Autocorrelation Analysis: A New and Improved Method for Branch Predictability Characterization

Jian Chen and Lizy K. John
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX, USA
chenjian@utexas.edu, ljohn@ece.utexas.edu

Abstract

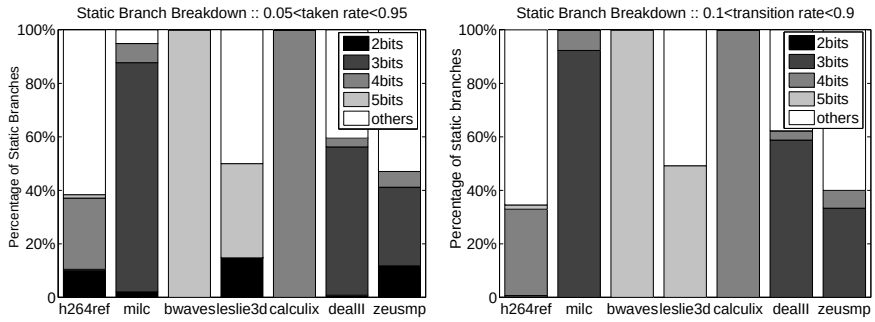
Branch predictability characterization not only helps to improve branch prediction but also helps to optimize predicated execution. Branch taken rate and branch transition rate have been proposed to characterize the branch predictability. However, these two metrics may misclassify branches with regular history patterns as hard-to-predict branches, causing an inaccurate and ambiguous view of branch predictability. In this paper, we utilize autocorrelation based analysis of branch history patterns and present two orthogonal metrics Degree of Pattern Irregularity (DPI) *and* Effective Pattern Length (EPL). *Unlike the existing taken rate or transition rate, DPI directly measures the regularity of the patterns in per-address branch history, and hence is more accurate in branch classification. On the other hand, EPL reveals the optimum branch history length for the easy-to-predict branches. The proposed metrics are evaluated with PAs, GAs, and Perceptron branch predictors, and the results show that on average, DPI improves the accuracy of hard-to-predict branch classification by up to* 17.7% *over taken rate and* 15.0% *over transition rate for the workloads in this study. It is also able to identify* 18.9% *more easy-to-predict branches compared with taken rate and* 12.8% *more compared with transition rate. The proposed metrics are valuable extension to the existing metrics for accurately characterizing branch predictability.*

1. Introduction

Classifying branches in terms of their predictability has been applied in many areas, including branch prediction, predicated execution, and benchmark cloning/synthesizing [12] etc. The accuracy of branch classification could significantly impact the outcomes of these applied areas. For example, branch predictors, especially hybrid branch predictors, leverage branch classification to improve prediction accuracy by steering branches to certain type of predictors most suitable for those branches [4]. A wrong classification of a branch may results in the degradation of prediction accuracy. The effectiveness of predicated execution also relies on accurately characterizing branch predictability. Predicated execution attempts to eliminate hard-to-predict branches via if-conversion during compilation [6]. However, if the if-converted branch can be easily and correctly predicted, predicated execution could cause significant performance degradation as it wastes time and energy in fetching and executing instructions from the false-path of the branch. Therefore, improving the accuracy of branch predictability characterization is an important step to improve the results of many applications.

The existing metrics for characterizing branch behaviors include branch taken rate and branch transition rate. Branch taken rate, proposed by Chang *et al*. [4], measures the taken frequency of a branch. Intuitively, branches with very high or very low taken rates are extremely biased toward one direction, therefore requires only a short branch history to produce high prediction accuracy. Whereas branches with near 50% taken rate may require longer branch history length to capture the execution pattern, and hence are considered to be hard-to-predict branches. On the other hand, branch transition rate, proposed by Haungs *et al*. [10], captures the frequency at which a branch changes direction between taken and not taken. Similar with the branch taken rate metric, branches with very high or very low transition rates are easy to predict, and branches with near 50% transition rate are hard to predict. However, the branch transition rate is able to prevent branches with history patterns like "101010" ("1" stands for taken and "0" for not-taken) from being falsely classified as hard-to-predict branches by branch taken rate, hence branch transition rate is more accurate and more widely used.

While both of these metrics help to reveal and characterize the branch behavior of workloads, they only provide an indirect and obscure view of the branch predictability, which causes problems from two aspects. First, these two metrics may not be able to identify all the easy-to-predict branches. In other words, some of the easy-to-predict branches may be misclassified as hard-to-predict by taken rate or transition rate. As shown in Figure 1(a), among the branches with taken



(a) (b)

Figure 1. Breakdown of static branches according to the number of history bits required to make a perfect prediction. Programs are from SPEC CPU2006 benchmark suite. (a) Breakdown of branches with *taken rate* between 0.05 and 0.95. (b) Breakdown of branches with *transition rate* between 0.1 and 0.9. The values of *taken rate* and *transition rate* used for this classification follow the ones used in [10].

rate between 0.05 and 0.95, which are typically considered to be the hard-to-predict branches, a large fraction of them can be perfectly predicted by using a branch history length less than 5 bits. The same is true for the branches with transition rate between 0.1 and 0.9. This observation is due to the fact that many branches may exhibit regular patterns with short periods. For instance, *milc* has a branch history pattern "110110110...", which means the branch has 0.667 taken rate and 0.667 transition rate, and is therefore considered to be hard-to-predict. Yet, a 3-bit history length is sufficient to make a perfect prediction for this branch. Neither taken rate nor transition rate is capable of differentiating these regular, easy-to-predict branches from the irregular and hard-to-predict ones, resulting in poor accuracy in identifying hard-to-predict branches. Second, both metrics suffer from the indirect and qualitative connection between their value and the corresponding history lengths. Specifically, for those branches classified as easy-to-predict, neither of these two metrics can provide a quantitative indication as to the number of history bits required to achieve high prediction rate. Therefore, it is difficult and inaccurate to estimate the branch predictor complexity from the branch predictability characterized with either of these two metrics.

To address these two issues of the existing metrics, this paper proposes to use *autocorrelation analysis* for branch predictability characterization. Autocorrelation analysis treats the history of a branch as a discrete sequence, and can reveal its pattern period by simply searching for the peak values in the autocorrelation of the branch history. Unlike branch taken rate or transition rate, autocorrelation analysis is able to identify the regular branch patterns at a much larger scope than taken rate or transition rate, hence can better differentiate hard-to-predict branches from easy-to-predict ones. In addition, it also enables the classification of the easy-to-predict branches in term of the branch pattern length, bridging the gap between the branch predictability and the predictor complexity. In particular, the contributions of this paper are as follows:

1. **Autocorrelation based Branch Classification:** Based on the autocorrelation analysis of the branch history, we propose two novel metrics for branch classification, i.e., the *Degree of Pattern Irregularity* (DPI), which measures the degree of the branch behavior deviating from regular branch pattern, and the *Effective Pattern Length* (EPL), which indicates the history length requirement for a conditional branch. We show that on average, DPI improves the accuracy of hard-to-predict branch classification by up to 17.7% over taken rate and 15.0% over transition rate. It also identifies 18.9% more easy-to-predict branches compared with taken rate and 12.8% more compared with transition rate. These proposed metrics are an important extension to the existing metrics in branch classification: taken rate examines the branch history *bit by bit*, and transition rate examines the history *two-bit by two-bit*; whereas DPI and EPL examine the branch history at a broader *pattern level*.

2. **Prediction Accuracy Estimation:** We present a first order analytical model to estimate the prediction accuracy of a per-address branch predictor based on the proposed DPI and EPL metrics. Experimental results show that the prediction rate estimated by the proposed model has an average error of 1.5% and a maximum absolute error of 7.1% compared with the simulated prediction rate. This new capability, for the first time, allows us to evaluate the impact of branch characteristics on the prediction accuracy without detailed simulation.

The rest of the paper is organized as follows. Section 2 gives the background about autocorrelation analysis. Section 3 describes the experiment methodology. Section 4 presents the DPI and EPL metrics, as well as the results of branch classification using these metrics. Section 5 lists the applications of the proposed metrics. Section 6 presents the prediction accuracy estimation model for PAs predictor. Section 7 shows the related work, and section 8 summarizes this paper.

2 Background

Autocorrelation is widely applied in signal processing and pattern recognition to find repeating patterns buried under noise. It is essentially the cross-correlation, i.e., sliding dot product, of a signal with itself [3]. Specifically, for a real-value discrete sequence of n elements $\{h(i)\}_{i=0}^{n}$, the autocorrelation of this sequence is calculated as:

$$R_{hh}(j) = \sum_{i=0}^{n} h(i)h(i-j) \quad (1)$$

where $j \in [0, n]$. In order to prevent undefined values outside the window $[0, n]$ from polluting the calculation, the sequence $\{h(i)\}_{i=0}^{n}$ is typically extended periodically to the left, creating a rotation effect in the coordinate window $[0, n]$ as the sequence slides to the right. As a result, the autocorrelation holds the following two properties [3]: a). Autocorrelation reaches its maximum value at the origin. In other words, for any delay i, $R_{hh}(0) \geq R_{hh}(i)$; b). If the discrete sequence is periodic with period T, its autocorrelation is also periodic with the same period T.

The combination of these two properties leads to the following inference: if the discrete sequence is periodic with period T, its autocorrelation reaches its maximum value at the delay T. This characteristic allows us to *identify the pattern period in a discrete sequence by simply searching for the peak value in the autocorrelation of the sequence*, which is one of the key advantages of applying autocorrelation in recognizing branch behavior patterns.

3 Experiment Methodology

In order to evaluate the proposed metrics, we use PIN [15], a dynamic x86 instrumentation tool, to instrument the workload and obtain the trace of conditional branches. This trace is then seamlessly fed to our branch analyzer, which is able to perform autocorrelation analysis on each static branch and simulate different types of branch predictors simultaneously.

The workloads of the experiment are composed of 27 programs from SPEC CPU2006 benchmark suite [1]. Each program is compiled to x86-ISA with base configurations. To reduce the simulation time, we use PinPoints [16], a SimPoint-like [9] tool built on top of PIN, to identify the representative simulation points. For each program, we simulate the dominant simulation points that cover 90% of the total weights, and each simulation point contains 100 million instructions. The aggregated statistics of the simulation points for each program are shown in Table 1. Note that two programs (*tonto*, *gamess*) from the benchmark suite are not included in this study because the PIN-based trace generator has some errors in these programs. Nevertheless, the listed programs are still representative for the entire benchmark suite according to benchmark similarity studied by Phansalkar *et al*. [17].

After developing the proposed metrics, we evaluate them using three different types of branch predictors to ensure that the observed properties are general and are not tied to a specific predictor. These branch predictors are: a per-address history predictor (PAs), a global two-level predictor (GAs) [21] and a global neural network predictor (Perceptron) [11], each with history length ranging from 1 to 16. (While there are many other branch predictors, we believe these three predictors provide enough confidence for the generality of the proposed metrics. Note that the goal of this paper is NOT to compete the existing branch predictors for higher prediction accuracy, but to create improved characterization metrics that could be applied in many areas.) For PAs and GAs, the size of Pattern History Table (PHT) is set to 64K entries, and remains constant across all history lengths, which means when history length is smaller than 16, the lower bits from branch address are combined together with branch history to index into the 64K-entry PHT. The branch history table (BHT) of PAs has 1024 entries, and also remains constant across all history lengths. To be consistent with PAs and GAs, the Perceptron predictor also contains 64K entries for the weights (which are used to dot-product with the branch history), with each weight 8-bit wide. In this paper, we only consider the conditional branches.

Table 1. Workloads and Branch Statistics

SPECCPU2006 (Points cover 90%weights)		Input	Static Branch #	Dynamic Branch #
CINT2006	mcf	ref	470	153353091
	gcc	166	7135	145078280
	perlbench	checkspam	46634	195356421
	bzip2	combined	2047	170176113
	gobmk	trevord	35005	161864828
	sjeng	ref	10830	208957597
	hmmer	retro	2923	224433233
	libquantum	1397	354	220161973
	h264ref	sss	13288	50744561
	omnetpp	ref	3870	74803965
	astar	rivers	918	83837997
	xalancbmk	xalanc	13426	244632666
CFP2006	bwaves	ref	565	63379429
	milc	ref	478	59817869
	zeusmp	ref	378	22270535
	gromacs	ref	321	23282041
	cactusADM	ref	763	3070633
	leslie3d	ref	3295	52895367
	namd	ref	1493	64360181
	soplex	ref	1596	64826986
	GemsFDTD	ref	241	12832561
	sphinx3	ref	5651	118766972
	calculix	ref	120	85279203
	dealII	ref	7700	186902861
	lbm	ref	48	9597189
	wrf	ref	2259	12007854
	povray	ref	1459	29840134

4 Autocorrelation Based Branch Classification

4.1 Branch History Autocorrelation

The unique properties of autocorrelation allow us to effectively identify the regular patterns inside a discrete sequence.

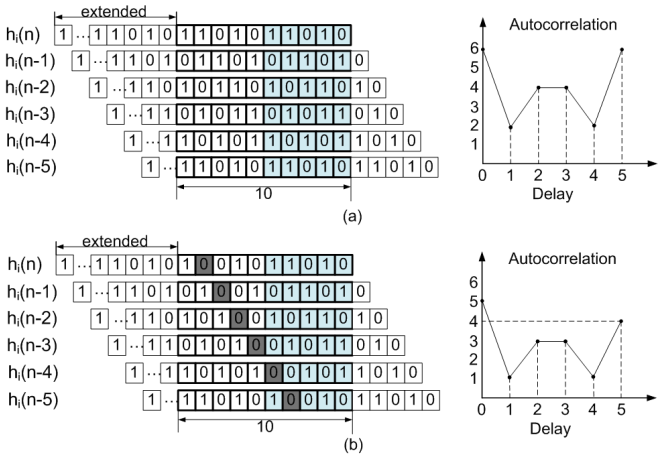


Figure 2. Autocorrelation Analysis for Branch History. (a) Autocorrelation for regular branch history with pattern "11010". The history has a length of 10, and has been extended. (b) Autocorrelation for irregular branch history. The deviating bit is highlighted with dark grey.

However, when it comes to branch behavior analysis, the discrete sequence becomes the branch history comprised of "0" (as not taken) and "1" (as taken). The autocorrelation of such history bit sequence preserves all the properties of a normal autocorrelation, yet it also brings two new implications pertaining to the context of branch prediction.

First, for the autocorrelation of periodic branch histories, the period of the history pattern indicates the required history length for accurately predicting the branch behavior. This is because by setting the history length to the pattern period, the branch predictor is guaranteed to be trained and accessed by the repetitive pattern, leading to near perfect prediction accuracy for these regular branches. The pattern period, on the other hand, can be easily identified by searching the autocorrelation of the branch history for the nearest maximum value around the origin. For example, as shown in Figure 2(a), the nearest maximum value around the origin of the autocorrelation occurs at the delay 5, which is the period of the regular pattern "11010", indicating the desired history length for this branch is 5. This value is essentially the x-coordinate difference between the origin and the nearest maximum value, which we refer to as the *Effective Pattern Length* (EPL).

Second, for the autocorrelation of irregular branch histories, the difference between the maximum value at the origin and the largest value off the origin reflects the amount of irregularity in the branch history. This can be understood by treating an irregular branch history as a regular branch history XORed with one or more history bits deviating from the regular pattern. Generally speaking, the more deviating history bits, the larger amount of irregularity in the history pattern. Meanwhile, the number of these deviating bits is reflected on

Pseudocode 1 Autocorrelation analysis of branch history

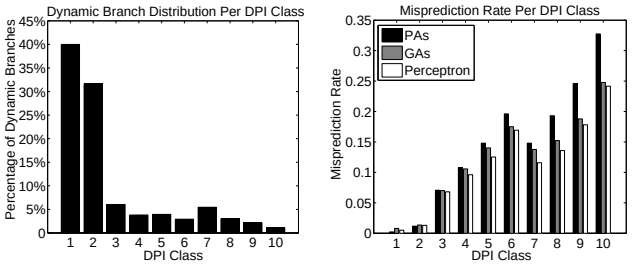
```
#define H /*the upper limit of pattern period*/
#define n /*dynamic accesses of branch i (n > 2 · H)*/
#define h_i(n) /*history pattern of branch i */
#define EPL_i /*EPL of branch i*/
#define DPI_i /*DPI of branch i*/

extend h_i(n) periodically until it's size is larger than n + H;
for j in ( 0 .. H)
{  R_hh(j) = Σ_{k=H}^{n+H} h(k)h(k − j); }
for j in ( 1 .. H) /* searching for the maximum off-origin value */
{  if( max < R_hh(j) )
     { max = R_hh(j); EPL_i = j; } }
DPI_i = (R_hh(0) − max)/n;
```

the difference between the two largest values of the autocorrelation. As shown in Figure 2(b), one bit highlighted with dark grey deviates from the periodic pattern, which causes the difference between the two largest autocorrelation values to be one. The irregularity measured by such difference is one of the main sources of branch misprediction. Therefore, the fraction of the irregularity over the total number of the branch dynamic accesses is the direct indicator of branch predictability, which we refer to as the *Degree of Pattern Irregularity* (DPI). Note that a regular branch history has a zero DPI value.

Both EPL and DPI of each conditional branch can be easily obtained with the autocorrelation analysis of branch history, as shown in Pseudocode 1. We only consider branches with a reasonably large number of dynamic accesses, preferably larger than twice the upper limit of the investigated pattern period. This is to prevent the artifacts created by the periodic extension of the history length from being mistakenly identified as history pattern by autocorrelation analysis. The complexity of the autocorrelation analysis involves computation and storage requirements. The calculation of autocorrelation appears to have $(n+1)\cdot H$ multiply-accumulate operations, where n is the length of branch history, and H is the specified upper limit of history pattern period. However, since branch history only consists of 0's and 1's, the multiplication can be replaced with simple logic AND operation. On the other hand, since we are only interested in the autocorrelation between 0 and H, it is not necessary to store the entire history of a branch before calculating the autocorrelation. In fact, the autocorrelation can be calculated based on slices of branch history, with each slice size less than H+1. It can be understood by transforming equation (1) to the following form:

$$R_{hh}(j) = \sum_{k=H}^{n+H} h(k)h(k-j) = \sum_{k=H}^{H+H} h(k)h(k-j) + \sum_{k=2H+1}^{2H+H} h(k)h(k-j) + .. + \sum_{k=m\cdot H+1}^{n+H} h(k)h(k-j)$$

where $0 \le j \le H$, and $m\cdot H < n < (m+1)\cdot H$. The



(a) (b)

Figure 3. (a) Percentage of dynamic branches per DPI class. (b) Prediction accuracy of the branches in each DPI class for PAs, GAs, and Perceptron.

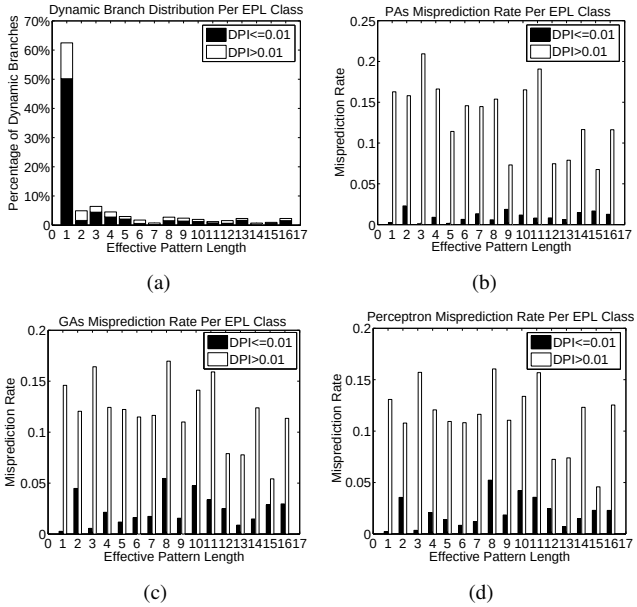
calculation of each item in the equation requires the storage of current history slice and the immediate previous history slice. This storage can be organized as a two-entry FIFO, with each entry size of $(H + 1)$ bits, resulting in a total storage requirement of $2(H + 1)$ bits for each static branch. As we finish calculating the autocorrelation of one history slice and move to the next, the new history slice is pushed in the FIFO. Together with the old history slice, we are able to calculate the autocorrelation of the new history slice. Compared to the storage requirements of taken rate (1 bit per static branch) and transition rate (2 bits per static branch), autocorrelation analysis requires more storage space, yet its impact on the profiling speed is insignificant as long as H is within a reasonable range.

With the two metrics DPI and EPL, we are able to classify branches from two orthogonal aspects simultaneously, and gain new insights into the branch behaviors in terms of their predictability and their implications on the requirement of branch history lengths.

4.2 Branch Classification

We now investigate the branch behaviors by classifying branches according to the proposed DPI and EPL metrics. The results shown in this subsection are based on the aggregated branch behaviors of all programs listed in Table 1.

In Figure 3(a), we classify the branches into 10 groups in terms of their DPI values. Class 1 has DPI value 0, representing the branches with regular history pattern. Class 2 to 6 have DPI values covering the ranges of (0,0.01], (0.01,0.02], (0.02,0.03], (0.03,0.04], (0.04,0.05], respectively; and class 7 to 10 have DPI values with the ranges of (0.05-0.10], (0.10-0.15], (0.15-0.20], (0.20-1] respectively. As shown in the figure, 40.0% of the total dynamic conditional branches fall in class 1, and 31.6% of them fall in class 2, making a total of 71.6% of dynamic branches with $DPI \leq 0.01$. The occupancies of the other classes are significantly lower, with each class less than 6.0%, meaning that most branches exhibit very small irregularity. Figure 3(b) shows the misprediction rate of the branches in each DPI class for PAs, GAs, and Perceptron predictors. As expected, there is an overall trend that the misprediction rate increases as the branch DPI increases. This trend holds true for all three different types of branch predictors, which demonstrates that DPI is an appropriate metric for branch predictability. More importantly, this figure also shows that the misprediction rates of the branches in DPI class 1 and 2 are drastically smaller than that of the branches in the rest DPI classes, which means branches with DPI less than 0.01 are the easy-to-predict branches. In addition, for these branches, there is very little difference between the misprediction rates achieved by different predictors. However, for branches with DPI larger than 0.01, PAs predictor consistently exhibits largest misprediction rate, followed by GAs and perceptron predictors. This is because the irregularity of the branches in these classes mainly comes from the correlation among multiple different branches, which can be captured by the global history [7], making global predictors more preferable over the per-address predictor for these branches. As a result, DPI allows us to classify the branch predictability in a clear and coherent way: branches with DPI less than 0.01 are the easy-to-predict branches, and are suitable for any type of predictors, preferably, the per-address predictor; whereas branches with DPI larger than 0.01 are the hard-to-predict branches, and are suitable for global branch predictors.



(a) (b) (c) (d)

Figure 4. (a) Dynamic branch classification according to EPL. (b) PAs misprediction rate for each EPL class. (c) GAs misprediction rate for each EPL class. (d) Perceptron misprediction rate for each EPL class.

In Figure 4(a), we classify the branches according to the EPL values ranging from 1 to 16. Like the branch distribution in DPI, most of the branches have small effective pattern length, and 62.4% of them have EPL of 1. Among these branches with EPL of 1, 80.4% of them are with DPI no larger

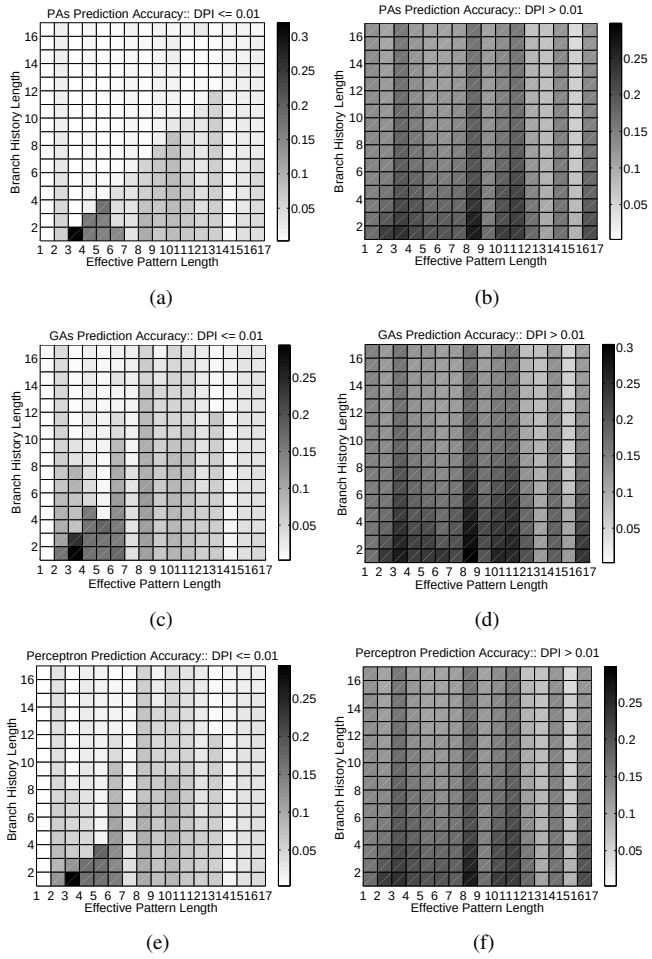


Figure 5. Colormap of prediction accuracy. The values associated with the colors are the misprediction rate.

than 0.01, meaning that these branches can be predicted accurately with very short branch history length. Figure 4(b)-(c) illustrate the optimal misprediction rates under PAs, GAs, and Perceptron predictors for the branches with different effective pattern lengths. These figures provide an additional view of the distribution of misprediction rate across the spectrum of EPL. Moreover, it also shows that EPL is irrelevant with minimum achievable misprediction rate for either regular or irregular branches. However, EPL does establish a connection between the easy-to-predict branches and their optimal history lengths, as shown in the colormap in Figure 5(a) (the darker the color, the higher the misprediction rate). When the history length of PAs reaches the effective pattern length of an easy-to-predict branch ($DPI <= 0.01$), the misprediction rate drops dramatically, meaning that EPL represents the optimal branch history length in PAs. Because of that, one can observe a clear diagonal edge in this colormap. However, when it comes to global predictors, such as GAs or perceptron predictor, the diagonal edge becomes blurred, as shown in Figure 5(c) and Figure 5(e). Yet, by setting the history length to EPL, one can still get a reasonable, though not necessary optimal, misprediction rate for the branches in that EPL class. On the contrary, for the hard-to-predict branches ($DPI > 0.01$), the optimal branch history length is irrelevant to EPL. As shown in Figure 5(b), 5(d) and 5(f), there is no clear cut value for optimal history length, and a large branch history length, preferably no less than 10, is always desirable for all three types of predictors. The relationship between EPL and the optimal history length of an easy-to-predict branch opens the possibility to precisely tailor the complexity of branch predictors.

4.3 Comparison with Conventional Metrics

Now that we have demonstrated that DPI is a useful metric for branch classification, we compare this metric with the conventional taken rate and transition rate metrics.

Figure6(a) shows the percentage of the branches classified as easy-to-predict branches among the branches with prediction rate larger than 95%. (In this paper, branches with more than 95% prediction rate are regarded as truly easy-to-predict, which is in line with the study by Kim *et al*. [13].) The higher the percentage, the more effective the metric in identifying the easy-to-predict branches. As shown in this figure, DPI consistently yields larger percentage than transition rate or taken rate across all three types of branch predictors, meaning that DPI is able to identify more truly easy-to-predict branches than taken rate or transition rate. In particular, for PAs prediction, DPI covers 12.1% more dynamic branches than taken rate and 8.7% more than transition rate. On the other hand, to evaluate the false-positives in branch classification, we also measure the percentages of the branches with prediction rate larger than 95% over the branches classified as easy-to-predict, as well as the percentages of the branches with prediction rate less than 95% over the branches classified as hard-to-predict. The higher the percentage, the more accurate the metric is, since less branches would be misclassified. As shown in figure 6(b), taken rate has slightly higher percentage than the other two metrics, meaning that taken rate has the least false-positives in classifying easy-to-predict branches. DPI has the lowest accuracy in classifying easy-to-predict branches, and its accuracy is less than 3.8% smaller than that of taken rate. However, considering the significant improvement in the effectiveness of classifying easy-to-predict branches, DPI is still able to correctly identify significant amount of truly easy-to-predict branches that otherwise would have been misclassified as hard-to-predict branches by taken rate, improving the accuracy in classifying hard-to-predict branches. As shown in Figure 6(c), DPI improves the accuracy of the hard-to-predict branch classification by up to 17.7% over taken rate, and 15.0% over transition rate. Overall, DPI, as a metric, is more effective and accurate in branch classification than the existing metrics.

Figure 7 further shows the comparison between the metrics on a per-program basis. Figure 7(a) shows the dynamic

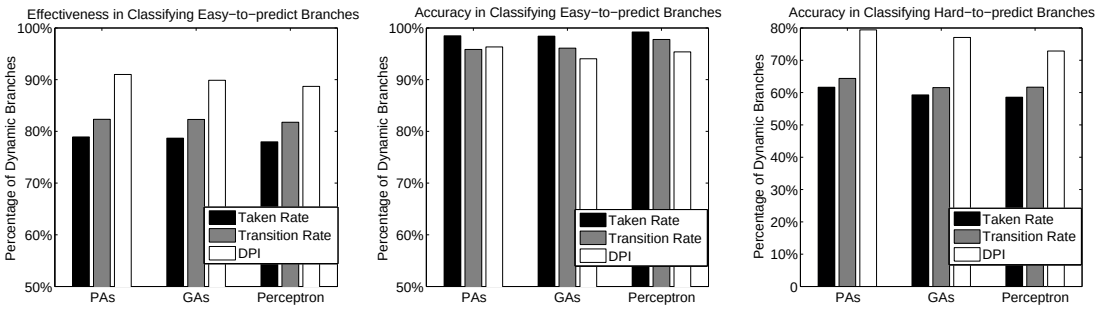


Figure 6. (a) The branches classified easy-to-predict among the branches with prediction rate $>$ 95%. The easy-to-predict branches are classified by taken rate $\in [0, 0.05) \bigcup (0.95, 0.1]$, transition rate $\in [0, 0.1) \bigcup (0.9, 0.1]$, or DPI$\in [0, 0.01]$. (b) The percentage of the branches with prediction rate $> 95\%$ among the branches classified as easy-to-predict. (c) The percentage of the branches with prediction rate $< 95\%$ among the branches classified as hard-to-predict.

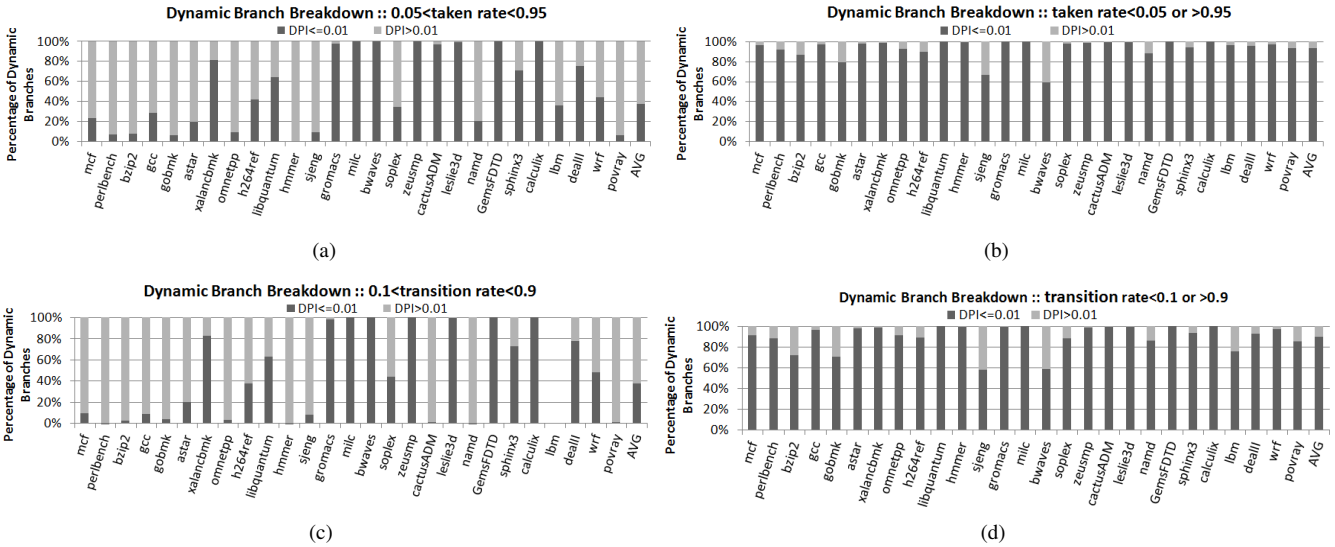


Figure 7. (a) Percentages of branches with DPI≤ 0.01 among the branches with $0.05 <$taken rate< 0.95. (b) Percentages of branches with DPI≤ 0.01 among branches with taken rate< 0.05 or > 0.95. (c) Percentages of branches with DPI≤ 0.01 among the branches with $0.1 <$transition rate< 0.9. (d) Percentages of branches with DPI≤ 0.01 among the branches with transition rate < 0.1 or > 0.9.

branch breakdown for branches with taken rate between 0.05 and 0.95. These dynamic branches are considered to be relatively hard-to-predict according to the taken rate metric [10]. However, a large fraction of these branches are classified as easy-to-predict according to the DPI metric. In particular, for *milc*, *bwaves*, *zeusmp*, *leslie3d*,*GemsFDTD* and *calculix*, all of the conditional branches that are initially classified as hard-to-predict according to taken rate are classified as easy-to-predict according to DPI. These branches have regular history patterns, and can reach near-zero misprediction rate with PAs predictor, which means they have been indeed misclassified by the taken rate. On average, we observe 37.7% of the branches with taken rate between 0.05 and 0.95 are misclassified as hard-to-predict. Similarly, for the transition rate, we observe an average of 36.5% of the branches with transition rate between 0.1 and 0.9 are misclassified as hard-to-predict according to DPI, as shown in the Figure 7(c). Figure 7(b) further illustrates how much DPI is in agreement with the taken rate when it comes to the classification of easy-to-predict branches. We observe an average 6.3% of the branches with taken rate larger than 0.95 or smaller than 0.05 are classified as hard-to-predict according to DPI. For transition rate, the number increases to 9.9%. This is inline with the previous observation that taken rate is more accurate in classifying easy-to-predict branches than transition rate. Overall, 71.6% of the total conditional branches are classified as easy-

to-predict with DPI, 60.2% with taken rate, and 63.5% with transition rate. That gives us 18.9% improvement over the taken rate, and 12.8% improvement over the transition rate. The largest observed improvement happens in program *milc*, where taken rate or transition rate can only identify 14.2% of the dynamic branches that are classified as easy-to-predict by DPI. In other words, DPI identifies 7.0 times more easy-to-predict branches than taken rate or transition rate does for *milc*. The reason that DPI is superior in branch classification is that it has a broader view of branch history when characterizing the branch behaviors. In fact, taken rate examines the branch history *bit by bit*, and transition rate examines the history *two-bit by two-bit*; whereas DPI examines the branch history at a *broader pattern level*. In that sense, DPI is a *superset* of the taken rate and transition rate, hence is able to achieve better branch classification results than the existing metrics.

5 Applications

As an important extension to the existing metrics, the proposed DPI and EPL metrics significantly improve the quality of branch classification, and can be applied in the fields where the conventional branch classification metrics are used. These fields include, but not limited to, the following.

Branch Prediction: The hybrid branch predictor is an important type of branch predictor that is able to achieve high prediction accuracy. Yet, the design space exploration for hybrid branch predictors is non-trivial. Haungs *et al*. suggests to use branch taken rate and transition rate to accelerate the design space exploration process [10]. Our proposed DPI and EPL metrics could also be applied in this area. More importantly, EPL provides a straightforward indication of the proper history length for each branch.

Predication: Predication attempts to improve the performance by eliminating hard-to-predict branches using if-conversion. Its benefit largely depends on the accuracy in identifying those branches. Compilers typically identify the candidates for predication by profiling the program and collecting the branches with taken rate near 50% [8] or with low prediction accuracy for a certain branch predictor [13][19]. Since the proposed DPI metric has higher accuracy in identifying hard-to-predict branches than taken/transition rate, it can replace taken rate or transition rate in identifying the hard-to-predict branches and improve the effectiveness of predication.

Benchmark Cloning and Synthesizing: Recently, benchmark cloning/synthesizing has become attractive as it miniaturizes the prohibitively large and ever-expanding benchmark programs meanwhile protects the proprietary workload information [12]. The quality of benchmark cloning relies on faithfully characterizing the control flow predictability of the original programs. To do so, taken rate was initially used by Bell *et al*. [2], and later replaced by transition rate in order to improve the quality of control flow cloning [12]. We argue that using DPI could further improve the quality of benchmark cloning since DPI metric has less false-positives in branch classification than existing metrics.

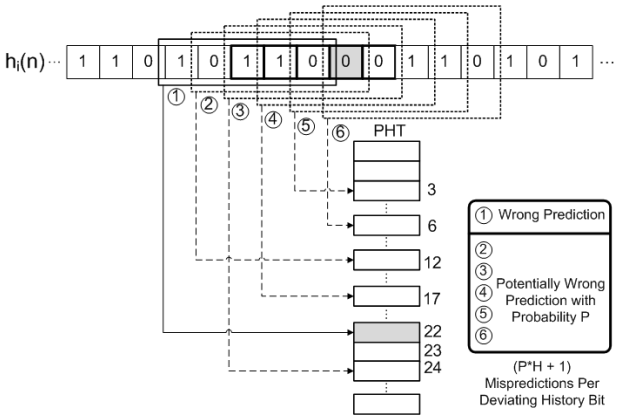


Figure 8. Misprediction mechanism on a Per-address Predictor with 5-bit history length.

The proposed metrics could also be applied in the fields of benchmark subsetting and program similarity analysis[17], as well as program-core mapping for heterogeneous multi-core processors [5]. Besides these existing application areas, the combination of DPI and EPI also opens the new possibility to estimate the prediction accuracy of a per-address predictor based on branch characteristics.

6 Case Study: Prediction Accuracy Estimation

As a case study, this section presents a first-order analytical model to estimate the prediction accuracy of a PAs predictor using the proposed EPL and DPI metrics. Such model decouples the estimation of branch misprediction from simulation, and is very useful in analytical performance modeling.

As mentioned previously, DPI measures the fraction of the history bits deviating from regular history pattern. These deviating bits are the major source of the misprediction in the per-address predictor, because the predictor is trained with repetitive patterns and will cause misprediction every time it runs into the deviating bit. For example, Figure 8 shows a branch history sequence with repetitive pattern "10110", and a deviating bit highlighted with grey color. The predictor will mispredict the deviating bit since the entry in PHT learns the pattern and predicts the next outcome should be "1" instead of "0". The deviating bit then enters branch history buffer, which will point to a PHT entry never visited by this branch before. This entry may be a cold entry that has never been visited by any other branches, or may be an entry already trained by other branches. Either way, there is a certain amount of probability that the prediction of the next outcome could be wrong. Note that such situation remains until the deviating bit has been shifted outside the history buffer if the history buffer is smaller than EPL of that branch. In this case, one deviating history bit could cause $P \cdot h + 1$ mispredictions (h stands for the branch history length of the predictor and P for the probability of the misprediction). Therefore, we can estimate the

number of mispredictions with the following equation:

$$Miss_{case1} = \sum_i DPI_i \cdot N_i \cdot (P_i \cdot h + 1) \qquad (2)$$

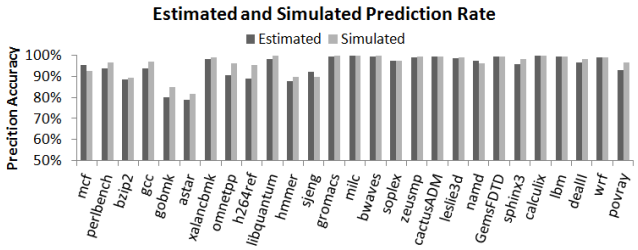
where N_i is the dynamic access number of branch i and P_i is the probability of the misprediction caused by the deviating bit. This equation gives reasonable estimation of the number of mispredictions only when h, though smaller than EPL of the branch, is still relatively large. Extremely small branch history length makes the predictor trained by only a small fraction of the history pattern, causing a large number of aliasing and interference unable to be modeled with this equation. In this study, we set h to be larger than 6. On the other hand, if the history buffer size is larger than the EPL of the branch, once the deviating bit has been shifted more than EPL bits in the buffer, the misprediction probability of the next outcomes of the branch would drop significantly, as shown in Figure 5(a). This is because the most recent history bits are now filled with basic regular pattern, which statistically dominates the prediction of the next outcome. Therefore, the estimation of the misprediction number in this case can be made by the following:

$$Miss_{case2} = \sum_i DPI_i \cdot N_i \cdot (P_i \cdot EPL_i + 1) \qquad (3)$$

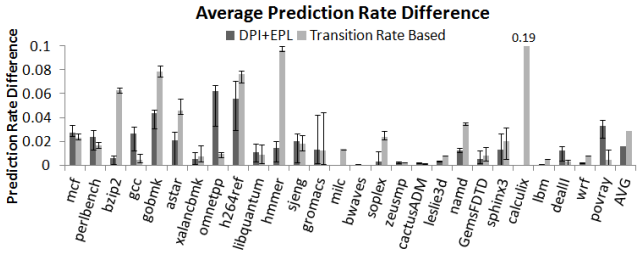
Let $U(n)$ be the unit step function, which equals 0 when $n < 0$, and 1 when $n \geq 0$. The total number of misprediction can be written as follows:

$$Miss_t = \sum_i DPI_i \cdot N_i \cdot (P_i \cdot EPL_i + 1) \cdot U(h - EPL_i) + \sum_i DPI_i \cdot N_i \cdot (P_i \cdot h + 1) \cdot U(EPL_i - h) \qquad (4)$$

Now the problem comes down to determine the probability of misprediction P_i. Although this probability is dependent on the nature of interference (constructive or destructive), if we treat the interference as unbiased random process, 50% is a statistically reasonable estimate for P_i. Therefore, with P_i being set to 50%, we are able to estimate the branch misprediction rate of PAs predictors using equation (4). Figure 9(a) shows the comparison between the simulated prediction rate and the estimated one at history length 15. The estimated prediction rate closely matches the simulated one. Figure 9(b) shows the average difference between the simulated prediction rate and the estimated one with history length ranging from 6 to 15. The smallest observed error is 0, and it occurs in *milc*. This is because *milc* has regular branch patterns that can be perfectly predicted with PAs predictor; meanwhile, regular patterns have zero DPI, leading to zero misprediction in our model. In addition, we also estimate the prediction rate using a heuristic based on transition rate. This heuristic assumes the branches with transition rate between 0.4 and 0.6 are predicted with 50% accuracy, and other branches are predicted



(a)



(b)

Figure 9. (a) Comparison of estimated and simulated prediction rate (History length 15). (b) Average prediction rate difference. The error bar shows the min and max difference in prediction rate (History length from 6 to 15).

perfectly. As shown in Figure 9(b), the estimation error of this heuristic varies significantly between programs, from less than 1% to 19%, indicating this heuristic is not stable. This is because for some programs, e.g. *calculix*, a large amount of branches with transition rate between 0.4 and 0.6 can be easily predicted, but this heuristic treats them as hard-to-predict branches. In contrast, the accuracy of our DPI-EPL-based model is not only higher, but more consistent across different programs. The average prediction rate error is 1.5%, about half of that of the transition-rate-based heuristic. Overall, the proposed analytical model estimates the prediction rate of PAs predictor with reasonable accuracy, and automatically lends itself to micro-architecture independent performance modeling. It also demonstrates the capability of EPL and DPI in understanding and characterizing the branch behavior.

7 Related Work

Our work is most related to the work done by Chang *et al*. [4] and by Haungs *et al*. [10]. Chang *et al*. [4] first applied branch classification to improve the performance of branch predictors. They used taken rate to differentiate branches and suggested to use static predictors for the extremely biased branches. Haungs *et al*. [10] proposed to use branch transition rate to classify branches. They demonstrated that the transition rate is able to discover easy-to-predict branches that otherwise would be misclassified by taken rate. However, taken rate and transition rate may not able to discover branches with regular behavior patterns that are easy-to-predict. Our proposed DPI

metric has a much broader view of branch behavior, and hence can characterize branch predictability more accurately.

Evers *et al*. [7] categorized the per-address branch predictability into three classes: loop-type branches, branches with repeating patterns, and branches with non-repeating patterns. They classified the branches by using a set of branch predictors to inspect the branch history patterns, but no general metrics were developed in their work. Our work shares some common grounds as both of these work recognize the importance of history pattern in classifying branch predictability. However, we develop a general metric to measure the branch pattern regularity, which provides a more efficient, accurate and coherent interface for branch classification.

Kim and Tyson [14] employed the dynamic working set characteristics of branches to classify branches. This working set metric attempts to characterize the path-based global branch correlation, whereas our proposed metrics target at measuring per-address branch patterns.

Thomas *et al*. [20] identified correlated branches by using runtime dataflow information, then used these correlated branches to improve prediction accuracy. Sazeides *et al*. [18] followed this line of research and used the subset of history most highly correlated with a given branch to improve predictor accuracy. These work targets at improving prediction accuracy by exploiting the branch correlation characteristics. In contrast, this paper does NOT attempt to devise a new scheme for prediction accuracy improvement, but rather to develop improved metrics for branch predictability characterization, which could be applied in a broad set of areas.

8 Conclusions

Branch predictability characterization is an important technique to understand branch behaviors. It is widely applied in the fields such as predicated execution and benchmark cloning. Based on the autocorrelation analysis of branch history patterns, this paper presents two new metrics, *Degree of Pattern Irregularity* (DPI) and *Effective Pattern Length*(EPL), for branch predictability characterization. Unlike existing taken rate or transition rate metrics, DPI directly measures the regularity of the patterns in per-address branch history, and hence is able to identify more easy-to-predict branches and significantly improve the accuracy of the classification of hard-to-predict branches. On the other hand, EPL reveals the optimum branch history length for the easy-to-predict branches. The combination of DPI and EPL provides a deeper insight into the branch behavior and allows us to accurately estimate prediction rate for a per-address predictor based on these metrics without detailed simulation. Our experiments show that DPI improves the classification of easy-to-predict branches by 18.9% compared with taken rate and 12.8% compared with transition rate for the workloads in this study. It also improves the accuracy of hard-to-predict branch classification by up to 17.7% over taken rate and 15.0% over transition rate. Overall, these proposed metrics significantly improve the quality of branch classification, and are valuable extension of the existing metrics.

References

[1] Spec cpu2006 benchmark suit. In *http://www.spec.org*.

[2] R. H. Bell, Jr. and L. K. John. Improved automatic testcase synthesis for performance model validation. In *ICS'05*, pages 111–120, 2005.

[3] E. O. Brigham. *The Fast Fourier Transform*, chapter 13. Prentice-Hall, 1974.

[4] P.-Y. Chang et al. Branch classification: a new mechanism for improving branch predictor performance. In *MICRO'94*, pages 22–31, 1994.

[5] J. Chen et al. Efficient program scheduling for heterogeneous multi-core processors. In *DAC'09*, pages 927–930, 2009.

[6] Y. Choi et al. The impact of if-conversion and branch prediction on program execution on the intel® itanium processor. In *MICRO'01*, pages 182–191, 2001.

[7] M. Evers et al. An analysis of correlation and predictability: what makes two-level branch predictors work. In *ISCA '98*, pages 52–61, 1998.

[8] M. U. Farooq et al. Compiler controlled speculation for power aware ilp extraction in dataflow architectures. In *HiPEAC'09*, pages 324–338, 2009.

[9] G. Hamerly et al. Simpoint 3.0: Faster and more flexible program analysis. In *Journal of Instruction Level Parallelism*, volume 7, pages 1–28, 2005.

[10] M. Haungs et al. Branch transition rate: a new metric for improved branch classification analysis. In *HPCA'00*, pages 241–250, 2000.

[11] D. Jimenez and C. Lin. Dynamic branch prediction with perceptrons. In *HPCA'01*, pages 197–206, 2001.

[12] A. Joshi et al. Automated microprocessor stressmark generation. In *HPCA'08*, pages 229–239, 2008.

[13] H. Kim et al. 2D-profiling: Detecting input-dependent branches with a single input data set. In *CGO'06*, pages 159–172, 2006.

[14] S. P. Kim et al. Analyzing the working set characteristics of branch execution. In *MICRO'98*, pages 49–58, 1998.

[15] C.-K. Luk et al. Pin: building customized program analysis tools with dynamic instrumentation. In *PLDI'05*, pages 190–200, 2005.

[16] H. Patil et al. Pinpointing representative portions of large intel®; itanium programs with dynamic instrumentation. In *MICRO '04*, pages 81–92, 2004.

[17] A. Phansalkar et al. Analysis of redundancy and application balance in the spec cpu2006 benchmark suite. In *ISCA'07*, pages 412–423, 2007.

[18] Y. Sazeides et al. The significance of affectors and affectees correlations for branch prediction. In *HiPEAC'08*, pages 243–257, 2008.

[19] B. Simon et al. Incorporating predicate information into branch predictors. In *HPCA'03*, pages 53–64, 2003.

[20] R. Thomas et al. Improving branch prediction by dynamic dataflow-based identification of correlated branches from a large global history. In *ISCA'03*, pages 314–323, 2003.

[21] T.-Y. Yeh and Y. N. Patt. A comparison of dynamic branch predictors that use two levels of branch history. In *ISCA'93*, pages 257–266, 1993.