despite the fact that this table looks like it has a simple pattern, with each row becoming one wider, this does not hold. Sadly due to size restrictions the end of the table is not included, but as shown in **Figure 8**, the frequency of the numbers start decreasing after 5.5. Creating a formula to find the sum of all the numbers in **Figure 9** is very challenging, because the numbers follow a complex pattern.

First, we must find a formula to describe the frequency of each number in the table at **Figure 8**. The data, as seen in **Figure 7**, appears to be describeable with an absolute value function. Through experimentation the following equation fits the data in **Figure 8**:

$$c(n) = -|2n - 11| + 10$$

To give an example of how to apply this, imagine that n equals 5. Evauating at 5, we see that we get 9. Looking at **Figure 8**, the number 5 appears 9 times within the table. Now that we have a way to determine how many times each number appears in each side of **Figure 9**, we can use this information to help us determine how many times each number appears within **Figure 9**. First, note that this table is similar to **Figure 4**, since both are tables taking the max between the two sides. Additionally, the "shells" appear in both, which could help us adapt our previous method for this problem. The width of the shells is described by $c(n)$, but there is one other piece of information required to determine the amount of each number in the shells - the index (location) of the shell. This is important because shells further in the table will have more numbers inside them since there is more space to fill. The index, given n, is sum of $c(n)$ from 1 to n. This can be written as the following equation:

$$i(n) = \begin{cases} \Sigma_{k=2}^{2n} c(\frac{k}{2}) & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases}$$

Now, we can simply subtract the square at $i(n-1)$ from the square at $i(n)$ to find the amount of times a certain number appears within **Figure 9**. The equation for this would be the following:

$$a(n) = i(n)^2 - i(n - \frac{1}{2})^2$$

To see this in action, lets find the amount of times 2 appears in **Figure 9**. $i(2)$ equals $c(1) + c(1.5) + c(2)$, which equals 6. $i(2 - \frac{1}{2})$ equals 3. $6^2 - 3^2$ equals 27, which is correct! Stepping back, we have just found an equation to find the frequency that a certain value is the max in a $3x1$ grid of nodes when the path length is two (essentially the max of two paths with length two). To generalize to a larger grid of nodes, specifically $5x5$, we can make one simple adjustment to $a(n)$ - Exchange the

square on each term with a variable representing the total paths of length two within the desired grid. This can be written as the following:

$$a(n,p) = i(n)^p - i(n - \frac{1}{2})^p$$

where p is the total paths of length two within the grid

The next critical insight is that the a function works for any path length, as long as it is supplied with an i function that describes the distribution of that path, and one other slight change. Instead of $\frac{1}{2}$, the a function should have $\frac{1}{l}$, where l is the path length. This is because since we are taking the average of the nodes in a path to find its value, the shift between numbers as shown in **Figure 9** is one over the path size (because we divide by path size when taking the average). Now, the a function should look like the following:

$$a(n,l,p) = i(n)^p - i(n - \frac{1}{l})^p$$

Since we are generalizing, we should also generalize the notation of the i function. We will do this by indexing it to the path length that it describes a distribution for. From now on, it will look like this:

$$i_l(n)$$

We should also redefine the a function with this new notation:

$$a(n,l,p) = i_l(n)^p - i_l(n - \frac{1}{l})^p$$

Using this new notation, we can actually write a function for the expected value of any path length using the following equation:

$$E(l,p) = \frac{\Sigma_{k=l}^{10l} a(\frac{k}{l}, l, p) \cdot k}{10^{lp}}$$

To explain this equation, I will go through each part. First, the bounds of summation. The reason that we are summing from l to $10l$ is pretty straight-forward, and involves the division by l inside the call to the a function. Since the step size of the possible values for a path with length l is $\frac{1}{l}$, we need a way to make the summation have a variable step size. This trick to set the step size to $\frac{1}{l}$ isn't too hard to understand if you think about it, but is challenging to explain. For now, it is best to just understand that these terms are setting the step size. Next, I will explain the call to the a function. Since the a function gives the amount of times a certain term appears within a certain contingency table, summing all of these multiplied by the value k gives the complete sum of all possible maximum paths with a certain length

in a given grid. Dividing this by the total amount of possible maximum paths, which is clearly 10^{lp}, we are able to find the expected value!

Unfortunately though, this equation though has two clear requirements that we do not have yet, a general form for $i_l(n)$ and a way to find p. The remainder of this essay will focus on these. We will start with looking at the class of i functions. First, lets make a slight modification to the i function so that it iterates correctly when given a different l value:

$$i_l(n) = \begin{cases} \Sigma_{k=l}^{ln} c_l(\frac{k}{l}) & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases}$$

This just passes the l indexing issue to the c function, so lets take a look there now (also from now on the c function will be indexed by l). First, lets notice that just as the constant distribution was not of the general form, the absolute value equation is also not of a general form. We know we are looking for something that tends to a normal distribution as l goes to infiniy. An absolute value graph will never do this since it always has straight edges, but a different function class does! Combinatorics are the answer. For example, Pascal's triangle tends to a normal distribution as you approach infinite rows, and it is highly likely we are looking for something similar. Sadly Pascal's traingle does not work for us though, because it deals with binomials, while we are working with variables that range from 1 to 10, meaning we need a sort of "10-nomial" distribution. This can be expressed as the coeffecients to the equation:

$$(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9)^l$$

To do this, we can use the multinomial theorum, but this is beyond the scope of this paper. Instead, we will just define the behaivior of $c_l(n)$. First, n is a number between 1 and 10 (inclusive), with a step size of $\frac{1}{l}$. For example, if the path length was 2, valid inputs would be 1, 1.5, 2, 2.5, 3,$\cdots$, 9.5, 10. The output of the c function is the coefficient in the expansion of the previously mentioned equation that is in the same position as the input. The sizes of the input set and the coefficient set are the same for any l, so there are no domain issues. Another way to think about the behaivior of the c function is that it is indexing a higher dimentional version of Pascal's Triangle. The l term is the row, and the n term is converted to an index for how far over in the row the desired number is. Below is a table showing this behaivior for small values of l.

Path Length (l)	**Outputs of $c_l(n)$ on the input set**
1	1 1 1 1 1 1 1 1 1 1
2	1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1
3	1 3 6 10 15 21 28 36 45 55 63 69 73 75 75 73 69 63 55 45 36 28 21 15 10 6 3 1

Figure 10: Output sets from $c_l(n)$ for different path lengths

Each of these rows again represents the width of shells in **Figure 9**. When the path length increases, the table in **Figure 9** becomes more dimentional. This makes it hard to visualize, but thinking about the row width in terms of the expansion of this equation is easier to do. Now, the only remaining challenge is finding a way to calculate the amount of paths of a given length in a $5x5$ grid (Refer to **Figure 5**).

For a path length of one, this is trivial. There are 25 paths of length one in the grid. For a path of length two, it is still somewhat trivial. It is easy to see that the paths can only connect two nodes, so there are only two types of path. Horizontal paths and vertical ones in this case. We can see that there are $4 \cdot 5$ horizontal paths, and $4 \cdot 5$ vertical ones. This means that in total there are 40 paths with length two, when these two cases are added together. Calculating the amount of paths with larger lengths is possible, but is beyond my ability. I tried to find the total number of paths using Pascal's triangle, since one property of each number in Pascal's triangle is that it represents the amount of paths that get to that number from the top of the triangle. Sadly, this does not help much since the paths only go in one direction for Pascal's triangle, but in our grid they go in any direction. Finding a meaningful generalization might be possible, but since the Pascal's triangle system does not need to account for overlapping paths, a generalization is a far reach.

Analysis and Conclusion

Unfortunately, the model developed leaves more questions than it answers. We do not know how to quickly calculate the c function, and we do not know at all how to calculate how many paths of a given length are within the grid. These problems can be solved, but are beyond the scope of this essay since they are large problems on their own.

Currently, we have derived the following. $E(l, p)$, which takes in path length and the amount of paths with that size in the $5x5$ grid. We have the functions that are nested within the E function, like the a, i, and c functions. These are all part of the E function, but are seperated to help with readability. These functions are hard to

test without solving the previously stated issues, but assuming the problem functions are defined, it would be more than possible to use this form to calculate the expected value of the optimal path.
For more the simple cases (path length one and path length two), we have managed to find closed form solutions solutions for this problem due to the fact that the c function and path lengths within the grid are simple. This allows us to evaluate the algorithm that created the data for the $l = 1$ case. The data states that for this case the average path weight was 9.95. The math states that the expected average path weight should be about 9.92429, and after recognizing the random noise, this a is significant correlation. Since we were unable to find numerical solutions for higher path lengths we can not do this comparison, but it is good to know that for the base case our method has been effective.