The State University of New York at Binghamton

Department of Computer Science

CS 520 – Spring 2019

Project #1: Branch Prediction

By

Antra Aruj

Honor Pledge: I have neither given nor received unauthorized aid on this test or assignment.
Student's electronic signature: Antra Aruj (sign by typing your name)

1. BIMODAL BRANCH PREDICTOR

a) Validation

I have validated the code against all the validation file given.

b)

- **Graphs:**

Below are the three graphs for gcc perl and jpeg. Each data point in these graphs is generated by taking a value of "m" and a given trace file. This is done using the script "bimodal_runs.sh"

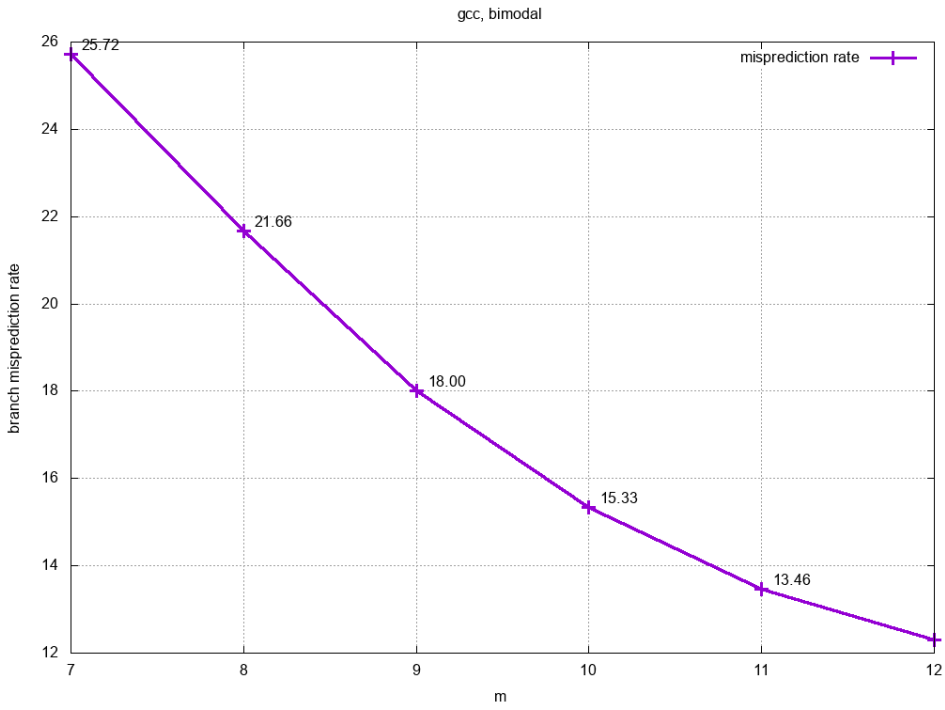

Figure 1: GCC Benchmark using Bimodal Predictor

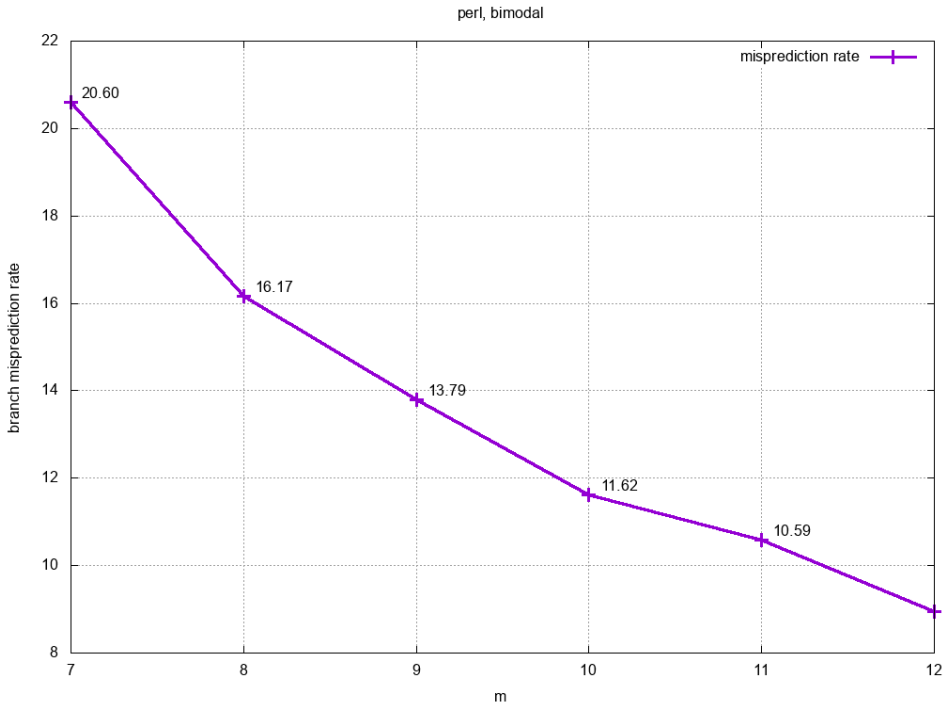

Figure 2: PERL Benchmark using Bimodal Predictor

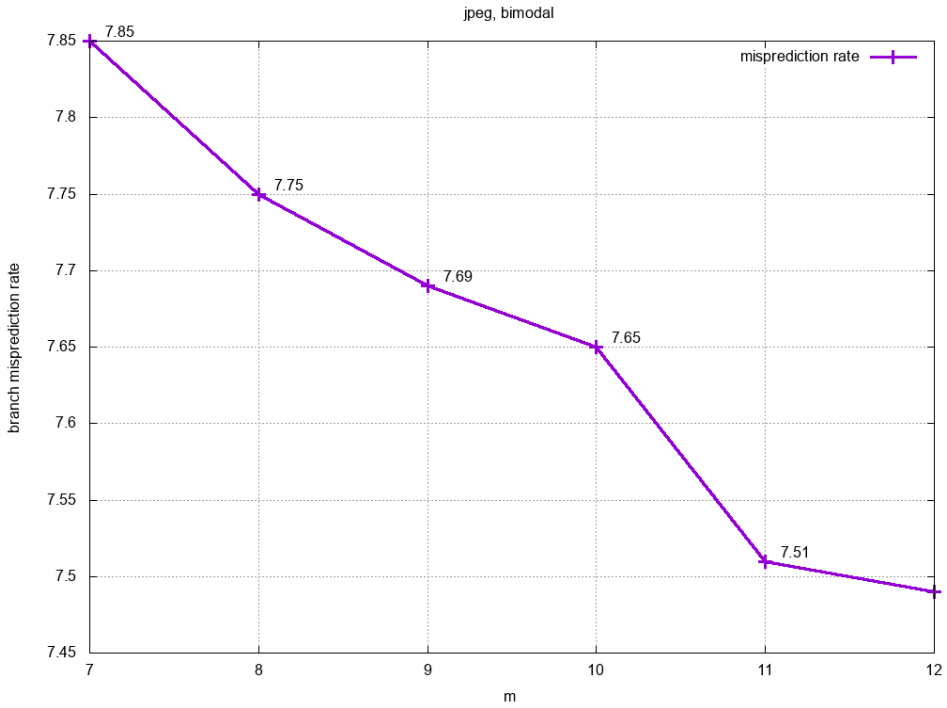

Figure 3: JPEG Benchmark using Bimodal Predictor

- **Analysis**

Similarities:

GCC:
Misprediction rate decreases as the size of prediction table (2^m) increases, therefore we can say misprediction rate is inversely proportional to m.

PERL
Misprediction rate decreases as the size of prediction table (2^m) increases, therefore we can say misprediction rate is inversely proportional to m.

JPEG:
Misprediction rate decreases as the size of prediction table (2^m) increases, therefore we can say misprediction rate is inversely proportional to m.

Differences:

GCC:
Misprediction rate for gcc is much higher than perl and jpeg for the same 'm'. The change in misprediction rate as 'm' increases has larger gaps. Therefore, we can conclude from the traces available to us that the GCC has worse misprediction rate.

PERL:
Misprediction rate for perl lies in between gcc and jpeg for the same 'm'. The change in misprediction rate as 'm' increases has smaller gaps compared to gcc and larger compare to jpeg. Therefore, we can conclude from the traces available to us that the perl has average misprediction rate.

JPEG:
Jpeg has the lowest misprediction rate. The difference in misprediction rate as 'm' increases is very small. Therefore, we can conclude from the traces available to us that the jpeg has best misprediction rate.

Also, the reduction is the prediction rate with increase in m is lowest for jpeg trace.

c) Design:

The design of branch predictor involves selection of parameters that uses least amount of memory and gives best numbers for mispredictions. I have conducted a parameter search and analyzed memory used for the parameter. I have tried to optimize the memory usage for gain in mispredictions.

We have been given a limit of 16KB of storage space of prediction table have been asked to identify the right set of parameters that will give lowest mispredictions rate.

For a 3-bit prediction counter and given m, we can calculate the memory usage of the prediction table.

Memory usage = 2^m * 3
for example, if we have,

m = 15
Memory usage = 2^15 * 3 = 12 KB

This would be feasible. Therefore, we know that our maximum value of m should be 15.
We ran the bimodal branch predictor in the range of "7 ≤ m ≤ 12" and plotted the prediction rate and memory usage as a function of m. These graphs are presented below.

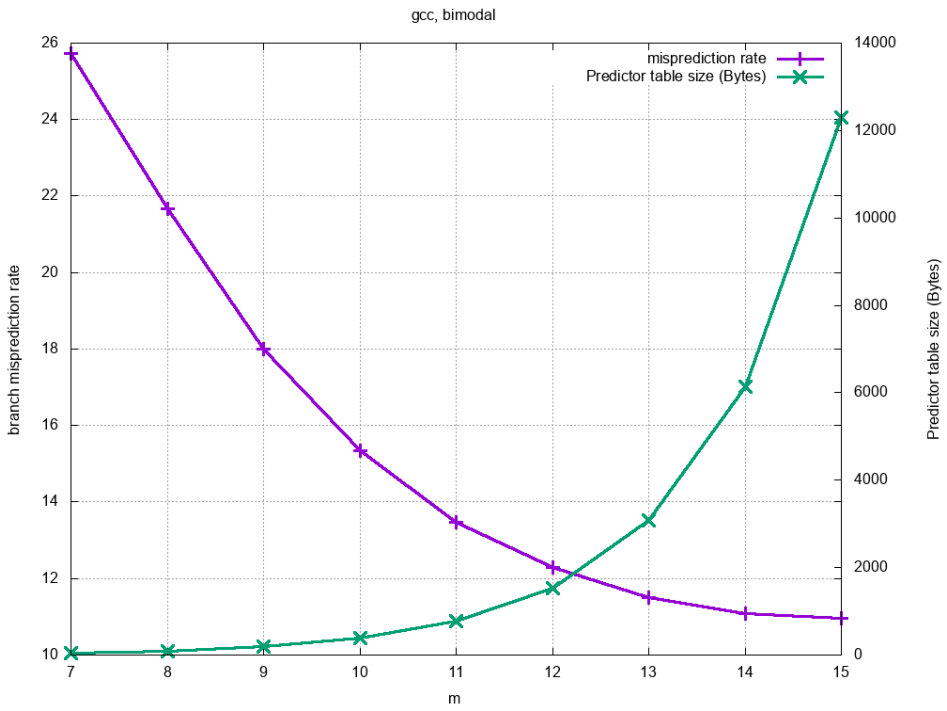

Figure 4: Predictor table memory usage and misprediction rate as a function of m for GCC trace

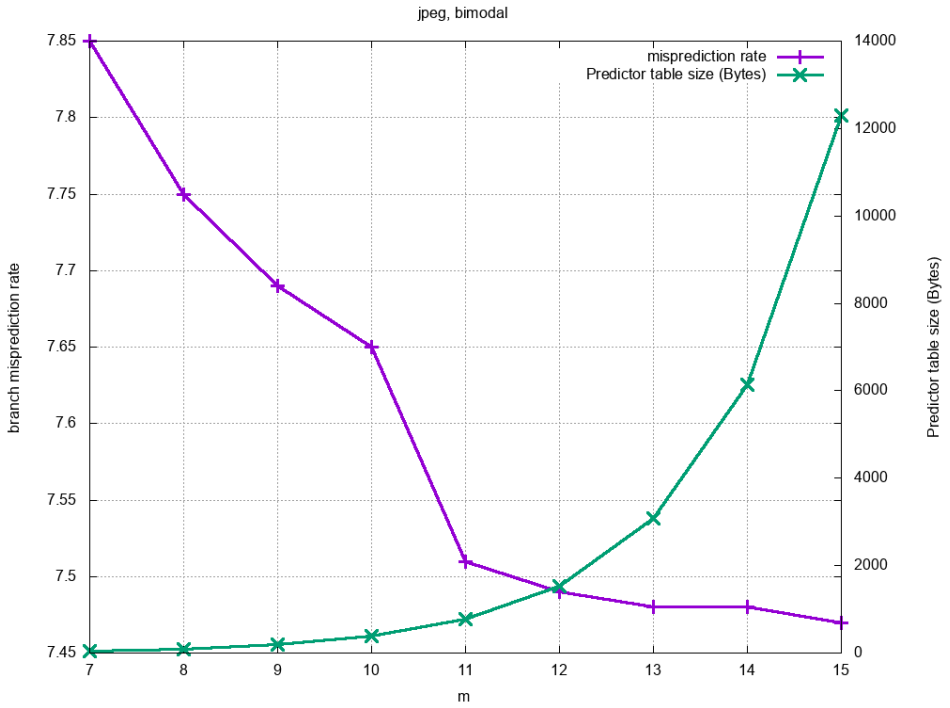

Figure 5:Predictor table memory usage and misprediction rate as a function of m for JPEG trace

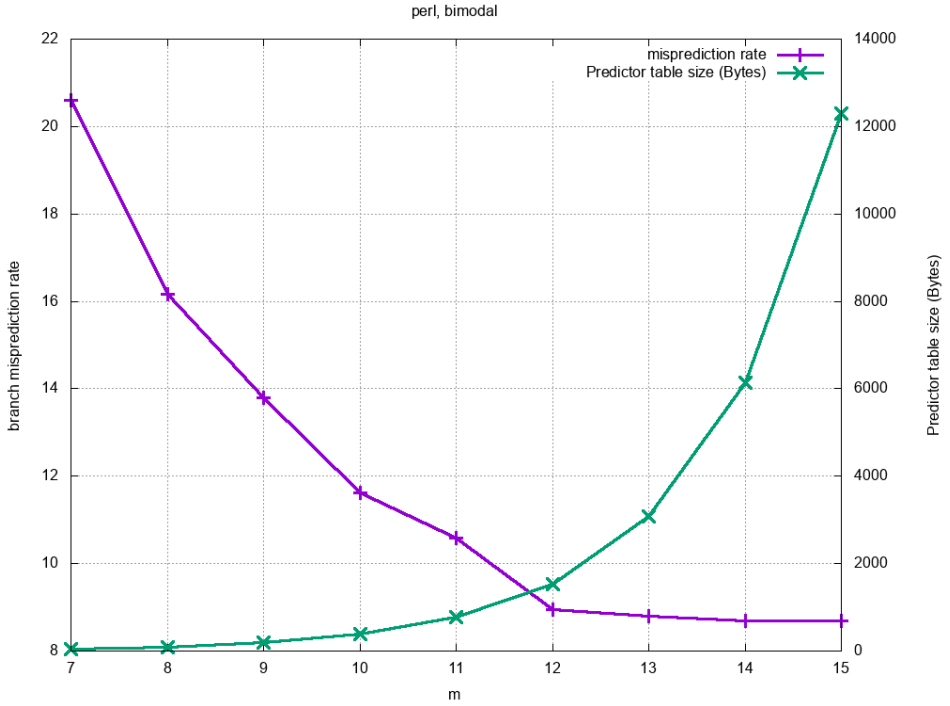

Figure 6:Predictor table memory usage and misprediction rate as a function of m for PERL trace

From above figure we can see that the memory increases as the size of the table increases and the misprediction decrease. However, after certain point the decrease in prediction rate much slower than increase in memory usage as m increases. The optimum value of m is 12 which gives a reasonable accuracy in reasonable memory usage.

2. GSHARE BRANCH PREDICTOR

a) Validation

I have validated the code against all the validation file given.

b)

- **Graphs:**

Below are the three graphs for gcc perl and jpeg. Each data point in these graphs is generated by taking a value of "m" and "n" and a given trace file. This is done using the script "gshare_runs.sh".
Range of m and n:
$7 \le m \le 12$
$2 \le n \le m$, n is even

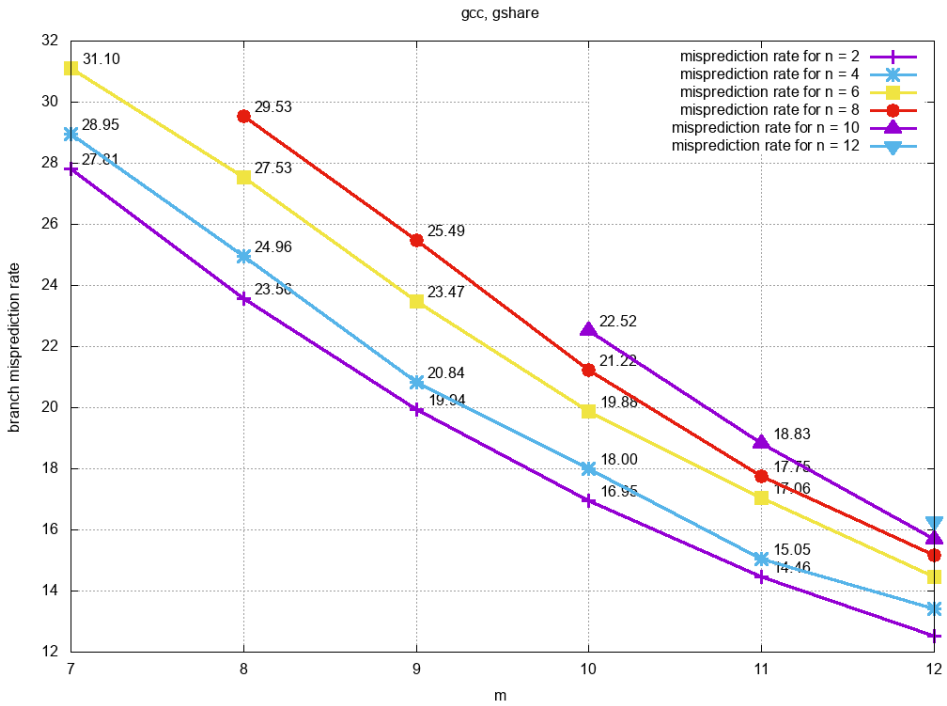

Figure 7:GCC Benchmark using Gshare Predictor

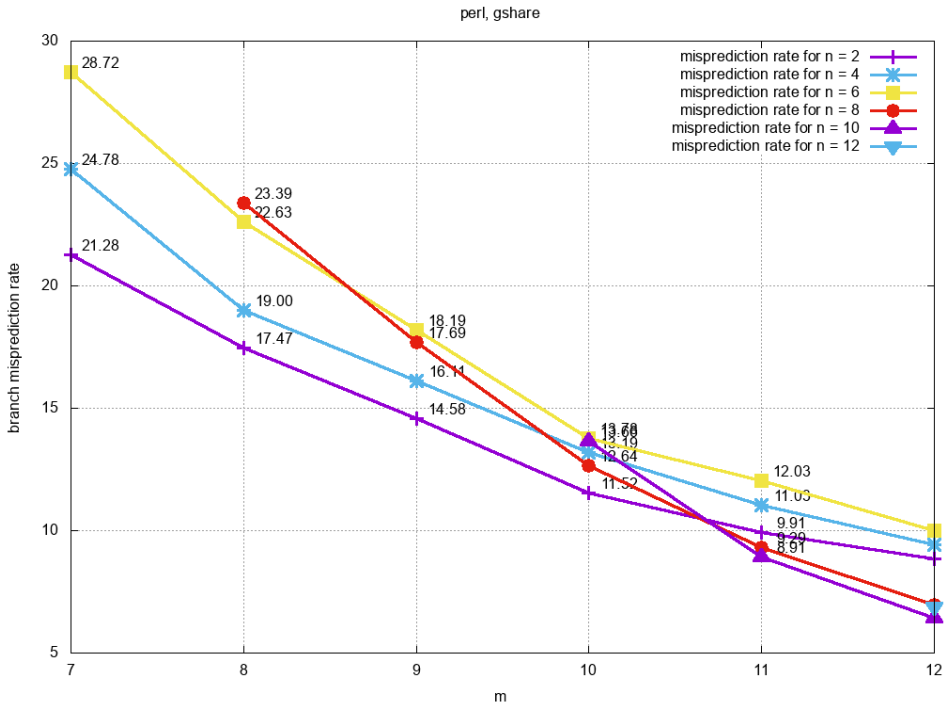

Figure 8:PERL Benchmark using Gshare Predictor

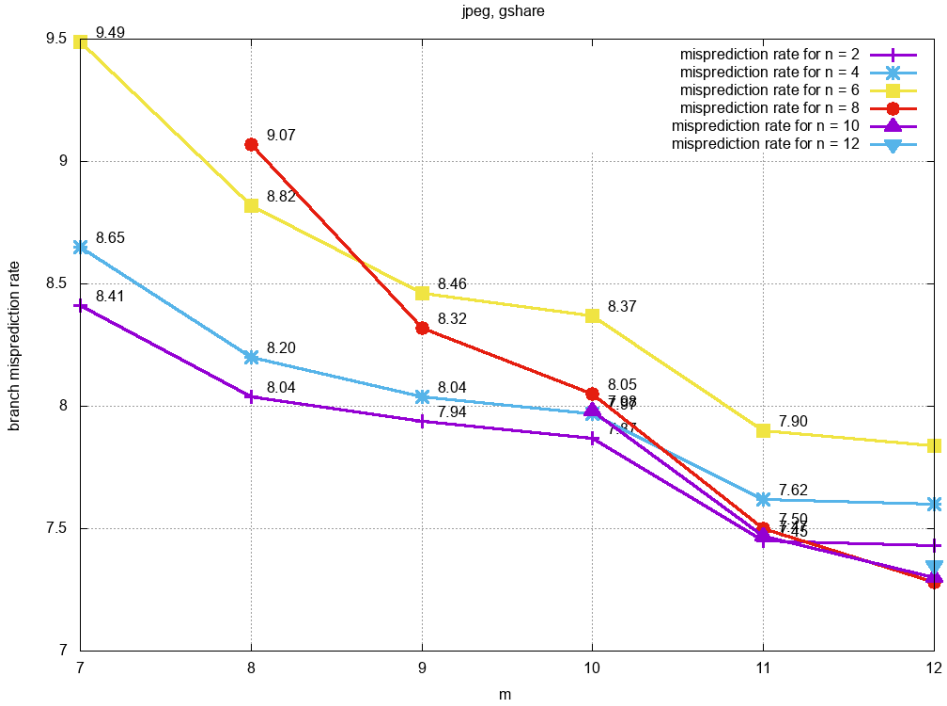

Figure 9:JPEG Benchmark using Gshare Predictor

- **Analysis:**

Similarities:

GCC

Misprediction rate decreases as m increases but it increases as n increases, therefore we can say misprediction rate is inversely proportional to m and directly proportional to n. It should also be noted that all the curves are converging as m is becoming larger.

PERL

Misprediction rate decreases as m but dependence on n is not clear because at lower m, and lower n, prediction is lower but at higher m and higher n, the prediction is better, therefore we can say misprediction rate is inversely proportional to m.

JPEG

Misprediction rate decreases as m and n increases together, therefore we can say misprediction rate is inversely proportional to m and n.

Differences:

GCC

For the same 'm' and 'n', misprediction rate for gcc is much higher than perl and jpeg. The change in misprediction rate as 'm' and 'n' increases have big differences. Therefore, we can conclude form the traces available to us that the GCC has worst misprediction rate.

For constant 'm' and increasing 'n' the misprediction rate increases.

PERL:

For the same 'm' and 'n', misprediction rate for perl lies in between gcc and jpeg. The change in misprediction rate as 'm' and 'n' increases have smaller gaps compared to gcc and larger compare to jpeg. Therefore, we can conclude form the traces available to us that the perl has average misprediction rate.

For constant 'm' and increasing 'n' the misprediction rate increases for few 'n's and then decreases and increases. From the graph obtained for perl trace we cannot be sure about the trends with the given combination of constant 'm' and increasing 'n'

JPEG:

Jpeg has the lowest misprediction rate. The difference in misprediction rate as 'm' and 'n' increases are very small. Therefore, we can conclude form the traces available to us that the jpeg has best misprediction rate.

For constant ‘m’ and increasing ‘n’ the misprediction rate increases for few ‘n’s and then decreases and increases. From the graph obtained for jpeg trace we cannot be sure about the trends with the given combination of constant ‘m’ and increasing ‘n’

c) Design:

For a 3-bit prediction counter and given m and n, we can calculate the memory usage of the prediction table.

Memory usage = 2^m * 3 + n
for example, if we have,

m = 15
n = 14
Memory usage = 2^15 * 3 + 14 ≈ 12 KB

This would be feasible. Therefore, we know that our maximum value of m should be 15.
We ran the gshare branch predictor in the range of “7 ≤ m ≤ 14” and “ 2 ≤ n ≤ m”
and plotted the prediction rate and memory usage as a function of m. These graphs are presented below.

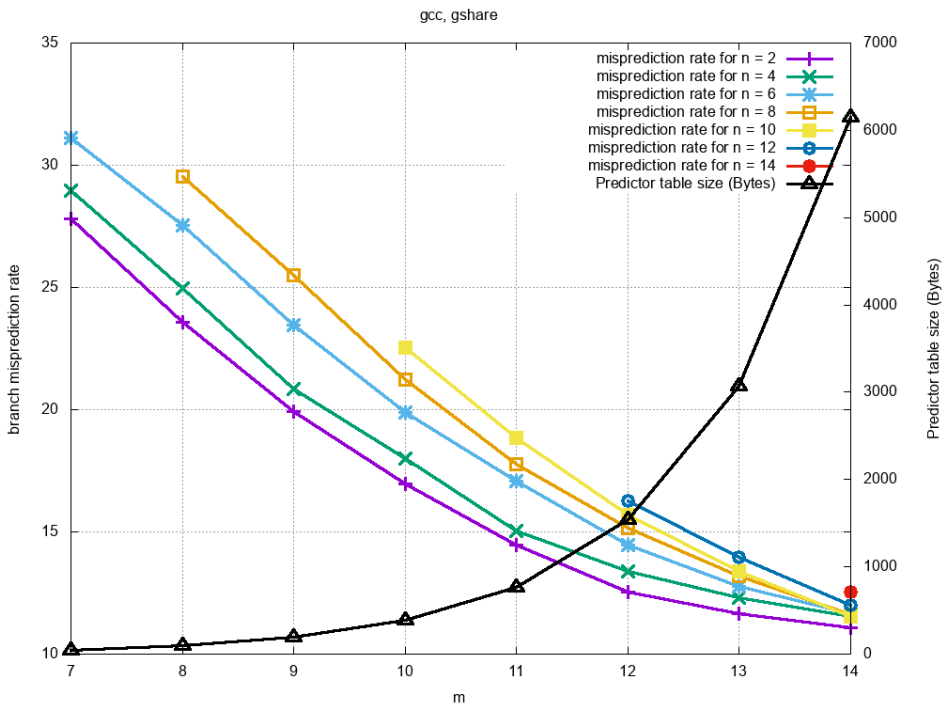

Figure 10:Predictor table memory usage and misprediction rate as a function of m and n for GCC trace

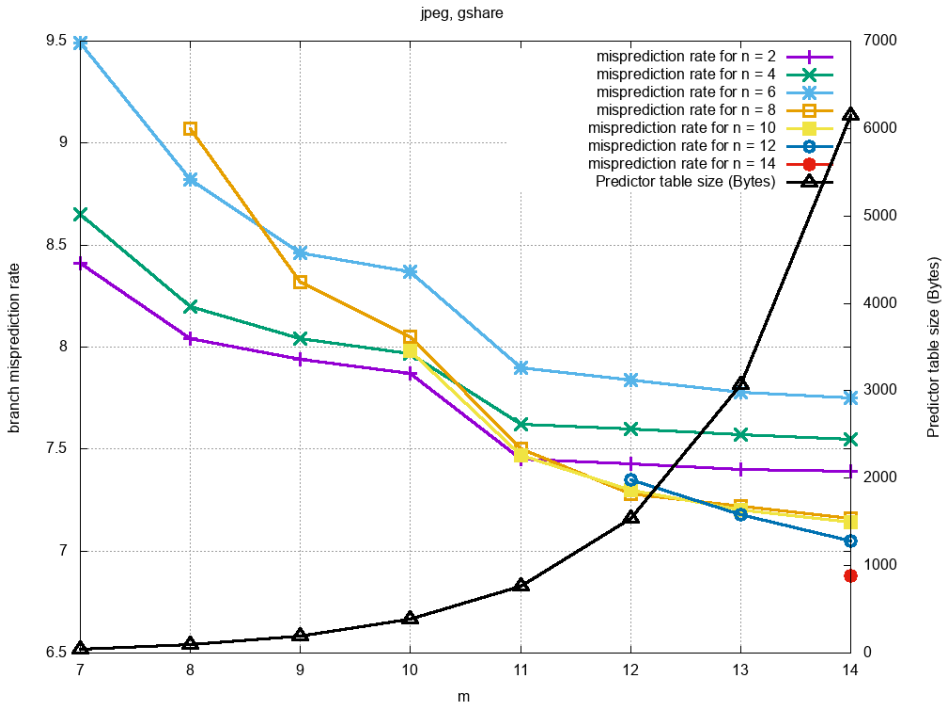

Figure 11:Predictor table memory usage and misprediction rate as a function of m and n for JPEG trace

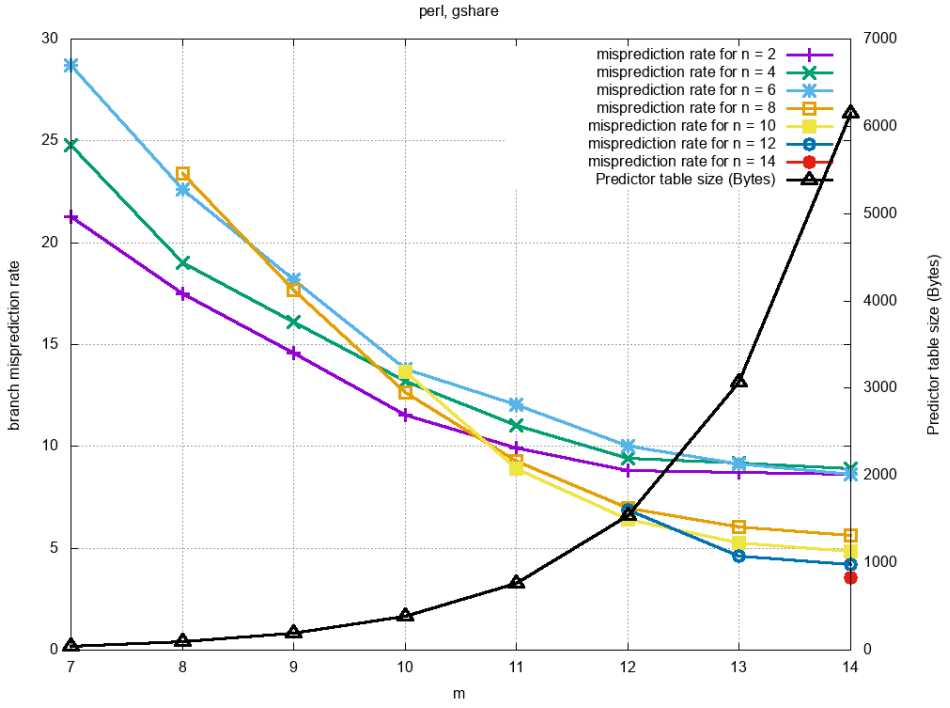

Figure 12:Predictor table memory usage and misprediction rate as a function of m and n for PERL trace

From above figures we can notice that for gcc trace the optimum parameter is m = 11 and n= 2, however for other two traces the m = 12 and n = 10 gives the better performance. Therefore, for gshare design depends on the type of the trace being used. However, if we had to pick a single design, it will be m = 12 and n=10.

3. HYBRID BRANCH PREDICTOR

a) Validation

I have validated the code against all the validation file given.