Improving Branch Prediction by considering Affectors and Affectees Correlations

Yiannakis Sazeides[1], Andreas Moustakas[2,⋆], Kypros Constantinides[2,⋆], and Marios Kleanthous[1]

[1] University of Cyprus, Nicosia, CYPRUS/HiPEAC
[2] University of Michigan, Ann Arbor, USA

Abstract. This work investigates the potential of *direction*-correlations to improve branch prediction. There are two types of *direction*-correlation: *affectors* and *affectees*. This work considers for the first time their implications at a basic level. These correlations are determined based on dataflow graph information and are used to select the subset of global branch history bits used for prediction. If this subset is small then *affectors* and *affectees* can be useful to cut down learning time, and reduce aliasing in prediction tables. This paper extends previous work explaining why and how correlation-based predictors work by analyzing the properties of *direction*-correlations. It also shows that branch history selected based on *direction*-correlations improves the accuracy of the limit and realistic conditional branch predictors, that won at the recent branch prediction contest, by up to 30% and 17% respectively. The findings in this paper call for the investigation of predictors that can efficiently learn correlations that may be non-consecutive (i.e. with holes between them) from long branch history.

1 Introduction

The ever growing demand for higher performance and technological constraints drive for many years the computer industry toward processors with higher clock rates and more recently to multiple cores per chip. Both of these approaches can improve performance but at the same time can increase the cycle latency to resolve an instruction, the former due to deeper pipelines and the latter due to inter-core contention for shared on-chip resources. Longer resolution latency renders highly accurate conditional branch prediction a necessity because branch instructions are very frequent in programs and need to be resolved as soon as they are fetched in a processor to ensure continuous instruction supply.

Today, after many years of branch prediction research and the two recent branch prediction championship contests [1, 2], the accuracies of the state of the art predictors are high but far from perfect. For many benchmarks the O-GEHL and L-TAGE predictors[3] [3, 4] have more than five misses per thousand

⋆ The author contributed to this work while at the University of Cyprus

[3] O-GEHL won the best practice award in the 2004 branch prediction contest and L-TAGE won the realistic track of the 2006 contest

instructions. Such a rate of misprediction, depending on the average branch resolution latency and other execution overheads, can correspond to a substantial part of the total execution time of a program. A recent study shows that the misprediction overhead for an 8-way out-of-order processor using an 8KB O-GEHL predictor, for SPECINT CPU2000 benchmarks, can be up to 50% and on the average 17% of the execution time [5]. Consequently, we believe there is still a need to further improve prediction accuracy. The challenge is to determine how to achieve such an improvement.

In the seminal work by Evers et al. [6] it is shown that choosing more selectively the correlation information can be conducive for improving branch prediction. In particular, using an exhaustive search is determined for a gshare [7] predictor that only a few, not necessarily consecutive, of the most recent branches are sufficient to achieve best prediction accuracy. Furthermore, is demonstrated that a correlation may exist between branches that are far apart. The same work, introduces two reasons for why global history correlation exists between branches: *direction* and *in-path* correlation, and divides *direction*-correlations into *affectors* and *affectees.* [4] These various types of correlations can mainly be derived by considering the data and control flow properties of branches. These causes of correlation are only discussed qualitatively in [6] to explain what makes two-level branch predictors work, no measurements of their frequency or quantification of their importance are given.

The work by [6] motivated subsequent prediction research with goal the selective correlation from longer global history. One of the most notable is perceptron based prediction [9] that identifies, through training, the important history bits that a branch correlates on. The success of perceptron based prediction provides a partial justification for the claims by [6] for the importance of selective correlation. However, it was never established that the dominant perceptron correlations correspond to *direction* or *in-path* correlation and therefore remains uncertain if indeed such correlations are important or whether predictors exploit them efficiently.

One other interesting work by [8] investigated the usefulness of *affectors* branches, one of the types of *direction*-correlation introduced by [6] . In [8] the affector branches are selected dynamically from the global history using data dependence information and are used to train an overriding tagged predictor when a baseline predictor performs poorly. The experimental analysis, for specific microarchitectural configurations and baseline predictors, show that this idea can potentially improve both prediction accuracy and performance. This work also provides the first concrete evidence that the *direction*-correlation is an important information for prediction. However, [8] did not examine the importance of *affectees.*

In this paper we investigate the significance for improving branch prediction accuracy using the two types of *direction*-correlation: affectors and affectees.

[4] In [6] the two types of *direction*-correlations are defined but not named. In [8] they referred to them as affectors and forerunners. In this work, for symmetry we decided to name the forerunners as affectees.

Our analysis is done at a basic level because it does not consider implementation issues for detecting affectors and affectees correlations. The primary objectives of this paper is to establish the extent that state of the art predictors learn *direction*-correlations, and determine how precise the detection of *direction*-correlations needs to be for best accuracy. Our evaluation uses the two winning predictors of the limit and realistic track of the recent championship prediction [2] and considers their accuracy when they use the global history as is versus the global history packed [8] to "ignore" the positions with no *direction*-correlation.
Contributions
The key contributions and findings of this paper are:

- A framework that explains why some branches are more important than others to correlate on. The framework can be used to precisely determine these branches based on architectural properties without regard to implementation.
- An experimental analysis of the potential of *direction*-correlations to improve branch prediction accuracy.
- An investigation of the position and the number of *direction*-correlations reveals that their behavior varies across programs. Also, is very typical for programs to have branches with the number of correlations ranging from few branches to several hundreds. The correlations can be clustered together but also be very far apart, i.e. correlations may not be consecutive and can have holes between them. Affectees are found to be more frequent than affectors.
- Demonstrate that for best accuracy both affectors and affectees correlations are needed. Their use can provide accuracy improvements of up to 30% for the limit predictor, and 17% for the realistic predictor
- Show that it is crucial to include in branch history *direction*-correlations that are detectable by tracking dependences through memory.
- Establish a need to further study predictors that can learn correlation patterns with and without holes from long branch history.

The remaining of the paper is organized as follows. Section 2 defines what affectors and affectees correlations are and discusses parameters that influences the classification of a branch as correlating. Section 3 presents the experimental framework. Section 4 discusses the experimental results of this study and establishes the significance of affectors and affectees. Section 5 discusses related work. Finally, Section 6 concludes the paper and provides directions for future work.

2 Affectors and Affectees

This section defines what affector and affectee branches are and provides intuition as to why these are important branches to select for correlation. It also discusses how the treatment of memory dependences influence the classification of a branch as an affector or affectee of another branch. Finally, a discussion is presented on how this correlation information can be used for prediction. Part of this discussion is based on earlier work [6, 8].

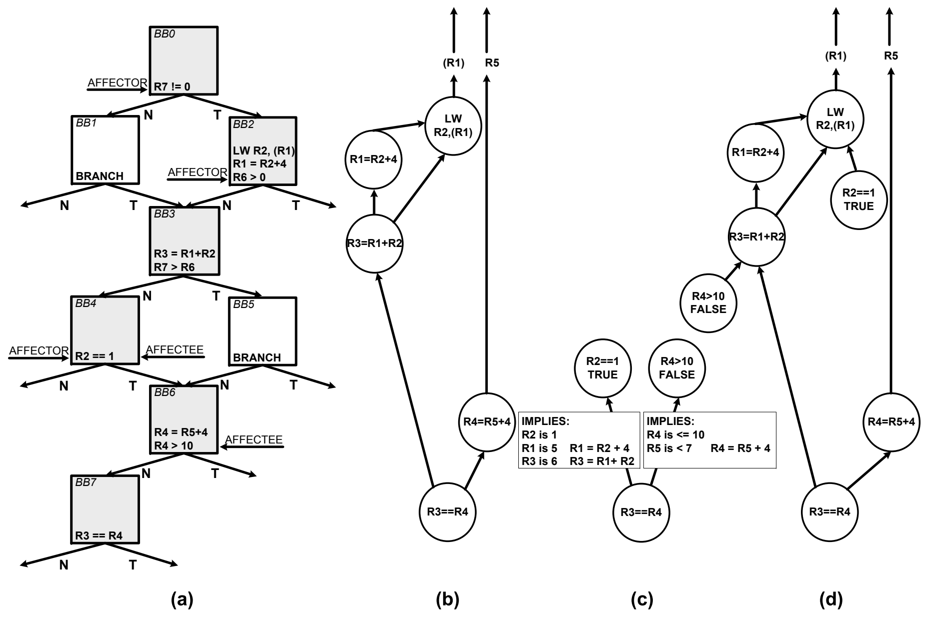

Fig. 1. (a) Example control flow graph, (b) affector graph, (c) affectee graph and (d) affector plus affectee graph

2.1 Definitions and Intuition

Affectors: A dynamic branch, A, is an affector for a subsequent dynamic branch, B, if the outcome of A affects information (data) used by the subsequent branch B. Affectors are illustrated using the example control flow graph in Fig. 1.a. Assume that the (predicted) program order follows the shaded basic-blocks and we need to predict the branch in the basic-block 7. The affector branches are all those branches that steer the control flow to the basic-blocks that contain instructions that the branch, in basic-block 7, has direct or indirect data dependence. In our example, these correspond to the branches in basic-blocks *BB0, BB2* and *BB4*. Effectively, the selected affector branches can be thought of as an encoding of the data flow graph leading to the branch to be predicted (this affector data flow graph is shown in Fig. 1.b). Predictors may benefit by learning affector correlations because when branches repeat with the same data flow graph they will likely go the same direction. Furthermore, affector correlations use a more concise branch history to capture the data flow graph leading to a branch and thus reduce learning time and table pressure for training a predictor.
Affectees: A dynamic branch, A, is affectee of a subsequent dynamic branch, B, if A is testing the outcome of an instruction C that can trace a data dependence to an instruction D in the data flow graph leading to B.[5] The direction of an affectee branch encodes, in a precise or imprecise manner, the values produced

[5] C and D can be the same instruction.

or yet to be produced by D and of other instructions in the data dependence graph from branch B to instruction D.

In the example in Fig. 1.a the branch in *BB7* has two affectees, the branches in *BB4* and *BB6*. More specifically, the branch *R2==1* in *BB4* is an affectee because it tests the outcome of the load instruction *LW R2,(R1)*, on which the branch *R3==R4* in *BB7* has an indirect data dependence (through the instructions *R3 = R1 + R2* in *BB3* and *R1 = R2 + 4* in *BB2*). Since the direction of *R2==1* is taken it implies that the test condition is true and consequently the value loaded from the load instruction *LW R2,(R1)* is *1*, the value produced by *R1 = R2 + 4* is *5* and the result of *R3 = R1 + R2* is 6. Therefore, in this case the direction of branch *R2==1* in *BB4* provides a precise encoding for the operand *R3* of the branch *R3==R4* in *BB7*. On the other hand, the false condition of the affectee branch *R4>10* in *BB6* is less precise and provides a range of possible values for the second operand *R4* of the branch.

Essentially, affectees provide an encoding for values consumed or produced in the dataflow graph leading to the branch to be predicted. The affectee value encodings for the example in Fig. 1.a are shown in Fig. 1.c. Note that a branch can be both affector and affectee of another branch depending on its dependences. An example of such branch is *R2==1* in *BB4* in Fig. 1.a.

Combo: It is evident that the combination of affectors and affectees can be more powerful than either correlation alone since affectees can help differentiate between branches with the same data affector data flow graphs but different input values. Similarly, affectors can help distinguish between same affectee graphs that correspond to different affector graphs. The combined affector and affectee data flow graph of our running example is shown in Fig. 1.d.

Section 4 investigates how the above types of correlations affect branch prediction accuracy. We believe that existing predictor schemes are able to learn data flow graphs, as those shown in Fig. 1, but they do this inefficiently using more history bits than needed. Therefore, they may suffer from cold effects and more table pressure/aliasing. Our analysis will establish how much room is there to improve them.

2.2 Memory Dependences

For accurate detection of the direction-correlations data dependences need to be tracked through memory. That way a branch that has a dependence to a load instruction can detect correlation to other branches through the memory dependence. Although, tracking dependences through memory may be important for developing a better understanding for the potential and properties of affectors and affectees correlations, it may be useful to know the extent that such precise knowledge is necessary. Thus may be interesting to determine how well predictors will work if direction-correlations detected through memory dependences are approximated or completely ignored.

We consider two approximations of memory dependences. The one tracks the dependence of address operands of a load instruction ignoring the dependence for the data. And the other does not consider any dependences past a load instruction, i.e. limiting a branch to correlations emanating from the most recent load instructions leading to the branch. These two approximations of memory dependences need to track register dependences whereas the precise scheme requires maintaining dependences between stores and load through memory. We will refer to the precise scheme of tracking dependences as *Memory*, and to the two approximations as *Address*, and *NoMemory*. In Section 4 we will compare the prediction accuracy of the various schemes to determine the importance of tracking accurately correlations through memory.

For the *Memory* scheme we found that is better to not include the address dependences of a load when a data dependence to a store is found (analysis not presented due to limited space). This is reasonable because the correlations of the data encode directly the information affecting the branch whereas the address correlations are indirect and possibly superfluous.

Recall that our algorithm for detecting direction-correlations does not consider implementation constraints. It is based on analysis of the dynamic data dependence graph of a program. The intention of this work is to establish if there is potential from using more selective correlation.

2.3 How to use Affectors and Affectees for Prediction

Based on the findings of this paper one can attempt to design a predictor grounds-up that exploits the properties exhibited by affectors and affectees correlations. That is also our ultimate goal and hopefully this paper will serve as a stepping stone in that direction. This, however, may be a non-trivial effort and before engaging in such a task may be useful to know its potential.

Therefore, in this paper we decided to determine the potential of affectors and affectees using unmodified existing predictors. We simply feed these predictors with the complete global history and with the history selected using affectors and affectees and compare their prediction accuracy. If this analysis reveals that the selective correlations have consistently and substantially better accuracy then may be worthwhile to design a new predictor.

The only predictor design space option we have is how to represent the selected bits in the global history register. In [8] they were confronted with a similar problem and proposed the use of *zeroing* and *packing*. Zeroing means set a history bit to zero if it is not selected while branches retain their original position in the history register. Packing moves all the selected bits to the least significant part of the history register while other bits are set to zero. Therefore, in packing selected branches lose their original position but retain their order. Our experimental data (not shown due to space constraints) revealed that packing had on average the best accuracy and is the representation we used for the results reported in Section 4.

Our methodology for finding the potential of affectors and affectees may be suboptimal because it uses an existing predictor without considering the properties exhibited in the global history patterns after selection. Another possible limitation of our study has to do with our definition of affectors and affectees.

Table 1. Benchmarks

SPECINT CPU2000	bzip200, crafty00, eon00, gap00, gcc00, gzip00, mcf00, perlbmk00, twolf00, vortex00, vpr00
SPECFP CPU2000	ammp00, equake00, fma3d00, galgel00, mesa00 mgrid00 sixtrack00, wupwise00
SPECINT CPU95	gcc95, go95, ijpeg95

Alternative definitions may lead to even more selective and accurate correlations. For instance by considering only affectees that trace dependences to load instructions. These and other limitations to be found may lead to increased potential and thus the findings of this study should be viewed as the potential under the assumptions and constraints used in the paper.

3 Experimental Framework

To determine the potential of affectors and affectees to increase branch prediction accuracy we used a functional simulation methodology using a simplescalar [10] derived simulator. A subset of SPEC2000 and three SPEC95 benchmarks, listed in Table 1, are used for our analysis. For the SPEC2000 benchmarks the early regions, of 10-100 million instructions, identified by sim-point [11] are used, whereas for SPEC95 complete runs of modified reference inputs are executed. Some SPEC2000 benchmarks are not included because they required large memory and/or long simulation time to track dependences, affectors and affectees. The selected SPEC95 benchmarks exhibit the higher misprediction rates with a 32KB L-Tage predictor among integer SPEC95 benchmarks.

Two predictors are used in the experimentation: a 32KB L-TAGE [12] predictor with maximum history length of 400 bits, and the GTL [4] predictor with 400 maximum history length for the GEHL component and 100000 maximum history length for the TAGE component.

For the experiments where selective correlation is used, the selection is applied to the 400 bit global history of the L-TAGE predictor and to the 400 bit history used to access the GEHL component of the GTL predictor. Selection was not used for the TAGE component of GTL because the memory required to track affectors and affectees for a 100000 global history were extremely large and beyond the memory capacities of todays servers.

The detection of affectors and affectees is done on-line using the dynamic data flow graph of a program. Unless stated otherwise, the default policy is to track correlations through memory dependences.[6]

[6] In the conference version of the paper [13] the term oracle was used to signify the precise tracking of memory dependences assumed for obtaining some of the results. The same assumption is used for this paper but the term is omitted to avoid confusion with an oracle off-line analysis for detecting affectors and affectees.

The algorithm used to determine affectors is the simple approximation proposed in [8]. A dynamic branch is an affector, of a branch to be predicted, if it is the last, in the dynamic program order, branch that executed before an instruction in the dataflow graph of the branch to be predicted.

The algorithm that detects affectees tracks the *sources* for each unique state, register or memory location, updated during a program's execution. Sources are the roots in the dynamic data dependence graph of each dynamic instruction. Sources are either dynamic instances of instructions with no inputs, like a move immediate, or locations with program data input, i.e. locations read but not updated by a program instruction. Each unique source contains a bit vector with n bits. Every time an instruction executes it computes the union of its input operand sources to produce the set of sources to be written in its destination. Every time a conditional branch executes all sources shift their bit vector by one.[7] Also, the sources of the branch set their least significant bit to indicate that this branch can trace a dependence to this source. The above imply that when bit i of a source is set then the ith most recent branch has a dependence to this source. To determine the affectees of a branch we determine the union of its operands sources and bitwise-or these sources bit vectors. All the positions that are set in the resultant bit vector correspond to the global branch history positions with a correlation.

4 Results

We present three sets of results, the first analyzes the properties of affectors and affectees, the second discusses the accuracy of the GTL predictor, and the third shows the accuracy of the L-TAGE predictor

4.1 Characterization of Affectors and Affectees

Fig. 2 and 3 show the cumulative distribution of dynamic branches according to the number of affector and affectee correlations they have. The number of correlations can not exceed 400 since we consider only correlations from the 400 most recent branches. We decided to analyze the behavior for the 400 most recent branches since the two predictors used in the study use a 400 entry global branch history register.

The results reveal that branches usually have much fewer affectors than affectees. For most benchmarks 80% of the branches have at most 30 affectors. According to the definition of affectors, this means that the computation that determines the outcome of a branch can be found in less than 30 out of the most recent 400 basic blocks preceded by a conditional branch. The outlier is *gcc00* where many branches have large number of affectors. The data about affectees

[7] A key optimization is to not shift all sources every time a branch executes but only the sources of the branch. The shift amount is determined based on the distance in branches between the current branch instruction and the last branch that updated the particular source.

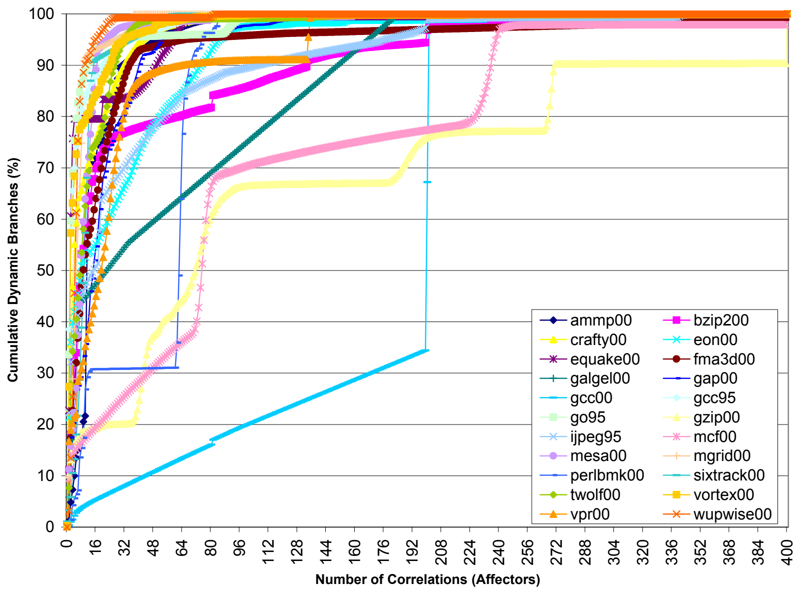

Fig. 2. Affectors distribution

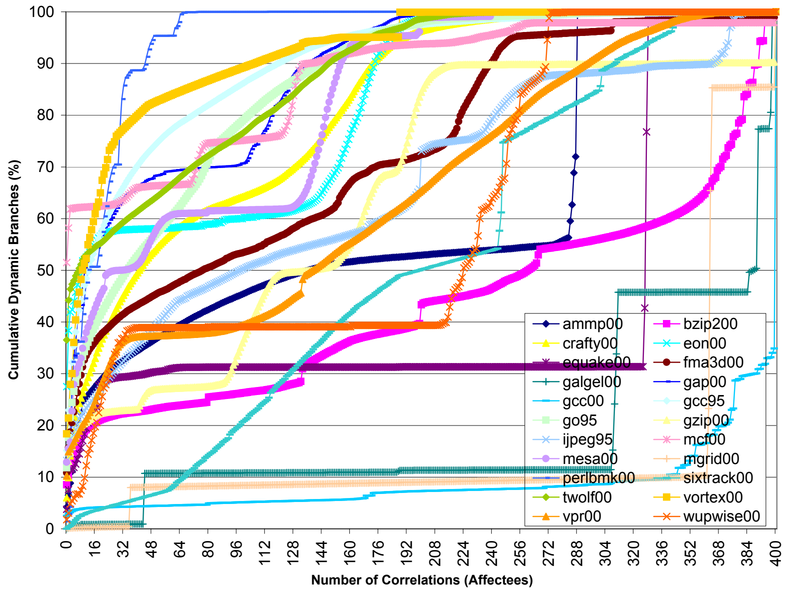

Fig. 3. Affectees distribution

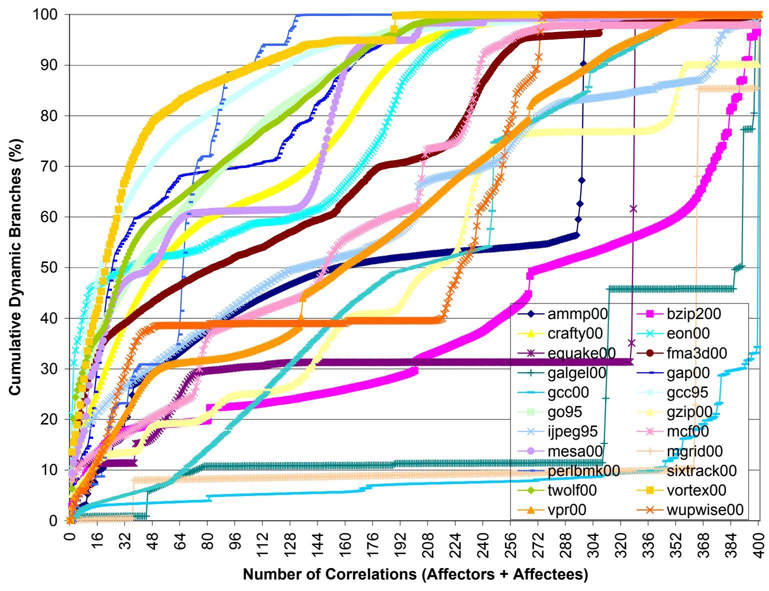

Fig. 4. Combined Affectors and Affectees distribution

correlations show clearly that for most programs 50% of the branches have 30 or more affectees. This means that a branch frequently checks information that directly or indirectly has been tested by at least 30 other out of the 400 most recent branches. The data also show few benchmarks, *bzip00, galgel00, gcc00,* and *mgrid00*, to have 300 or more affectee correlations.

The above observations suggest that the dynamic dataflow graphs of branch instructions are usually small and shallow (implied by the small number of affectors), and branches often share part of their dynamic data flow graphs with other branches (indicated by the large number of affectees).

The graph in Fig. 4 shows the distribution of the branches when we consider both affectors and affectees correlations. Overall, the data show that there are more correlations when we consider affectors and affectees in combination (compare Fig. 4 against Fig. 2 and 3). Nonetheless, the results for *ALL* benchmarks reveal that there are many branches that have much less than maximum number correlations. Therefore, if: (a) affectors and affectees are the dominant types of correlation that predictors need to learn, and (b) existing predictors are unable to use only the relevant part of history, then these data suggest that there may be room for improving prediction.

In Fig. 5 we attempt to give more insight by presenting the dominant patterns of correlation when we consider the combination of affectors and affectees. The figure shows for six benchmarks, *twolf00*, *bzip00*, *ammp00*, *crafty00*, *perlbmk00* and *equake00* what are the most frequent 1000 patterns of correlations. To help the reader we present these top patterns sorted from top to bottom according to

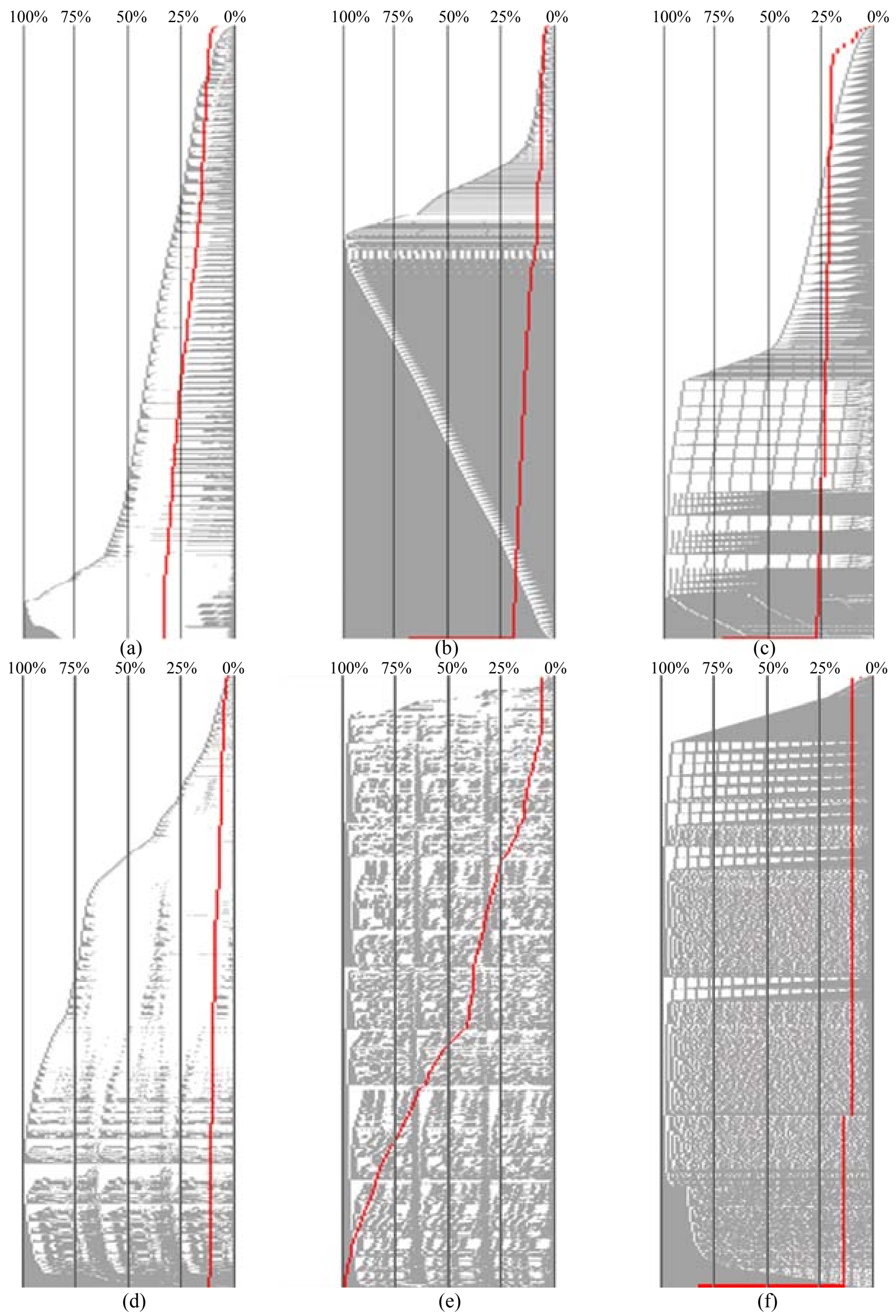

Fig. 5. Most frequent correlation patterns for (a) twolf00, (b) bzip00, (c) ammp00, (d) crafty00, (e) perlbmk00, and (f) equake00

Table 2. Representative Benchmarks for Correlation Patterns

Benchmark	Representative of Benchmarks
twolf00	vpr00, gcc95 and go95
bzip00	gcc00, gzip00, mcf00 and ijpeg95
ammp00	galgel00, mgrid00 and sixtrack00
crafty00	mesa00
perlbmk00	eon00, fma3d00, gap00, vortex00 and wupwise00
equake00	-

the oldest position with a correlation (i.e. the most recent correlation position is to the right). The curve that cut-across each graph represents from top to bottom the cumulative branch distribution of the patterns. This line is not reaching 100% since we only display the top 1000 patterns. A given pattern has a gray and white part representing the bit positions with and without correlations. To help the reader we present patterns with 100 positions where each position corresponds to 4 bits (a position is set to one if any of its corresponding four bits is set). These six graphs are representative of the remaining benchmarks we considered in this paper as shown in Table 2. Benchmark *equake00* has a unique behavior with very few dominant correlations patterns. For the following discussion we define the length of a correlation pattern to be the oldest position with a correlation.

One of the main observation from these data is that branch correlations are not always consecutive, there are *holes* between correlated branches. These holes can be of any size and a given correlation pattern can have one or more holes. The hole behavior varies across benchmarks, for *twolf00 and crafty00* like benchmarks is dominant whereas for *bzip00* like benchmarks they occur less frequently. Within a benchmark there can be both sparse and dense patterns.

More specifically, the results indicate that virtually always correlation patterns include at least few of the most recent branches (for each benchmark almost all patterns have at the right end - most recent branches - few positions set). Also, it is observed across almost all benchmarks that for a given correlation length the pattern with all positions set is very frequent. However, for *twolf00* like benchmarks many patterns have correlations that occur at the beginning and at the end of the pattern with all the branches in the middle being uncorrelated. Benchmark *crafty00* exhibits similar behavior with *twolf00* except that some correlations may exist in the middle. Another remark for *bzip00*, *ammp00* and *equake00* like benchmarks, is that they have many branches with correlations distributed over all 100 positions (bottom pattern in Fig. 5 for *bzip00*, *ammp00* and *equake00* accounts for over 40% of the patterns). Finally, *perlbmk00* like benchmarks are distinct because of few but often long correlation patterns.

Provided it is important to predict by learning precisely the above correlations, the results suggest that there is a need for predictors that can learn efficiently patterns with holes.

Another key observation from Fig. 5 is that correlation patterns occur usually across all history lengths. These underlines the need for predictors to be capable

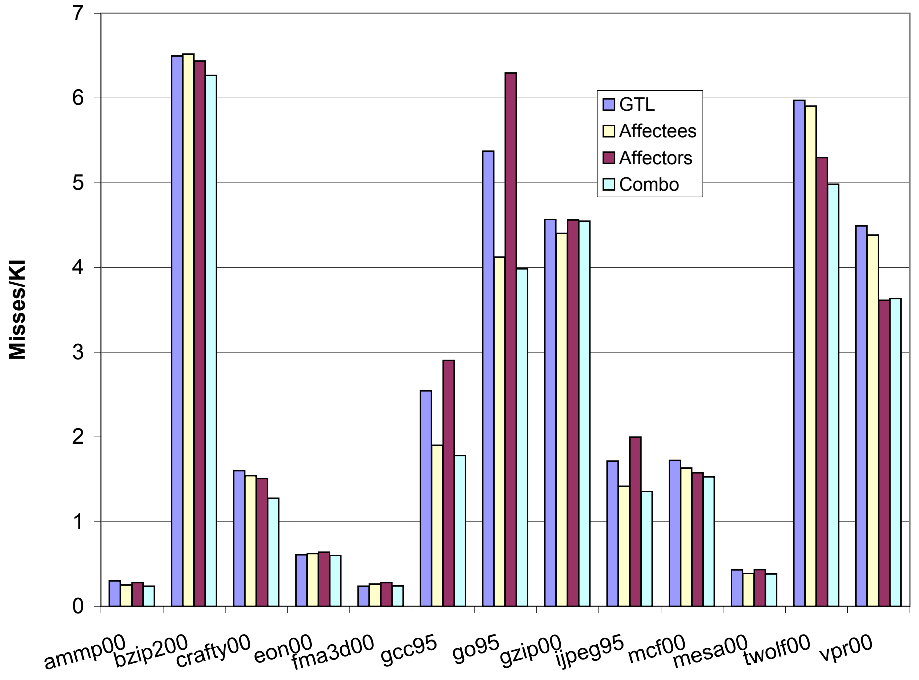

Fig. 6. GTL accuracy with selective correlation

of predicting with variable history length. The distribution of patterns according to length is similar to the distribution in Fig. 4. Assuming is important to learn precisely the correlation patterns, the exponential like cumulative distributions of correlation lengths, for most benchmarks, suggests that most prediction resources should be devoted to capture correlations with short history length and incrementally use less resources for longer correlations. This observation clearly supports the use of geometric history length predictors [14].

The above observations may represent a call for predictors that can handle both geometric history length and holes. As far as we know no such predictor exists today. In the next section we attempt to establish the potential of such a predictor using two existing geometric history length predictors that are accessed with selected history, with holes, using affectors and affectees correlations. In the remaining paper we only present data for the benchmarks that exhibited at least 0.25 misses per one thousand instructions. The other benchmarks displayed minimal sensitivity to the predictor used and for the sake of graph clarity are omitted.

4.2 GTL Results

Fig. 6 shows the accuracy of the GTL predictor when accessed with full global history, only with affectors correlations, only with affectees, and with the combi-

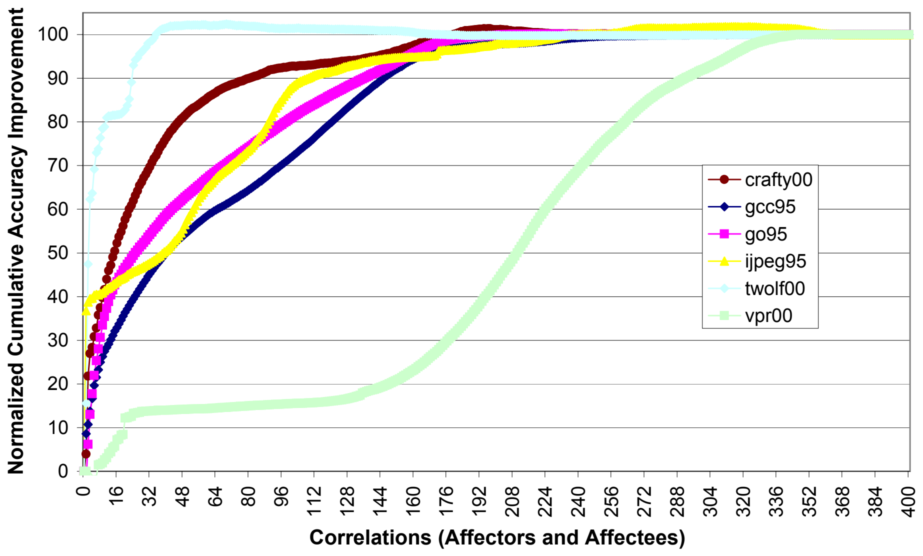

Fig. 7. Number of Correlations vs Accuracy Improvement

nation of affectors and affectees. The data show that the combination of affectors and affectees provides the best performance. It is always the same or better than GTL and almost always better than each correlation separately. The exception are *gzip00* and *vpr00* where the combination does slightly worse than using only affectees and affectors respectively. This can happen when the one type of correlation is sufficient to capture the program behavior and the use of additional information is detrimental. The improvement provided by combining affectors and affectees is substantial for several benchmarks. In particular, for *crafty00, gcc95, go95, ijpeg95, twolf00, and vpr00* it ranges from 15% to 30%. The data clearly support the claim by [6] that *direction*-correlation is one of the basic types of correlations in programs that predictors need to capture. For the remaining paper we present results for experiments that combine affectors and affectees since they provide the best overall accuracy.

Fig. 7 shows the normalized cumulative improvement in prediction accuracy when using affectors and affectees over GTL as a function of the number of correlations. This is shown only for the benchmarks that experienced the largest accuracy improvement when using affectors and affectees. To illustrate how to interpret the graph consider *crafty00*. The *Combo* configuration in Fig. 6 reduces mispredictions of crafty by 20%. The data in Fig. 7 indicate that 90% of this improvement is due to correlations patterns that include less than 75 affectors and affectees. In general, the data in Fig. 7 reveal that most of the improvement from selective correlation is due to better prediction accuracy for the branches that have fewer than 100 branch correlations. This may indicate that the GTL

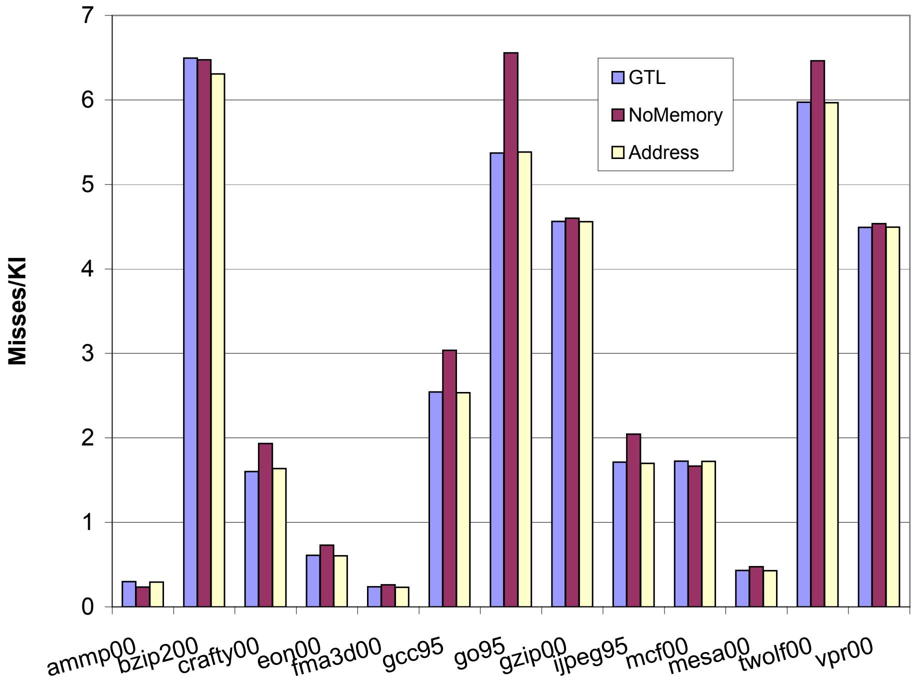

Fig. 8. Significance of Memory Dependences

predictor may be slow to learn or using more table resources than necessary for such branches. For all benchmarks there is little improvement for branches with over 300 correlations. This may suggest that the more bits in a correlation pattern the closer the resemblance to the global history register and thus little room for improvement from selective correlation.

Fig. 8 shows the prediction accuracy when we combine affectors and affectees but with no correlations through memory. For each benchmark we present three results, the GTL predictor with full history, the affectors and affectees with no correlations past load instructions (NoMemory), and with correlations past load instructions using their address dependences (Address). The data show that there is very little improvement to gain when we do not consider correlations through memory dependences. The data indicate that an approximation of memory dependences using addresses dependences offers very little improvement. This underlines that important correlations from the data predecessors of load instructions are needed for improved accuracy.

The data show that selective correlation using the combination of affectors and affectees can provide substantial improvement in prediction accuracy. The results also show that correlations past memory instructions are important and that address dependences provide a poor approximations of the data dependence correlations. Overall, we believe the data suggest that may be worthwhile inves-

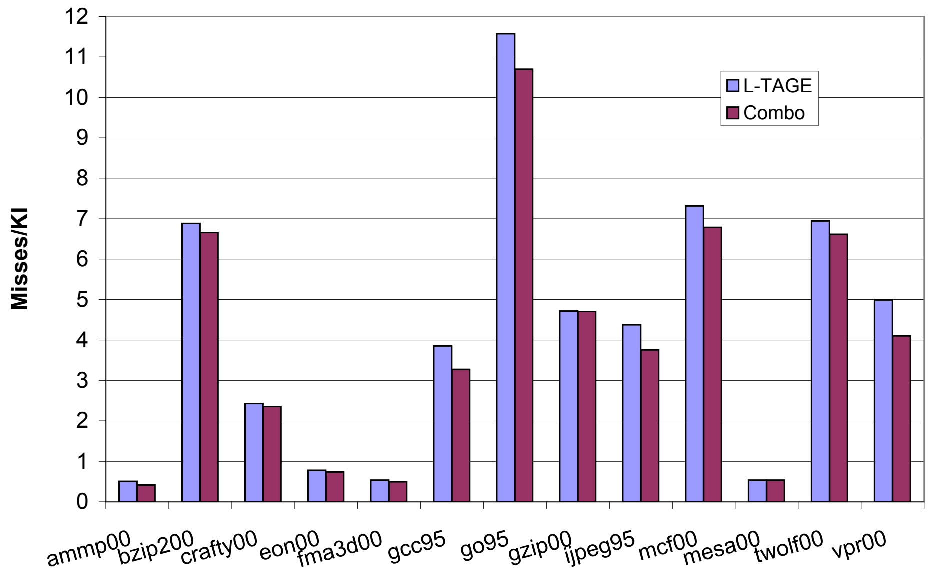

Fig. 9. L-TAGE accuracy with selective correlation

tigating the development of a predictor that is capable of learning correlations from long history with holes. These conclusions are true for GTL an unrealistically large predictor that demonstrate that the improvements are not mere accident but due to basic enhancements in the prediction process. However, we are interested to know if these observations hold for a realistic predictor. Next we consider selective correlation for a 32KB L-TAGE predictor.

4.3 L-TAGE Results

Fig. 9 shows the prediction accuracy for a 32KB L-TAGE when accessed using the complete global history (L-TAGE) and with selective history using the combination of affectors and affectees (Combo). The results show that selective correlation with affectors and affectees can also improve the accuracy of the L-TAGE predictor at a realistic size. The amount of improvement is significant for several benchmarks. In particular, for *gcc95, ijpeg95, and vpr00* is above 15% (for vpr 17%). We believe that these improvements call for the design of a predictor that can exploit *direction*-correlations.

The amount of improvements for L-TAGE are smaller as compared to GTL. However, one should recall that GTL is a completely different predictor not simply a bigger L-TAGE predictor. We also performed analysis of the importance of correlations through memory and the data suggest, similarly to GTL, that it is necessary to include such correlations for better accuracy.

5 Related Work

Since Smith [15] proposed the first dynamic table based branch predictor, innovation in the field of prediction has been sporadic but steady. Some of the key milestones are: correlation-based prediction [16] that exploits the global and or local correlation between branches, hybrid prediction [7] that combines different predictors to capture distinct branch behavior, variable history length [17] that adjusts the amount of global history used depending on program behavior, the use of perceptrons [9] to learn correlations from long history, geometric history length prediction [14] that employs different history lengths that follow a geometric series to index the various tables of a predictor, and partial tagging [18] of predictor table entries to better manage their allocation and deallocation. The above innovations have one main theme in common: the correlation information used to predict a branch is becoming increasingly more selective. This facilitates both faster predictor training time and less destructive aliasing. Our paper extends this line of work and shows that there is room for further improvement if we could select correlations with holes out of long history.

The importance for selective correlation is first established in the work by Evers et al. [6]. In that paper it is shown that a predictor that selectively correlates on few bits from the global history register can outperform a predictor that correlates on the entire global history register. The paper argues that the improvement is due to a reduction in the number of correlation patterns that need to learned which leads to faster training and less aliasing. However, the findings in [6] are based on an off-line oracle analysis. Fern et al. [19] proposed a possibly implementable on-line predictor based on the principles of dynamic decision trees capable of learning and correlating on a subset of history bits. An initial evaluation of this predictor revealed comparable performance to equal sized *Gap* [16] and *Pap* [16] predictors.

A return-history-stack [20] is a method that can introduce holes in the branch history. In broad terms, a return history stack pushes in a stack the branch history register on a call and recovers it on a return, thus introducing holes in the history. A return history stack is shown to be useful for a trace predictor [20] and offers modest improvements for a direction branch predictor [21]. This suggests that there are many cases where branches executed in a function are often no significant to correlate on for the branches that execute after the function return.

In two recently organized branch prediction championships [1, 2] researchers established the state of the art in branch prediction. In 2006, the L-TAGE global history predictor [12] was the winner for a 32KB budget. L-TAGE is a multi-table predictor with partial tagging and geometric history lengths that also includes a loop predictor. In the 2006 championship limit contest the GTL predictor [4] provided the best accuracy. GTL combines GEHL [14] and L-TAGE predictors using a meta-predictor. The GEHL global history predictor [14] employs multiple components indexed with geometric history length. Our paper uses the L-TAGE and GTL predictors to examine our ideas to ensure that observations made are not accidental but based on basic principles. The use of longer history is central to these two predictors and the analysis in this paper confirmed the need and usefulness for learning geometrically longer history correlations.

Several previous paper explored the idea of improving prediction by encoding the data flow graphs leading to instructions to be predicted. They use information from instructions in the data flow graph [22–26], such as opcodes, immediate values, and register names, to train a predictor. Effectively these papers are implementing variations of predictors that correlate on affector branches. In [26], they consider using the live in values of the dataflow graphs when they become available and in [23] they examined the possibility of predicting such values. The inclusion of actual or predicted live-in values is analogous to the correlation on affectee branches of such values, since the predicted or actual outcome of affectee branches represents an encoding of the live-in values.

Mahlke and Natarajan [27] performed profiling analysis to determine simple correlation functions between register values and branch outcomes. Instructions are inserted in the code by the compiler to dynamically compute the branch direction according to the derived functions. In our view, this work also attempts to implement a variation of affectees correlation since the functions supply analogous information to what can be provided by affectee branches.

6 Conclusions and Future Work

In this paper we investigate the potential of selective correlation using affectors and affectees branches to improve branch prediction. Experimental analysis of affectors and affectees revealed that many branches have few correlations and often the correlations have holes between them. Prediction using selective correlation, based on affectors and affectees, is shown to have significant potential to improve accuracy for a both a limit and a realistic predictor. The analysis also shows that correlations past memory instruction are needed for best accuracy. Overall, our study suggests that may be worthwhile to consider the design of a realistic predictor that can exploit the properties exhibited by affectors and affectees correlation patterns by learning correlations with and without holes from long history.

A possible venue for future work is to train the tables of TAGE like predictors, that contain multiple prediction tables, with branch history with holes. The challenge is to decide what are going to be the holes in the branch history since different benchmarks have different hole patterns. To design efficiently such scheme it may be useful to first investigate and determine what is the relation between dynamic program properties and holes.

One other direction of work is to focus on difficult to predict branches and investigate their correlation patterns with increasingly longer history. Such an analysis will reveal the importance of selective correlation to distant correlations.

Another possible direction of future work, is to investigate which affectors and affectees are more important. A decision-tree based approach [5, 19] can be used to establish such classification. Such an analysis can be useful for better understanding and hopefully further reduce the correlations required for best prediction.

Finally, the approach proposed in this paper can be applied to static branch prediction, and to other types of predictors, such as value and dependence predictors.

Acknowledgments This work is partially supported by an Intel research grant and the University of Cyprus. Yiannakis Sazeides would like to thank Ronny Ronen, Roni Rosner, Avi Mendelson, Pierre Michaud, Hans Vandierendonck and Veerle Desmet for their encouragement and feedback on earlier versions of this work. The authors like also to thank the anonymous reviewers for their constructive critique and suggestions that helped improve the presentation of this manuscript.

References

1. Wilkerson, C., Stark, J.: Introduction to JILP's Special Edition for Finalists of the Championship Branch Prediction (CBP1) Competition. Journal of Instruction-Level Parallelism **7** (2005)
2. Jiménez, D.A.: The Second Championship Branch Prediction Competition. Journal of Instruction-Level Parallelism **9** (2007)
3. Seznec, A.: Genesis of the O-GEHL Branch Predictor. Journal of Instruction-Level Parallelism **7** (2005)
4. Seznec, A.: The Idealistic GTL Predictor. Journal of Instruction-Level Parallelism **9** (2007)
5. Desmet, V.: On the Systematic Design of Cost-Effective Branch Prediction. (PhD Thesis, University of Ghent, Belgium 2006)
6. Evers, M., Patel, S.J., Chappel, R.S., Patt, Y.N.: An Analysis of Correlation and Predictability: What Makes Two-Level Branch Predictors Work. In: 25th International Symposium on Computer Architecture. (June 1998)
7. McFarling, S.: Combining Branch Predictors. Technical Report DEC WRL TN-36, Digital Western Research Laboratory (June 1993)
8. Thomas, R., Franklin, M., Wilkerson, C., Stark, J.: Improving Branch Prediction by Dynamic Dataflow-based Identification of Correlated Branches from a Large Global History. In: 30th International Symposium on Computer Architecture. (June 2003) 314–323
9. Jimenez, D.A., Lin, C.: Dynamic Branch Prediction with Perceptrons. In: 7th International Symposium on High Performance Computer Architecture. (Feb. 2001)
10. Burger, D., Austin, T.M., Bennett, S.: Evaluating Future Microprocessors: The SimpleScalar Tool Set. Technical Report CS-TR-96-1308, University of Wisconsin-Madison (July 1996)
11. Perelman, E., Hamerly, G., Biesbrouck, M.V., Sherwood, T., Calder, B.: Using SimPoint for Accurate and Efficient Simulation. In: International Conference on Measurement and Modeling of Computer Systems. (2003)
12. Seznec, A.: The L-TAGE Branch Predictor. Journal of Instruction-Level Parallelism **9** (2007)

13. Sazeides, Y., Moustakas, A., Constantinides, K., Kleanthous, M.: The Significance of Affectors and Affectees Correlations for Branch Predicion. In: International Conference on High Performance Embedded Architectures and Compilers. (January 2008) 243–257
14. Seznec, A.: Analysis of the O-GEometric History Length branch predictor. In: 32nd International Symposium on Computer Architecture. (2005)
15. Smith, J.E.: A Study of Branch Prediction Strategies. In: 8th International Symposium on Computer Architecture. (May 1981) 135–148
16. Yeh, T.Y., Patt, Y.N.: Two-Level Adaptive Branch Prediction. In: 24th International Symposium on Microarchitecture. (November 1991) 51–61
17. Juan, T., Sanjeevan, S., Navarro, J.J.: Dynamic History-Length Fitting: A third level of adaptivity for branch prediction. In: 25th International Symposium on Computer Architecture. (June 1998) 155–166
18. Michaud, P.: A PPM-like, Tag-based Predictor. Journal of Instruction-Level Parallelism **7** (2005)
19. Fern, A., Givan, R., Falsafi, B., Vijaykumar, T.N.: Dynamic feature selection for hardware prediction. Journal of Systems Architecture **52**(4) (2006) 213–234
20. Jacobson, Q., Rottenberg, E., Smith, J.E.: Path-Based Next Trace Prediction. In: 30th International Symposium on Microarchitecture. (December 1997) 14–23
21. Gao, F., Sair, S.: Exploiting Intra-function Correlation with the Global History. In: SAMOS. (2005)
22. Farcy, A., Temam, O., Espasa, R., Juan, T.: Dataflow analysis of branch mispredictions and its application to early resolution of branch outcomes. In: 31st International Symposium on Microarchitecture. (December 1998) 59–68
23. Thomas, R., Franklin, M.: Using Dataflow Based Context for Accurate Value Prediction. In: 2001 International Conference on Parallel Architectures and Compilation Techniques. (September 2001) 107–117
24. Sazeides, Y.: Dependence Based Value Prediction. Technical Report CS-TR-02-00, University of Cyprus (February 2002)
25. Constantinides, K., Sazeides, Y.: A Hardware Based Method for Dynamically Detecting Instruction Isomorphism and its Application to Branch Prediction. In: 2nd Value Prediction Workshop. (2004)
26. Chen, L., Dropsho, S., Albonesi, D.H.: Dynamic Data Dependence Tracking and its Application to Branch Prediction. In: 9th International Symposium on High Performance Computer Architecture. (February 2003) 65–76
27. Mahlke, S., Natarajan, B.: Compiler Synthesized Dynamic Branch Prediction. In: 29th International Symposium on Microarchitecture. (December 1996) 153–164