Assignment 4

Effect on Throughput and Cycles per Instruction for Non-Branch Prediction and Branch Prediction Programs with Varying Equation Parameters

Justin Clark

Introduction

The goal of this assignment is to determine how clock cycles per instruction and throughput (instructions per clock cycle) are affected by varying parameters of the equations used to determine program performance. There are two parts to this assignment. Part A analyzes the performance of a program without branch prediction behavior while Part B analyzes the performance of a program with branch prediction. The target variables and parameters to be varied in each part are as follows:

TARGET VARIABLES

C_{AVG}	The average number of clock cycles per instruction of a program
H	The throughput of the program (otherwise known as instructions per clock cycle)

PART A VARIABLES

P_b	The probability that a given instruction is a branch instruction
b	The branch penalty in clock cycles
P_t	The probability that a branch is taken

Equations:

$$C_{AVG} = 1 + P_b P_t b$$

$$H = \frac{1}{C_{AVG}}$$

PART B VARIABLES

c	The reduced penalty associated with a branch prediction that is correct
P_c	The probability of a correct branch prediction

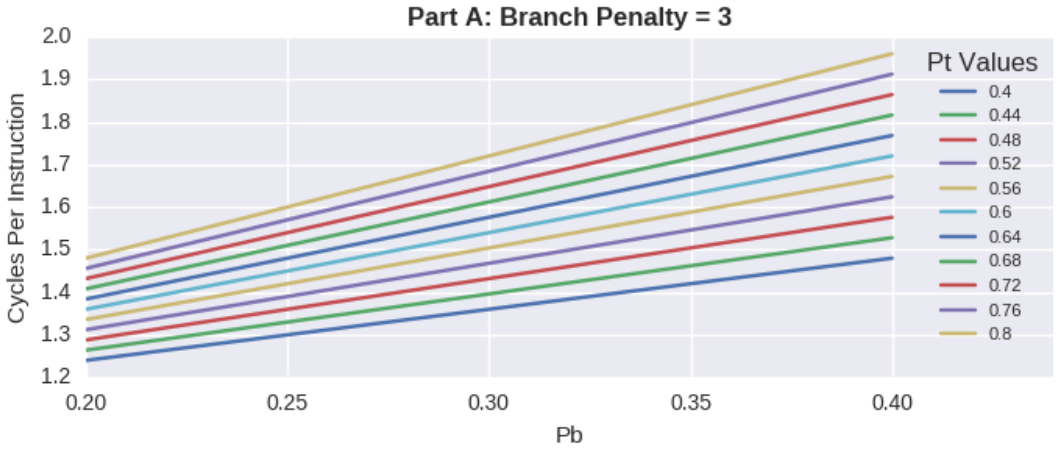
Equations:

$$C_{AVG} = 1 + P_b b - P_b P_c b + P_b P_t P_c c$$

$$H = \frac{1}{C_{AVG}}$$

Part A: Non-branching program

There are two graphs for the throughput of a program and two graphs for the clock cycles per instruction of a program. Each one of the two graphs in each category represents a family of curves for a program with either a branch penalty of 3 clock cycles or a branch penalty of 4 clock cycles.



Pt

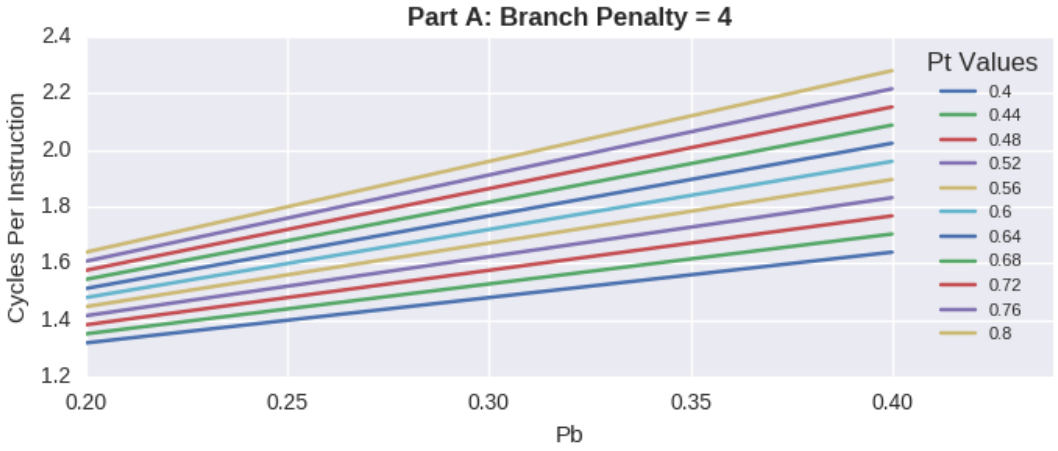
	0.2	0.22	0.24	0.26	0.28	0.3	0.32	0.34	0.36	0.38	0.4
0.4	1.24	1.264	1.288	1.312	1.336	1.36	1.384	1.408	1.432	1.456	1.48
0.44	1.264	1.2904	1.3168	1.3432	1.3696	1.396	1.4224	1.4488	1.4752	1.5016	1.528
0.48	1.288	1.3168	1.3456	1.3744	1.4032	1.432	1.4608	1.4896	1.5184	1.5472	1.576
0.52	1.312	1.3432	1.3744	1.4056	1.4368	1.468	1.4992	1.5304	1.5616	1.5928	1.624
0.56	1.336	1.3696	1.4032	1.4368	1.4704	1.504	1.5376	1.5712	1.6048	1.6384	1.672
0.6	1.36	1.396	1.432	1.468	1.504	1.54	1.576	1.612	1.648	1.684	1.72
0.64	1.384	1.4224	1.4608	1.4992	1.5376	1.576	1.6144	1.6528	1.6912	1.7296	1.768
0.68	1.408	1.4488	1.4896	1.5304	1.5712	1.612	1.6528	1.6936	1.7344	1.7752	1.816
0.72	1.432	1.4752	1.5184	1.5616	1.6048	1.648	1.6912	1.7344	1.7776	1.8208	1.864
0.76	1.456	1.5016	1.5472	1.5928	1.6384	1.684	1.7296	1.7752	1.8208	1.8664	1.912
0.8	1.48	1.528	1.576	1.624	1.672	1.72	1.768	1.816	1.864	1.912	1.96

The major observations in this part are as follows:

1. The curves are upward sloping, i.e. larger values of P_b lead to higher cycles per instruction.
2. As P_b increases, the distance between curves also increases.
3. The curves seem linear.
4. The larger the P_t, the higher the cycles per instruction.

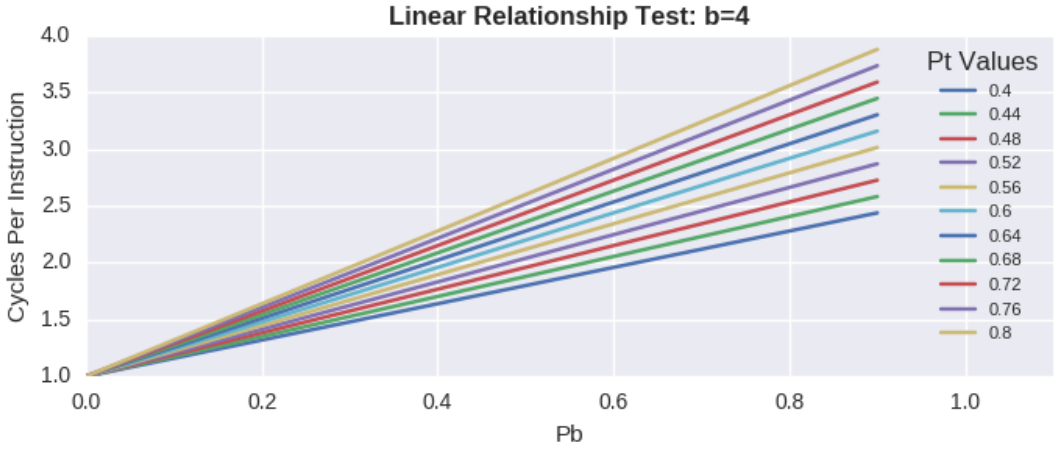
The explanations to these observations are as follows:

1. In the equation, P_b must lead to higher values as it increases since it is in an increasing term. So, this sort of behavior is expected.
2. This is due to the fact that P_b and P_t are multiplied. When P_b is smaller (left side of the graph), their interaction in the equation leads to a small difference between curves. At the right side of the graph, the cycles per instruction changes more drastically between curves.
3. This will be investigated further after the next part.
4. Similar to (1), this should be expected given the equation in the Introduction.

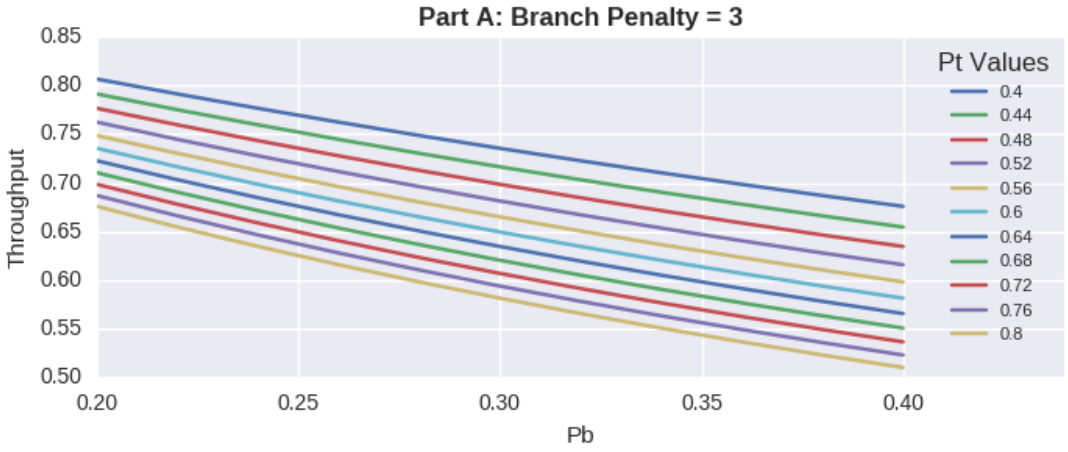


Pt \ Pb	0.2	0.22	0.24	0.26	0.28	0.3	0.32	0.34	0.36	0.38	0.4
0.4	1.32	1.352	1.384	1.416	1.448	1.48	1.512	1.544	1.576	1.608	1.64
0.44	1.352	1.3872	1.4224	1.4576	1.4928	1.528	1.5632	1.5984	1.6336	1.6688	1.704
0.48	1.384	1.4224	1.4608	1.4992	1.5376	1.576	1.6144	1.6528	1.6912	1.7296	1.768
0.52	1.416	1.4576	1.4992	1.5408	1.5824	1.624	1.6656	1.7072	1.7488	1.7904	1.832
0.56	1.448	1.4928	1.5376	1.5824	1.6272	1.672	1.7168	1.7616	1.8064	1.8512	1.896
0.6	1.48	1.528	1.576	1.624	1.672	1.72	1.768	1.816	1.864	1.912	1.96
0.64	1.512	1.5632	1.6144	1.6656	1.7168	1.768	1.8192	1.8704	1.9216	1.9728	2.024
0.68	1.544	1.5984	1.6528	1.7072	1.7616	1.816	1.8704	1.9248	1.9792	2.0336	2.088
0.72	1.576	1.6336	1.6912	1.7488	1.8064	1.864	1.9216	1.9792	2.0368	2.0944	2.152
0.76	1.608	1.6688	1.7296	1.7904	1.8512	1.912	1.9728	2.0336	2.0944	2.1552	2.216
0.8	1.64	1.704	1.768	1.832	1.896	1.96	2.024	2.088	2.152	2.216	2.28

The major observances in this graph are the same as in the previous graph considering the values of P_b and P_t are the same. The difference arises in the value of b. Notice that all the values in the table above are slightly higher than in the table from the previous section. This is due the behavior of b in the equation for C_{AVG}. Since b is multiplied by P_b and P_t, one would expect a higher b to lead to higher cycles per instruction.

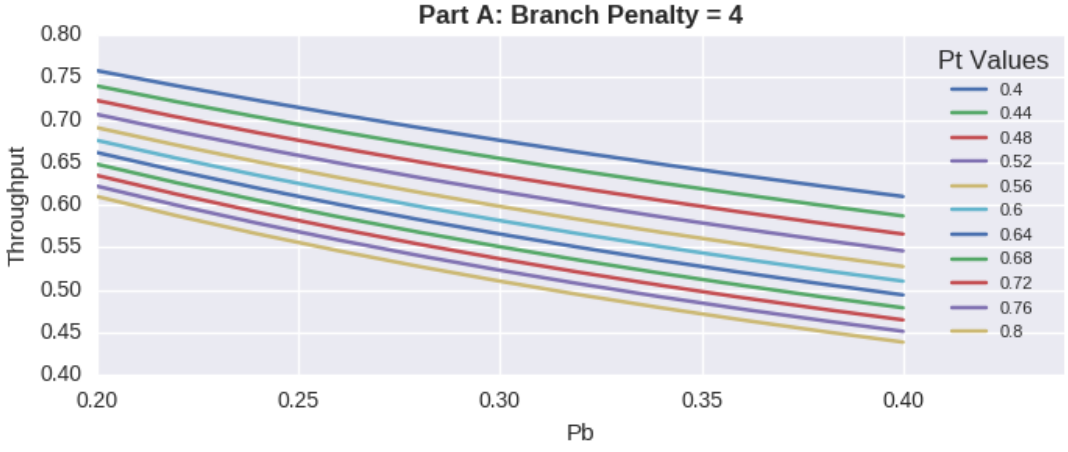


As stated in the previous section, I wanted to investigate the seemingly linear curves. Even varying P_b from $0.0 - 0.9$, we notice the graph shows a linear relationship. After further thought, each curve *should* be linear because each curve is a function of only one variable (P_b) with all other variables being held constant.



Pt

	0.2	0.22	0.24	0.26	0.28	0.3	0.32	0.34	0.36	0.38	0.4
0.4	0.8064516	0.7911392	0.7763975	0.7621951	0.748503	0.7352941	0.7225434	0.7102273	0.698324	0.6868132	0.6756757
0.44	0.7911392	0.7749535	0.7594168	0.7444908	0.7301402	0.7163324	0.7030371	0.6902264	0.6778742	0.6659563	0.6544503
0.48	0.7763975	0.7594168	0.7431629	0.7275902	0.7126568	0.698324	0.6845564	0.6713212	0.658588	0.6463289	0.6345178
0.52	0.7621951	0.7444908	0.7275902	0.71144	0.6959911	0.6811989	0.6670224	0.6534239	0.6403689	0.6278252	0.6157635
0.56	0.748503	0.7301402	0.7126568	0.6959911	0.6800871	0.6648936	0.6503642	0.6364562	0.6231306	0.6103516	0.5980861
0.6	0.7352941	0.7163324	0.698324	0.6811989	0.6648936	0.6493506	0.6345178	0.6203474	0.6067961	0.5938242	0.5813953
0.64	0.7225434	0.7030371	0.6845564	0.6670224	0.6503642	0.6345178	0.6194252	0.6050339	0.5912961	0.5781684	0.5656109
0.68	0.7102273	0.6902264	0.6713212	0.6534239	0.6364562	0.6203474	0.6050339	0.5904582	0.5765683	0.5633168	0.5506608
0.72	0.698324	0.6778742	0.658588	0.6403689	0.6231306	0.6067961	0.5912961	0.5765683	0.5625563	0.5492091	0.5364807
0.76	0.6868132	0.6659563	0.6463289	0.6278252	0.6103516	0.5938242	0.5781684	0.5633168	0.5492091	0.5357908	0.5230126
0.8	0.6756757	0.6544503	0.6345178	0.6157635	0.5980861	0.5813953	0.5656109	0.5506608	0.5364807	0.5230126	0.5102041



Pt

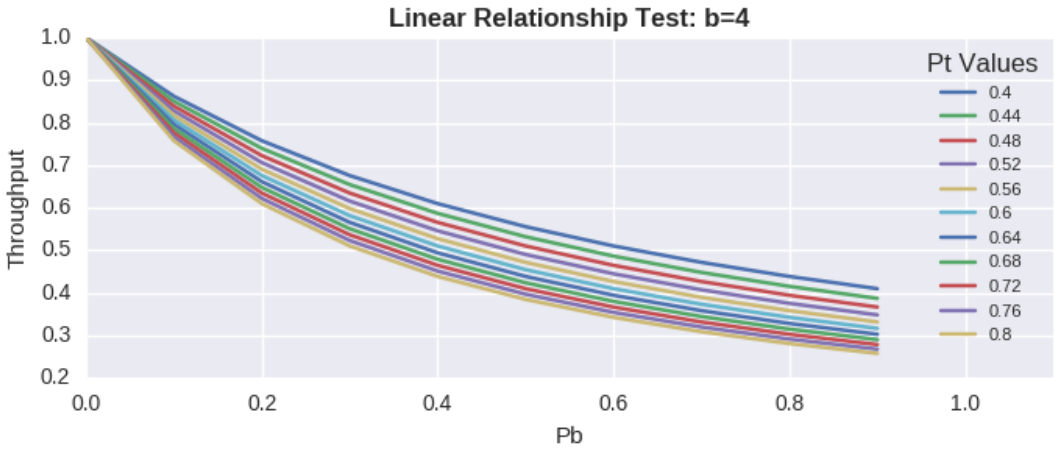
	0.2	0.22	0.24	0.26	0.28	0.3	0.32	0.34	0.36	0.38	0.4
0.4	0.7575758	0.739645	0.7225434	0.7062147	0.6906077	0.6756757	0.6613757	0.6476684	0.6345178	0.6218905	0.6097561
0.44	0.739645	0.7208766	0.7030371	0.6860593	0.6698821	0.6544503	0.6397134	0.6256256	0.612145	0.599233	0.5868545
0.48	0.7225434	0.7030371	0.6845564	0.6670224	0.6503642	0.6345178	0.6194252	0.6050339	0.5912961	0.5781684	0.5656109
0.52	0.7062147	0.6860593	0.6670224	0.6490135	0.6319515	0.6157635	0.6003842	0.5857545	0.5718207	0.5585344	0.5458515
0.56	0.6906077	0.6698821	0.6503642	0.6319515	0.6145526	0.5980861	0.582479	0.5676658	0.5535872	0.5401901	0.5274262
0.6	0.6756757	0.6544503	0.6345178	0.6157635	0.5980861	0.5813953	0.5656109	0.5506608	0.5364807	0.5230126	0.5102041
0.64	0.6613757	0.6397134	0.6194252	0.6003842	0.582479	0.5656109	0.5496922	0.534645	0.5203997	0.5068938	0.4940711
0.68	0.6476684	0.6256256	0.6050339	0.5857545	0.5676658	0.5506608	0.534645	0.5195345	0.5052546	0.4917388	0.4789272
0.72	0.6345178	0.612145	0.5912961	0.5718207	0.5535872	0.5364807	0.5203997	0.5052546	0.4909662	0.4774637	0.464684
0.76	0.6218905	0.599233	0.5781684	0.5585344	0.5401901	0.5230126	0.5068938	0.4917388	0.4774637	0.4639941	0.4512635
0.8	0.6097561	0.5868545	0.5656109	0.5458515	0.5274262	0.5102041	0.4940711	0.4789272	0.464684	0.4512635	0.4385965

The two graphs on the previous page show the throughput as opposed to cycles per instruction. If we look closely, a more curved relationship can be seen in these graphs. Since these graphs show the reciprocal of of cycles per instruction, we would expect to make (almost) opposite observations. Here are some corresponding observations of throughput:

1. The curves are *downward* sloping.
2. The distance between curves still seem to increase slightly with increasing P_b, but also note that the distance between curves increases as P_t increases.
3. The curves are more obviously *curvilinear* for throughput than cycles per instruction.
4. Larger values of P_t lead to *lower* throughput.
5. The larger branch penalty b leads to a lower throughput.

These observations should be expected when taking the reciprocal of cycles per instruction in the previous part. Taking the reciprocal of a perfectly linear function will map it to a curved function. It will also turn an increasing function to a decreasing function.

With this limited view of of the graphs, I thought it would be good to expand our observation window of P_b as in the previous section. My hypothesis was that it may accentuate the relationships between variables.

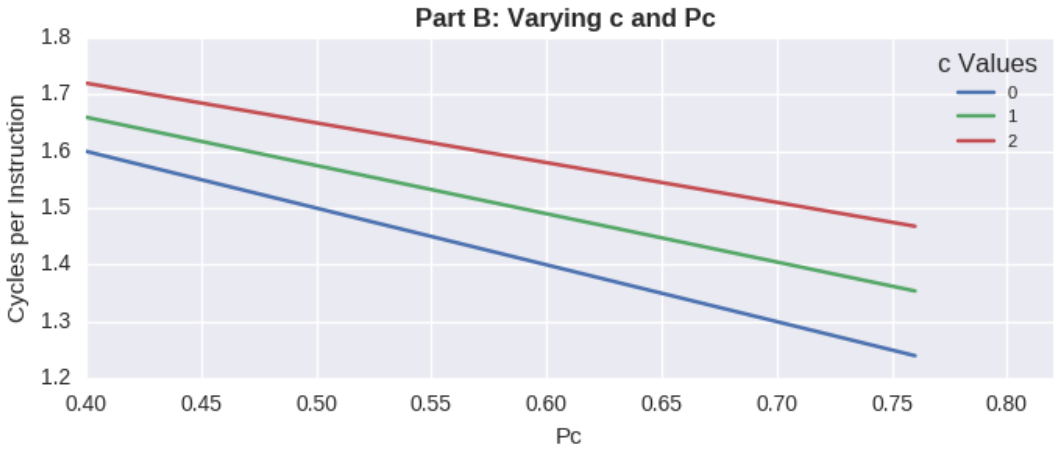


We do, in fact, see the relationship between variables amplified here. There is a clearly curvilinear relationship between P_b and throughput. One of the interesting observances is that the difference between curves remains relatively constant after about $P_b = 0.3$. This could be due to the fact that after this point, the relationship more closely resembles a linear relationship and the differences start to increase at a decreasing rate. Let's dig more into why this is the case. If we rearrange the equation we have:

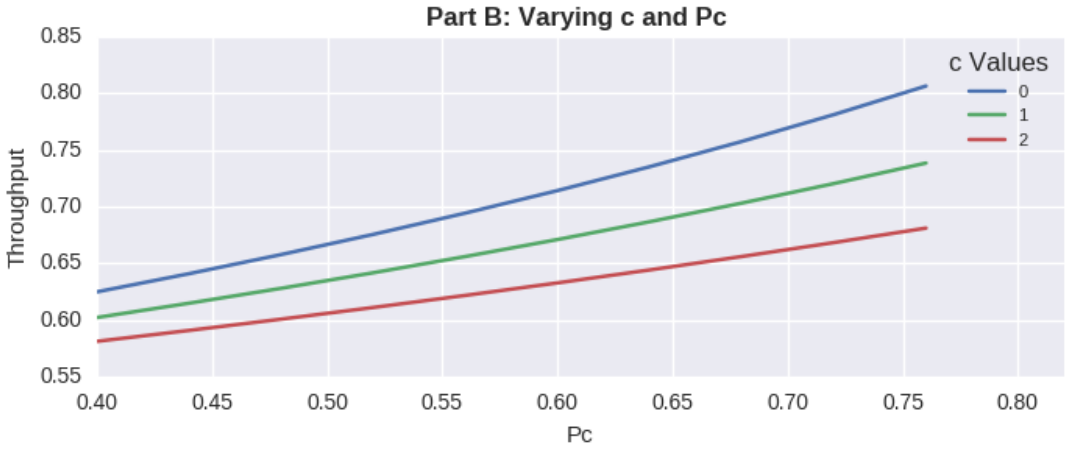
$P_b = \frac{H-1}{bP_t}$ and the partial derivative, $\frac{\partial P_b}{\partial P_t} = \frac{1-H}{bP_t}$ shows that as P_t increases, the rate of change of P_b decreases. Also, the partial derivative, $\frac{\partial P_b}{\partial H} = \frac{1}{bP_t}$ shows that there is a constant rate of increase in P_b as H decreases (with b and P_t held constant). This is the reverse of how one might look at these variables, but both derivatives together support the observation.

Part B: Branching Program

There is only one graph for cycles per instruction and one graph for throughput in this part since b, P_b, and P_t are all held constant. Each graph represents a family of curves where each curve represents a constant value of c with varying P_c. Note that $P_b = 0.25$, $P_t = 0.60$, and $b = 4$.



	0.4	0.44	0.48	0.52	0.56	0.6	0.64	0.68	0.72	0.76
0	1.6	1.56	1.52	1.48	1.44	1.4	1.36	1.32	1.28	1.24
1	1.66	1.626	1.592	1.558	1.524	1.49	1.456	1.422	1.388	1.354
2	1.72	1.692	1.664	1.636	1.608	1.58	1.552	1.524	1.496	1.468



	0.4	0.44	0.48	0.52	0.56	0.6	0.64	0.68	0.72	0.76
0	0.625	0.6410256	0.6578947	0.6756757	0.6944444	0.7142857	0.7352941	0.7575758	0.78125	0.8064516
1	0.6024096	0.6150062	0.6281407	0.6418485	0.656168	0.6711409	0.6868132	0.7032349	0.7204611	0.7385524
2	0.5813953	0.5910165	0.6009615	0.6112469	0.6218905	0.6329114	0.6443299	0.656168	0.6684492	0.6811989

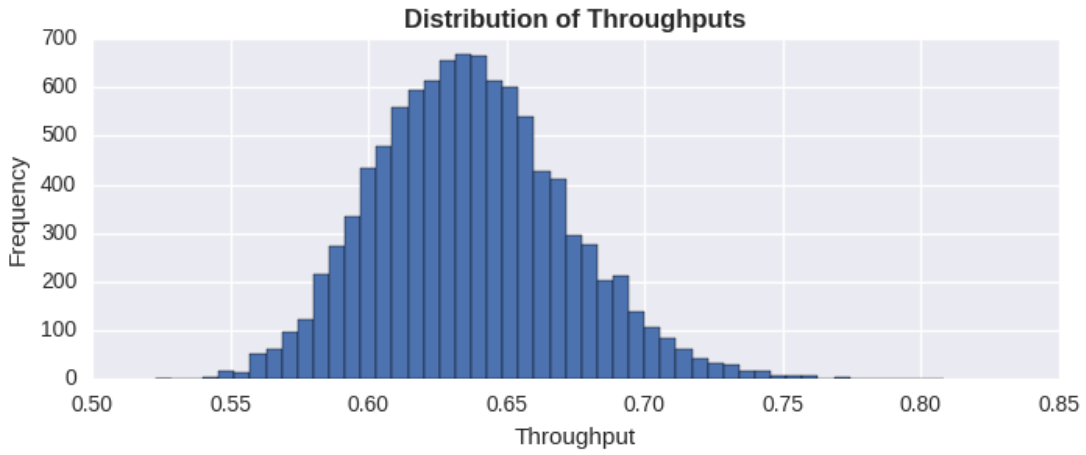
Observations about the graphs on the previous page are as follows:

1. Cycles per instruction decreases as P_c increases (and vice versa for throughput).
2. With P_b, P_t, and b all held constant, cycles per instruction decreases perfectly linearly as P_c increases and throughput increases curvilinearly (as one would expect given our investigation in part A).
3. The difference between the curves increases as P_c increases in both the cycles per instruction and throughput graphs.

The equation of average cycles per instruction for branching programs is a little more complicated than for non-branching equations. The first thing to note is that there is actually a decreasing term in the equation which accounts for a correct branch prediction. As the probability of a correct branch prediction P_c increases, one would expect the average clock cycles per instruction to go down.

One thing I would like to investigate further is what a *distribution* of throughputs might look like if P_c were a random variable. Of course, it is useful to know the *average* throughput, but it is also useful to know the distribution of throughputs so computer experts might determine confidence intervals for program throughput overtime with unknown and varying P_c.

The best way to represent the distribution is a histogram showing how throughput varies with many different values of P_c and a constant value of c. My method is as follows: use Monte Carlo simulation to sample 10,000 random values of P_c from Gaussian distribution $N(\mu = 0.5, \sigma = 0.1)$ and hold $c = 1$. The frequency distribution of throughput is as follows:



We could also run this simulation for the other variables in H to determine best and worst case scenarios for each of the variables and determine an overall confidence interval for our program's performance through time.

Conclusion

There are many ways to approach the analysis of programs that exhibit branching behavior. Overall, I found it interesting to see how each individual variable affected the cycles per instruction and throughput of a program. Additionally, it was interesting to look at the distribution of throughputs when one of the variables was treated as a random variable. This gives us a little more insight into the possible throughputs of a program as opposed to the average. Both of these approaches are valuable.

Code used to generate graphs and tables was written in Python and will be submitted with this document.