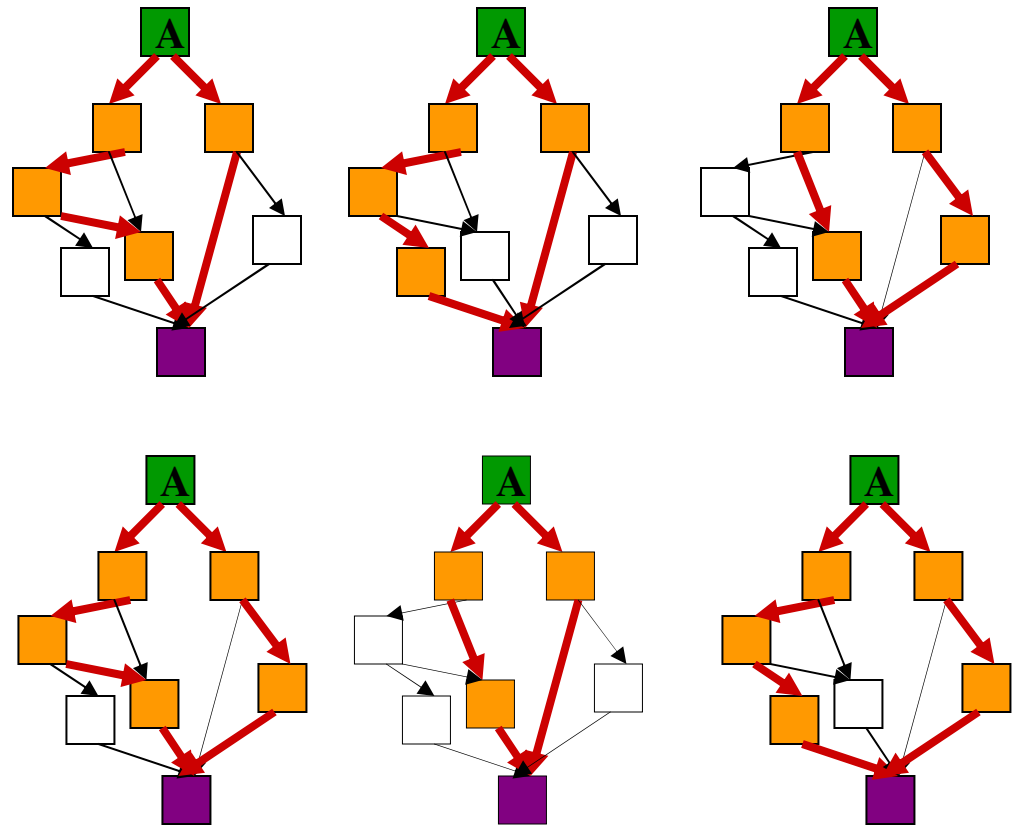
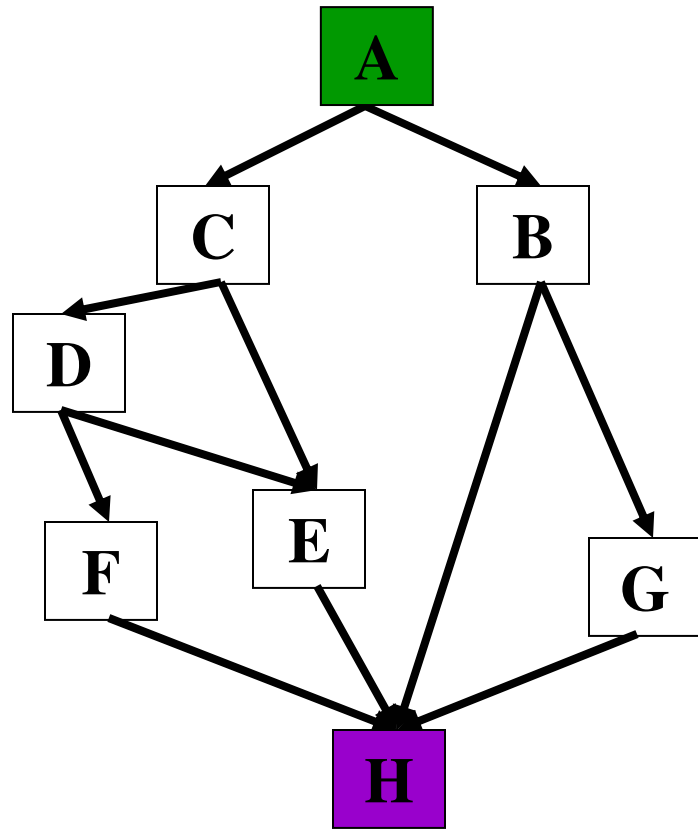


- Frequently executed path
- Not frequently executed path
- Hard to predict path

Diverge-Merge Processor (DMP)

[MICRO 2006, TOP PICKS 2007]



→ Frequently executed path **■ diverge-branch** **■ executed block** **■ CFM point**
→ Not frequently executed path

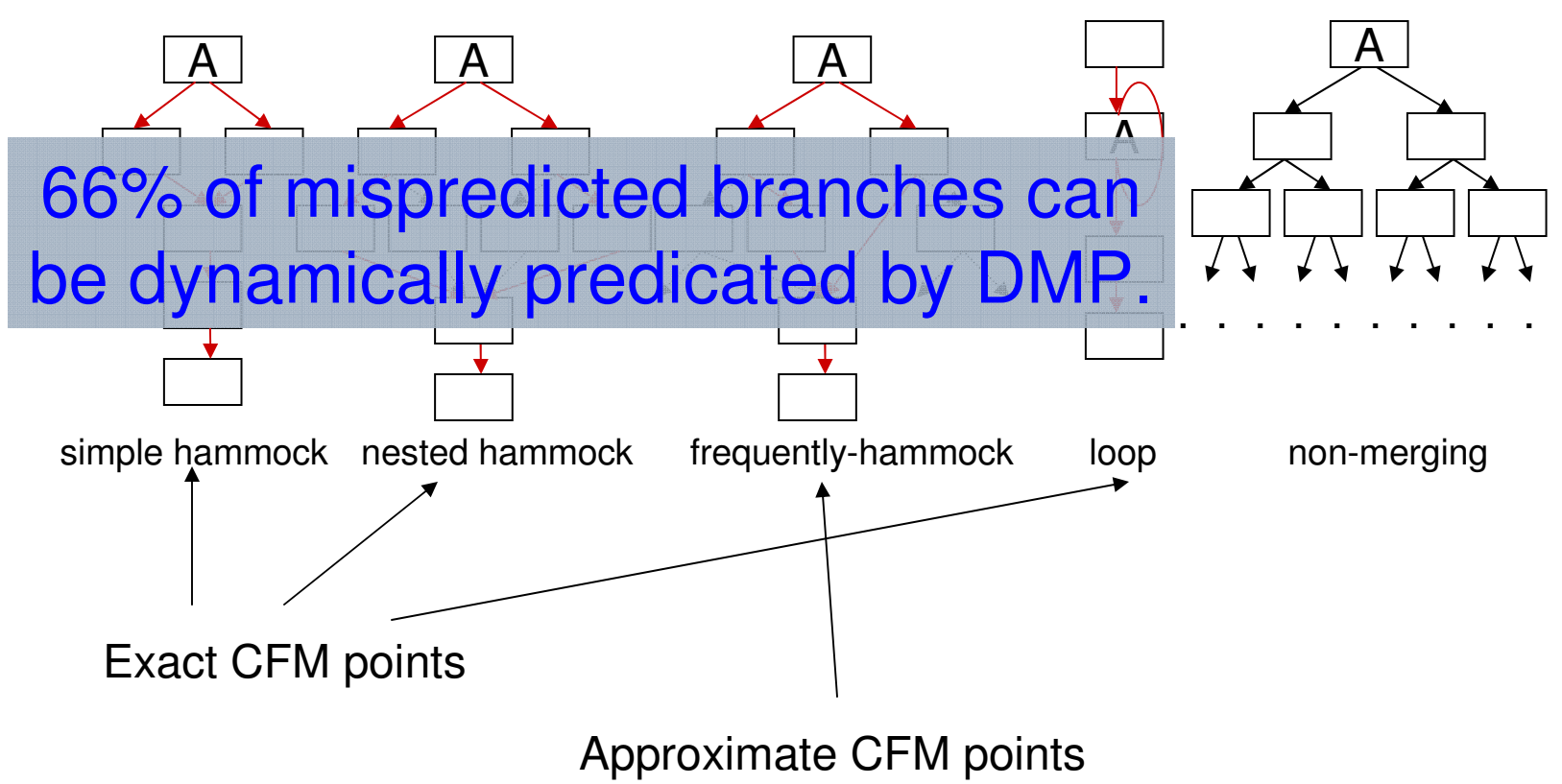
Profile-Assisted Compiler Support for Dynamic Predication in Diverge-Merge Processors

Hyesoon Kim
José A. Joao
Onur Mutlu*
Yale N. Patt

HPS Research Group
University of Texas at Austin



Control-Flow Graphs



Diverge-Merge Processor (DMP)

- DMP can dynamically predicate **complex branches** (in addition to simple hammers).
- The **compiler identifies**
 - Diverge branches
 - Control-flow merge (CFM) points
- The **microarchitecture decides when** and **what** to predicate dynamically.

Why hardware *and* compiler?

- ❑ Compiler-centric solution (static predication):
predicated ISA, not adaptive,
applicable to limited CFG.

- ❑ Microarchitecture-only solution:
complex, expensive, limited in scope.

- ❑ Compiler-microarchitecture interaction
 - Each one does what it is good at.

Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points

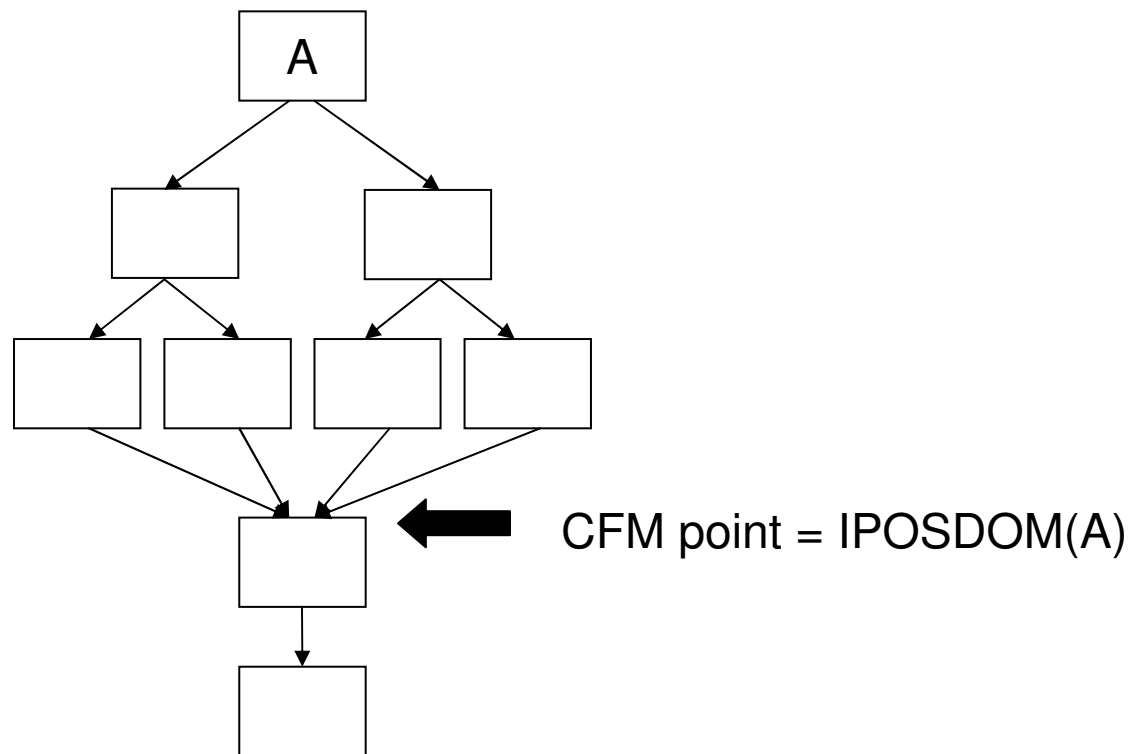


Select Diverge Branches and CFM points

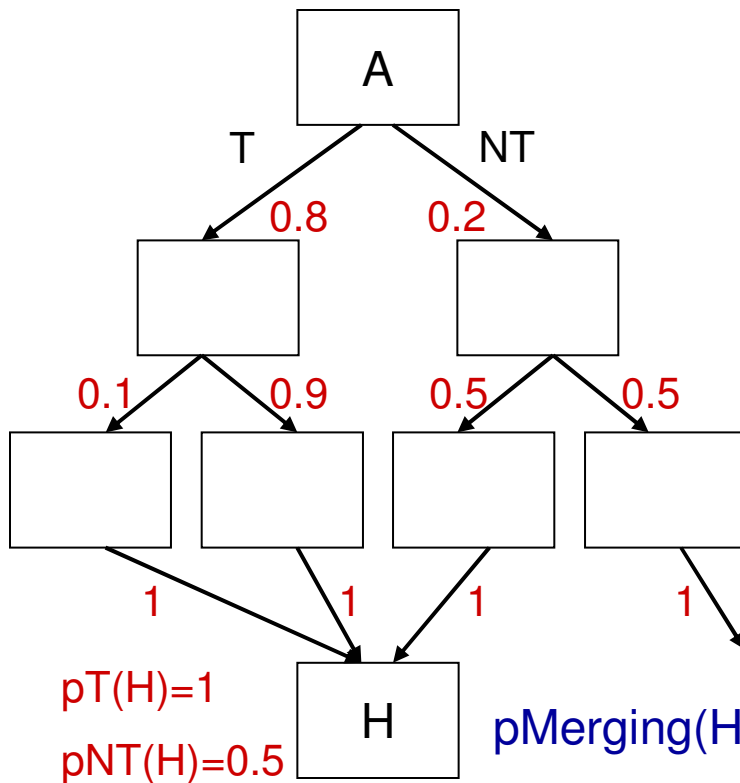


Code generation: mark the selected diverge branches and CFM points (ISA extensions)

Simple/Nested Hammocks: Alg-exact



Frequently-Hammocks: Alg-freq



- Use edge-profiling frequencies

$pT(X)$: conditional probability of reaching basic block X if branch A is taken

$pNT(X)$: conditional probability of reaching basic block X if branch A is not taken

- Stop at IPOSDOM or MAX_INSTR

- Compute $pMerging(X) = pT(X) * pNT(X)$

$pMerging(H) = 1 * 0.5 = 0.5$ ← CFM point candidate !

Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points

- Simple/nested hammocks
- Frequently-hammocks
- Chains of CFM points
- Return CFM points
- Short hammocks
- Diverge loop branches



Select Diverge Branches and CFM points

Code generation: mark the selected diverge branches and CFM points (ISA extensions)

Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points



Select Diverge Branches and CFM points

- Heuristics

- Cost-benefit model

Code generation: mark the selected diverge branches and CFM points (ISA extensions)

Heuristics-Based Selection

Motivation:

minimize overhead: wrong path

maximize benefit: control-independence

Do not select:

- CFM points too far from the diverge-branch
- Hammocks with too many branches
on each path
- Approximate CFM points with
a low probability of merging

Cost-Benefit Model-Based Selection

Confidence	Estimation	Cost	Benefit	Compared to baseline
High	-	-	-	Same
Low	Inaccurate	Overhead	None	Lose
	Accurate	Overhead	No flush	possibly WIN

$A = \text{accuracy of the conf. estimator} = \text{mispredicted} / \text{low_conf.}$

$$\text{cost} = (1 - A) \times \text{overhead} + A \times (\text{overhead} - \text{misprediction_penalty})$$

Cost-Benefit Model-Based Selection

Select if $cost < 0$

$A = accuracy\ of\ the\ conf.\ estimator = mispredicted / low_conf.$

$$cost = (1 - A) \times overhead + A \times (overhead - misprediction_penalty)$$

Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points



Select Diverge Branches and CFM points

- Heuristics
- Cost-benefit model



Code generation: mark the selected diverge branches and CFM points (ISA extensions)

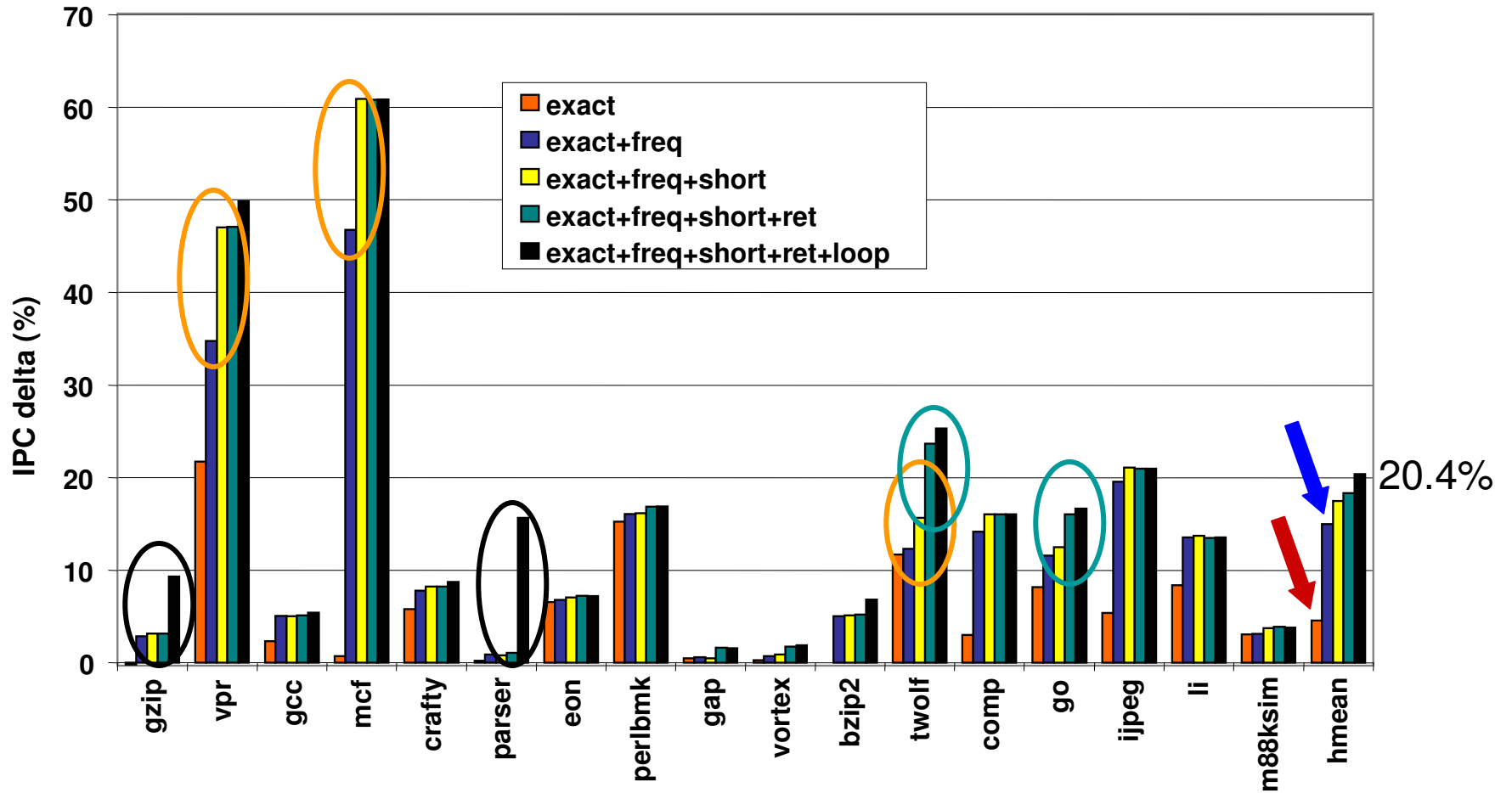


Methodology

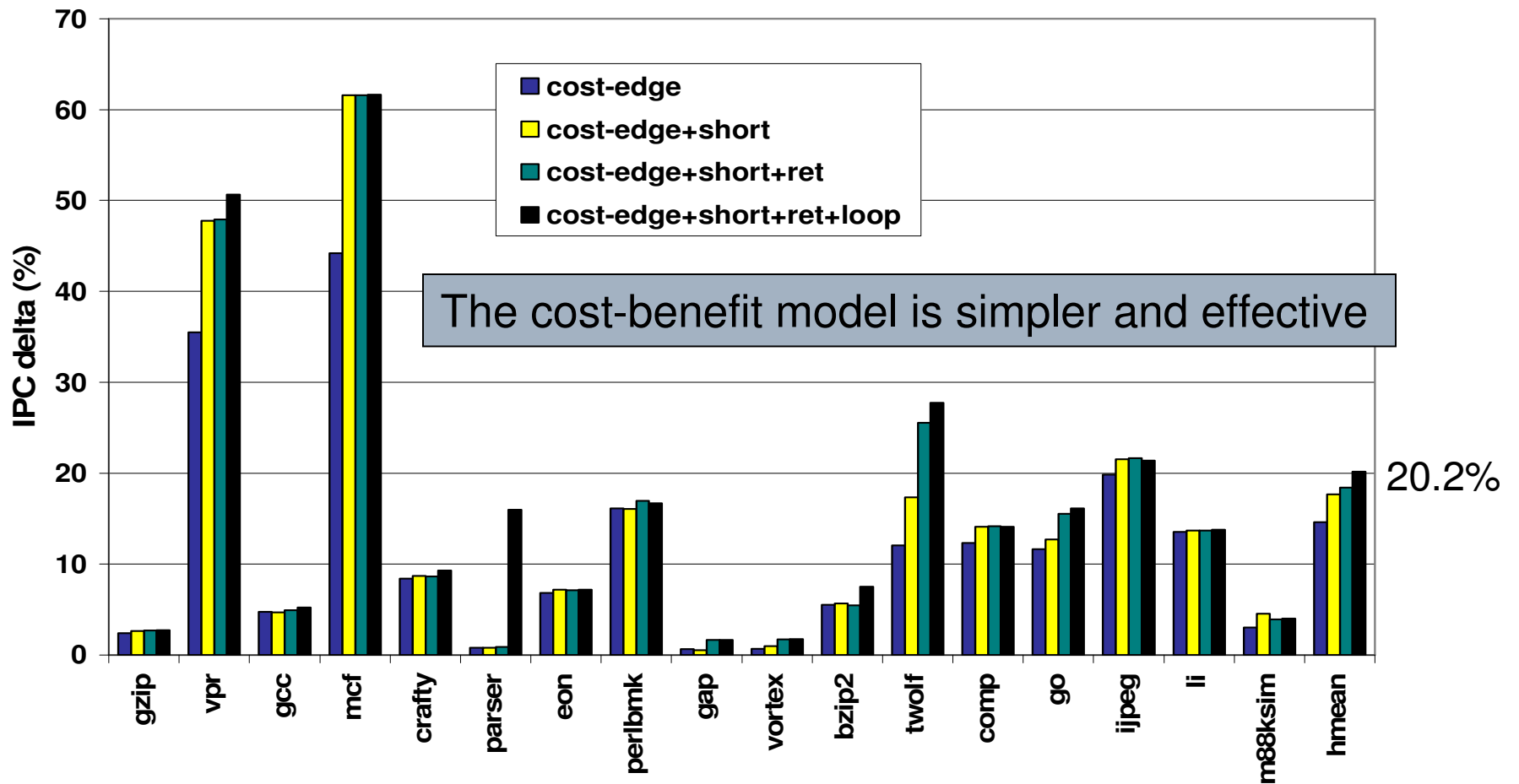
- All compiler algorithms implemented on a binary analysis and annotation toolset.

- Cycle-accurate execution-driven simulation of a DMP processor:
 - Alpha ISA
 - Processor configuration
 - 16KB perceptron predictor
 - Minimum 25-cycle branch misprediction penalty
 - 8-wide, 512-entry instruction window
 - 2KB 12-bit history enhanced JRS confidence estimator
 - 32 predicate registers, 3 CFM registers
 - 12 SPEC CPU 2000 INT, 5 SPEC 95 INT

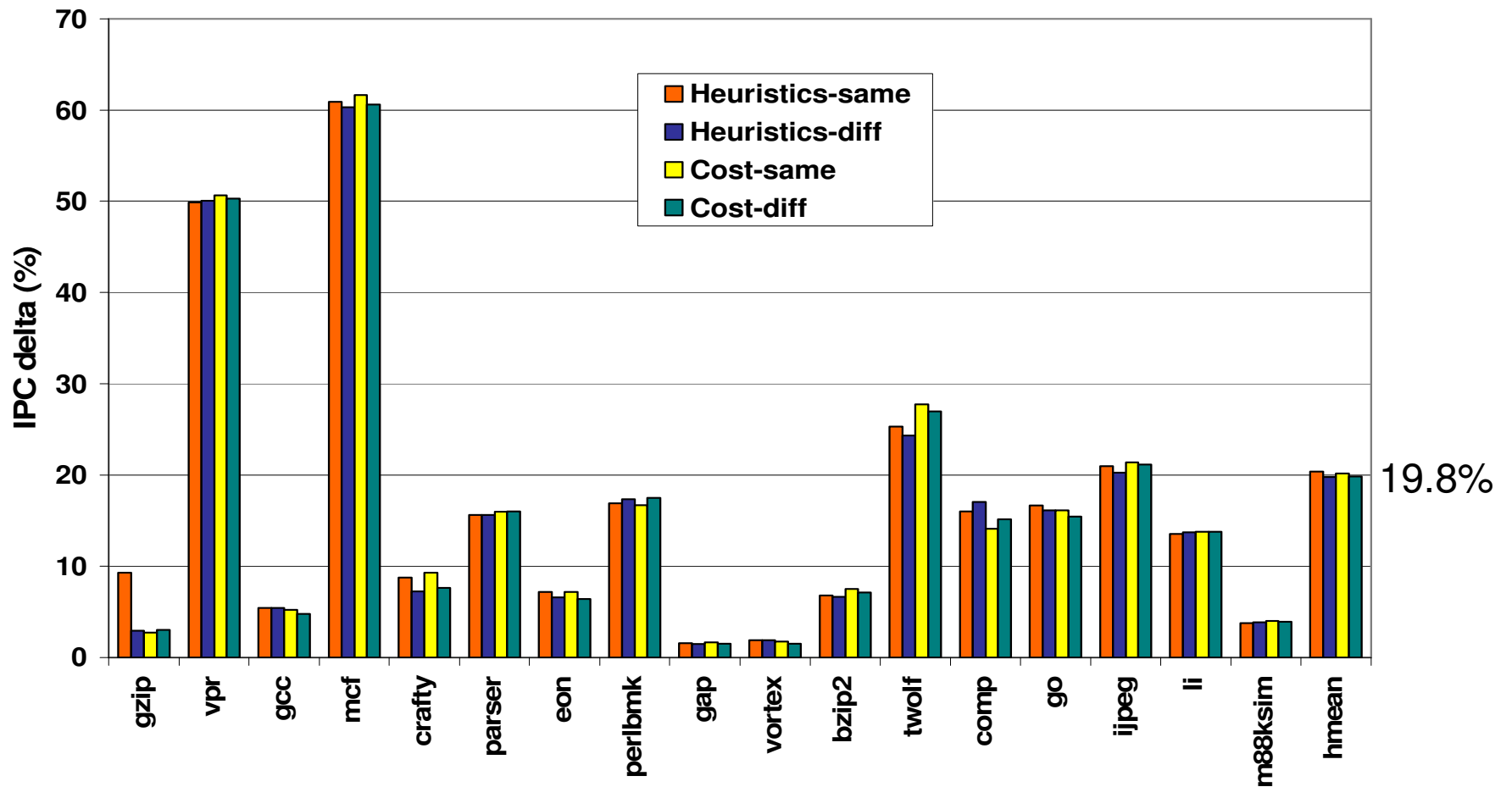
Heuristic-Based Selection



Cost-Benefit-Based Selection



Input Set Effects



Conclusion

- ❑ Compiler-microarchitecture interaction is good!
- ❑ DMP exploits frequently-hammocks.
- ❑ We developed new algorithms that select beneficial diverge-branches and CFM points.
- ❑ We proposed a new cost-benefit model for dynamic predication.
- ❑ DMP and our algorithms improve performance by 20%.

Thank You!

Questions?