NC State University

Department of Electrical and Computer Engineering

ECE 563: Fall 2017

Project #2: Branch Prediction

by

Kevin Volkel

NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

Student's electronic signature: __________________Kevin Volkel__________
(sign by typing your name)

Course number: ________563_________
(463 or 563 ?)

Abstract:

A branch predictor simulator was constructed in order to investigate various branch prediction methods and their impacts on three different traces. The simulator can be configured to behave as a bimodal predictor, a ghsare predictor, and a hybrid predictor that combines bimodal and gshare prediction. For each of these three prediction configurations, the simulator can be configured to have a branch target buffer if it is desired. This report studies the behavior of two of the predictors in detail, specifically the bimodal predictor and the gshare predictor. For each of these two predictors, the effect of various configurations (such as the number of predictor table entries) for each trace is studied. After studying the design space of these two branch prediction methods, an “optimal” configuration for each trace and predictor is determined.

Bimodal Predictor Results:

This section presents results obtained from exploring the design space of the bimodal predictor for each of the trace files. For these simulations the branch target buffer was not used. Figures 1-3 below present the misprediction rate vs m ($\log_2$(number of prediction table entries)) for each trace file. For each trace file the value of m is varied from 7 to 12 and the misprediction rate is gathered for each value. Figures 4-6 below show the results of varying the value of m from 1 to 16. The max value of m in these figures is 16 because that is the highest value of m allowed by the 16 kB constraint for finding the optimal performance. The purpose of these graphs is to observe the misprediction rate over a wide range of m in order to determine the point at which there is insignificant performance improvement with increases in m. Table 1 below shows the optimal bimodal configurations for each trace, along with the corresponding table size.

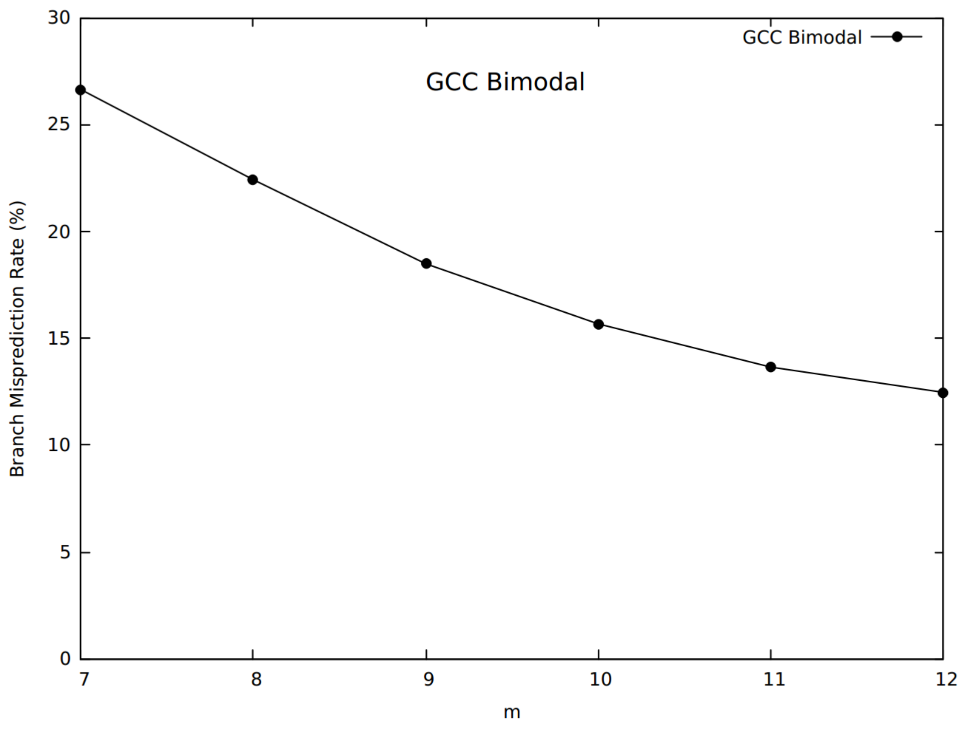

Figure 1: Branch Misprediction Rate vs m for the gcc trace and bimodal predictor.

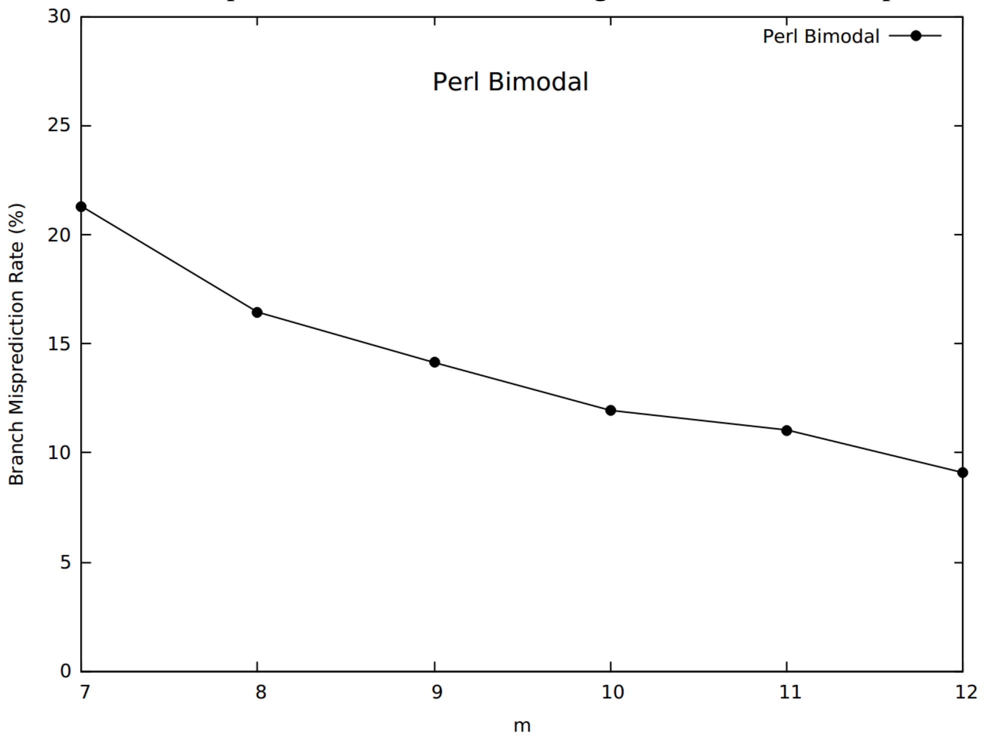

Figure 2: Branch Misprediction Rate vs m for the perl trace and bimodal predictor.

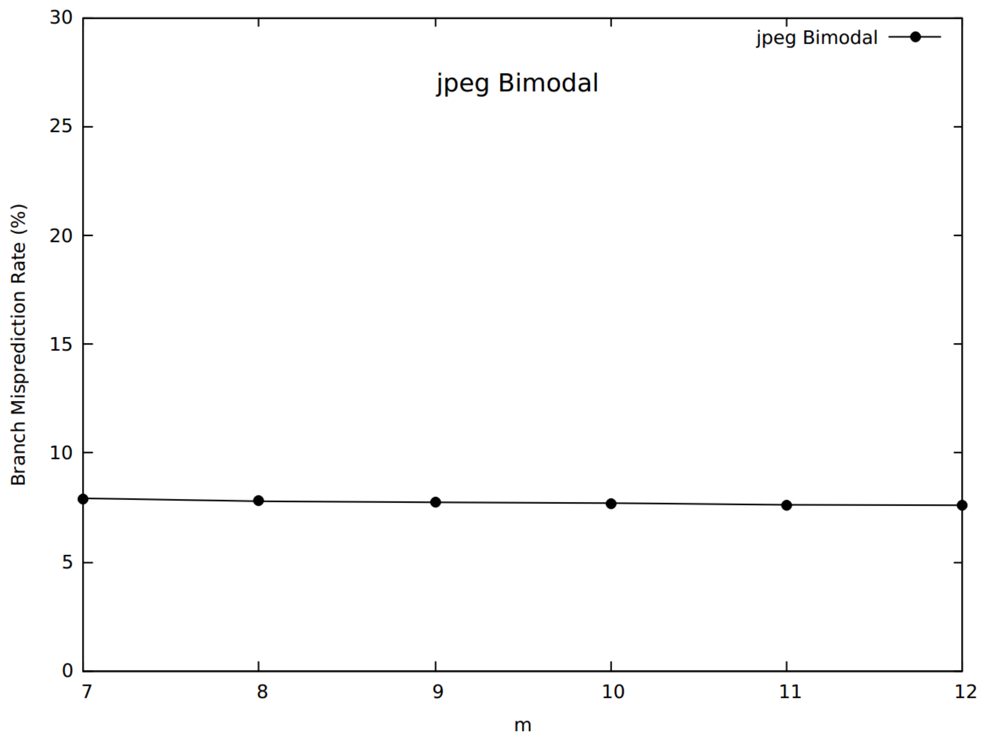

Figure 3: Branch Misprediction Rate vs m for the jpeg trace and bimodal predictor.

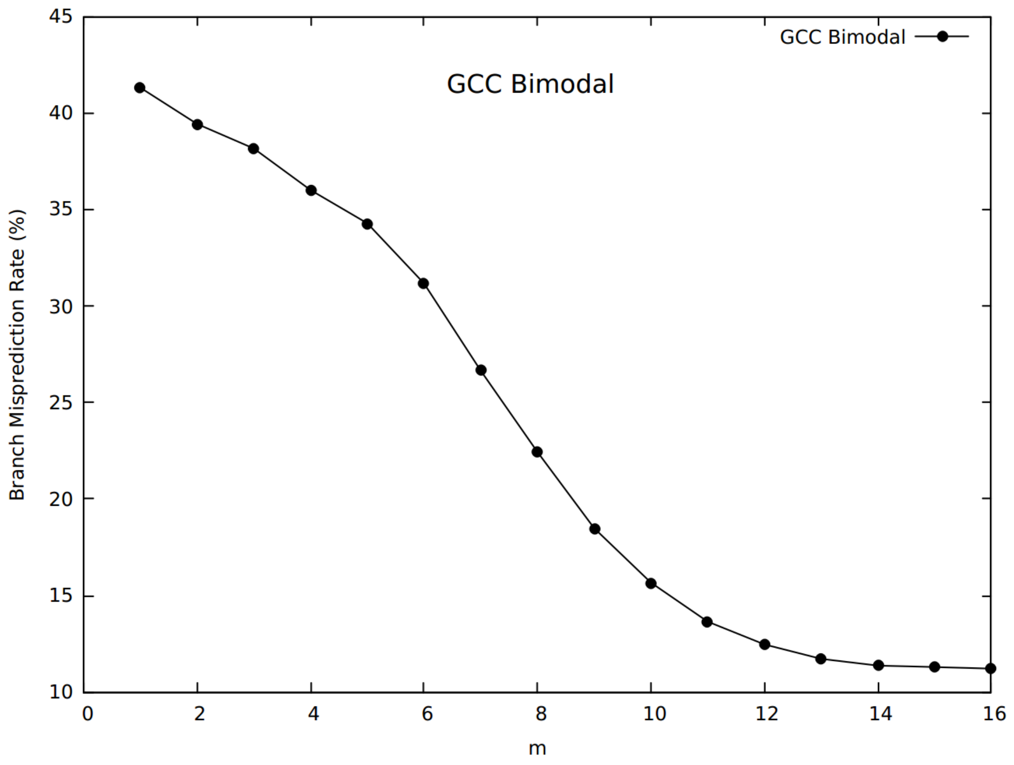

Figure 4: Branch Misprediction Rate vs m for the gcc trace and bimodal predictor.

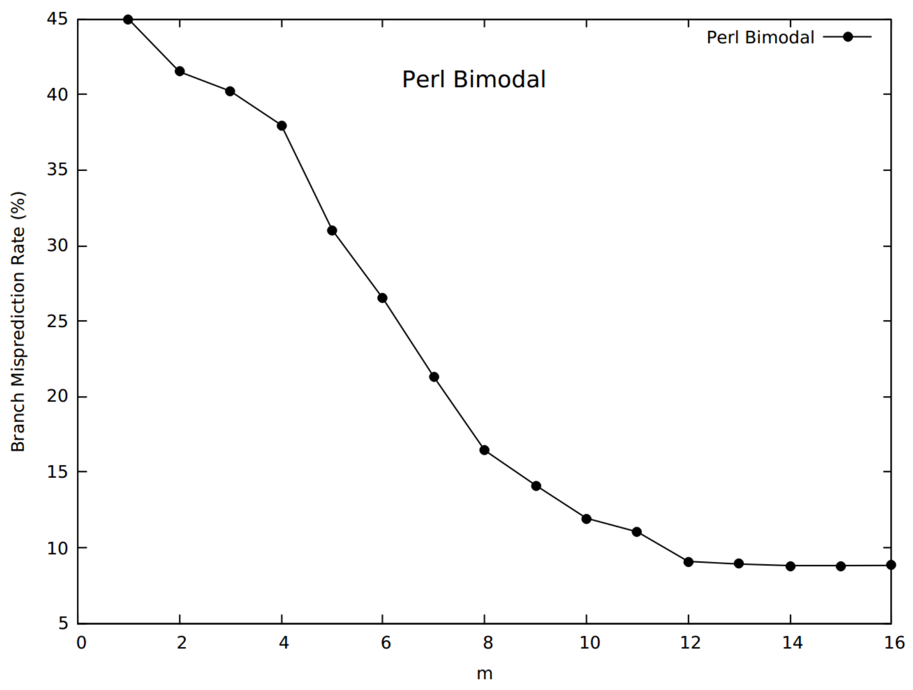

Figure 5: Branch Misprediction Rate vs m for the perl trace and bimodal predictor.

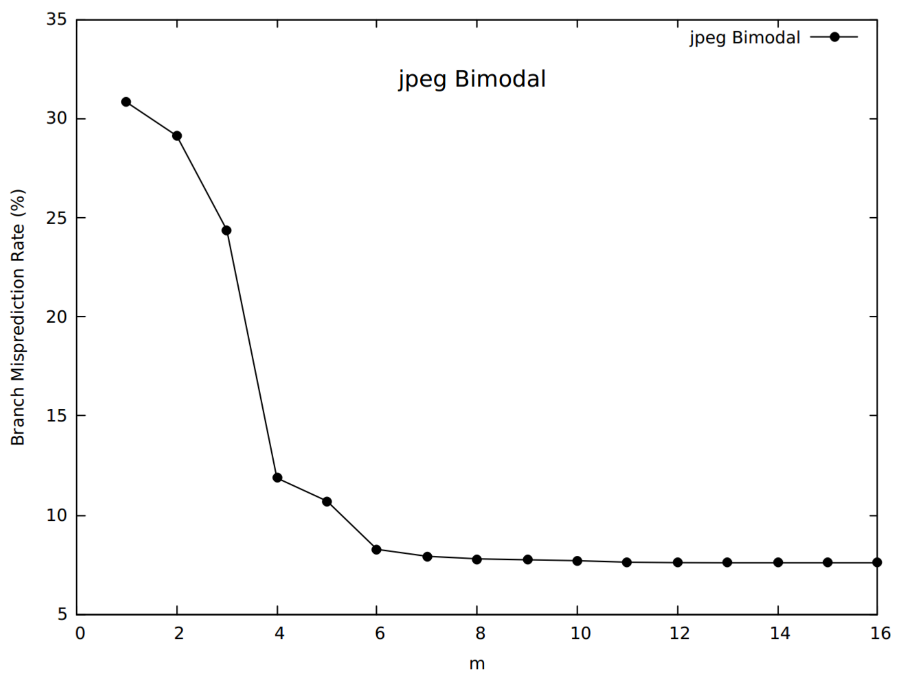

Figure 6: Branch Misprediction Rate vs m for the jpeg trace and bimodal predictor

Table 1: Optimal value of m for each trace with corresponding misprediction values and sizes.

Trace	m	Prediction Table Size	Misprediction Rate (%)
gcc	14	4 kB	11.37
perl	12	1 kB	9.09
jpeg	7	32 B	7.92

From the first 3 figures it can be seen that the behavior of the misprediction rate vs m is different for each of the traces. For the gcc and perl trace, the misprediction rate decreases rapidly until m values are larger than 11, but for the jpeg trace the misprediction rate remains relatively constant as the value of m is increased. The jpeg misprediction rate actually is not constant, it just decreases at a much slower rate compared to the other traces. The reason why the jpeg misprediction rate does not decrease a notable amount from m=7 to m=12 is because this trace does not have a sequence of branches that results in a lot of interference in the prediction table when m=7, and if there is little interference when m=7, then increasing m to 12 will not result in a large performance increase since increasing m only helps with decreasing the amount of branches that interfere with each other. Although increasing m for the jpeg trace does not lead to large performance increase, increasing m to larger values for the gcc and perl traces results in a large decrease in misprediction rate. This large decrease in misprediction rate is most likely due to the traces having many unique branches that index to the same spot in the prediction table when m=7, and as the value of m increases, the amount of branches that index to the same spot in the table decreases.

The last 3 figures (Figures 4-6) show plots of the misprediction rate vs the value of m for m in the rang of 1 to 16. The vale of m=16 is the highest allowable value if the prediction table size is budgeted to 16 kB. These plots are used to determine the point at which increasing m does not provide any more significant increases in prediction accuracy. For the gcc trace this point occurs at m=14, for the perl trace this point occurs at m=12, and for the jpeg trace this point occurs at m=7. These values of m with their corresponding prediction table size and misprediction rates are shown in Table 1. As expected from the previous discussion, the optimal value for m for the jpeg trace is lower than the optimal number for the gcc and perl traces. One last interesting observation is the differences in the prediction rates for the optimal values of m for each trace. The gcc trace has the highest “minimal” misprediction rate with 11.37 % while the jpeg trace has the lowest “minimal” misprediction rte of 7.92 %. This is most likely caused by the nature of the branches that occur in the two traces. For example, the jpeg has branches that are more predictable from a simple 2 bit counter e.g. a pattern of branches such as TTTTTT. While the gcc trace may have more sophisticated branching patterns that are not easily predicted from a simple 2 bit counter e.g. a pattern such as NTNTNT.

Gshare Results:

This section presents results obtained from simulations with the gshare predictor. The values of m were constrained to $7 \leq m \leq 12$ and the values of n were constrained to $2 \leq n \leq m$. Figures 7-9 below present the simulation results for each trace using these constraints. Figures 10-12 present graphs that are used to determine the best gshare configuration for each trace, and Table 2 shows the “optimal” configuration along with relevant information such as predictor size and misprediction rate for the “optimal” setting.

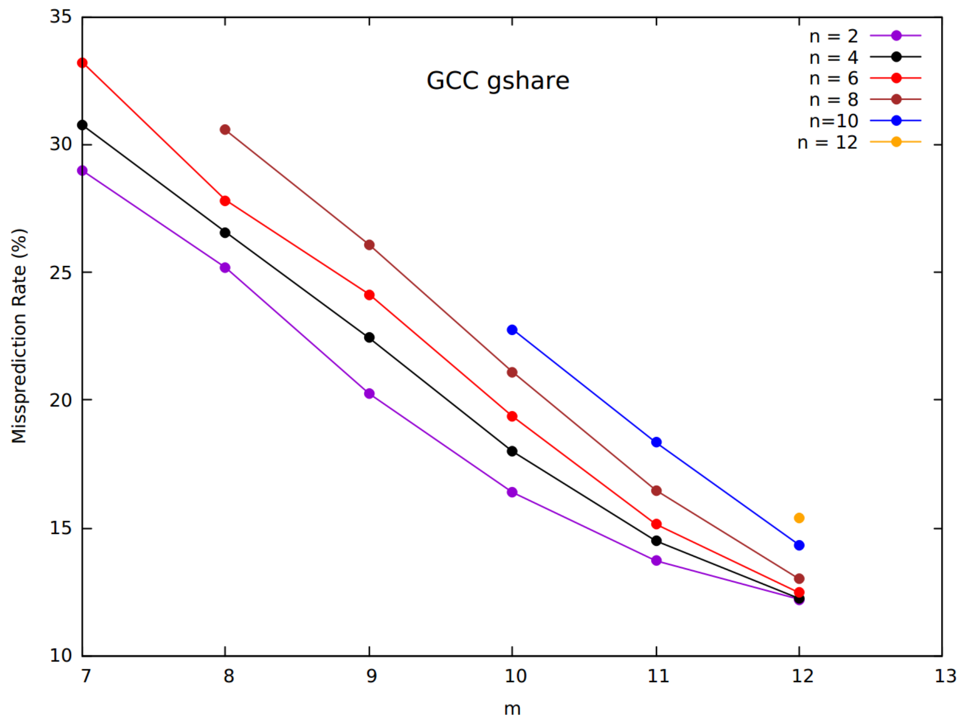

Figure 7: Results of the simulation for the gcc trace using the gshare predictor.

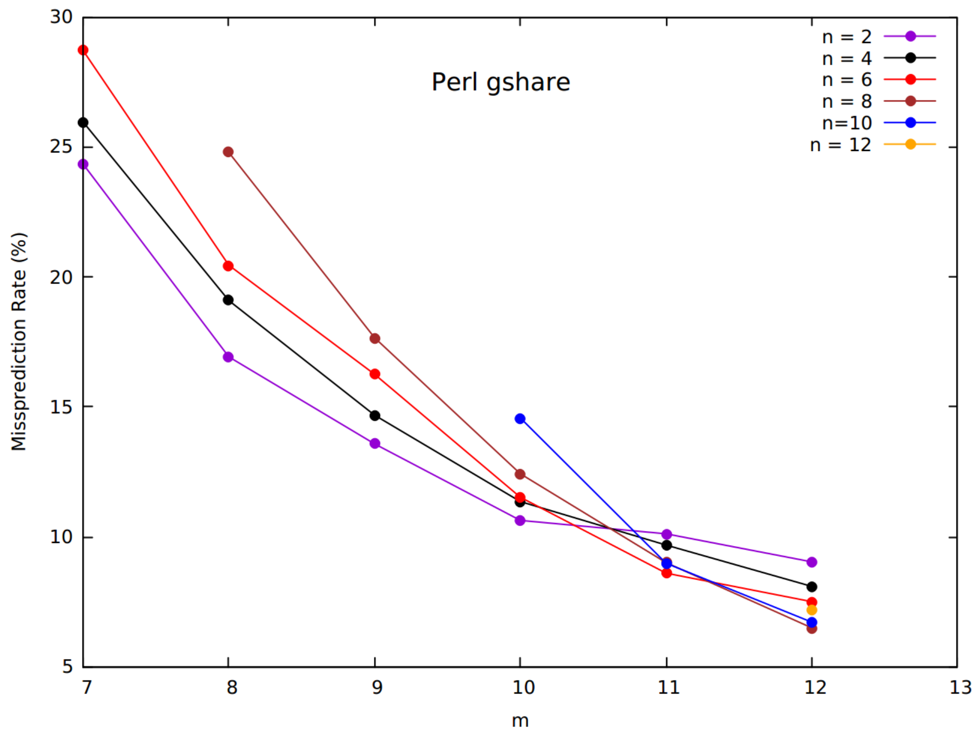

Figure 8: Results of the simulation for the perl trace using the gshare predictor.

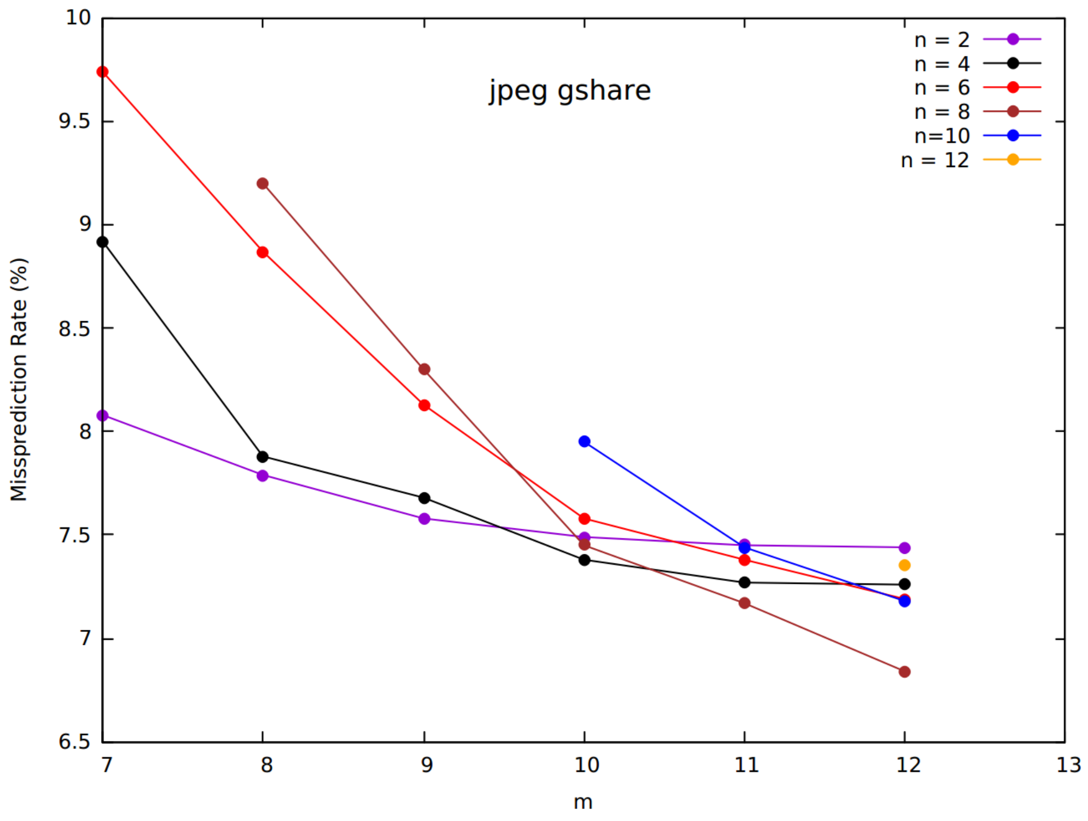

Figure 9: Results of the simulation for the jpeg trace using the gshare predictor.

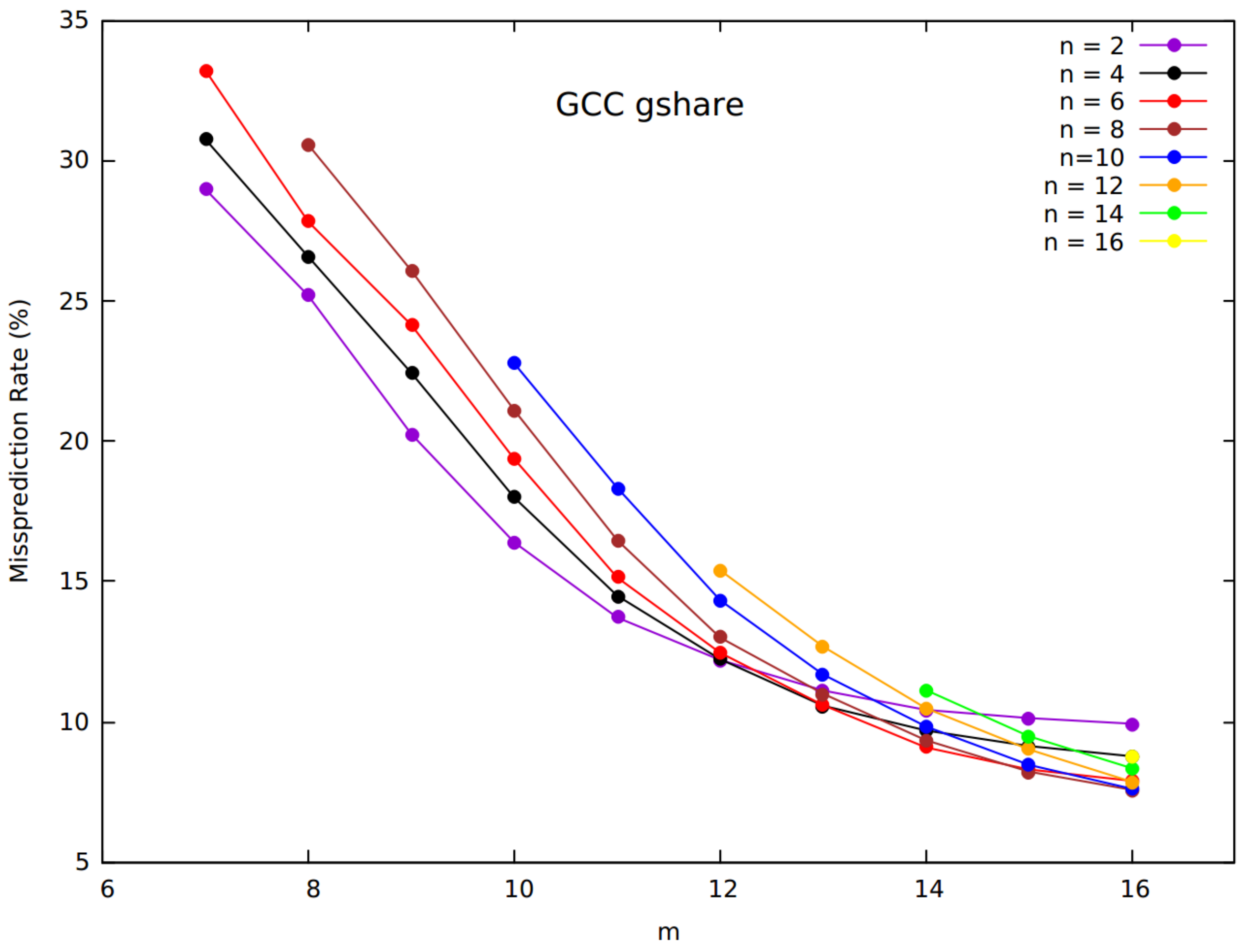

Figure 10: Graph used to determine the “optimal” configuration for the gshare predictor for the gcc trace.

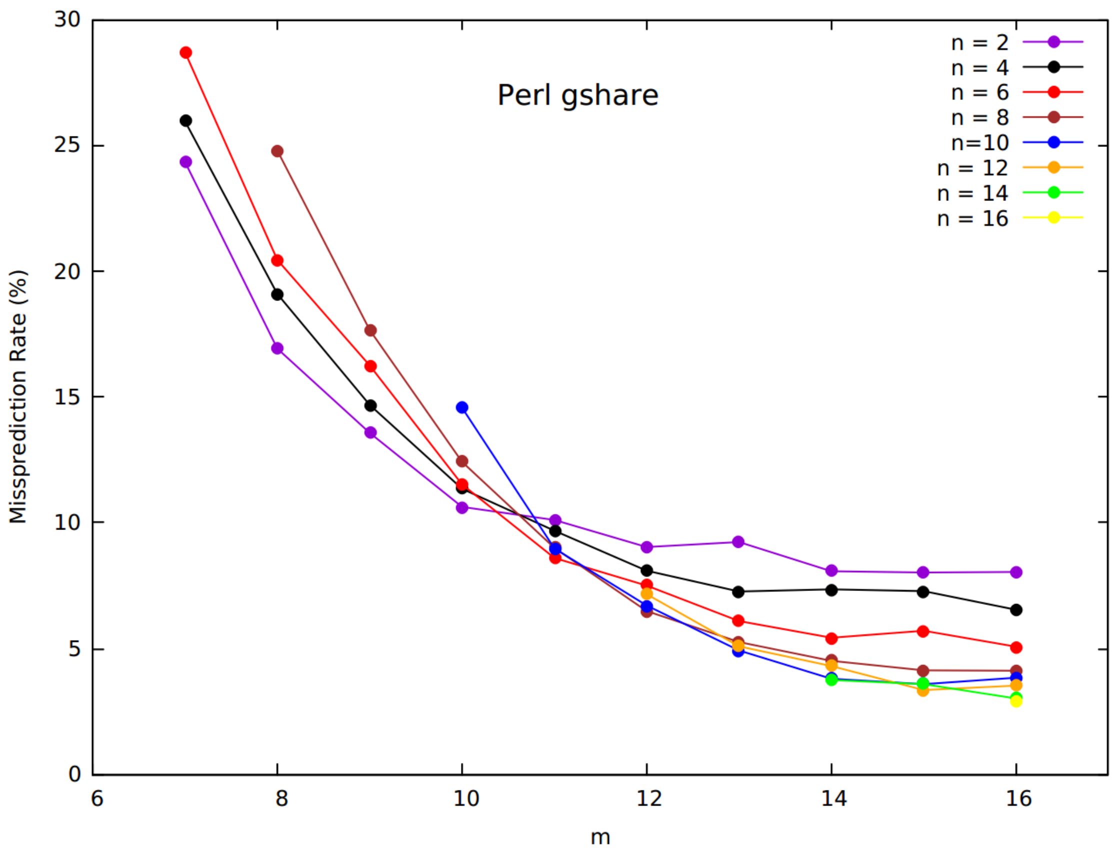

Figure 11: Graph used to determine the “optimal” configuration for the gshare predictor for the perl trace.

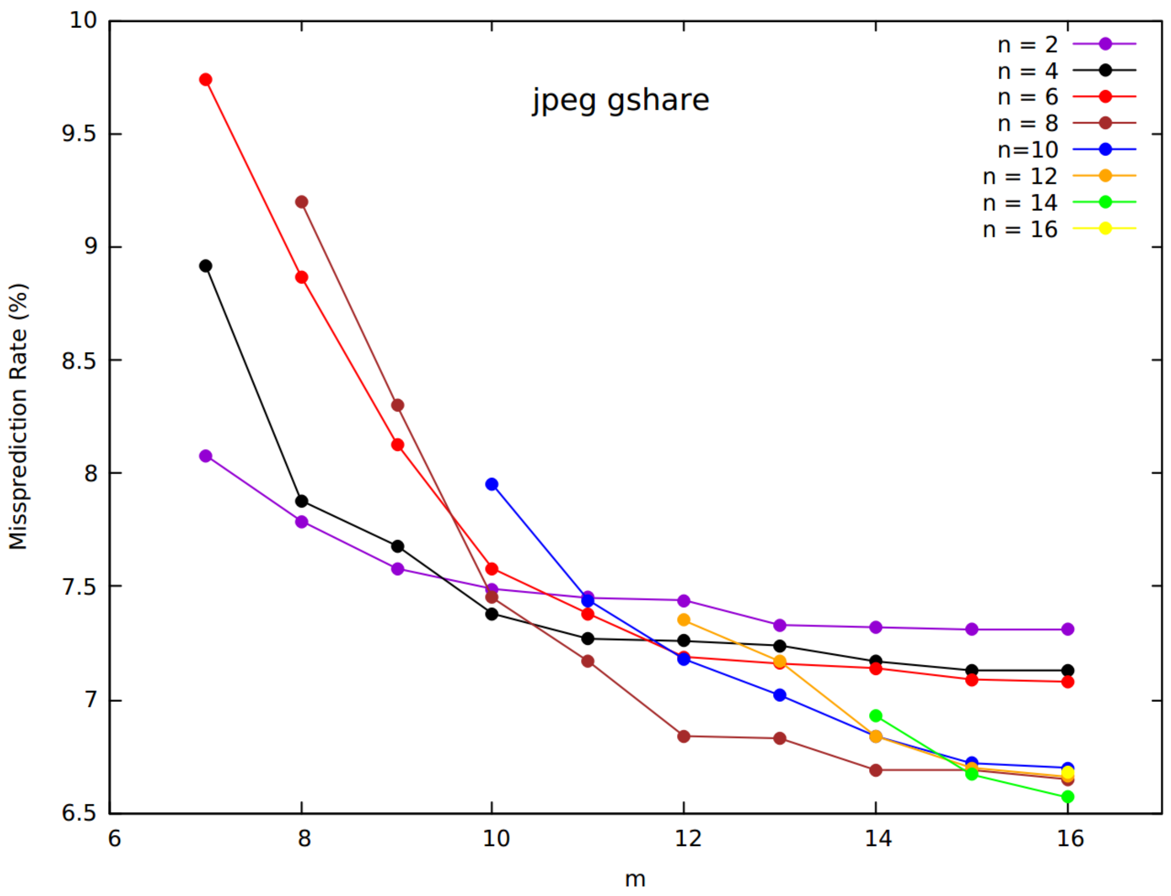

Figure 12: Graph used to determine the “optimal” configuration for the gshare predictor for the jpeg trace.

Table 2: Optimal value of m and n for each trace with corresponding misprediction values and sizes.

Trace	m	n	Prediction Table Size	Misprediction Rate (%)
gcc	16	8	16 kB	7.57
perl	14	10	4 kB	3.80
jpeg	14	8	4 kB	6.69

From Figures 7-9, it can be seen that for all values of n and for each trace, as the value of m increases the misprediction rate decreases. This is expected because as the value of m increases the number of predictor table entries increases, decreasing the interference that may occur between indexes. The impact of n on the misprediction rate may seem complex, but for each trace the curves for each value of n follow a similar pattern. This pattern is that when m is equal to 7, the lowest value of n provides the best prediction, but as the value of m increases, the higher values of n begin to give better prediction performance. An explanation of this may be that when m is a low value (e.g. 7) and n is a value close to m (e.g. 6) the predictor table index is mostly composed of the XORing of the global history register and the lower 7 bits of the PC value. If this happens, then the information in the branch's PC that differentiates two specific branches may be lost, allowing for more interference to happen in the prediction table. This idea is backed up by the behavior of the curves as m increases for higher values of n (n=6, n=8, n=10). When m increases for each trace, the higher values of n begin to give better predictions than the lower values of n. This is because as m increases for a certain value of n, the index to the prediction table begins to incorporate more information about the specific branch because more of the index is made of the branch's PC reducing the amount of interference that occurs between branches. On top of allowing for more information about a specific branch, increasing the value of m allows for the incorporation of branch history information into the top n bits of the index with the information that identifies a specific branch leading to an overall better prediction. From the graphs it can be seen that the gcc trace has the highest minimum misprediction rate, while the perl and jpeg traces have the lowest. Although the gcc trace has the highest minimum misprediction rate, increasing the value of m further may help greatly considering the misprediction rate is decreasing at a trend that has a relatively steep slope. When compared to the bimodal prediction of each trace, the gshare predictor can provide a lower minimal misprediction rate, but the gshare predictor requires a larger predictor size in order to achieve it.

Figures 9-11 show plots of the misprediction rates for various values of m and n, where m and n are now allowed to increase to 16. These plots can be used to determine the "optimal" configuration for a gshare predictor under a size constraint of 16 kilobytes. For the jpeg trace, the lowest misprediction rate occurs at m=16 and n=14, and has a value of 6.57%. Although this is the lowest misprediction rate, a much smaller prediction table can be used to attain a prediction rate that is only 1.018 times 6.57%. This occurs when m=14 and n=8 and this will be the configuration that "minimizes" prediction error. For the gcc trace, the misprediction rate decreases steadily and does not level out as the maximum size is approached. For this reason the optimal configuration will be chosen to be m = 16 n = 8. For the perl trace, the misprediction rates begin to level out at m = 14 and n=10, and this will be the "optimal" configuration chosen for this trace. The results for the "optimal" configurations of the gshare predictor for each trace reveal that perl requires more branch history information while gcc and jpeg do not.