

Deriving Limits of Branch Prediction with the Fano Inequality

Jeremy Singer, Gavin Brown and Ian Watson
University of Manchester, UK

June 19, 2006

Abstract

Information theory provides tools and techniques to measure the fundamental limits of predictability. Apart from this formal basis, researchers have attempted to construct so-called ‘optimal’ predictors in order to conduct limits studies for different aspects of program predictability (such as branch outcomes and method return values).

This paper reports on two case studies for branch prediction limits. The first study duplicates an earlier analysis of the branch predictability of the Quicksort algorithm by Mudge et al [15]. Whereas they design an optimal predictor for the algorithm, we use information theory and arrive at the same result with considerably less effort. The second study is the information theoretic analysis of the Championship Branch Prediction traces. Our analysis establishes the limits

of branch prediction for these dynamic execution traces. This allows us to ascertain how close each championship candidate gets in relation to the absolute limit.

The main insight of this paper is that predictability limits studies do not need to devise optimal predictors, but simply apply information theoretic measures, particularly the Fano inequality.

1 Introduction

Modern architecture is designed to execute program fragments in a *speculative* manner. Processors precompute results that are potentially required at some point in the future. However this is not entirely straightforward since program control flow is often non-sequential. Certain instructions cause program fragments to be repeated or skipped in the dynamic execution trace. In general, these instructions are known as *branch instructions*. A branch instruction can have one of two possible outcomes, based on the result of some computation. The *taken* outcome changes the program counter and causes non-sequential control flow. The *non-taken* outcome simply causes control flow to fall through to the subsequent instruction. Such conditional branch outcomes must be predicted in order to support speculation via pre-execution of the most likely instruction stream.

Many dynamic execution features can be predicted. These include branch outcomes [18], memory load values [11], method return values [16] and general

purpose register values [10]. This paper focuses on branch prediction, which is the simplest case since the answer is always a boolean value. Prediction of values from larger ranges is a more difficult problem [17]. Programs are predictable since their execution exhibits distinct *phases* which generally involve some periodicity. This periodicity occurs at the finest granularity due to high-level control flow structures like loops. Periodicity has also been observed over millions of instructions [5]. This may be more difficult to understand, but it is often caused by the modelling of real world data and processes which contain intrinsic redundancy.

Existing limits studies [18, 15, 6] for instance, involve particular predictors, which are employed to derive *empirical* limits of predictability. However, it is crucial to note that the *fundamental* limits of predictability must be predictor independent (or at least, they should only depend on the performance of the optimal predictor, and that differs for each program; indeed for each element that is to be predicted within a program!)

The key question to address is: ‘What is an optimal predictor?’ An *oracle* predictor is assumed to be omniscient, thus it should always achieve 100% accuracy. Realistic predictors observe a number of events through a finite window. Their scope is necessarily limited, unlike the potentially infinite scope of events observable by the oracle predictor. Such realistic predictors may be less accurate than the oracle, but there is still the concept of optimality given a fixed event window. That is to say, given a fixed set of input events to a predictor, what is the best possible prediction accuracy

that can be achieved? So whereas previous limits studies fixed both the predictor *inputs* and the prediction *algorithm*, this limits study will only fix the predictor inputs, and leave the algorithmic details unspecified.

Information theory can be used to derive fundamental limits of predictability, based on measures such as *entropy*. Information theory provides mathematical techniques to characterize optimal prediction without the need to construct particular predictor models. The main contribution of this paper is the insight that the performance of optimal predictors can be bounded using information theory, without the knowledge of how such optimal predictors are built.

The rest of paper is structured as follows: Section 2 lays the mathematical foundation by reviewing crucial concepts from information theory. It explains the Fano inequality and shows how this relates entropy with predictor accuracy. Section 3 studies the Quicksort algorithm. It reports how the results of a previous analysis of branch predictability may be duplicated using information theory. Section 4 studies the Championship Branch Prediction traces. It reveals how optimal predictors that use just local or global branch history would perform in comparison to the original championship candidates. Section 5 considers related work and Section 6 concludes.

2 Background

Information theory [4] provides a rich mathematical framework for analysis of data sources. Originally developed by Shannon in the context of secure communications and cryptography, over the past five decades it has been applied in fields as diverse as medical image processing, machine learning, and financial markets analysis. This paper explores how these ideas might be related to branch prediction.

2.1 Entropy and Conditional Entropy

The fundamental measure in information theory is *entropy*, which explicitly quantifies the information content in a given source of data: the more randomness or ‘unpredictability’ in the data source, the higher the entropy value. As an example consider a device emitting symbols drawn according to a random variable X , from the finite alphabet of possible symbols, \underline{X} . If we assume each successive symbol x is independent of the previous ones, the *unconditional entropy* is defined as,

$$H(X) = - \sum_{x \in \underline{X}} p(x) \log(p(x)) \quad (1)$$

where $p(x)$ is the probability of symbol x being produced. Note that all logarithms are taken to base 2. In practical terms, $p(x)$ can be calculated

with frequency counts, i.e.:

$$p(x) = \frac{\text{number of occurrences of symbol } x}{\text{total number of symbols seen}} \quad (2)$$

In this work, we consider the device as a branch instruction within a computer program, emitting a value each time it is executed. We employ the convention that value 0 corresponds to ‘branch not taken’ and value 1 corresponds to ‘branch taken’. Each successive execution may result in a change of the machine state, so we consider the random variable defining the branch behavior to have changed also: the variable now is X , while on the next execution it is Y , and we wish to compute the uncertainty in Y given that we know X . In this case it is clear that successive branch outcomes are *not* independent of one another, and what we have in fact could be considered as a *time series* of values. An unconditional entropy measurement will not take this into account, therefore we use *conditional entropy*,

$$H(Y|X) = - \sum_{x \in \underline{X}} p(x) \sum_{y \in \underline{Y}} p(y|x) \log(p(y|x)). \quad (3)$$

The required probabilities can again be computed from frequency counts:

$$p(y|x) = \frac{\text{number of times } y \text{ follows } x}{\text{number of occurrences of } x} \quad (4)$$

Conditional Entropy (CE) takes a minimum value of zero and a maximum value of $\log(|\underline{Y}|)$. In the case of branches, there are just two values, so

$\underline{Y} = \{0, 1\}$ and the maximum value is $\log(2) = 1$. It measures the uncertainty we have in the next branch outcome, given that we know the current outcome. If values are produced uniformly at random over the set of possible symbols \underline{Y} , then equation 3 will converge in the limit of large sequences to $\log(|\underline{Y}|)$.

We have so far only considered the case when we take *one* element of history into account for predicting the next value, such that the random variable X is defined over the ‘history’ set $\underline{X} = \{0, 1\}$.

If we used the previous *two* elements of history, then the random variable X would be defined over the history set $\underline{X} = \{00, 01, 10, 11\}$, for a three element history $\underline{X} = \{000, 001, \dots, 111\}$, and so on¹. As one might expect, if we add more history then there should be more information for our predictor to use, and its accuracy should be higher. Accordingly, if we compute CE with history $\underline{X} = \{00, 01, 10, 11\}$, it will always be less than or equal to that with history $\underline{X} = \{0, 1\}$. The exact *amount* of information added when we include that extra element of history is quantified by the reduction in the conditional entropy. An important question now is, ‘how much accuracy can we gain using this extra information?’. It turns out that a further result from information theory can answer this, which we will discuss in the next section.

¹Since we are just trying to predict the *next* single value, \underline{Y} is always defined over $\{0, 1\}$, even if the history set \underline{X} varies.

2.2 Fano's Inequality

One of the most important developments for information theory is Fano's inequality [7]. If the misprediction rate on any given data source is p (in the range $[0, 1]$), then

$$p \geq \frac{H(Y|X) - h(p)}{\log(|\underline{Y}| - 1)} \quad (5)$$

Or rearranged as it is more commonly shown in the Information Theory literature,

$$h(p) + p \log(|\underline{Y}| - 1) \geq H(Y|X) \quad (6)$$

where $h(p)$ is the *binary entropy function*, shown in Figure 1, defined as

$$h(p) = -(p \log(p) + (1 - p) \log(1 - p)) \quad (7)$$

Figure 2 shows the Fano inequality plotted for boolean random variables ($|\underline{Y}| = 2$). Note that the vertical axis is plotted as 'prediction accuracy' which is computed as $(1 - \text{minimum misprediction rate})$. If a boolean data source is *completely random*, its conditional entropy will be 1.0; since it is random we would expect to have no better than 50/50 chance in predicting it—accordingly Fano's inequality is equal to 0.5 at this point. If the CE is lower (less randomness), then we may be able to get higher than 50% accuracy; if it is completely un-random, for example $(1, 1, \dots, 1)$ then we have $H(Y|X) = 0.0$, and the potential to achieve 100% accuracy. Suppose a hypothetical data source has a conditional entropy of 0.75, and the particular

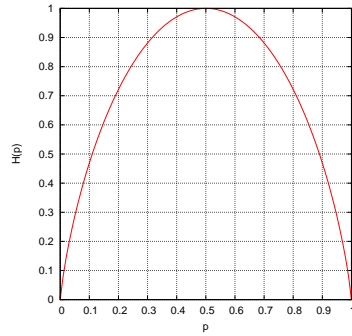


Figure 1: The binary entropy function

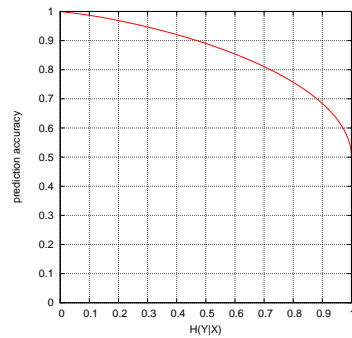


Figure 2: The Fano inequality for boolean data

(hypothetical) scheme that was used to predict the data values had an accuracy of 70%. Fano’s inequality shows that the limit of achievable accuracy for $H(Y|X) = 0.75$ is approximately 78%. It is mathematically impossible to create a predictor that, using the same history length, can achieve above this level—the data source contains a degree of randomness that cannot be resolved by *any* predictor.

The next section explores how this bound can be used in practice to examine the limits of branch prediction.

```

function quicksort(list  $q$ )
  var list  $less, pivotList, greater$ 
  if length( $q$ )  $\leq 1$ 
    return  $q$ 
  select a pivot value  $pivot$  from  $q$ 
  for each  $x$  in  $q$  except  $pivot$ 
    if  $x \leq pivot$  then add  $x$  to  $less$       /*  $b_0$  */
    if  $x > pivot$  then add  $x$  to  $greater$     /*  $b_1$  */
  add  $pivot$  to  $pivotList$ 
  return concatenate(quicksort( $less$ ),  $pivotList$ , quicksort( $greater$ ))

```

Figure 3: Simplified quicksort algorithm in pseudocode, adapted from Wikipedia

3 Quicksort Study

This section considers exact analysis of branch predictability for the *quicksort* algorithm. We use information theory to duplicate the results of an earlier analysis by Mudge et al [15].

Figure 3 presents a simple quicksort algorithm. Note that for the sake of simplicity of presentation, we do not give an in-place quicksort algorithm. This simple version allocates new lists each time a new pivot is selected, which is rather inefficient but does not affect the branching behaviour of the algorithm. Each quicksort recursive function call selects a pivot element from the input list, then divides the list into two sublists, one containing elements that are less or equal to the pivot, and the other containing elements that are greater than the pivot. Then the pivot element is positioned in between these two sections, since it is in its final position for the correctly sorted

list. Then we divide and conquer for the two sections on either side of the correctly positioned pivot.

Most branches in the quicksort algorithm are easy to predict. Mudge et al claim that they are effectively 100% predictable if enough execution history is provided. There are only two branches that are difficult to predict (b_0 and b_1) since their outcomes are entirely data dependent. These branches compare different elements to the pivot and control whether to move the elements into either the first (*less*) or second (*greater*) sublists.

There are a number of assumptions underlying the analysis of Mudge et al. We enumerate them explicitly here:

1. The n numbers to be sorted are distinct.
2. Each possible initial ordering is equally likely.
3. Each subarray of each iteration is in random order.
4. The overall branch prediction performance of the algorithm coincides with the performance in one iteration of the algorithm on a sufficiently large array.

The rest of the section examines two different methods for calculating the predictability limit of the quicksort algorithm. Section 3.1 reviews Mudge's method. He constructs an optimal predictor and analyses its behaviour. Section 3.2 applies information theoretic analysis to the quicksort algorithm and derives the same optimal behaviour.

3.1 Optimal Predictor Analysis

Mudge et al [15] design an *optimal predictor* for the two data dependent branch outcomes in the quicksort algorithm. Their predictor ‘maintains a running count of the proportion of elements examined so far that are greater than the pivot, and compares this quantity to 0.5 to decide which way to predict the next branch.’ Basically, if the majority of elements seen so far has been greater than the pivot, then they predict that the new element will also be greater, and vice versa. In the event of a tie they make a random guess.

Their predictor effectively estimates the *rank* (or final position in sorted list) of the pivot online. They estimate whether the pivot is above or below the median value in the list. They claim that it is possible to do this with arbitrarily high accuracy from the first \sqrt{n} elements for large enough n . The predictions made while this estimate is being computed form a negligible fraction $1/\sqrt{n}$ of the total number of predictions. Thus the scheme’s performance approaches the situation where we know the pivot rank ahead of time. Since the list is completely randomized, then if the normalized pivot rank $p = \text{pivot rank}/n$, it is the case that p will be uniformly distributed over $[0, 1]$ as n becomes large, and the branch prediction rate will be $\max(p, 1 - p)$, as shown in Figure 4. The expected success rate is $\int_0^{0.5} (1-p) dp + \int_{0.5}^1 p dp = 0.75$.

So the optimal branch predictability of quicksort is 75%. Mudge et al present some empirical studies to show that real-world predictors approach this theoretical upper limit from below.

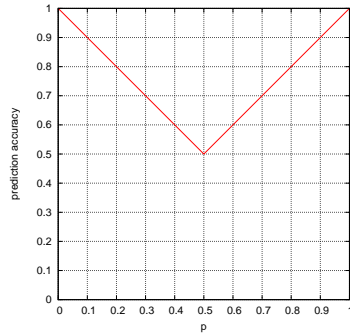


Figure 4: Prediction rate for Mudge's optimal predictor

3.2 Information Theoretic Analysis

In order to analyse the problem with information theory, it is necessary to extract the entropy of the branches in question. To be precise, we need to compute the entropy of the sequence of branch outcomes. For each branch, there are two possible outcomes. Assume that 0 is 'not taken' and 1 is 'taken'. So each branch outcome trace will be a sequence of bits. It is possible to compute the entropy of this bit stream in standard manner, using the entropy equation 1 specialized for two outcomes.

In Mudge et al's description of the problem, the normalized pivot rank p (where $0 \leq p \leq 1$) relates directly to the probability of the branch being taken or not taken. For branch b_0 (the first branch) $p_0 = p$ and $p_1 = 1 - p$. For b_1 (the second branch) $p_0 = 1 - p$ and $p_1 = p$.

Thus we can plot pivot rank against branch outcome trace entropy, as shown in Figure 1. (Note that this figure is simply the binary entropy function, equation 7.) Since we are assuming the probabilities a priori, then the

entropy is entirely independent of the size of the outcome trace. Also note that since the plot is symmetric about $p = 0.5$ then it is true for both b_0 and b_1 .

We have already reviewed how conditional entropy can be related to prediction accuracy using the *Fano inequality*. Since the elements in the array are guaranteed to be entirely randomly distributed, then there is no relationship between successive array elements. So the conditional entropy value should be the same as the unconditional entropy, for any order. That is, $H(Y|X) = H(Y)$ where X is the branch history and Y is the current outcome.

Recall the Fano inequality from Figure 2. The equation can be written as:

$$H(Y|X) = h(1 - m) \tag{8}$$

where $H(Y|X)$ is the conditional entropy of the branch outcome trace and h is the binary entropy function, see equation 7 and Figure 1, and m is the theoretical maximum prediction accuracy, where

$$m = (1 - \text{minimum misprediction rate}) \tag{9}$$

Now, it is required to map from pivot rank directly to prediction accuracy. This is easy to visualize using Figures 2 and 1, by eliminating H to derive the curve in Figure 4.

Below we outline the equivalent algebraic argument. We can rearrange

the Fano inequality as

$$m = 1 - \min(h^{-1}(H(Y|X))) \quad (10)$$

The inverse binary entropy function is given as $h^{-1}()$. The min function is required since the inverse binary entropy function has two solutions. We use min to select the larger value for maximum accuracy, to give the Fano inequality as shown in Figure 2.

Now, recall that for our situation, $H(Y|X) = H(Y)$ and the entropy is given by $h(p)$ where p is the pivot rank. Hence, substituting for $H(Y|X)$ in equation 10 gives

$$m = 1 - \min(h^{-1}(h(p))) \quad (11)$$

and since, for the binary entropy function, $h(p) = h(1 - p)$, then

$$m = 1 - \min(p, 1 - p) \quad (12)$$

which can be rearranged as

$$m = \max(p, 1 - p) \quad (13)$$

which gives us exactly the same predictability bound that Mudge derived. If we assume that all pivot ranks are equiprobable, then the mean predictability is given by the area under the curve, which is 0.75, as calculated by the

integral in Section 3.1. So our information theoretic analysis derives the same result as Mudge et al, without needing to construct an optimal predictor.

4 Championship Branch Prediction Study

This section describes an information theoretic study of the Championship Branch Prediction test data set. We compute theoretical upper limits of branch prediction, and compare these limits with the actual performance of the championship candidate branch predictors.

The *Championship Branch Prediction* (CBP) competition was held in 2004 [1, 19]. The organizers (from Intel Research) distributed a set of execution trace files and a C++ source code evaluation framework for branch prediction. Competitors were invited to design and implement predictors within the supplied framework. Then submitted predictors were tested using a previously undistributed set of execution trace files, and ranked according to performance. Branch prediction performance was measured in terms of *mispredicts per thousand instructions* (MPKI). The winning predictor [9] achieved a score of 2.574 MPKI on the undistributed traces.

The distributed trace files are still available from the CBP website [1]. There are 20 trace files, each records 30 million x86 instructions, including both user and system activity. The traces are split into four different categories, with five traces per category. The four categories are

1. DIST-INT: integer benchmarks from the SPEC CPU 2000 suite.

2. DIST-FP: floating-point benchmarks from the SPEC CPU 2000 suite.
3. DIST-MM: multimedia code, origin unspecified.
4. DIST-SRV: server code, origin unspecified.

The CBP evaluation framework gives a MKPI score per trace file. The overall score for a category, or indeed for the whole set of traces, is obtained by taking the arithmetic mean of MPKI scores for each trace file in the set.

We have analysed these files using our information theoretic techniques, in order to calculate the theoretical minimum misprediction score (MPKI). Recall that in order to compute the absolute limit of predictability, our analysis requires that the inputs to the branch predictor are fixed ahead of time. We conduct two studies, using different types of inputs. The first study restricts inputs to local branch history. That is, for each branch, the history of previous outcomes of that branch are used to predict the next outcome. The second study restricts inputs to global branch history. That is, for each branch, the history of previous outcomes of any branch are used to predict the next outcome. For both studies, we investigate the effect of allowing increasing amounts of history ‘bits’, where each bit represents a single outcome. So an n -bit local/global history predictor can use the last n outcomes of this/any branch to predict the next outcome, respectively.

Note that most existing branch prediction schemes lie somewhere in between these two extremes. Generally branch prediction schemes use a combination of local and global context. So these two studies effectively look

at extremal points, whereas real branch predictors operate in between these two extremes.

The procedure to compute theoretical minimum MPKI scores is as follows. We instrument the CBP prediction framework code to dump out branch outcomes as bits to a file. For the local study, we have one file per static branch instruction, whereas for the global study we have a single file for all branch instructions. These files are analysed to determine conditional entropies of various orders. The n th order conditional entropy is needed to compute the optimal prediction rate for n bits of history, whether local or global.

The minimum mispredict rate can be calculated from the conditional entropy by using Fano inequality. Since the Fano inequality equation is non-invertible, then we have to do this calculation numerically, as a table-lookup rather than algebraically. This gives us a misprediction rate per branch instruction. Then we have to multiply this number by the proportion of all instructions that are branches (reported by the CBP framework), then multiply by 1000 to get a MKPI score.

In the case of global history, the above calculation is sufficient. However for local history, it is necessary to do some normalization to get an average mispredict rate over all branches. Each branch has its own MPKI score. This score is then weighted according to its magnitude in relation to the total number of branches in the trace file. This allows us to compute an average misprediction rate over all branches.

Figures 5 and 6 plot the results for the local and global branch history studies respectively. The local study uses history lengths between 8 and 64. The global study only uses history lengths between 8 and 32. It was prohibitively expensive to compute conditional entropies for the global history traces for more than 32nd order. Note that an average trace file contains 3.9 million branches.

It is interesting to see how MPKI decreases as history length increases. It is also clear to see the order-of-magnitude difference in accuracy between local and global histories. Figure 7 highlights this, as it plots the mean MPKI scores over all categories, for both local and global histories. Another interesting point is the different performance of various trace categories. Loh has already studied these variations in some detail [12]. Our analysis confirms many of his findings.

1. The floating-point traces are most predictable on all predictors. Loh states this is due to the small number of distinct branches (the branch footprint) in typical floating-point programs.
2. The integer and multimedia traces have similar performance.
3. The server traces benefit greatly from increases in predictor hardware budget. In our case, this corresponds to increase in global history length. Loh states that server programs are not intrinsically difficult to predict, the main challenge is the large branch footprint.

It is interesting to compare the CBP MPKI scores of existing branch

predictors with the optimal scores computed above. The popular *gshare* predictor [13] achieves a mean score of 5.301 MPKI on the distributed trace set, which would be possible using an optimal local history predictor with 8 bits of history, or an optimal global history predictor with 16 bits of history. The CBP champion predictor [9] achieves a mean score of 2.823 MPKI on the distributed trace set, which corresponds with 8 bits of local history or 24 bits of global history for the optimal predictor.

However, note that both *gshare* and the CBP champion use other inputs apart from local or global history. For instance, the *gshare* predictor uses the address of the current branch as an input. This address is XORed with the global history. It should be possible to compute the entropy of these additional inputs and use the Fano inequality to determine how close to optimal the *gshare* scheme achieves, for its particularly specialized window of inputs.

Note that the CBP regulations constrain branch predictor candidates to operate within a fixed 64 KB storage budget. This size limit obviously has an impact on prediction accuracy, generally caused by the aliasing problem [14, 6]. Our information theoretic analysis is currently unable to model storage constraints. Future research will be directed towards incorporating such real-world limitations into the predictability bounds analysis.

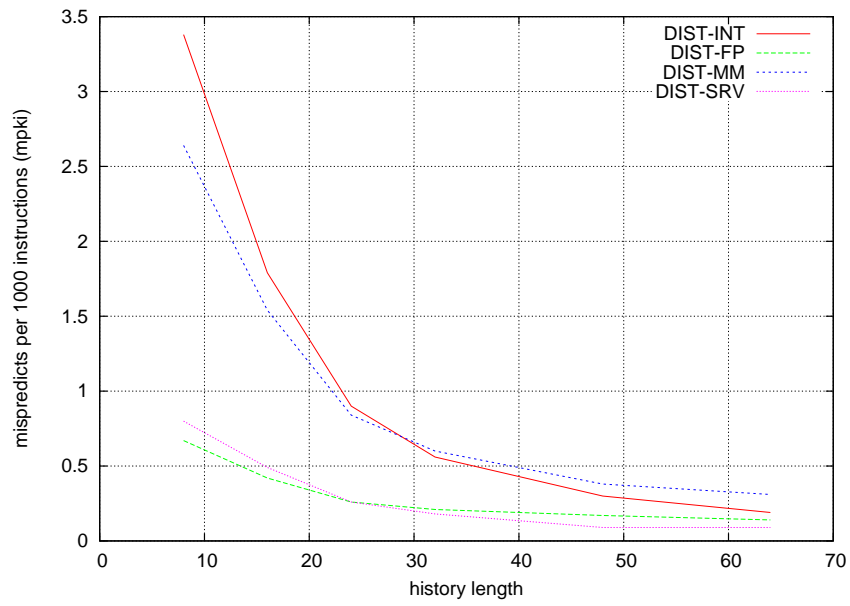


Figure 5: Minimum misprediction rates with local history

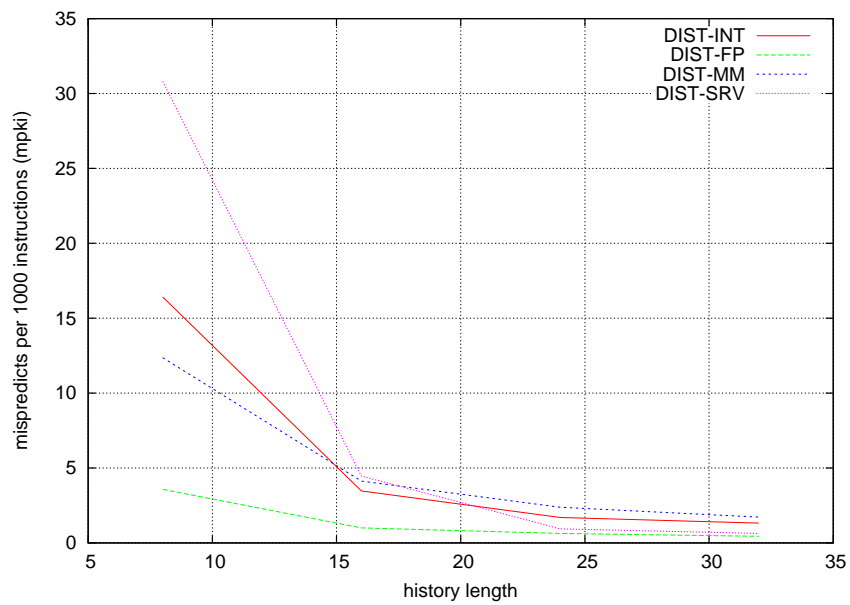


Figure 6: Minimum misprediction rates with global history

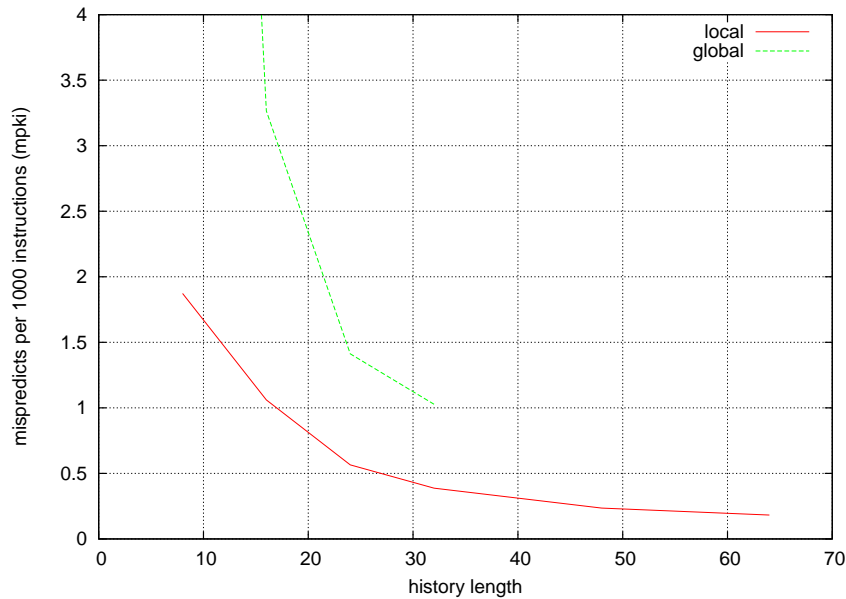


Figure 7: Mean minimum misprediction rates

5 Related Work

Many authors have contributed limits studies for branch prediction. Smith [18] presents an early overview of different schemes and their performance. Twenty years later, Eden gives a more modern overview [6].

Some authors have applied optimal predictors from data compression to the branch prediction problem [2, 8]. The standard example is ‘prediction by partial matching’ (PPM). Chen et al comment that optimal predictors establish limits to general branch prediction performance. However, they are still required to make certain assumptions about the predictor structure and inputs, so their claims of optimality are subject to this predictor structure and input set. These optimal prediction studies are all based on existing,

or new, branch predictor models. These models are simulated to measure their branch prediction performance. So the limit depends on the predictor model, inputs and subject programs. In contrast our technique derives a limit independent of the model. It only depends on inputs to a black box predictor. Again we rely on subject programs, but these are selected from a broad, industry-standard scope [12].

One previous study has related measured the entropy of the CBP trace files [12]. However Loh adopts a very pragmatic approach. He measures the ratio between the `bzip2` compressed trace of branch outcomes and the uncompressed trace. The ratio gives the average compressibility of the branch sequence, which allows him to estimate the minimum number of bits to represent each branch's information. This is his entropy estimate. In contrast to Loh's ad-hoc measurements, our technique for calculating branch outcome entropy uses proper information theoretic formalisms. Loh uses his entropy results relatively, to compare between the four different categories of CBP trace. He finds that DIST-MM has the highest entropy and DIST-FP has the lowest. He states that greater entropy should imply greater prediction difficulty. Our study confirms many of his findings.

More generally, there is a small amount of research that applies information theory to derive limits for different dynamic behaviour of programs. For instance, Clark et al [3] use information theoretic analysis to bound secure information leakage in simple imperative programs. Also some of our earlier work [17] focuses on entropy measurements for method return value

predictability to support speculative thread-level parallelism.

6 Conclusions

This paper has developed an information theoretic analysis in order to derive limits for branch prediction. This is an improvement on earlier branch prediction limits studies, which simply constructed supposedly optimal predictors and measured their performance. Whereas these earlier studies fixed both the branch predictor algorithm and the subject program, our new analysis is more generic—instead of fixing the predictor algorithm it fixes the inputs. It measures the entropy of these inputs over the dynamic execution trace of the subject program. The Fano inequality is used to relate entropy to optimal predictability. So effectively, our study treats the optimal branch predictor as a black box, with known inputs and output. This is less constrained than previous studies. One limitation of this analysis is that it is unable to model real-world constraints such as storage, timing and power. However in limits studies, these secondary features are often assumed to be unimportant. Effectively they are engineering constraints rather than fundamental theoretical limits.

Information theory has proved useful for branch prediction limits studies. However these limits studies treat branch predictors as black boxes and give no clues as to their makeup. The next question to address is: can information theory also be useful for designing a new generation of branch predictors?

References

- [1] The 1st JILP championship branch prediction competition, 2004. <http://www.jilp.org/cbp>.
- [2] I.-C. K. Chen, J. T. Coffey, and T. N. Mudge. Analysis of branch prediction via data compression. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 128–137, 1996.
- [3] D. Clark, S. Hunt, and P. Malacaria. Quantified interference for a while language. *Electronic Notes in Theoretical Computer Science*, 112:149–166, 2005.
- [4] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, 1991.
- [5] E. Duesterwald and C. Dwarkadas. Characterizing and predicting program behavior and its variability. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pages 220–231, 2003.
- [6] A. N. Eden. *Of limits and myths in branch prediction*. PhD thesis, University of Michigan, 2001.
- [7] R. M. Fano. *Transmission of Information: A Statistical Theory of Communications*. MIT Press & John Wiley and Sons, New York, 1961.

- [8] E. Federovsky, M. Feder, and S. Weiss. Branch prediction based on universal data compression algorithms. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 62–72, 1998.
- [9] H. Gao and H. Zhou. Adaptive information processing: An effective way to improve perceptron predictors. *Journal of Instruction Level Parallelism*, 7, Apr 2005. <http://www.jilp.org/vol17>.
- [10] M. H. Lipasti and J. P. Shen. Exceeding the dataflow limit via value prediction. In *Proceedings of the 29th Annual International Symposium on Microarchitecture*, pages 226–237, 1996.
- [11] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen. Value locality and load value prediction. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 138–147, 1996.
- [12] G. H. Loh. Simulation differences between academia and industry: A branch prediction case study. In *International Symposium on Performance Analysis of Software and Systems*, pages 21–31, 2005.
- [13] S. McFarling. Combining branch predictors. Technical Report TN-36, Digital Western Research Laboratory, Jun 1993.
- [14] P. Michaud and A. Seznec. A comprehensive study of dynamic global history branch prediction. Technical Report RR-4219, INRIA-Rennes, Jun 2001.

- [15] T. Mudge, I.-C. Chen, and J. Coffey. Limits to branch prediction. Technical Report CSE-TR-282-96, Feb 1996.
- [16] J. T. Oplinger, D. L. Heine, and M. S. Lam. In search of speculative thread-level parallelism. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pages 303–313, 1999.
- [17] J. Singer and G. Brown. Return value prediction meets information theory. In *Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages*, 2006. To appear in ENTCS.
- [18] J. Smith. A study of branch prediction strategies. In *Proceedings of the 8th Annual Symposium on Computer Architecture*, pages 135–148, 1981.
- [19] C. Wilkerson and J. Stark. Special issue: Championship branch prediction. *Journal of Instruction Level Parallelism*, 7, Apr 2005. <http://www.jilp.org/vol7>.