Extending Correlation in Branch Prediction Schemes

Lucian N. VINTAN$^{*)}$, Colin. EGAN$^{**)}$
$^{*)}$University “L. Blaga”, Dept. of Comp. Sc., ROMANIA, E-mail: vintan@cs.sibiu.ro
$^{**)}$University of Hertfordshire, Dept. of Comp. Sc., UK, E-mail: c.1.egan@herts.ac.uk

Abstract

The main aim of this research is to propose a new Two-Level Adaptive Branch Prediction scheme, based on additional correlation information. Conventional two-level adaptive branch prediction exploits the correlation between the outcome of a branch and the path followed through a program to reach the branch. Typically the program path is identified by recording whether each branch on the path is taken or not taken. Unfortunately, this limited information is insufficient to allow one path to a branch to be distinguished from other potential paths to the same branch. In this paper, we explore the benefits of adding sufficient information, in the form of successive branch addresses, to uniquely identify each program path. We use trace-driven simulation to compare our modified branch prediction scheme with a conventional GAp two-level predictor and demonstrate that our new predictor performs better than the conventional GAp scheme at the same level of hardware complexity.

Key Words *Branch prediction, Two-Level Adaptive Branch Prediction, Trace driven simulation*

1. Introduction

As the average instruction issue rate and depth of the pipeline in multiple-instruction-issue (MII) processors increase, accurate dynamic branch prediction becomes more and more essential. Very high prediction accuracy is required because an increasing number of instructions are lost before a branch misprediction can be corrected. As a result even a misprediction rate of a few percent involves a substantial performance loss.

If branch prediction is to improve performance, branches must be detected within the dynamic instruction stream, the direction taken by each branch must be correctly predicted and the branch target address must be correctly predicted. Furthermore, all of the above must be completed in time to fetch instructions from the branch target address without interrupting the flow of new instructions to the processor pipeline. A classic Branch Target Cache (BTC) [Hen96] achieves these objectives by holding the following information for previously executed branches: the address of the branch instruction, the branch target address and information on the previous outcomes of the branch. Branches are then predicted by using the PC address to access the BTC in parallel with the normal instruction fetch process. As a result each branch is predicted while the branch instruction itself is being fetched from the instruction cache. Whenever a branch is detected and predicted as taken, the appropriate branch target is then available at the end of the instruction fetch cycle, and instructions can be fetched from the branch target in the cycle immediately after the branch itself is fetched. Straightforward prediction mechanisms based on the previous history of each branch give a prediction accuracy of around 80 to 95% [Hen96]. This success rate proved adequate for scalar processors, but is generally regarded as inadequate for MII architectures.

The requirement for higher branch prediction accuracy in MII systems and the availability of additional silicon area led to a dramatic breakthrough in the early 90s with branch prediction success rates as high as 97% [Yeh92] being reported. These high success rates were obtained using a new set of prediction techniques known collectively as Two-Level Adaptive Branch Prediction that were developed independently by Yale Patt’s group at the University of Michigan [Yeh91] and by Pan, So and Rahmeh from IBM and the University of Texas [Pan92]. Two-Level Adaptive Branch Prediction uses two levels of branch history information to make a branch prediction. The first level consists of a History Register [HR] that records the outcome of the last k branches encountered. The HR may be a single global register, HRg, that records the outcome of last k branches executed in the dynamic instruction stream or multiple local history registers, HRl, that record the last k outcomes of the specific branch being predicted. The second level of the predictor known as the Pattern History Table (PHT) records the behaviour of a branch during previous occurrences of the first level predictor.

It consists of an array of two-bit saturating counters, one for each possible entry in the HR. 2^k entries are therefore required if a global PHT is provided, or many times this number if a separate HR and therefore PHT is provided for each branch PC. Although a single term is usually applied to the new predictors, this is misleading. Since the first level predictor can record either global or local branch history information, two distinct prediction techniques have in fact been developed. The global method exploits correlation between the outcome of a branch and the outcome of neighbouring branches that are executed immediately prior to the branch. In contrast, the local method depends on the assertion that the outcome of a specific instance of a branch is determined not simply by the past history of the branch, but also by the previous outcomes of the branch when a particular branch history was observed.

The main aim of this paper is to propose an improved global Two-Level Adaptive Branch Prediction scheme. Conventional global Two-Level Adaptive Branch Predictors [Pan92] exploit the correlation between the outcome of a branch and the dynamic path followed through a program to reach the branch. The program paths are identified by recording in the HRg whether each branch on the path is taken or not. Unfortunately, this information is insufficient to uniquely identify a program path. In our branch predictor we therefore record both the outcome and address of each branch on a program path. This additional information makes it possible to retrace the path taken to reach a branch and therefore identifies a unique path through the code. We use trace driven simulation to compare our improved predictor that uses this additional path information with a conventional global predictor.

2. An Improved Branch Predictor

Two-level adaptive branch prediction significantly reduces the number of incorrect branch predictions. Unfortunately, however, some branches are still difficult to predict correctly. For example, in the Stanford benchmarks there are a significant number of "hard-to-predict" branches whose direction cannot be determined by examining either the HRg or HRl bit patterns. In these cases, with identical HR values, the branch is almost equally likely to be taken or not taken. Furthermore, increasing the length of HR has little impact.

In theory, the adaptive nature of two-level adaptive branch prediction should help. Ideally, with a given HR pattern the predictor should correctly predict taken in some phases of the program and then adapt to predict not taken in other phases. Unfortunately, in practice, the extent of this dynamic adaptation appears to be minimal. As observed by Sechrist et al [Sec95], "The role of adaptivity at the second level of two-level branch prediction schemes is more limited than has been thought." It therefore appears that in these difficult-to-predict cases insufficient correlation information is fed to the predictor.

Earlier, we observed that the values stored in HRg do not identify a unique program path leading to each branch. Suppose, for example, that the final bit in HRg is set to logic "1", indicating that the branch executed immediately before the branch being predicted was taken. This final bit only indicates that one of perhaps several branches targeting the basic block containing the next branch has been taken. Since only the fall-through path from the immediately preceding basic block has been eliminated, the actual program path is indeterminate. As a result multiple program paths can map into a single HR bit pattern.

The correlation information available can be improved by recording not only the outcome of each branch but also the address of each branch instruction.[1] In this way additional correlation information can be provided for the predictor. Note that simply recording the address of each branch executed is insufficient to uniquely identify each path as can be seen from the following simple example:

```
        Bcc label
           :
label:  Bcc loop
```

Providing there are no intervening branches, the outcome of the first branch must also be recorded if the path is to be correctly identified.

Our improved branch predictor (MPAg), shown in Figure 1, makes full use of improved path information. A single global History Register (HRg) records both the outcome and the address of the last k branches; however, only the eight least significant bits of each PC are recorded to save bits. A fully associative Pattern History Table (PHT) is accessed by concatenating the PC address with the HRg. Each PHT entry holds the branch target address, prediction bits in the form of a two-bit saturating counter and LRU (Least Recently Used) bits used by our replacement algorithm.

A conventional two-level adaptive predictor would use the PC plus HRg to directly index a PHT consisting of an array of two-bit counters. Not surprisingly, the size of these PHT arrays is often excessive. With HRg extended to record full path information, the size of such a PHT would have been prohibitive. We have therefore chosen to implement our PHT as a fully-associative cache. As a result, although the size of each individual entry is increased, the total cost of the PHT is significantly reduced. Clearly a direct-mapped or a set-

[1] Alternatively, the requirement to also store the branch outcomes can be removed by saving the address of the instruction executed immediately after each branch instead of the actual branch address.

associative implementation would have been equally appropriate.

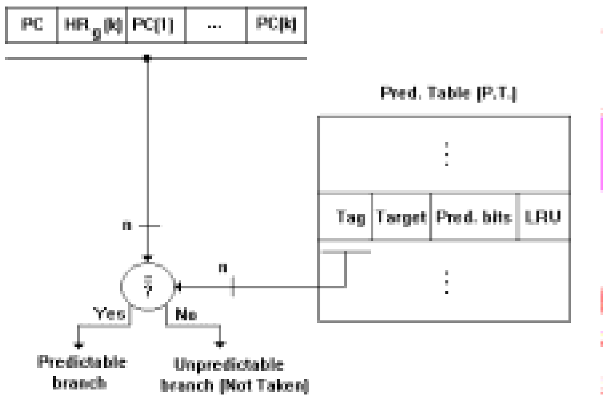

Figure. 1 A fully associative modified GAp (MGAp) scheme

We have implemented an MPP (Minimum Performance Potential Replacement) replacement algorithm similar to the one presented in [Per93]. The algorithm replaces the entry having the minimum product of the probability of reference, as given by the LRU field, and the probability of the branch being taken, as given by the prediction counter. As a result branches that are predicted as "not taken" tend to be replaced. Since any branches not held in the PHT is predicted as "not taken" by default, the effect is to minimise mispredictions caused by PHT replacements.

We compare our modified predictor (MPAg) with a conventional two-level adaptive scheme (Figure 2). This predictor would be classified as a GAp predictor in the Patt classification [Yeh92]. Again to reduce the size of the PHT and to provide a realistic comparison we have chosen to implement the PHT as a fully-associative cache.

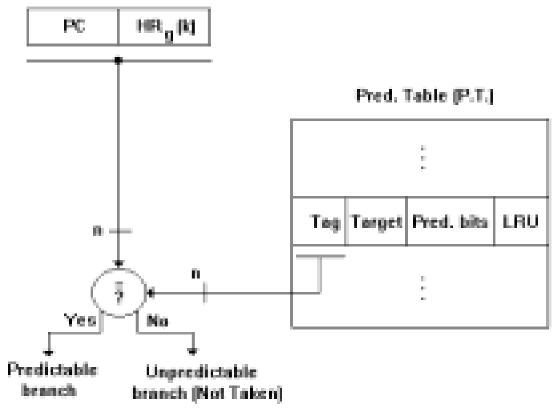

Figure. 2 A fully associative GAp scheme

3. Evaluation of Branch Predictor Schemes

3.1 Benchmark Programs

Our simulation work uses the Stanford integer benchmark suite, a collection of eight C programs designed by Professor John Hennessy to be representative of non-numeric code, while at the same time being compact. The benchmarks are computationally intensive with an average dynamic instruction count of 273,000. About 18% of the instructions are branches of which around 76% are taken. Some of the branches in these benchmarks are known to be particularly difficult to predict; see for example Mudges' detailed analysis [Mud96] of the branches in *quicksort*.

The benchmarks were compiled using a C compiler developed at the University of Hertfordshire the HSA (Hatfield Superscalar Architecture) [Ste97]. Instruction traces were then obtained using the HSA instruction-level simulator, with each trace entry providing information on the branch address, branch type and target address. These traces were used to drive a standalone branch predictor developed at the University of Sibiu that was used to simulate the branch predictors investigated in this paper. The trace-driven simulator is highly-configurable, the most important parameters being the number of HRg bits and the size of the PHT. As output the simulator generates the overall prediction accuracy, the number of incorrect target addresses and other useful statistics; see for example Table 1.

3.2. Simulation Results

First, we evaluated our MGAp predictor using a fully-associative PHT with 100 entries; see Table 1. HRg records the history of from one to five branches (k = 1 to 5). Since a total of nine bits are used to record each branch - eight bits for the least significant bits of the PC address and one bit to record the branch outcome - the size of HRg varies from 9 to 45 bits. As well as recording the number of correct and incorrect branch predictions, Table 1 also records the number of mispredictions caused by incorrect branch targets. This source of mispredictions could be almost completely eliminated by adding an address stack [Kae91] to hold the return addresses for subroutine return instructions. The last column in Table 1 records the total number of replacements, NR, that have taken place in the PHT during each simulation run. NR is a useful metric of branch interference within the PHT. Not surprisingly NR increases as the number of branch patterns recorded in the PHT increases. As more branches are added to each path, the number of paths associated with each branch increases. This in turn increases the pressure for entries in the PHT and the total number of replacements.

Table 2 is derived from Table 1 and records the most successful configuration for each benchmark. The average prediction accuracy, using the most successful configuration in each case, is 87.12%, a figure that rises to around 90% if incorrect branch targets are removed. Interestingly, with three exceptions, the highest success rates are obtained with paths consisting of only a single branch (k=1). Furthermore, in all cases the highest prediction accuracy is achieved in the absence of a large number of replacements (NR), and in all but two cases with an NR of zero.

Two opposing factors are at work here. First, the misprediction rate would be expected to fall as the length of the path recorded is increased. However, as path lengths are increased, more entries are required in the PHT. As a result more paths are evicted, and the number of mispredictions increases. This explanation is strongly supported by our results. In general, as the path length is increased the prediction success rate improves until the number of replacements in the PHT becomes significant. For example, *permute* experiences no PHT replacements and is the only benchmark to achieve its highest prediction accuracy with a path length of five.

We repeated our simulations with a conventional GAp prediction scheme (Table 3). In order to compare configurations with identical hardware costs, the length of HRg was increased in increments of nine bits rather than one. The average prediction success rates for the two predictors are compared in Table 4. The MGAp scheme achieves a maximum average success rate of 86.03% with a path length of two, while the conventional GAp scheme achieves an 83.74% success rate with a path length of one. Furthermore at every level of hardware complexity, the MGAp scheme outperforms the GAp scheme.

In our second set of experiments we concentrated exclusively on those branches in the Stanford benchmarks that are inherently “difficult to predict.” We consider a branch to be difficult to predict if both the local and global branch contexts used in conventional two-level predictors provide insufficient information to avoid a high misprediction rate. By the local context we mean HRl or the previous history of the branch being predicted while by the global context we mean HRg or the history of the branches executed immediately prior to the branch being predicted. Consider, for example, a specific branch from the program *perm* (Table 5). Here neither the local context, HRl, or the global context, HRg, provide sufficient information to allow accurate prediction. In general we consider branches that are mispredicted at least 20-30% of the time by conventional two-level techniques to be difficult to predict.

In Figure 3 to Figure 10 we compare the performance of the two predictors, GAp and MGAp, on the difficult-to-predict branches using seven of our benchmarks. The eighth benchmark *matrix* is excluded since it contains no difficult-to-predict branches. In the case of our MGAp predictor, the index in each figure represents the number of branches (PC + outcome) that make up the path recorded in HRg. In contrast, in the case of the conventional GAp predictor, k represents 9 bits of branch history information. As can be observed from the figures, our MGAp scheme generally outperforms the conventional GAp predictor that has identical hardware costs.

4. Conclusions and Discussion

In this paper we have simulated the performance of a modified GAp branch predictor based on complete program path information. Complete path information allows the branch predictor to uniquely identify the program path used to reach each branch and therefore potentially reduces the number of mispredictions.

Conventional two-level adaptive branch predictors implement the PHT as an array of two bit counters that increases exponentially in size as the length of HRg is increased. In many configurations this can leads to very large storage arrays. For example, using an 18 bit HRg and the 8 least significant bits of the PC address would require a PHT size of 2 ** 26 or over 64 million entries. By configuring our PHT as a cache rather than as a huge array, we significantly reduce the cost of the whole branch predictor. Furthermore, our configuration makes it possible to utilise full-path information at a reasonable cost, since the length of HRg is no longer the critical factor in determining the size of the PHT. Instead the size of the PHT is largely determined by the number of distinct program paths that must be stored to ensure accurate branch prediction. Finally, the more precise full path information, should allow different paths to be identified using a smaller number of distinct bit patterns in HRg.

The preliminary results presented in this paper are most encouraging, with our modified predictor generally outperforming the conventional GAp predictor. The prediction accuracies are smaller than those observed by other researchers using the Spec benchmarks. There are two reasons for this. Branches in very large benchmarks such as Spec tend to suffer a smaller percentage of mispredictions while the predictor is being trained. Furthermore, other researchers postulate unrealistically large PHT arrays, while the associative PHT simulated in this paper has only a hundred entries.

Particularly useful information has been gleaned regarding the interaction between path length and the number of replacements required in the PHT. The next stage of our research is to investigate our MGAp predictor using a wider range of parameters in our trace driven simulator and, in particular, to investigate increasing the size of our PHT to reduce the number of entry replacements.

References

[Hen96] **Hennessey, J. L. and Patterson, D. A.** *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 2nd ed., 1996.

[Kae91] **Kaeli, D. R. and Emma, P. G.** *Branch History Table Prediction of Moving Target Branches Due to Subroutine Returns*, ISCA-18, May 1991, pp34-42;

[Mud96] **Mudge T.N., Chen, I. K. and Coffey, J. T.** *Limits of Branch prediction,* Technical Report, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan, USA, January 1996, pp16.

[*Pan92]* **Pan S.T., So K. and Rahmeh J.T.** *Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation*, ASPLOS V Conference, Boston, October 1992, pp76-84.

[Per93] **Perleberg C. and Smith A. J**. *Branch Target Buffer Design and Optimisation*, IEEE Transactions on Computers, No. 4, 1993.

[Sec95] **Sechrest S., Lee C. and Mudge T.** *The Role of Adaptivity in Two-Level Adaptive Branch Prediction,* Micro-28, Ann Arbor, Michigan, November 1995, pp264-269.

[Ste97] **Steven G. B., Christianson, D. B., Collins, R., Potter, R. and Steven, F. L.** *A Superscalar Architecture to Exploit Instruction Level Parallelism*, Microprocessors and Microsystems, Vol. 20, No 7, March 1997, pp391-400.

[Yeh91] **Yeh T. and Patt Y.** *Two-Levels Adaptive Training Branch Prediction,* Micro-24, Albuquerque, New Mexico, November 1991, pp51-61.

[*Yeh92]* **Yeh T. and Patt Y.** *Alternative Implementations of Two-Level Adaptive Branch Prediction,* ISCA - 19, Gold Coast, Australia, 1992, pp124-134.

Acknowledgements

This work was supported in part by the Romanian Ministry of Research and Technology grant MCT No. 4086/1998 and by the Romanian National Council of Academic Research grant CNCSU No. 391/1998.

Table 1. Modified GAp Predictor (MGAp - 100 entries)

Benchmark	HRg	Number of Branches	Prediction Accuracy	Incorrect Predictions	Incorrect Branch Targets	Not Taken Branches	Number of Replacements (NR)
sort	9	12601	9441 (74.92%)	2929 (23.24%)	231 (1.83%)	4414 (35.03%)	0
sort	18	12601	9031 (71.67%)	3325 (26.39%)	245 (1.94%)	4414 (35.03%)	862
sort	27	12601	8590 (68.17%)	3849 (30.55%)	162 (1.29%)	4414 (35.03%)	2493
sort	36	12601	8234 (65.34%)	4277 (33.94%)	90 (0.71%)	4414 (35.03%)	3380
sort	45	12601	7935 (62.97%)	4619 (36.66%)	47 (0.37%)	4414 (35.03%)	3989
bubble	9	41216	35174 (85.34%)	6042 (14.66%)	0 (0.00%)	10140 (24.60%)	0
bubble	18	41216	35109 (85.18%)	6107 (14.82%)	0 (0.00%)	10140 (24.60%)	0
bubble	27	41216	35107 (85.16%)	6116 (14.84%)	0 (0.00%)	10140 (24.60%)	0
bubble	36	41216	34520 (83.75%)	6696 (16.25%)	0 (0.00%)	10140 (24.60%)	0
bubble	45	41216	34478 (83.65%)	6738 (16.35%)	0 (0.00%)	10140 (24.60%)	44
matrix	9	21341	20607 (96.56%)	733 (3.43%)	1 (0.00%)	703 (3.29%)	0
matrix	18	21341	20601 (96.53%)	739 (3.46%)	1 (0.00%)	703 (3.29%)	0
matrix	27	21341	20595 (96.50%)	745 (3.49%)	1 (0.00%)	703 (3.29%)	0
matrix	36	21341	20589 (96.48%)	751 (3.52%)	1 (0.00%)	703 (3.29%)	0
matrix	45	21341	20583 (96.45%)	757 (3.55%)	1 (0.00%)	703 (3.29%)	0
perm	9	54819	42828 (78.13%)	5272 (9.62%)	6719 (2.26%)	10862 (9.81%)	0
perm	8	54819	47857 (87.30%)	5282 (9.64%)	1680 (3.06%)	10862 (19.81%)	0
perm	27	54819	48010 (87.58%)	5129 (9.36%)	1680 (3.06%)	10862 (19.81%)	0
perm	36	54819	49303 (89.94%)	3417 (6.23%)	2099 (3.83%)	10862 (19.81%)	0
perm	45	54819	50321 (91.79%)	2316 (4.22%)	2182 (3.98%)	10862 (9.81%)	0
tower	9	37930	33043 (87.12%)	1305 (3.44%)	3582 (9.44%)	9153 (24.13%)	0
tower	18	37930	32778 (86.42%)	1315 (3.47%)	3837 (10.12%)	9153 (24.13%)	0
tower	27	37930	32701 (86.21%)	1265 (3.34%)	3964 (10.45%)	9153 (24.13%)	0

tower	36	37930	32622 (86.01%)	1280 (3.37%)	4028 (10.62%)	9153 (24.13%)	0
tower	45	37930	32694 (86.20%)	1302 (3.43%)	3934 (10.37%)	9153 (24.13%)	44
queens	9	38462	30511 (79.33%)	7932 (20.62%)	19 (0.05%)	19181 (49.87%)	0
queens	18	38462	31074 (80.79%)	7369 (19.16%)	19 (0.05%)	19181 (49.87%)	0
queens	27	38462	31075 (80.79%)	7368 (19.16%)	19 (0.05%)	19181 (49.87%)	158
queens	36	38462	29061 (75.56%)	9391 (24.42%)	10 (0.03%)	19181 (49.87%)	4612
queens	45	38462	26521 (68.95%)	11931 (31.02%)	10 (0.03%)	19181 (49.87%)	8709
tree	9	32887	28122 (85.51%)	3510 (10.67%)	1255 (3.82%)	8721 (26.52%)	0
tree	18	32887	28188 (85.71%)	3477 (10.57%)	1222 (3.72%)	8721 (26.52%)	0
tree	27	32887	28216 (85.80%)	3547 (10.79%)	1124 (3.42%)	8721 (26.52%)	37
tree	36	32887	27922 (84.90%)	3876 (11.79%)	1089 (3.31%)	8721 (26.52%)	548
tree	45	32887	26016 (79.11%)	5848 (17.78%)	1023 (3.11%)	8721 (26.52%)	3309
puzzle	9	204527	193579 (94.65%)	10946 (5.35%)	2 (0.00%)	18576 (9.08%)	13
puzzle	18	204527	193183 (94.45%)	11342 (5.55%)	2 (0.00%)	18576 (9.08%)	3322
puzzle	27	204527	190037 (92.92%)	14488 (7.08%)	2 (0.00%)	18576 (9.08%)	7349
puzzle	36	204527	187417 (91.63%)	17109 (8.37%)	1 (0.00%)	18576 (9.08%)	10576
puzzle	45	204527	185285 (90.59%)	19241 (9.41%)	1 (0.00%)	18576 (9.08%)	13074

Table 2. Modified GAp Predictor (MGAp) - best predictions

Benchmark	HRg	Number of Branches	Prediction Accuracy	Incorrect Predictions	Incorrect Branch Targets	Not Taken Branches	Number of Replacements (NR)
sort	9	12601	9441 (74.92%)	2929 (23.24%)	231 (1.83%)	4414 (35.03%)	0
bubble	9	41216	35174 (85.34%)	6042 (14.66%)	0 (0.00%)	10140 (24.60%)	0
matrix	9	21341	20607 (96.56%)	733 (3.43%)	1 (0.00%)	703 (3.29%)	0
perm	45	54819	50321 (91.79%)	2316 (4.22%)	2182 (3.98%)	10862 (9.81%)	0
tower	9	37930	33043 (87.12%)	1305 (3.44%)	3582 (9.44%)	9153 (24.13%)	0
queen	18	38462	31074 (80.79%)	7369 (19.16%)	19 (0.05%)	19181 (49.87%)	0
tree	27	32887	28216 (85.80%)	3547 (10.79%)	1124 (3.42%)	8721 (26.52%)	37
puzzle	9	204527	193579 (94.65%)	10946 (5.35%)	2 (0.00%)	18576 (9.08%)	13

Table 3. Conventional GAp Predictor - 100 entries

Benchmark	HRg	Number of Branches	Prediction Accuracy	Incorrect Predictions	Incorrect Branch Targets	Not Taken Branches	Number of Replacements (NR)
sort	1	12601	9354 (74.23%)	3027 (24.02%)	220 (1.75%)	4414 (35.03%)	0
sort	9	12601	7924 (62.88%)	4569 (36.26%)	108 (0.86%)	4414 (35.03%)	3390
bubble	1	41216	35166 (85.32%)	6047 (14.67%)	3 (0.01%)	10140 (24.60%)	0
bubble	9	41216	33499 (81.28%)	7717 (18.72%)	0 (0.00%)	10140 (24.60%)	2640
matrix	1	21341	20613 (96.59%)	725 (3.40%)	3 (0.01%)	703 (3.29%)	0
matrix	9	21341	20577 (96.42%)	763 (3.58%)	1 (0.00%)	703 (3.29%)	0

perm	1	54819	36998 (67.49%)	6060 (11.05%)	11761 (21.45%)	10862 (19.81%)	0
perm	9	54819	48188 (87.90%)	2331 (4.25%)	4300 (7.84%)	10862 (19.81%)	0
tower	1	37930	33432 (88.14%)	1419 (3.74%)	3079 (8.12%)	9153 (24.13%)	0
tower	9	37930	32923 (86.80%)	1348 (3.55%)	3659 (9.65%)	9153 (24.13%)	0
queen	1	38462	3033 (78.88%)	8101 (21.06%)	22 (0.06%)	19181 (49.87%)	0
queen	9	38462	26903 (69.95%)	11546 (30.02%)	13 (0.03%)	19181 (49.87%)	7508
tree	1	32887	28027 (85.22%)	3617 (11.00%)	1243 (3.78%)	8721 (26.52%)	0
tree	9	32887	25308 (76.95%)	6400 (19.46%)	1179 (3.59%)	8721 (26.52%)	3890
puzzle	1	204527	192455 (94.10%)	12070 (5.90%)	2 (0.00%)	18576 (9.08%)	0
puzzle	9	204527	182695 (89.33%)	21832 (10.67%)	0 (0.00%)	18576 (9.08%)	15059

Table 4 Conventional GAp versus MGAp

HRg length	Conventional GAP Average Prediction Success Rate (%)	Conventional GAP Average Prediction Success Rate (%)
1	83.74	-
9	81.43	85.19
18	74.43	86.03
27	66.15	85.39

Table 5 The behaviour of branch 35 belonging to "perm" benchmark

HRg	HRl	Taken (%)	Not taken (%)
101	10	1680 (67%)	839 (33%)
11101	10	1680 (67%)	839 (33%)
11111101	10	840 (59%)	579 (41%)
01011101	10	840 (76%)	260 (24%)

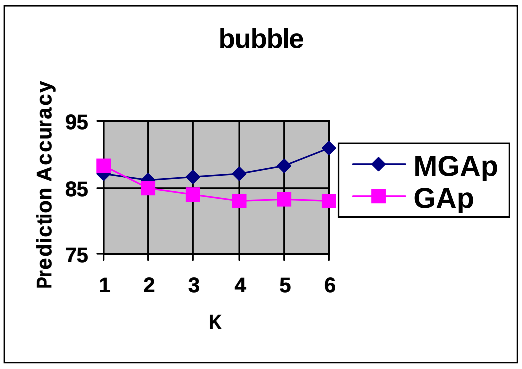

Figure 3. MGAp vs. GAp

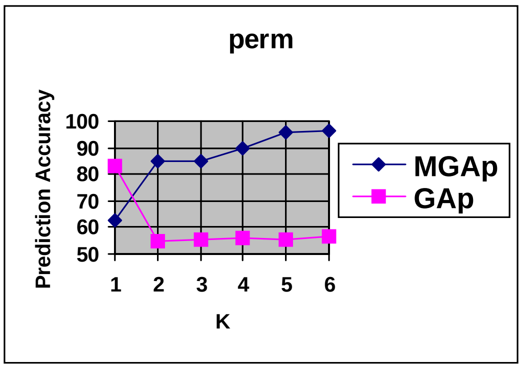

Figure 4. MGAp vs. GAp

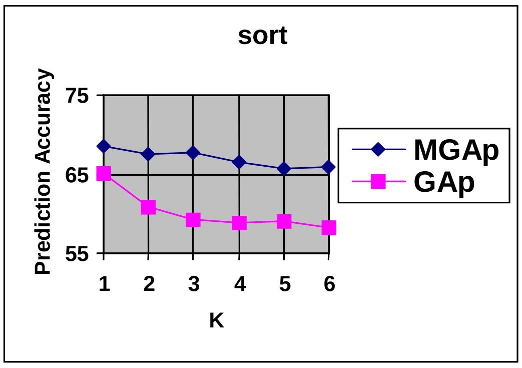

Figure 5. MGAp vs. GAp

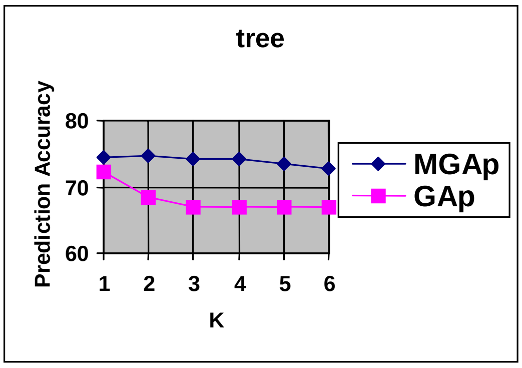

Figure 8. MGAp vs. GAp

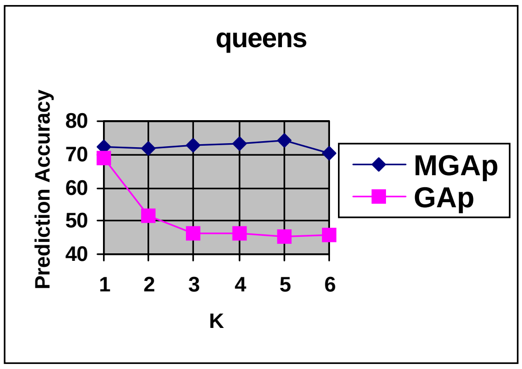

Figure 6. MGAp vs. GAp

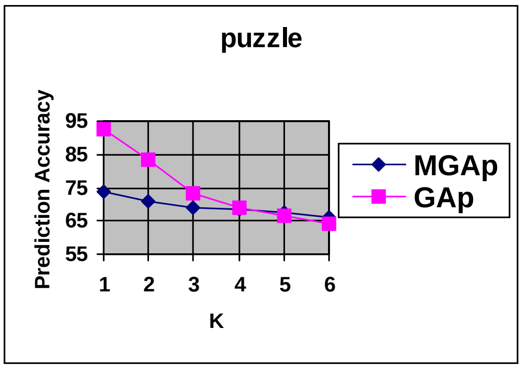

Figure 9. MGAp vs. GAp

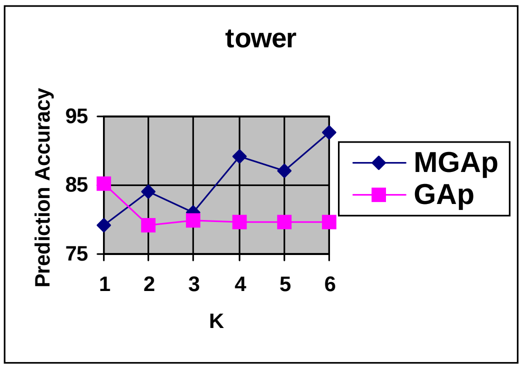

Figure 7. MGAp vs. GAp

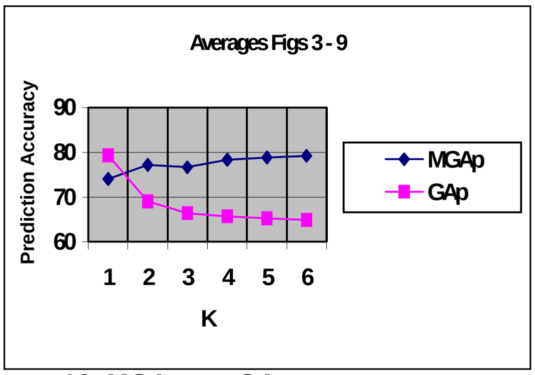

Figure 10. MGAp vs. GAp