2008－ARC－179 (11)
2008／8／6

分岐予測精度改善のための決定的な分岐フィルタ機構

三輪 忍† 中條拓伯†

プログラム中には，常に同じアドレスへと分岐する分岐，すなわち，制御が決定的な分岐が数多く存在する．これらの分岐に関する情報は，グローバル履歴や分岐命令のアドレスの履歴といった，パス情報には必要ない．これらの分岐をパス情報から除けば，同数の分岐履歴でより長いパスを表すことができ，分岐予測器などの精度が改善すると期待できる．そこで本稿では，これらの分岐をフィルタする機構を提案する．提案機構を gshare 予測器に適用した結果，競合の影響が大きく，予測精度はほとんど改善しないことがわかった．

Deterministic Branch Filter Mechanism for Improving Branch Prediction Accuracy

Shinobu Miwa† and Hironori Nakajo†

There are a lot of branches which always jump the constant address in the program, which are called *deterministic branches*. Their information is not required by path information such as a global branch history or a history of branch address. Their information is eliminated from path information, then, a shorter history can represent a longer path. If a branch predictor uses path information without information of deterministic branches, the prediction accuracy would be improved. Therefore, we propose a mechanism to eliminate these branches. The proposal mechanism is adapted to gshare branch predictor, then, the prediction accuracy is hardly improved because of a lot of interferences.

1. はじめに

近年の集中的な研究により，**分岐予測**のヒット率は大幅に向上した．しかし，その予測精度はアプリケーションの性質に強く依存する．たしかに，多くのアプリケーションではほぼ 100% 近いヒット率が示される．しかしその一方で，9 割程度のヒット率しか得られないアプリケーションの存在も，決して例外的というわけではない．そのため，分岐予測のヒット率の改善は，いまだにマイクロアーキテクチャにおける重要な課題であり続けている．

分岐予測には，さまざまな**実行パス情報**が利用される[3),4)]．**gshare 予測器**[5)]では，**PHT**（Pattern History Table）と呼ばれるテーブルのインデクスの一部に，**グローバル分岐履歴**（global branch history）を用いている．予測器によっては，グローバル履歴だけでなく，**アドレスの履歴**も利用するものもある[4)]．

予測に利用するパス情報の長さは，予測精度とハードウェア量の双方に多大な影響を与える[3)]．高い予測精度を実現するには，パス情報は基本的には長い方がよい．予測する分岐と相関のある分岐に関する情報，すなわち，予測に有用な情報を含む可能性が増すからである．その一方で，パス情報は PHT のインデクスなどに利用されるため，パス情報を長くすると，それにともなってハードウェア量も増加してしまう．

ハードウェア量を抑えつつ高い予測精度を実現するには，パス情報を保持する**グローバル履歴レジスタ**（Global History Register. 以下 **GHR** とする）などのバッファは，有用な情報だけを保持することが望ましい．

プログラム中には常に同じアドレスへと分岐する分岐，すなわち，制御が**決定的な分岐**（deteriministic branch）[8)]が数多く存在する．決定的な分岐は，基本的には，パス情報に必要ない．というのは，簡単に言ってしまえば，他の制御が決定的な命令，すなわち，分岐命令以外の命令と同じと考えてよいからである．決定的な分岐をパス情報から取り除けば，パスの持つ意味を維持しつつ，同数の分岐履歴でより長いパスを表すことができる．そうした履歴を利用すれば，予測器はより高い予測精度を実現できるだろう．

本稿の構成は以下の通りである．まず次章において，決定的な分岐がパス情報にとって不要なこと，そして，それをパス情報から取り除くことによって分岐予測精度が改善することを詳しく説明する．続く 3 章において提案する決定的な分岐フィルタ機構を説明し，その評価を 4 章で行う．5 章で関連研究について述べ，6 章でまとめる．

† 東京農工大学
Tokyo University of Agriculture and Technology

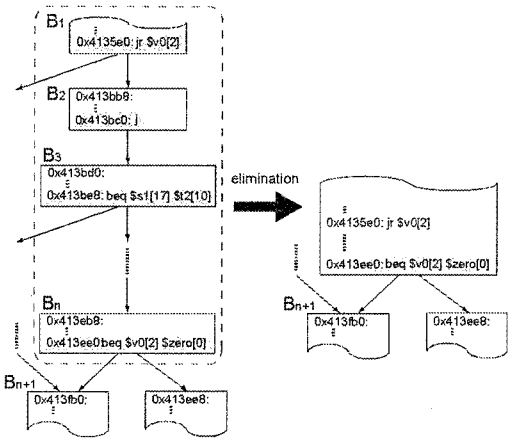

図 1 決定的な分岐の除去の持つ意味

2. 決定的な分岐

プログラムには，常に同じアドレスへと分岐する分岐，すなわち，決定的な分岐が存在する．direct jump はその代表例である．決定的な分岐は，次に実行すべき命令をただ一つに定める．そのような意味において，決定的な分岐は分岐命令以外の命令と等価である．

決定的な分岐は，以下の 3 つに分類される．

静的なもの

DIRECT direct jump.

動的なもの

UNTAKEN 常に成立しない分岐．後述するように，極端な偏りのある不成立分岐命令ともいう[6]．

UNI-TARGET 常に成立し，かつ，常に同じアドレスへと分岐する条件分岐，および，無条件分岐．前者は，後述するように，極端な偏りのある成立分岐命令とも呼ばれる[6]．

以下ではまず，これらの分岐をパス情報から取り除くことの持つ意味について詳しく説明する．続いて，そうした分岐がプログラム中にどの程度存在するかを示し，最後に，それらを除いた場合，理想化された予測器の精度がどの程度改善するかを示す．

2.1 決定的な分岐の除去が持つ意味

決定的な分岐は，次に実行される命令が常に同じである．そのため，制御フロー上では，基本的には分岐命令以外の命令と同じと考えてよい．

図 1 に例を示す．左図は，SPEC CINT 2000 ベンチマークの *mcf* の制御フロー・グラフの一部である．コロンの左側は命令のアドレスを，右側は命令を表している．簡単のため，図には分岐命令のみ記載してある．命令 j, jr, beq は，それぞれ，jump 命令，jump register 命令，branch equal 命令である．したがって，"0x4135e0 : jr $v0[2]"は，アドレス "0x4135e0"は，レジスタ "$v0[2]" を使用する jump register 命令であることを表す．

実行パス，すなわち，基本ブロックの系列は，分岐命令の系列によって代表できる．例えば，基本ブロックが $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow \cdots$ と実行され，B_n に至ったとしよう．このような実行パスは，分岐命令のアドレスの履歴などを用いて，0x4135e0 → 0x413bc0 → 0x413be8 → ⋯ と表される．逆に，パス情報がこのようなものだった場合は，$B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow \cdots$ と実行されたことに他ならない．

ここで，分岐命令 0x4135e0，0x413bc0，0x413be8 が決定的だったとしよう．これらの分岐に代表される基本ブロック $B_1, B_2, B_3, \cdots$ は，常にこの順序で実行され，他のブロックへ制御が移ることがない．そのため，右図のように，これらの基本ブロックを 1 つのトレースとし，それを末端の基本ブロック B_n の分岐 0x413ee0 で代表してよい☆．このようにすれば，トレースを表現するために必要だった複数個の分岐は，1 つで済むことになる．

こうした変換は，静的に行うのは難しいことに注意されたい．次節で示すように，決定的な分岐の多くは動的に決定的である．より多くのトレースについてこのような変換を行うためには，分岐についての動的な情報が必要となる．

このように，パス情報から決定的な分岐を除くという操作には，パスの持つ情報を維持しつつ，パスを代表する分岐を減らす意味がある．

非決定的な分岐の除去

その一方で，非決定的な分岐を除くと，トレースの持つ情報は失われてしまう．上述の制御フロー・グラフにおいて，非決定的な分岐 0x413ee0 を除くと，基本ブロック B_n の次にどこへ制御が移るのかわからなくなる．仮に，B_{n+1} に制御が移ったとし，そのようなトレースを B_{n+1} の分岐で代表したとしよう．すると，そのトレースとは別の，他から B_{n+1} に至ったパスと区別がつかなくなってしまう．

グローバル履歴上では，決定的か否かによらず，すべての無条件分岐が常に同じ値を示す．そのため，すべての無条件分岐をパス情報から除くという，一見妥当なポリシーも考えられる．しかし，このポリシーには上述のような問題があり，その影響は 4 章で評価する．

2.2 決定的な分岐の割合

SPEC CINT 2000 の 11 本のベンチマーク・プログラムにおける決定的な分岐の割合を**図 2** に示す．グラ

☆ 紙面の都合で詳しく述べることはできないが，決定的な分岐をもつ基本ブロックでも統合できないケースもある．本来，そうしたケースでは決定的な分岐の除去を控えるべきである．ただ，そうしたケースは稀であり，除去を限定しても予測精度はほとんど変わらなかったため，本稿では考慮しない．

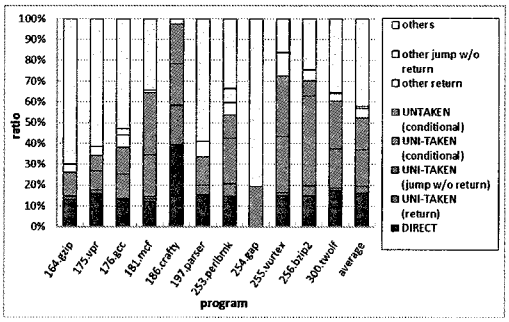

図 2 決定的な分岐の割合

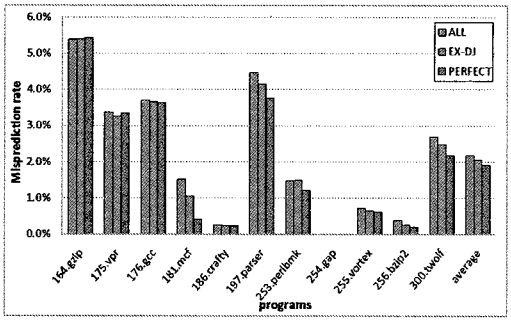

図 3 決定的な分岐がすべて除かれた場合の予測精度（履歴長：17）

フの横軸はプログラム（一番右は平均）を，縦軸は決定的な分岐の割合を表す．ただし，図の割合は動的分岐に対するものである．測定環境は 4.1 節で述べる環境と同様とした．

グラフより，静的に決定的な分岐（DIRECT）は，ほとんど存在しないプログラムも一部あるが，全体としては大きな割合を占めている．その割合は，最も多い *crafty* では 39.2% にものぼる．平均すると，16.2% の分岐が静的に決定的である．

それに対し，動的に決定的な無条件分岐（UNI-TARGET（return, jump））の割合はそれ程大きくない．一部のプログラムでは UNI-TARGET（return）が 5 ~ 20% 程度存在するものの，動的に決定的な無条件分岐の割合は平均すると 3.44% に過ぎない．

ただし，この割合は今後増加する可能性がある．評価に用いたプログラムは，すべて C 言語で記述されており，オブジェクト指向言語で記述されたものは 1 つもない．プログラム中の間接分岐の割合は，仮想関数の使用などにより，後者の方が多いことが知られている．そのため，オブジェクト指向言語が普及するにつれ，動的に決定的な無条件分岐の割合も増加すると考えられる．

動的に決定的な条件分岐命令（UNTAKEN, UNI-TARGET（conditional））もまた，大きな割合を占める．*gzip* では 11.2% と少ないものの，最も多い *vortex* では 55.8% にものぼる．平均では 32.6% の条件分岐が動的に決定的である．

合計すると，決定的な分岐の割合は 52.2% であった．このように，およそ半分の分岐は決定的である．

2.2.1 決定的な分岐の除去が予測精度に与える影響

履歴長 17 の ideal gselect 予測器において，グローバル履歴から決定的な分岐を取り除いた場合の予測精度を**図 3** に示す．3 本の棒グラフは，左から順に，全分岐の履歴を用いた場合（ALL），DIRECT のみ除いた場合（EX-DJ），そして決定的な分岐すべてを除いた場合（PERFECT）を表す．

グラフより，決定的な分岐を除くことで，ほとんどのプログラムで予測精度が改善する．特に，*mcf* においては，ALL と比べ 1.09 ポイント，EX-DJ と比べ 0.63 ポイント向上する．平均では，ALL，EX-DJ と比べ，それぞれ，0.26，0.14 ポイント向上する．

このように，競合がなければ，決定的な分岐をパス情報から除くことで予測精度は改善する．それでは次は，これらの分岐をフィルタする機構について述べる．

3. 決定的な分岐フィルタ機構

前章で述べたように，決定的な分岐は以下の 3 つに分類される．

静的なもの

DIRECT direct jump.

動的なもの

UNTAKEN 常に成立しない分岐．

UNI-TARGET 常に成立し，かつ，常に同じアドレスへと分岐する分岐．

これら 3 つを，それぞれ以下のように検出する．

静的なもの

DIRECT プリデコードによって検出する．

動的なもの

UNTAKEN BTB ミスによって検出する．BTB には，過去に成立したことがない分岐は登録されない．BTB ミス率は 1% 以下と低いため，BTB にミスする分岐は常に成立しないものとしてよい[7]．

UNI-TARGET return 以外の決定的な無条件分岐/条件分岐は，以下で述べるように，BTB を拡張して検出する．簡単のため，決定的な return は本稿では検出の対象としない．

提案する機構の構成を**図 4** に示す．図は，決定的な分岐をグローバル履歴から取り除く場合のみを示しているが，アドレスの履歴から除く場合も同様である．

UNI-TARGET を検出するため，当該分岐が決定的であるか否かを表す 1b のフラグ（D-flag）を BTB に追加する．1 は決定的であることを，0 はないことを表す．フラグの初期値は 1 とする．

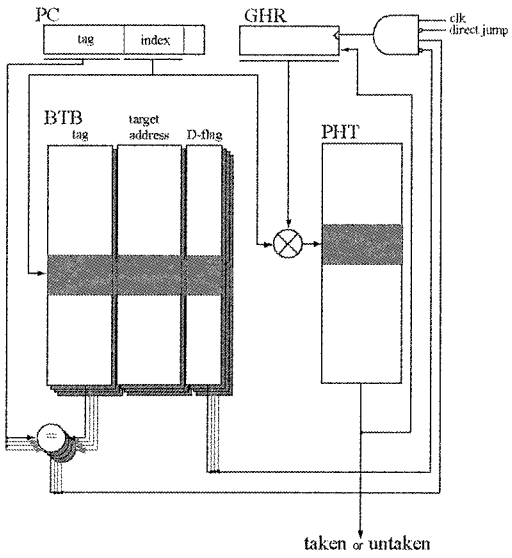

図 4　決定的な分岐フィルタ機構

表 1　各モデルのハードウェア量毎に最適な履歴長

	8KB	16KB	32KB	64KB
ALL	15	16	17	18
EX-DJ	15	16	17	17
EX-DJ&DB	10	11	12	13
EX-NC&DB	10	10	11	13
EX-J&DB	10	10	11	12
EX-DB	10	11	12	13
PERFECT	10	11	12	13

フラグは，分岐が決定的ではないことが判明した時点で更新する．すなわち，分岐方向が（成立から不成立に）変わった場合，または，BTB に登録されている分岐先アドレスが更新された場合にフラグを 0 にする．このようにすれば，BTB のエントリがリプレースされない限り，フラグが 1 の時は常に，その時点まではその分岐が決定的だったことになる．

分岐を取り除くため，本機構の GHR の入力クロックは，上述のようにして検出した信号とクロックとを AND する．このようにして，提案機構は決定的な分岐が GHR に登録されるのを防ぐ．

4. 評　　価

提案機構をシミュレータに実装し，評価を行った．以下ではその結果を述べる．

4.1 評価環境

以下の 7 つのモデルについて評価する．

ALL　全分岐の履歴を用いて予測するモデル．

EX-DJ　DIRECT を除いた履歴を用いるもの．

EX-DJ&DB　DIRECT，および，決定的な条件分岐命令を除いたもの．

EX-NC&DB　上述の履歴からさらに，call 以外の無条件分岐命令を除いたもの．非決定的な無条件分岐であっても，それが call 以外ならば除かれる．後述するように，予測器によっては，このような履歴を用いた方がよいことが報告されている[6]．

EX-J&DB　さらに call 命令も除いたもの．すなわち，履歴は，決定的か否かによらず，無条件分岐を一切含まない．

EX-DB　すべての決定的な分岐を除いたもの．EX-J&DB とは異なり，非決定的と検出された無条件分岐は履歴に含まれる．

PERFECT　決定的な分岐を完全に取り除いたもの．この性能が本提案の上限を与えることになる．

予測器は gshare とする．

表 1 に，各モデルのハードウェア量毎に最適化された履歴長を示す．ただし，ハードウェア量は予測器のそれのみを表す．提案機構のハードウェア量は，BTB のエントリが 1b 増加するだけと軽微なため，以降の評価に含めていない．

提案機構による決定的な分岐の検出能力は，BTB のヒット率に依存する．そこで次節では，BTB の set 数を 0.5K から 16K まで変えて評価する．なお，way 数はいずれの場合も 4 とする．

本機構は，フィルタ機構など[1),6),7)]と同様，PHT における競合を緩和するためにも利用できる．条件分岐命令が BTB を参照し，そのフラグが 1 だった場合，PHT を参照することなく，次も同じ分岐先に分岐するものとしてよい．フラグによる予測がヒットし続ける限り，その分岐は PHT を更新する必要がない．

こうした利用も可能であるが，本稿ではこのような分岐も PHT によって予測する．これは，決定的な分岐を除く効果のみを評価するためである．なお，無条件分岐については，通常通り，PHT を使用することなく成立と予測する．

上述のモデルを SimpleScalar ツールセット (ver. 3.0) の sim-bpred シミュレータに実装し，評価を行った．測定には，SPEC CINT2000 の 11 本のプログラムを使用し，入力セットには train を用いた．プログラムは先頭から 100M 命令を実行した．

4.2 評価結果

最適な BTB サイズ

BTB のセット数を 0.5K ~ 32K まで変えた時の，提案機構を用いたモデル（EX-DB）の予測精度を**図 5** に示す．グラフの横軸はハードウェア量，縦軸は予測ミス率である．

グラフより，BTB サイズが増加するにつれ，決定的な分岐の検出能力が向上し，その結果，予測精度が改善しているのがわかる．そして，予測精度の改善は 8K set でほぼ上限に達する．set 数が 8K の場合の予測精度と PERFECT のそれとの差は，64KB のハードウェ

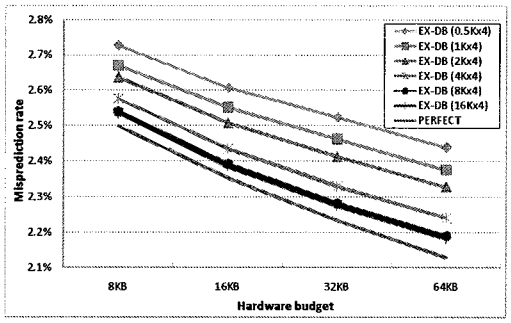

図 5 BTB サイズに対する予測精度

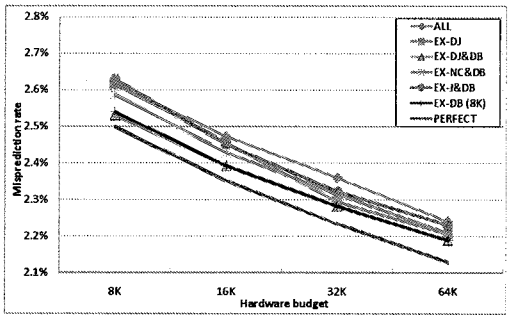

図 6 各モデルの予測精度

ア量においては 2.86%，8KB では 1.64% であった．

現在のプロセッサの BTB は，set 数が 0.5K 程度であり，それと比べると 8K set はあまり現実的でない．しかし，BTB は，Intel の次世代プロセッサ Nehalem では階層化される予定[2] であり，今後大容量化が進むと予想される．8K set の BTB は，下位のそれとして将来的に実現される可能性はある．

各モデルの予測精度

前節で述べた，7 つのモデルの予測精度を**図 6** に示す．グラフの見方は先程と同様である．BTB の set 数はすべて 8K とした．

グラフより，PERFECT を除く 6 つのモデルの中では，ほとんどのハードウェア量において，決定的な分岐すべてを除いたもの（EX-DB）が最も高い予測精度を示す．特に，Alpha 21264 の予測器と同程度のハードウェア量（8KB）においては，すべての分岐の履歴を用いた場合（ALL）と比べて 2.85%，DIRECT のみを除いた場合（EX-DJ）と比べて 2.92% 改善している．また，EX-NC&DB，EX-J&DB と比べ，それぞれ，1.83%, 3.39% 改善する．

ただし，その他のハードウェア量では，他のモデルとの差はほとんどない．64KB においては，EX-DJ と比べて 0.74% の改善に留まっている．Alpha 21464 の予測器と同程度のハードウェア量（32KB）においては効果がさらに薄く，0.55% 改善しているに過ぎない．

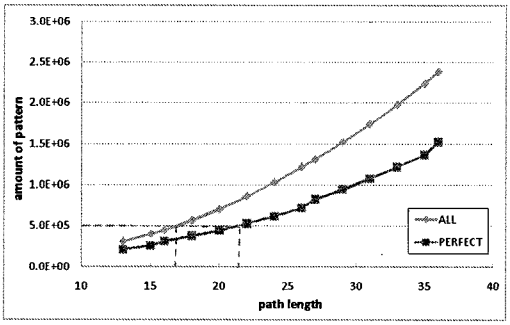

図 8 パスの長さとパターン数

この程度の改善率なら EX-DJ で十分である．

また，ideal gselect の場合とは異なり，現実の gshare においては，決定的な分岐を完全に取り除いた（PERFECT）としても予測精度はあまり変わらない．32KB では予測精度が，ALL と比べて 0.13 ポイント，EX-DJ と比べて 0.061 ポイント向上しているだけである．

このように，決定的な分岐をすべて除き，少ない分岐で長いパスを表現できるようになっても，予測精度は数 % 程度しか改善しない．これは，後述するように，決定的な分岐を除くと，単純に履歴を長くした場合と同様，競合が増加してしまうためである．競合を除去以前と同程度に抑えようとすれば，結局は以前と同程度の長さのパスしか表現できない．

各プログラムにおける効果

32KB における，プログラム毎の予測ミス率を**図 7** に示す．横軸はプログラム（一番右は平均）を，縦軸はミス率を表す．プログラム毎の 7 本の棒グラフは，左から順に，前節で述べた 7 つのモデルに対応する．

グラフより，決定的な分岐の除去は，一部のプログラムで影響が大きいものの，平均するとほとんど効果がない．*mcf* のように 1.06 ポイント精度が改善するものがある一方で，*gcc* のように 0.68 ポイント悪化するものもある．しかし，平均すると，ALL と EX-DB との差は 0.078 ポイントしかない．EX-DJ との差はさらに小さく，0.013 ポイントに過ぎない．

パスの長さとパターン数

ALL，および，PERFECT における，履歴が表しているパスの長さと（分岐，履歴）の組み合せパターン数との関係を**図 8** に示す．決定的な分岐を除いた履歴は，前述のように，実際には履歴長以上の長さのパスを表している．同じ履歴長であっても，パス上の決定的な分岐の数により，履歴が表すパスの長さにはばらつきがあるため，PERFECT では履歴長毎のパス長を中央値で代表している．

グラフより，どちらの履歴もパスが長くなるにつれ，組み合わせパターン数も増加する．ただし，増加速度は異なり，PERFECT の方が緩やかな傾向を示す．表 1 に示したように，32KB の ALL においては，履歴長

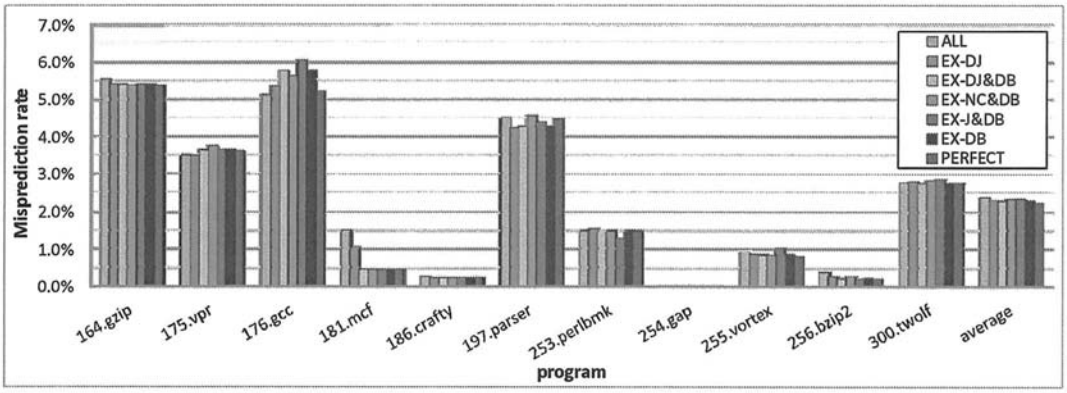

図 7 各プログラムにおける予測精度（32KB）

は 17 が最適である．それと競合の発生率，すなわち，組み合わせパターン数を同程度に抑えようとすれば，PERFECT の履歴を，長さ 21 ~ 22 のパスを表す程度にまで短くしなければならない．このように，履歴が表す実際のパス長がほとんど変わらないため，決定的な分岐を除いても予測精度はほとんど変わらない．

5. 関連研究

決定的な分岐を利用した予測器に，吉瀬らの Bimode-Plus 予測器[6] がある．吉瀬らの定義では，常に成立する/しない条件分岐命令を，それぞれ，極端な偏りのある成立/不成立分岐命令という．そして，Bimode 予測器に以下の 2 つの改良を施している．

(1) 極端な偏りのある分岐を予測器による予測の対象から外すことで，予測器における競合を緩和する．
(2) call 以外の無条件分岐命令，および，極端な偏りのある分岐を GHR に登録しないことで，予測精度を改善する．

まず，(1) については，本稿とは目的が異なる．4.1 節で述べたように，本機構でも競合を緩和することは可能である．しかし，本稿を通して述べてきたように，我々の目的は競合の緩和にあるのではない．決定的な分岐をパス情報から取り除くことの効果を確かめることにある．

(2) については，2.1 節で述べたように，非決定的な無条件分岐を除くことは，定性的には問題がある．非決定的な無条件分岐も除くのであれば，評価の項で示したように，その差がわずかとはいえ，決定的な無条件分岐のみを除いた方がよい．

なにより文献 6) では，2 章で述べたような，決定的な分岐をパス情報から取り除く意味や効果は，ほとんど述べられていない．これらの点について詳述し，実験的に確かめたという点で，吉瀬らの研究と本研究はまったく異なる．

6. おわりに

プログラム中には決定的な分岐が多数存在する．決定的な分岐は，分岐命令以外の命令と同様，パス情報には必要ない．これらの分岐をパス情報から除けば，より短いパス情報でより長いパスを表すことができ，分岐予測精度などが改善すると期待できた．

そこで本稿では，決定的な分岐をフィルタする機構を提案した．提案機構を gshare 予測器に適用した結果，競合の影響が大きく，すべての決定的な分岐を除いても予測精度はほとんど改善しないことがわかった．分岐予測精度を改善するには，パス情報を工夫するよりも，競合の影響を減らすことが重要である．

謝辞 本研究の一部は文部科学省 共生情報工学推進経費による．

参考文献

1) Chang, P. Y. et al.: Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference, *PACT96*, pp. 48–57 (1996).
2) Intel Corp.: *First the Tick, Now the Tock : Next Generation Intel Microarchitecture (Nehalem)* (2008).
3) Jiménez, D. A. et al.: Dynamic Branch Prediction with Perceptrons, *HPCA-7*, pp. 197–206 (2001).
4) Jiménez, D. A. et al.: Fast Path-Based Neural Branch Prediction, *MICRO-36*, pp. 243–252 (2003).
5) McFarling., S.: Combining branch predictors, Wrl technical report tn-36, Digital Equipment Corporation (1993).
6) 吉瀬謙二ほか: Bimode-Plus 分岐予測器の提案, 情報処理学会論文誌, ACS 10, pp. 85–102 (2005).
7) 斎藤史子ほか: BTB のエントリ有無を参照した分岐予測器, *SACSIS2004*, pp. 261–268 (2004).
8) 三輪忍ほか: 圧縮されたパス情報を用いた分岐予測手法, *SACSIS2008*, pp. 255–263 (2008).