Project 2: Branch Prediction

October 28, 2016

CONTENTS

1 Exploring the Bimodal Predictor

1.1 Plot 1

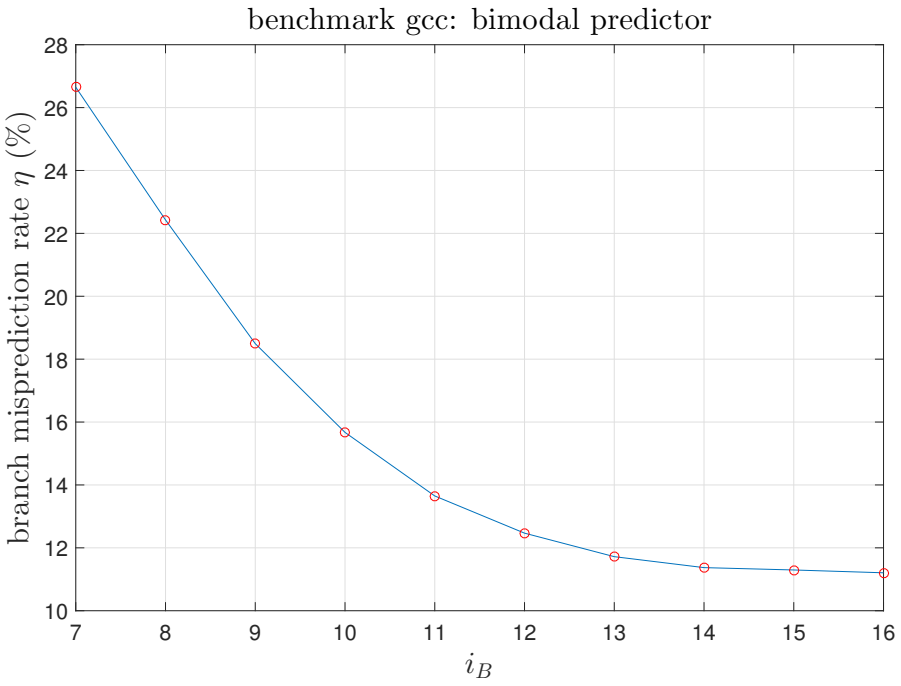

Figure 1.1: Bimodal Predictor Performance in benchmark gcc

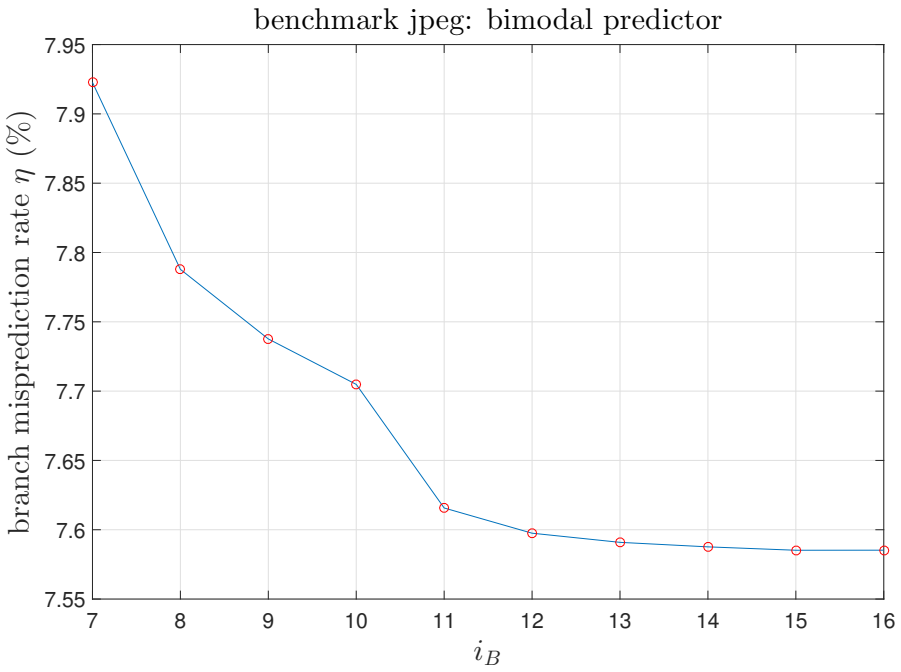

Figure 1.2: Bimodal Predictor Performance in benchmark jpeg

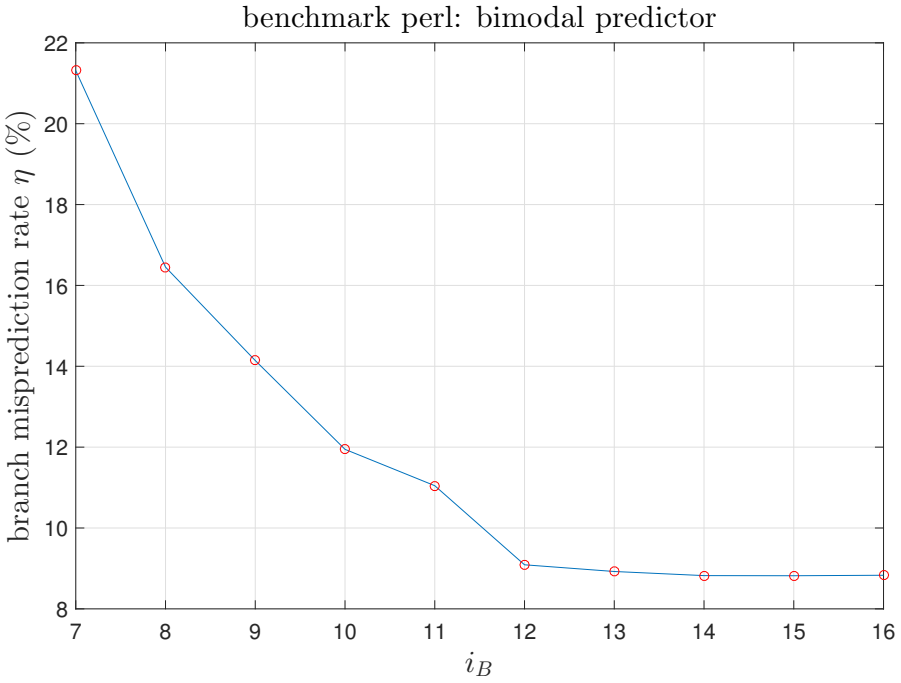

Figure 1.3: Bimodal Predictor Performance in benchmark perl

1.2 Analysis

From Figure 1.1, 1.2 and 1.3, we have observations that branch misprediction rate decreases, nearly exponentially, as the index width, i_B, increases of which index in PC is used to index the counters in bimodal predictor. Because different branches may index the same entry in the prediction table, which is called "interference", the longer the index is, the smaller the side-effect of interference is and thus the smaller the misprediction rate is. However, after a certain value of i_B, the descending of misprediction rate slows down - diminishing returns. From Figure 1.1, 1.2 and 1.3, the point is $i_B = 12$.

1.3 Design

In order to take both misprediction rate and size in to consideration, **misprediction rate and logarithm size production** of a bimodal predictor is defined as follows,

$$\textbf{misprediction rate and logarithm size production} = \text{misprediction rate} \times \log_2(\text{predictor storage size}) \tag{1.1}$$

Apparently, the smaller misprediction rate and logarithm size production is, the better predictor is.
Figure 1.4 shows the misprediction rate and logarithm size production of bimodal predictor under different benchmarks and their average.

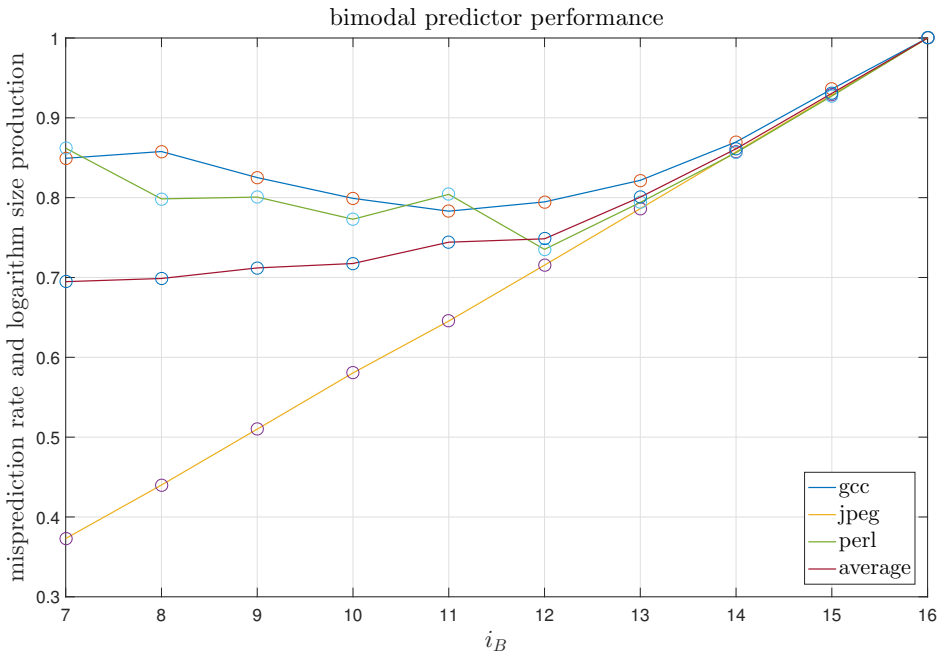

Figure 1.4: Bimodal Predictor Performance (after normalization)

From the average line in the Figure 1.4, before the diminishing returns ($i_B = 12$), the misprediction rate and logarithm size production is nearly flat, and therefore I'll choose $i_B = 10$ or $i_B = 12$ as final design of bimodal predictor.

2 Exploring the Gshare Predictor

2.1 Plot 2

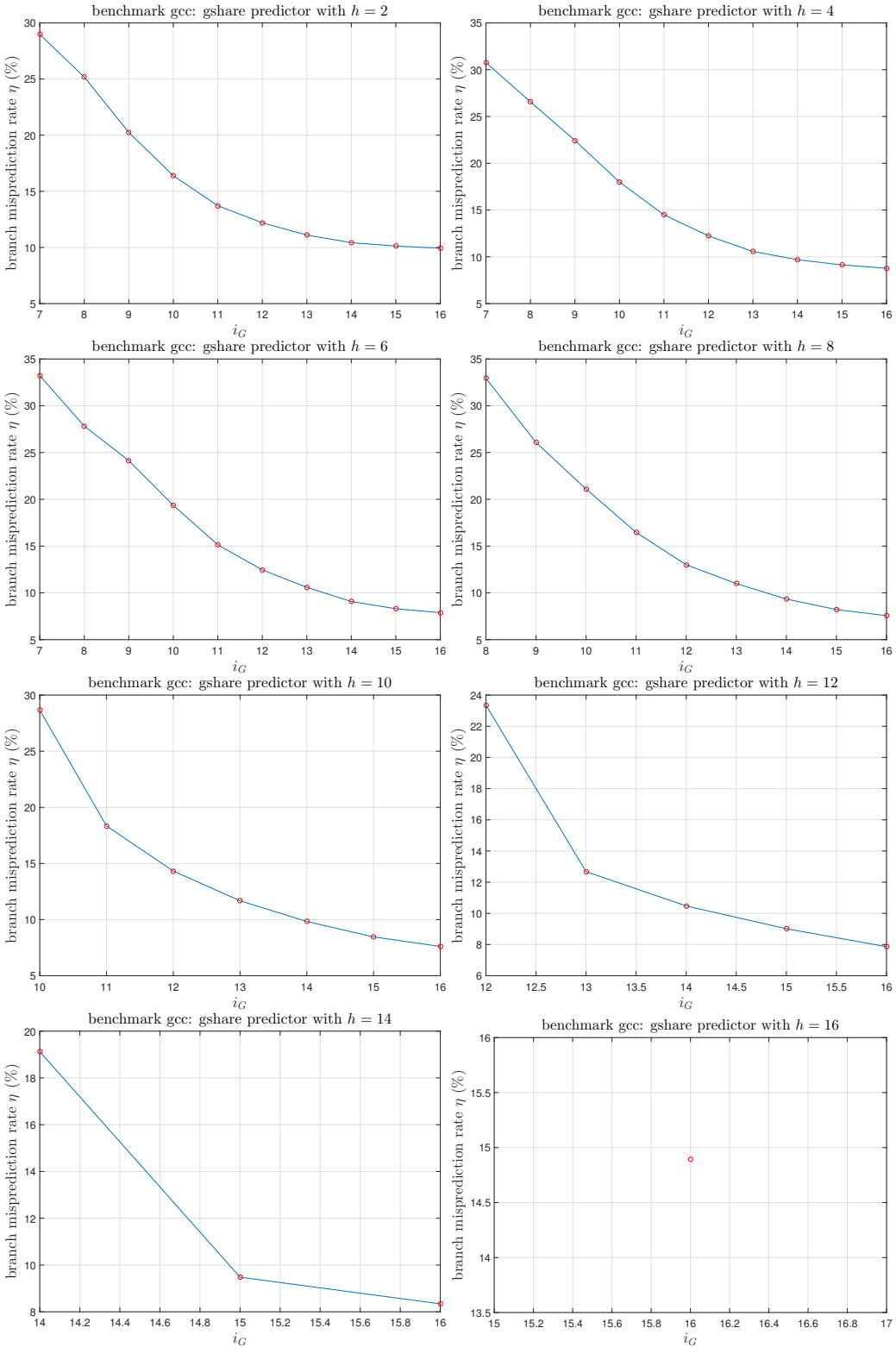

Figure 2.1: Gshare Predictor Performance in benchmark gcc

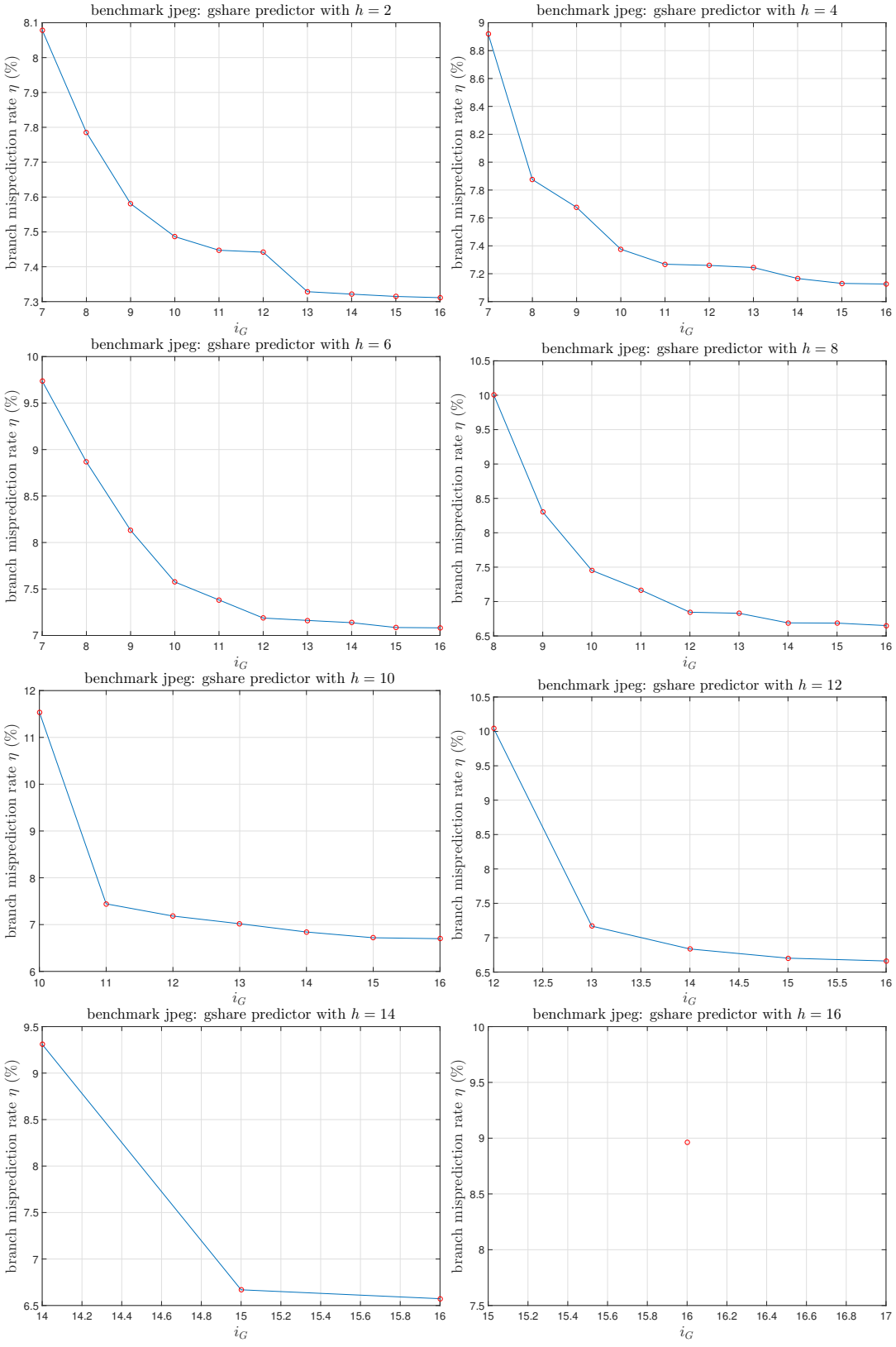

Figure 2.2: Gshare Predictor Performance in benchmark jpeg

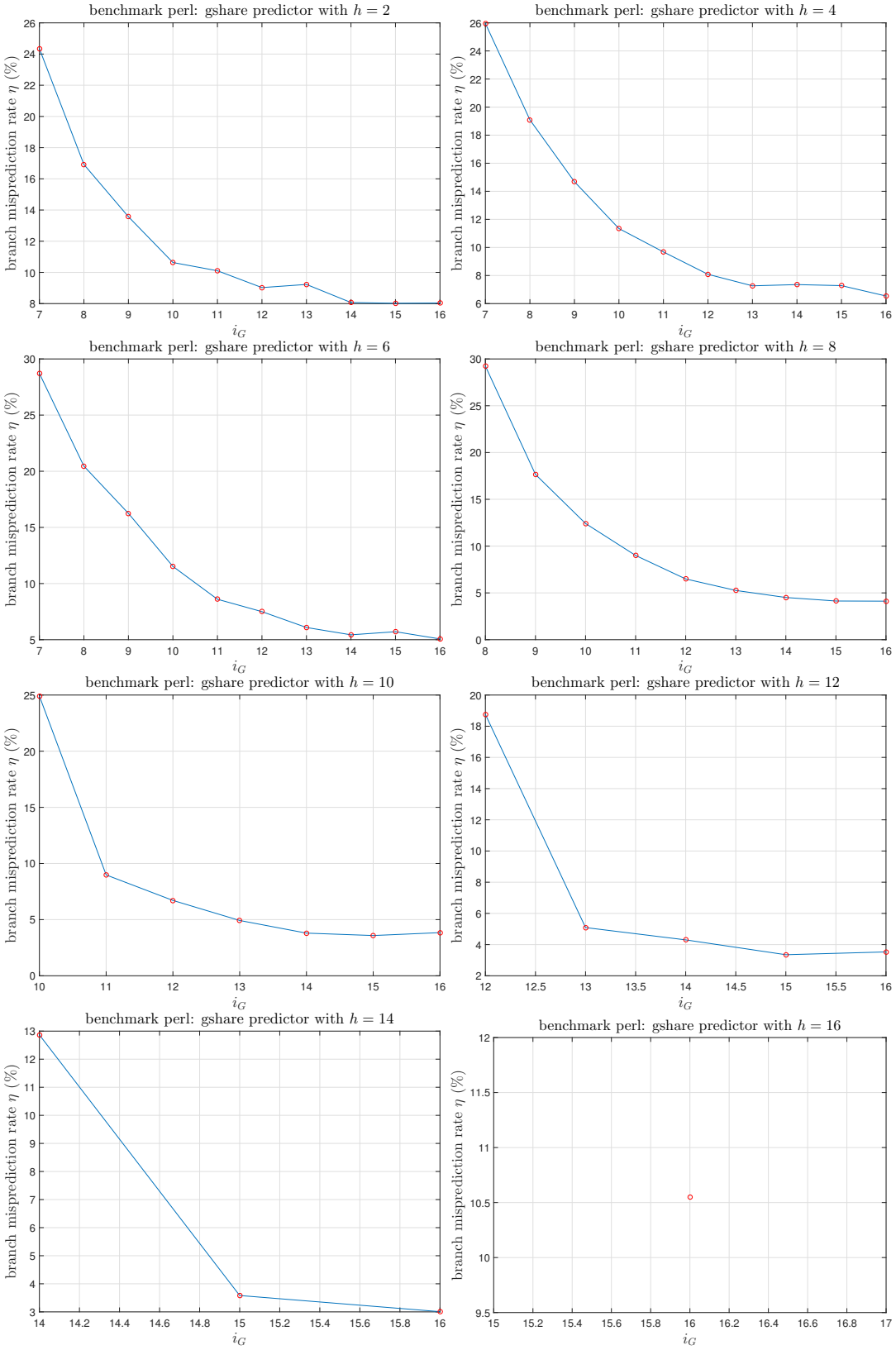

Figure 2.3: Gshare Predictor Performance in benchmark perl

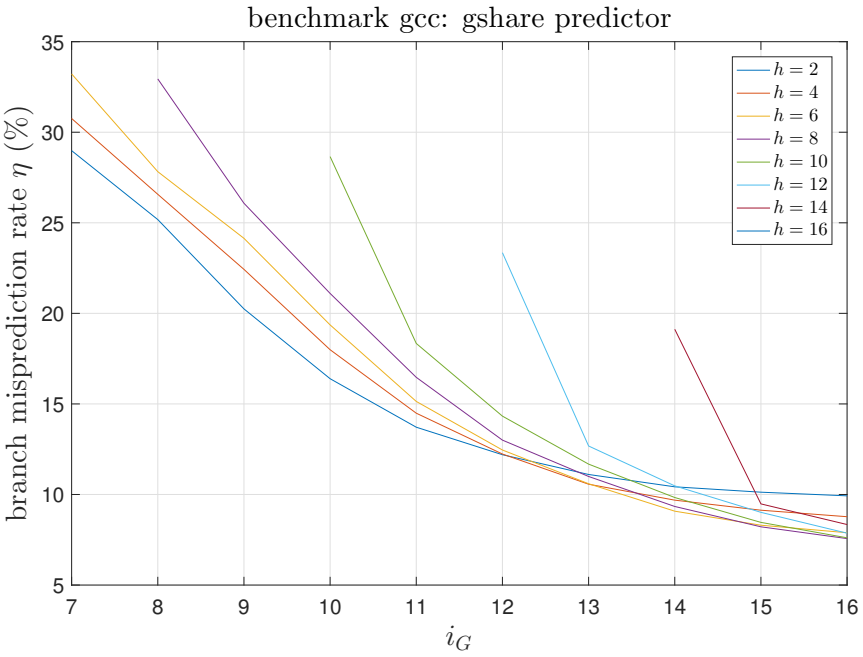

Figure 2.4: Gshare Predictor Performance in benchmark gcc

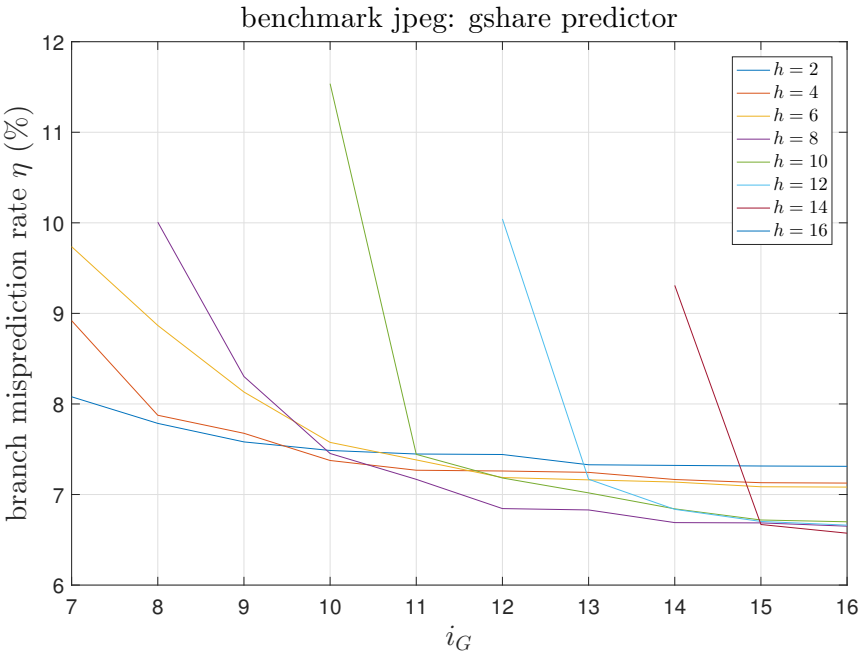

Figure 2.5: Gshare Predictor Performance in benchmark jpeg

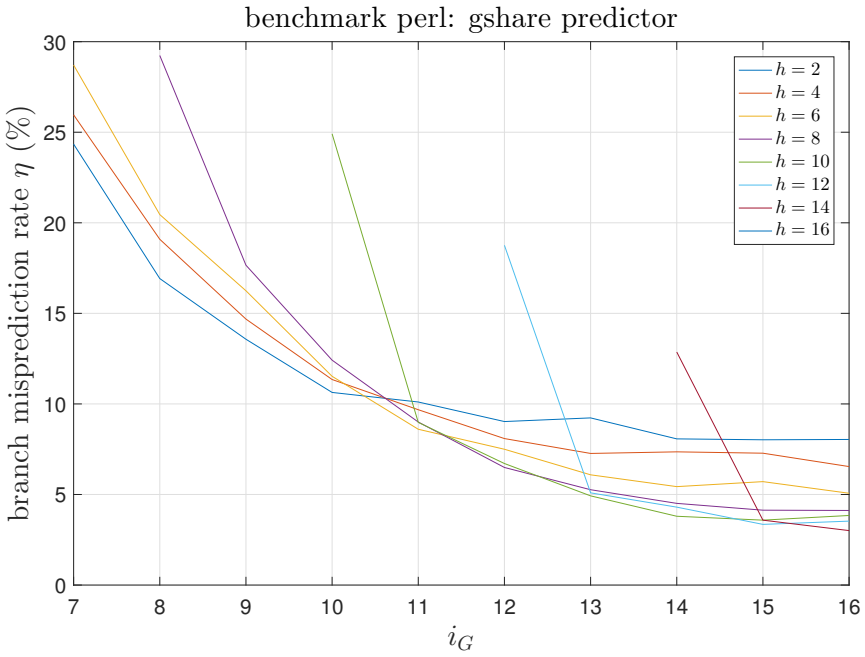

Figure 2.6: Gshare Predictor Performance in benchmark perl

2.2 Analysis

From Figure 2.1, 2.2 and 2.3, we have observations that branch misprediction rate decreases, nearly exponentially, as the index width, i_G, increases in gshare predictor. The reason may be the same as that of bimodal predictor, which is "interference". However, after a certain value of i_B, the descending of misprediction rate slows down - diminishing returns. From Figure 2.1, 2.2 and 2.3, the point is $i_B = 12$.
In Figure 2.4 (benchmark gcc), higher length of global history register, h, yields a higher misprediction rate before diminishing returns but yields a lower misprediction rate after diminishing returns. Besides, the speeds of descending of misprediction rate versus i_G are nearly the same among different h.
In Figure 2.5 (benchmark jpeg), most higher length of global history register, h, yields a lower misprediction rate. Besides, the speed of descending of misprediction rate versus i_G increases as h increases.
Figure 2.6 (benchmark perl) looks like the combination of Figure 2.4 and Figure 2.5. Higher length of global history register, h, yields a higher misprediction rate before diminishing returns but yields a lower misprediction rate after diminishing returns. Besides, the speed of descending of misprediction rate versus i_G increases as h increases.

2.3 Design

The concept of **misprediction rate and logarithm size production** is used again here.
Figure 2.7 shows the misprediction rate and logarithm size production of gshare predictor under different benchmarks.
Figure 2.8 shows average the misprediction rate and logarithm size production of gshare predictor.

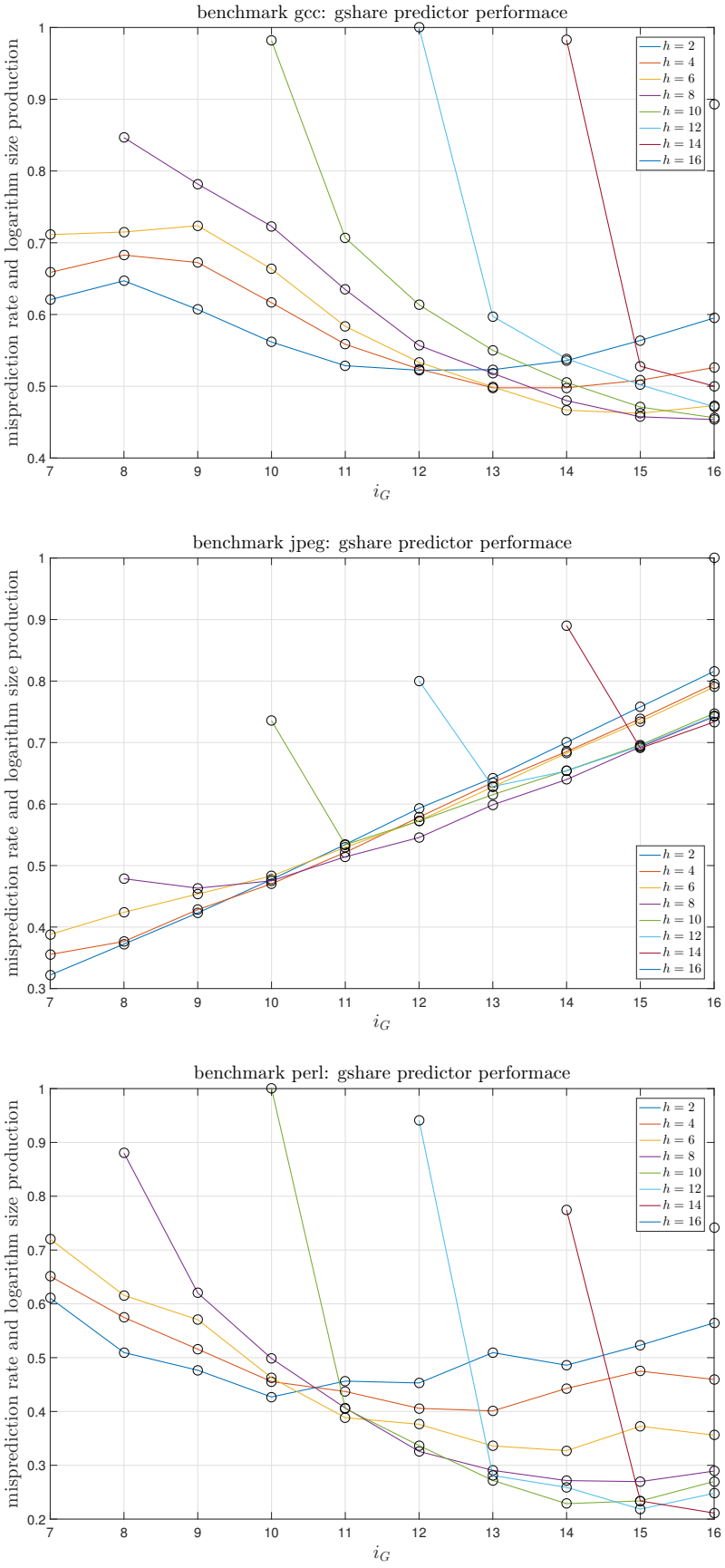

Figure 2.7: Gshare Predictor Performance in different benchmarks (after normalization)

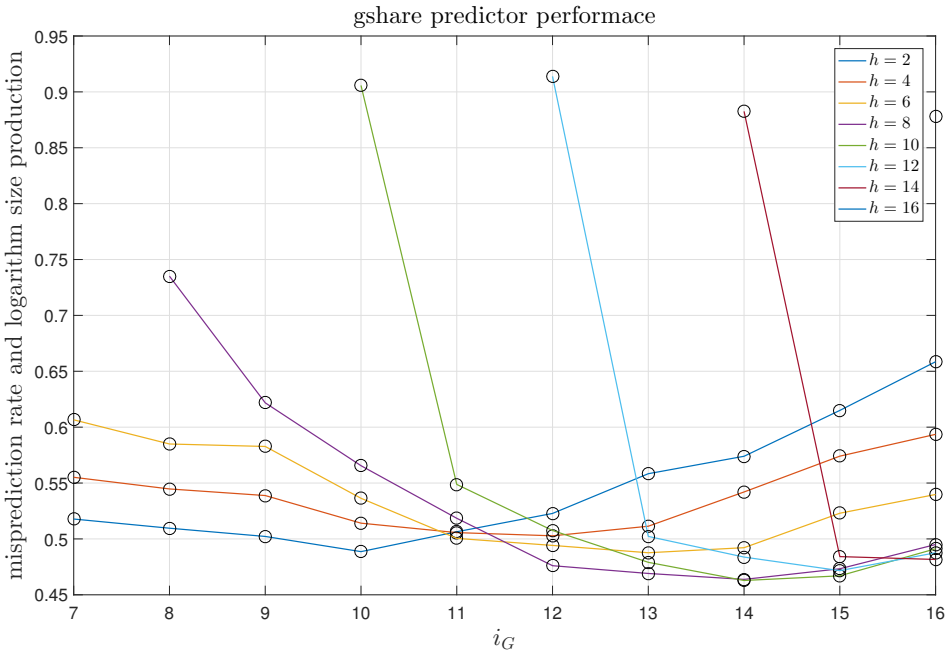

Figure 2.8: Gshare Predictor Performance (after normalization)

Because the smaller the misprediction rate and logarithm size production is, the better the predictor is, and from Figure 2.8, I'll choose $i_G = 12$ and $h = 8$ as my design of gshare predictor.

3 Exploring the Branch Target Buffer

3.1 C code

The C code of branch target buffer is provided in **btb.h** and **btb.c**.

4 Exploring the Hybrid Predictor

4.1 C code

The C code of branch chooser table is provided in **bct.h** and **bct.c**. The C code of hybrid predictor is provided in **bp.h** and **bp.c**.