Improving Inlining Decisions in the Open Research Compiler

Peng Zhao
José Nelson Amaral

University of Alberta
Edmonton, AB, Canada

http://www.cs.ualberta.ca/~amaral
amaral@cs.ualberta.ca

Yet Another Paper on Inlining?

What is new?

Adapt Decisions to Benchmark Sizes

Aggressive for small benchmarks, careful for large ones.

Use *Cycle* *Density* to Control Code Bloat

A correction to the temperature heuristic in ORC.

What is left to do?

Investigated why remainder procedures not inlined.

Next: **Partial Inlining** and **Recursive Procedure Inlining.**

Open Research Compiler

Cydrome → Ragnarok → Mongoose → Osprey →

Cydrome: 1984-88

Ragnarok: 1989-1995
PRE, 64 bits

Mongoose: 1995-1998
SSA, LNO, IPA

Osprey: 1999-2000
Itanium I

→ Pro64 → ORC 1.0 → ORC 2.1

Pro64: 2000
Open Source

ORC 1.0: Jan 2002
Perform. Code Gen.

ORC 2.1: July 2003

Why inline?

Eliminate function call overhead

Building stack frame, passing parameters, ...

Increase scope for code analysis

Better identification for loop optimizations

Improve code placement

Affine code can be placed nearby

Why not to inline?

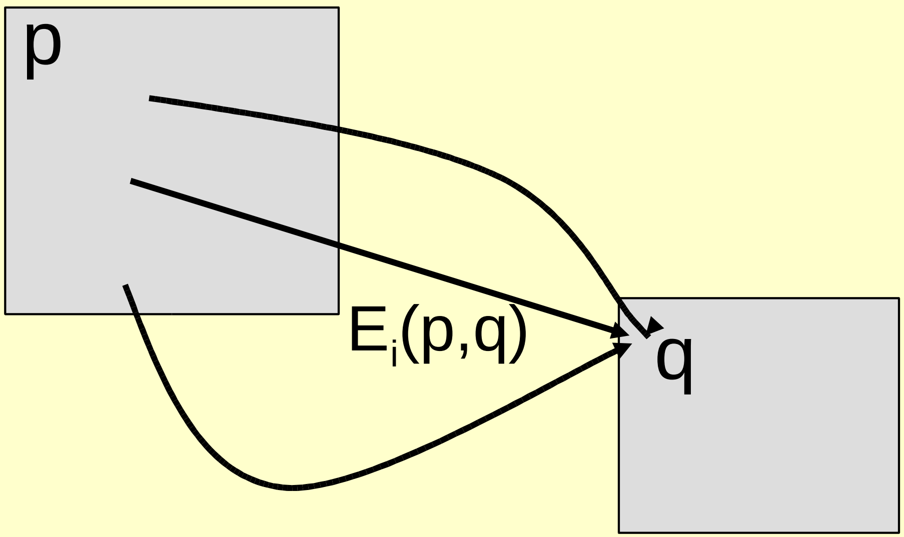
Code bloat

Negative instruction cache effects

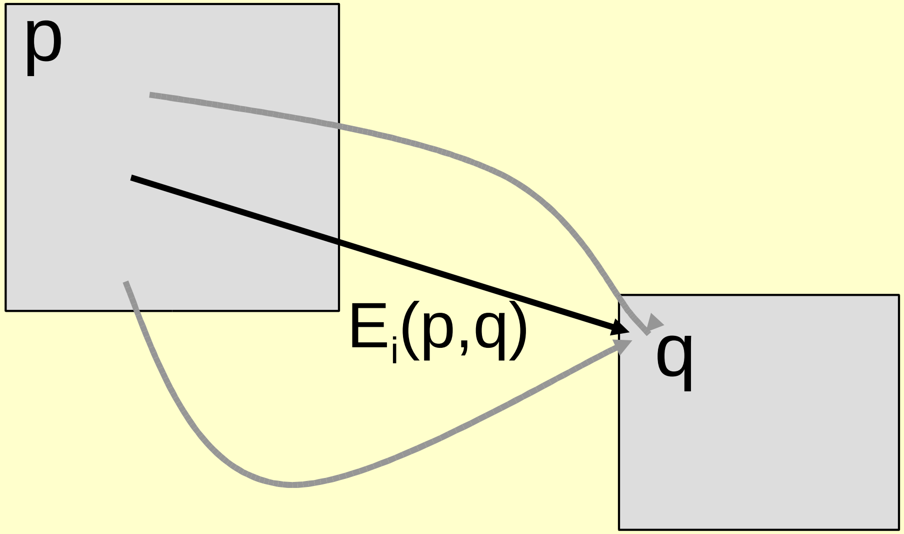
Compiler resources

Some analysis may choke on large procedures

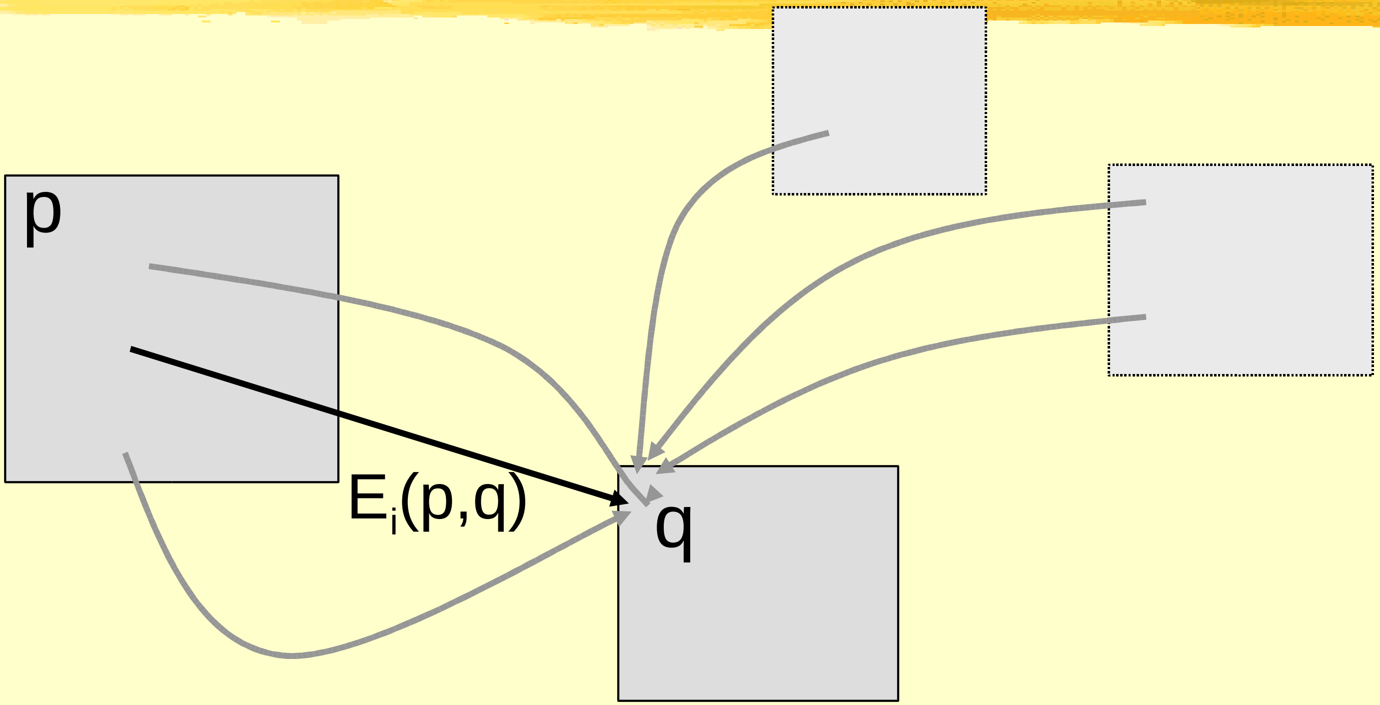
Original ORC Inlining Heuristic



Original ORC Inlining Heuristic

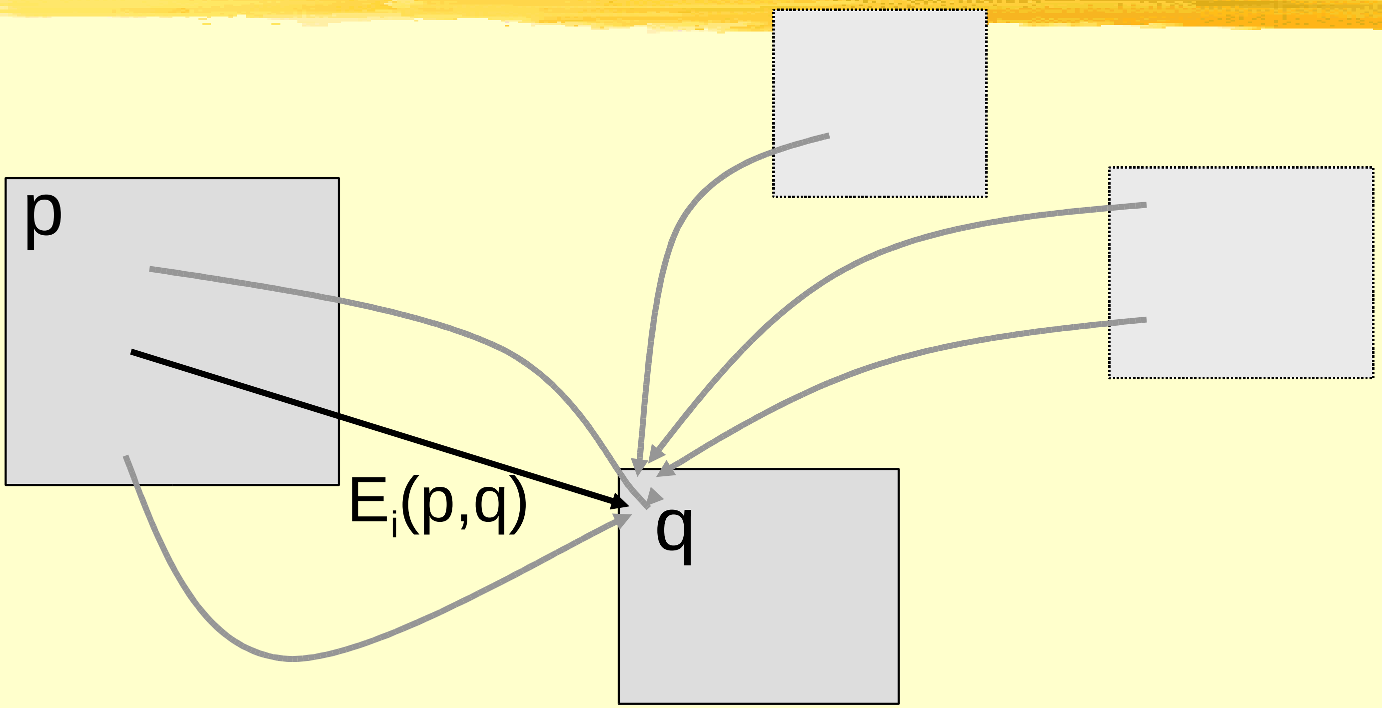


Original ORC Inlining Heuristic



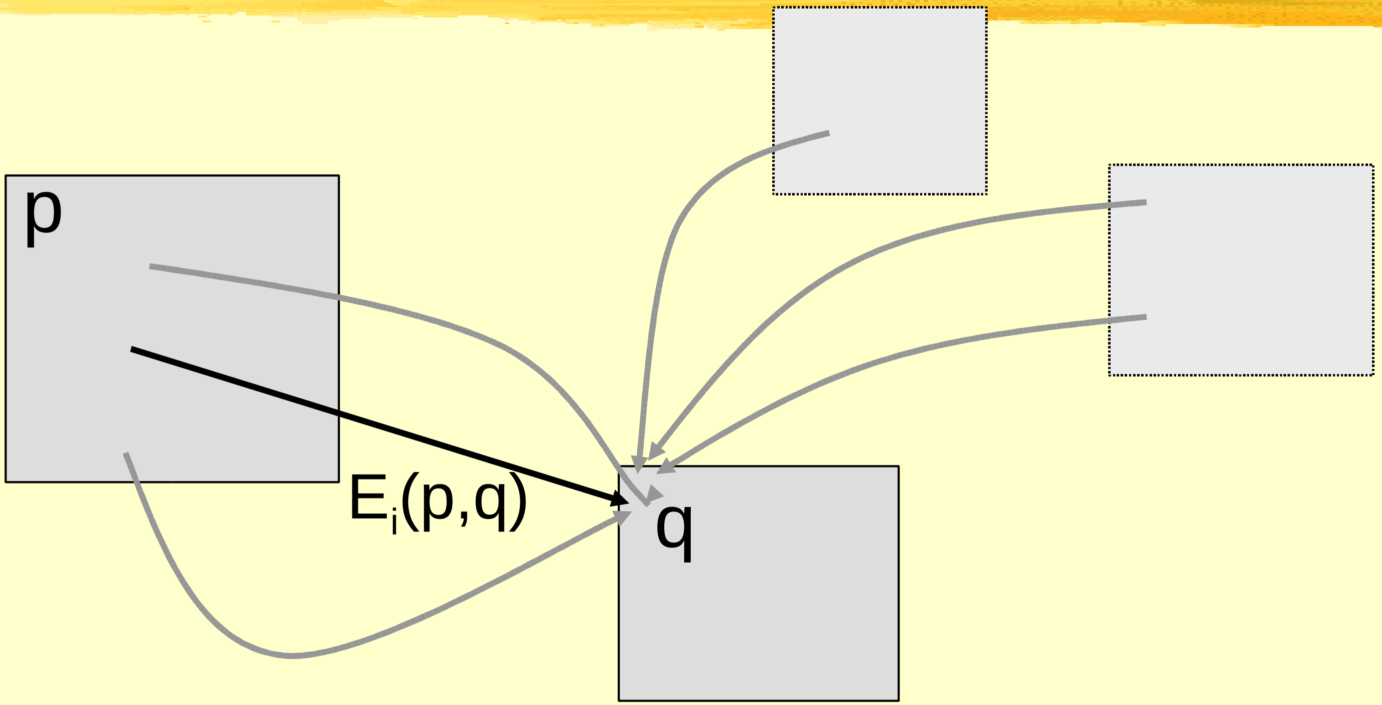
Temperature[$E_i(p,q)$] = ?

Original ORC Inlining Heuristic



$$cycle\_ratio[E_i(p,q)] = \frac{cycle\_count(q)}{Total\_cycle\_count} \times \frac{call\_freq[E_i(p,q)]}{call\_freq(q)}$$

Original ORC Inlining Heuristic



$$size\_ratio(q) = \frac{size(q)}{Total\_application\_size}$$

Original ORC Inlining Heuristic

$$\text{Temperature}[E_i(p,q)] = \frac{\text{cycle\_ratio}[E_i(p,q)]}{\text{size\_ratio}(q)}$$

$$\text{cycle\_ratio}[E_i(p,q)] = \frac{\text{cycle\_count}(q)}{\text{Total\_cycle\_count}} \times \frac{\text{call\_freq}[E_i(p,q)]}{\text{call\_freq}(q)}$$

$$\text{size\_ratio}(q) = \frac{\text{size}(q)}{\text{Total\_application\_size}}$$

Original ORC Inlining Heuristic

$$\text{Temperature}[E_i(p,q)] = \frac{\text{cycle\_ratio}[Ei(p,q)]}{\text{size\_ratio}(q)}$$

Edges called often are hot. Good!

$$\text{cycle\_ratio}[E_i(p,q)] = \frac{\text{cycle\_count}(q)}{\text{Total\_cycle\_count}} \times \frac{\text{call\_freq}[E_i(p,q)]}{\text{call\_freq}(q)}$$

$$\text{size\_ratio}(q) = \frac{\text{size}(q)}{\text{Total\_application\_size}}$$

Original ORC Inlining Heuristic

$$\text{Temperature}[E_i(p,q)] = \frac{\text{cycle\_ratio}[Ei(p,q)]}{\text{size\_ratio}(q)}$$

Functions that execute longer are hot. Good!

$$\text{cycle\_ratio}[E_i(p,q)] = \frac{\text{cycle\_count}(q)}{\text{Total\_cycle\_count}} \times \frac{\text{call\_freq}[E_i(p,q)]}{\text{call\_freq}(q)}$$

$$\text{size\_ratio}(q) = \frac{\text{size}(q)}{\text{Total\_application\_size}}$$

Original ORC Inlining Heuristic

$$\text{Temperature}[E_i(p,q)] = \frac{\text{cycle\_ratio}[Ei(p,q)]}{\text{size\_ratio}(q)}$$

Even if they are not called often. Bad!

$$\text{cycle\_ratio}[E_i(p,q)] = \frac{\text{cycle\_count}(q)}{\text{Total\_cycle\_count}} \times \frac{\text{call\_freq}[E_i(p,q)]}{\text{call\_freq}(q)}$$

$$\text{size\_ratio}(q) = \frac{\text{size}(q)}{\text{Total\_application\_size}}$$

Original ORC Inlining Heuristic

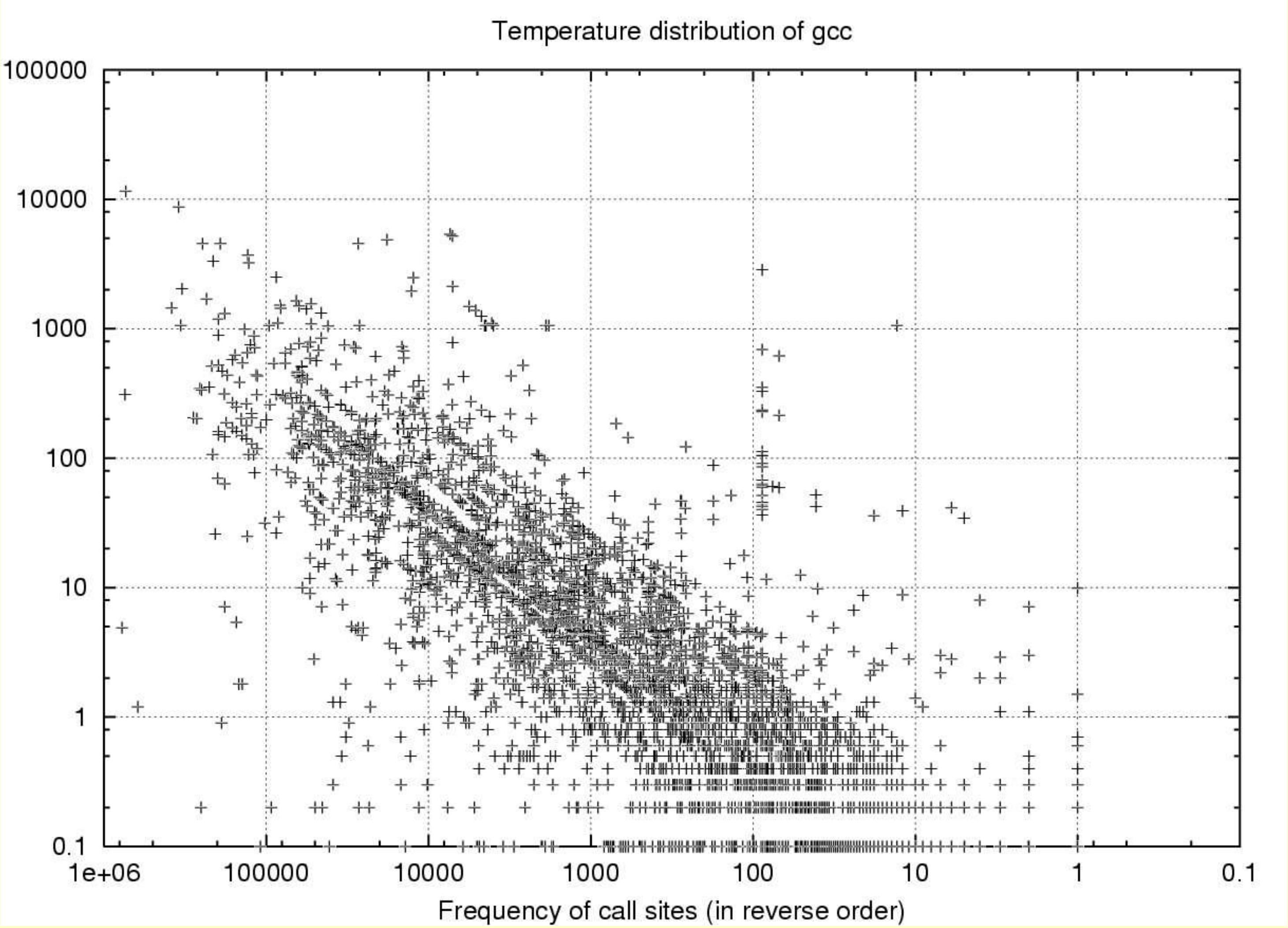
$$\text{Temperature}[E_i(p,q)] = \frac{\text{cycle\_ratio}[Ei(p,q)]}{\text{size\_ratio}(q)}$$

Small Code ==> small functions are cold. Bad!

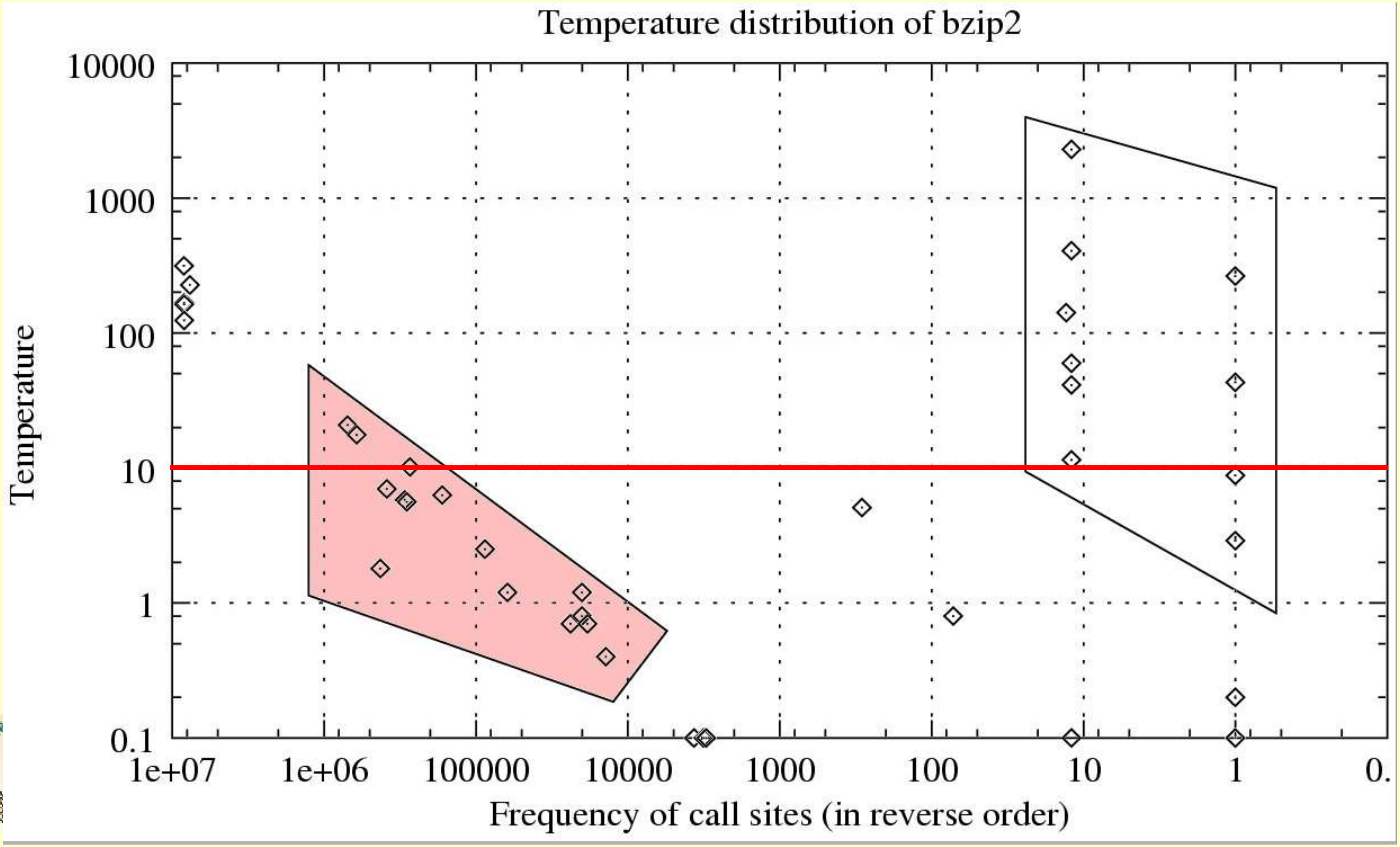
$$\text{cycle\_ratio}[E_i(p,q)] = \frac{\text{cycle\_count}(q)}{\text{Total\_cycle\_count}} \times \frac{\text{call\_freq}[E_i(p,q)]}{\text{call\_freq}(q)}$$

$$\text{size\_ratio}(q) = \frac{\text{size}(q)}{\text{Total\_application\_size}}$$

Tempereature Distribution for gcc



Temperature Distribution for BZIP2



Adapting the ORC Heuristic to Benchmark Size

Empirical classification of benchmarks (based on SPEC):

		Temperature Threshold
Small: < 10,000 AST Nodes	→	1
Medium: Anything in between	→	50
Large: > 250,000 AST Nodes	→	120

“ Heavy” Procedures

A procedure call that:

- is <u>hot</u> in the original ORC heuristic
- but that is <u>not called often</u>

must have <u>high trip count loops</u>.

We call these heavy procedures.

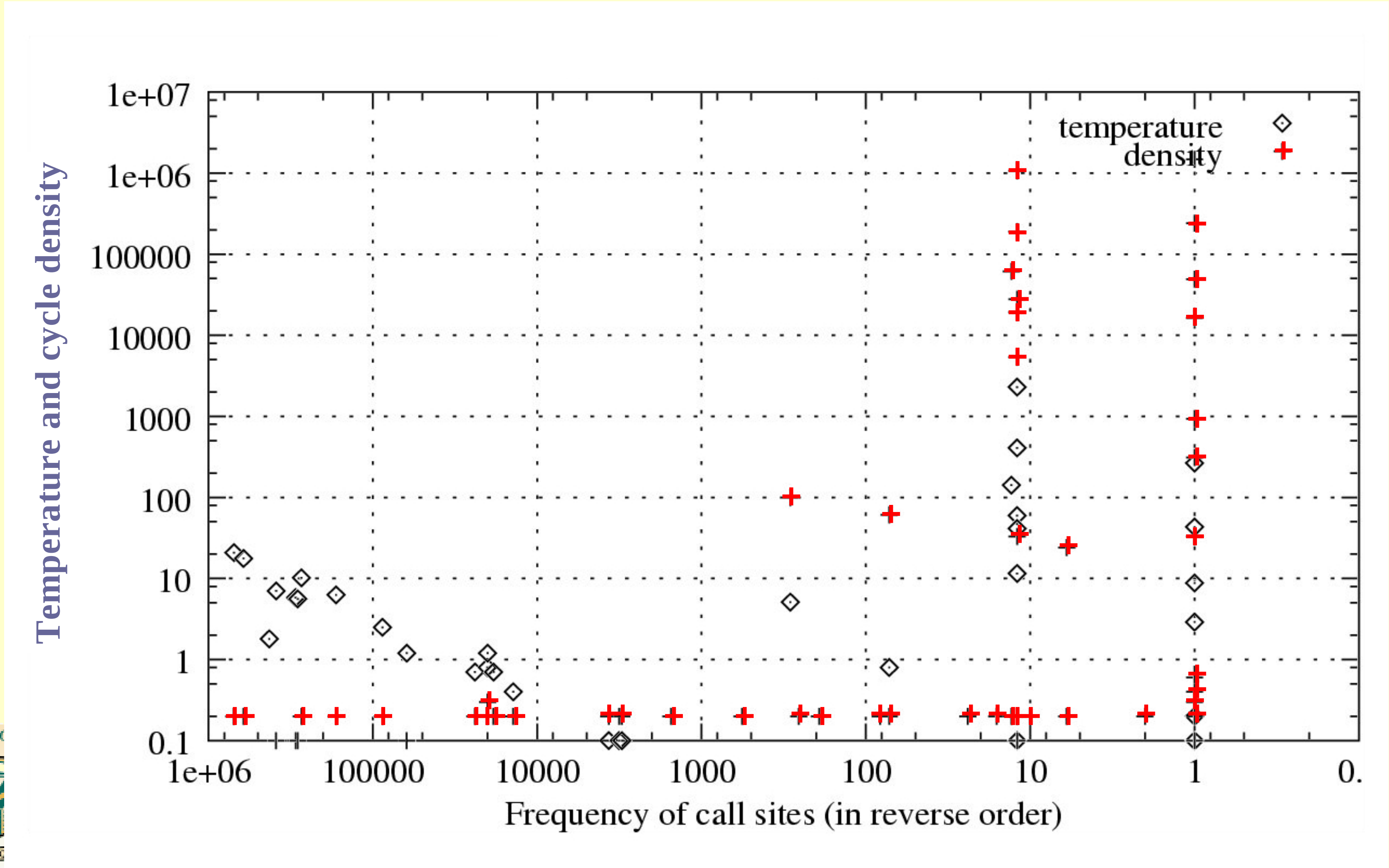
We introduce the cycle density heuristic to fix the ORC inlining decisions for heavy procedures.

Cycle Density

$$\text{cycle\_density}(q) = \frac{\text{cycle\_count}(q)}{\text{frequency}(q)}$$

High cycle density indicates a heavy procedure.

Temp. ×Cycle Density (BZIP2)



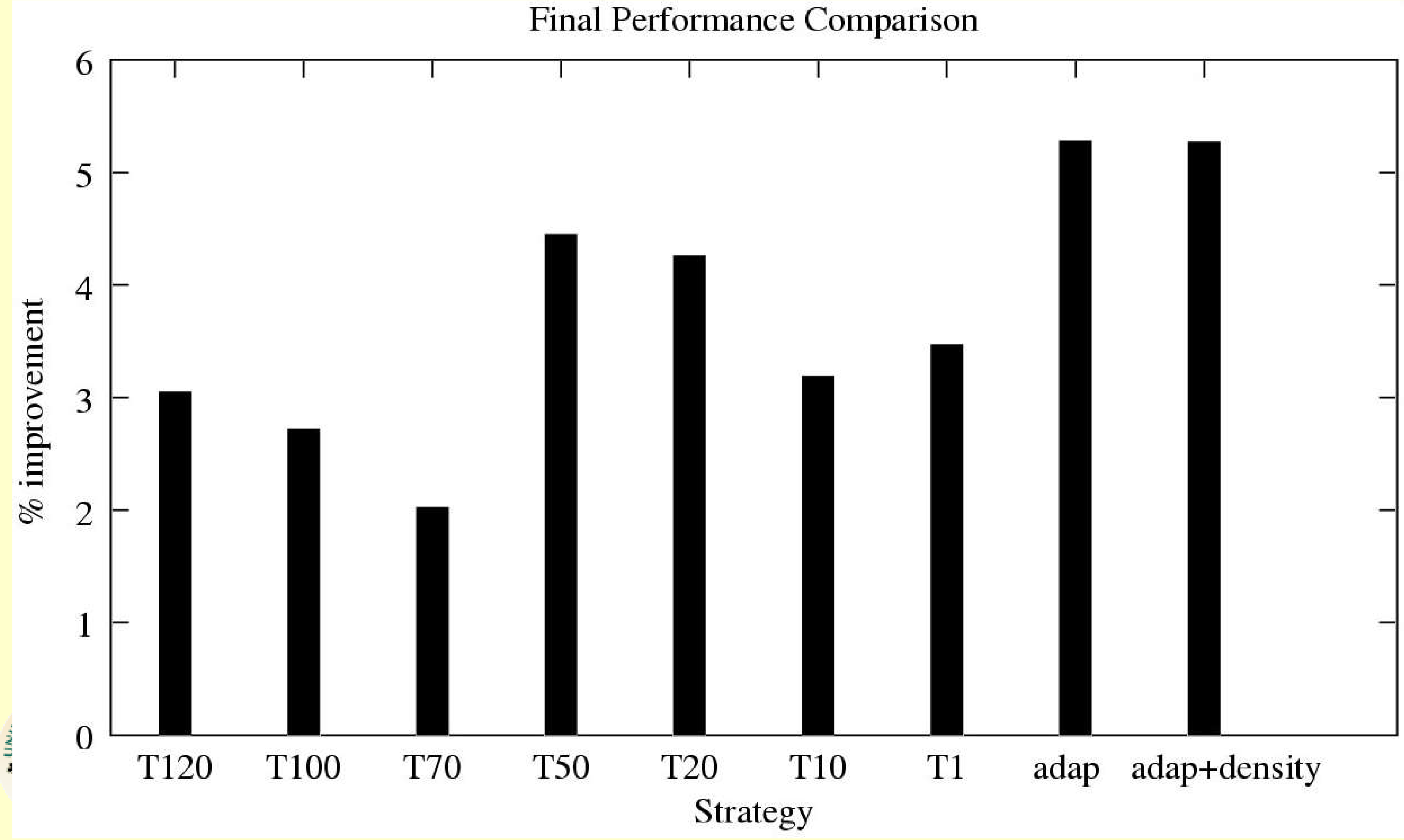
Experimental Study

SPEC2000 benchmarks, except EON.

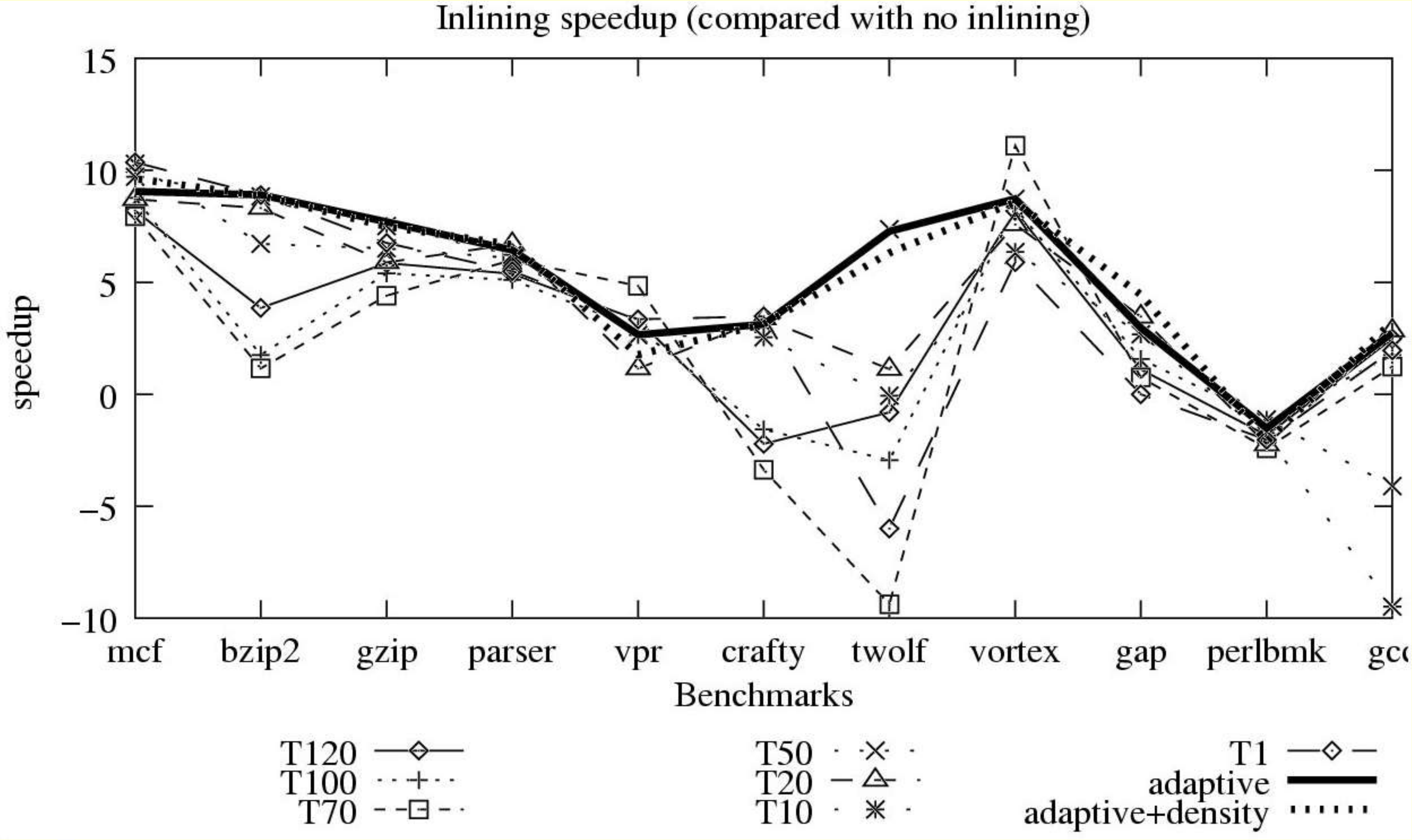
Runtime on an Itanium-I (733 MHz,
1 GB mem, RH-Linux 7.1)

Compilation on a dual Pentium III (600 MHz,
512 MB mem, RH-Linux 7.2)

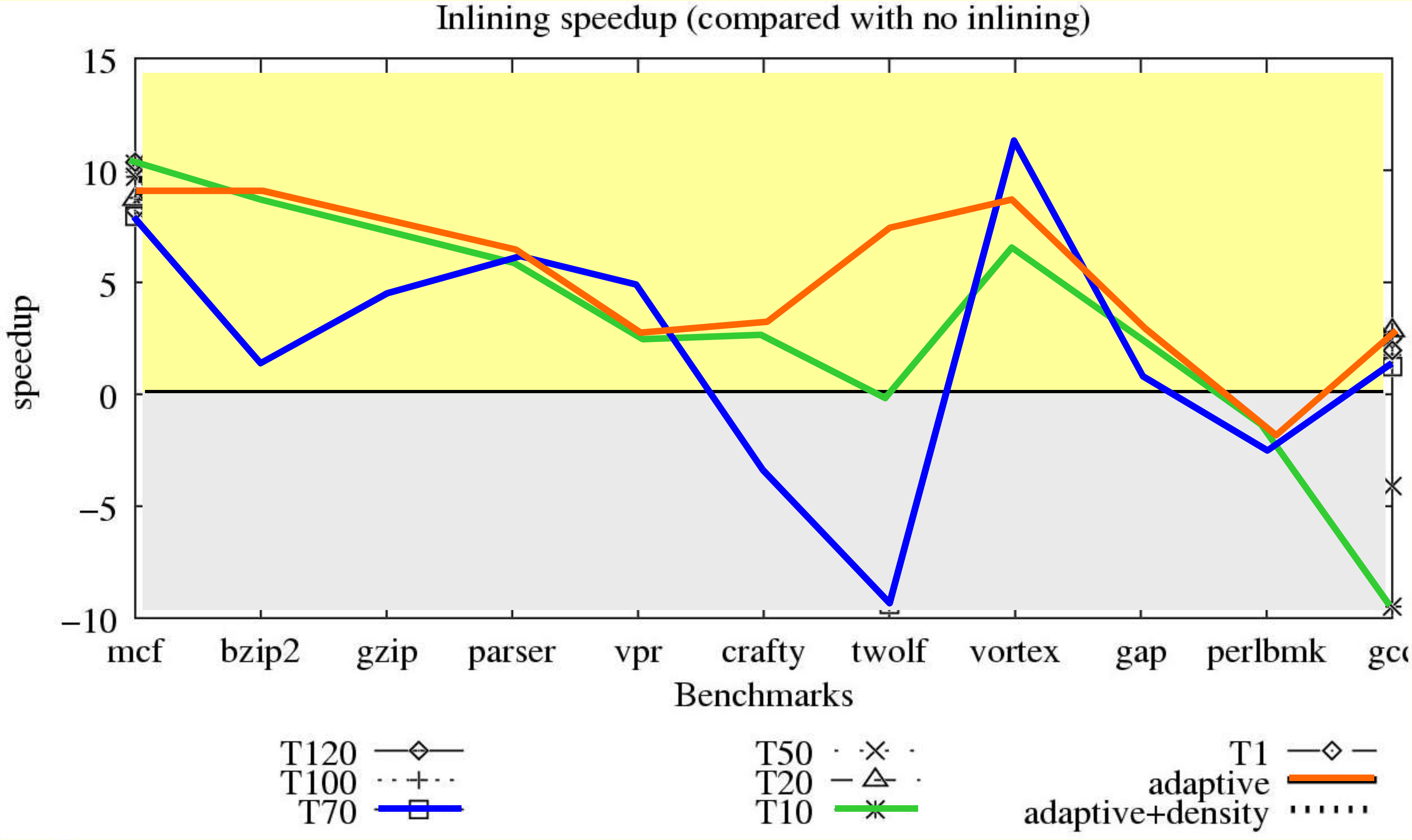
Average Performance Improvement on SPEC2000



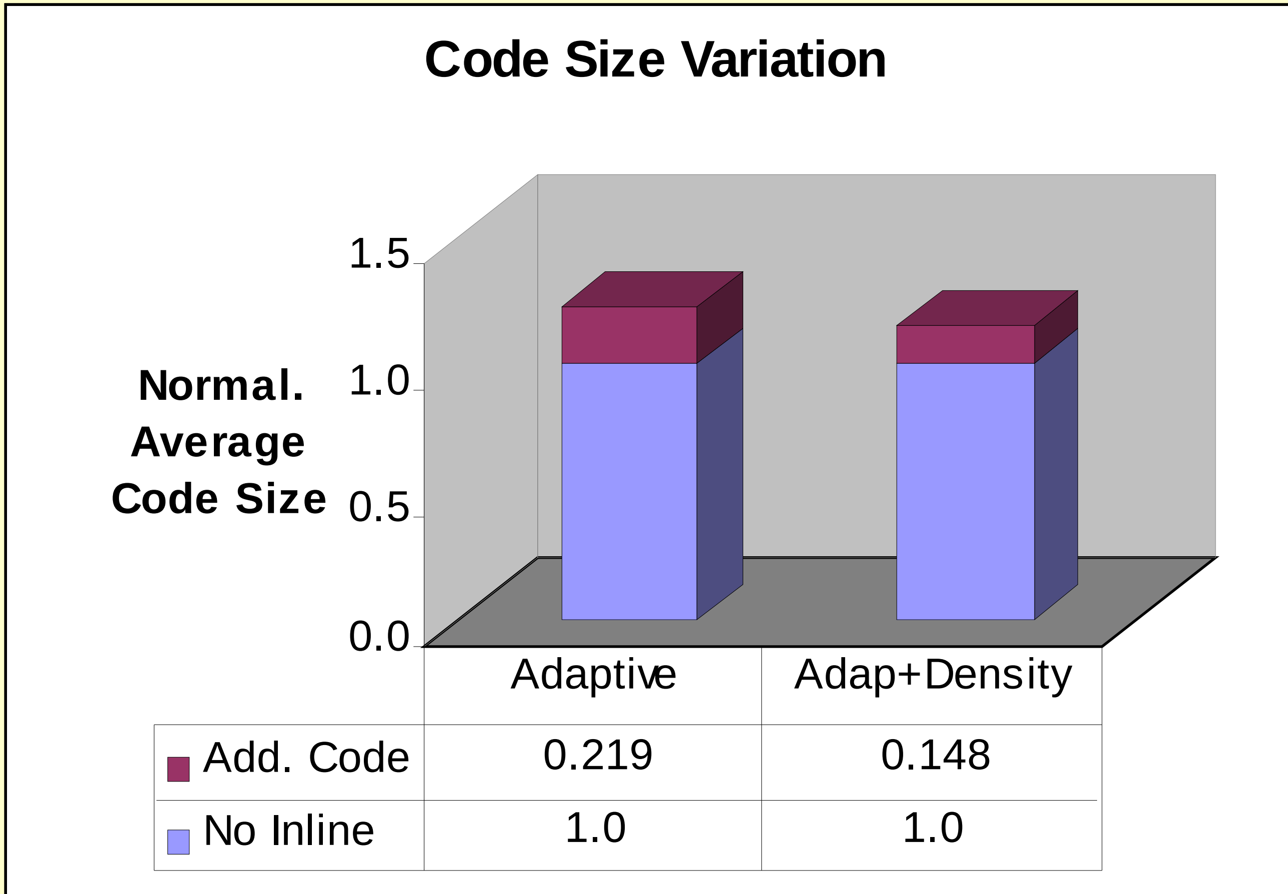
Inlining Speedup



Inlining Speedup

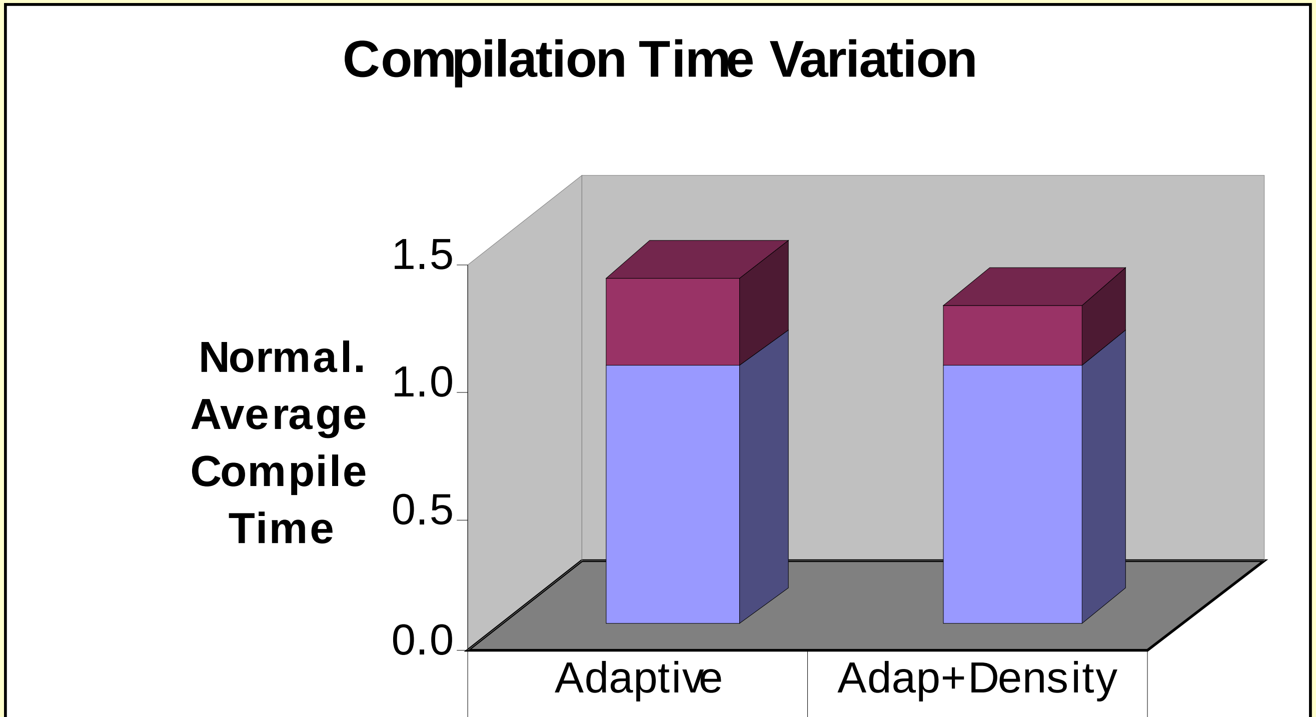


Effect of Cycle Density on Code Size



	Adaptive	Adap+Density
Add. Code	0.219	0.148
No Inline	1.0	1.0

Effect of Cycle Density on Compilation Time



	Adaptive	Adap+Density
Add. Comp. Time	0.343	0.24
No Inline	1.0	1.0

Can We Inline Further?