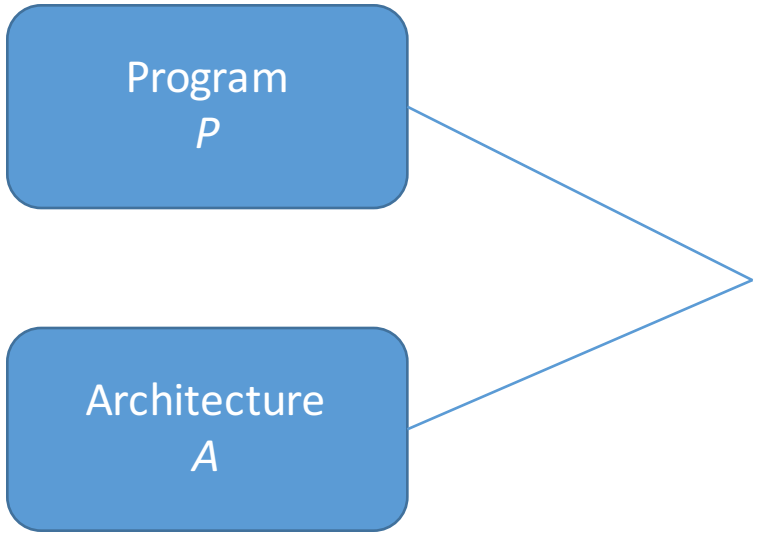
Precise Analysis of Private and Shared Caches for tight WCET estimates

Kartik Nagar

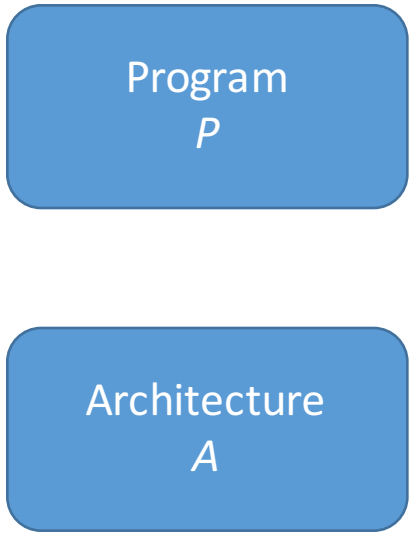
Advisor : Y N Srikant

The WCET Problem



WCET Problem : To determine W such that
W ≥ Actual Execution Time of P on A

The WCET Problem

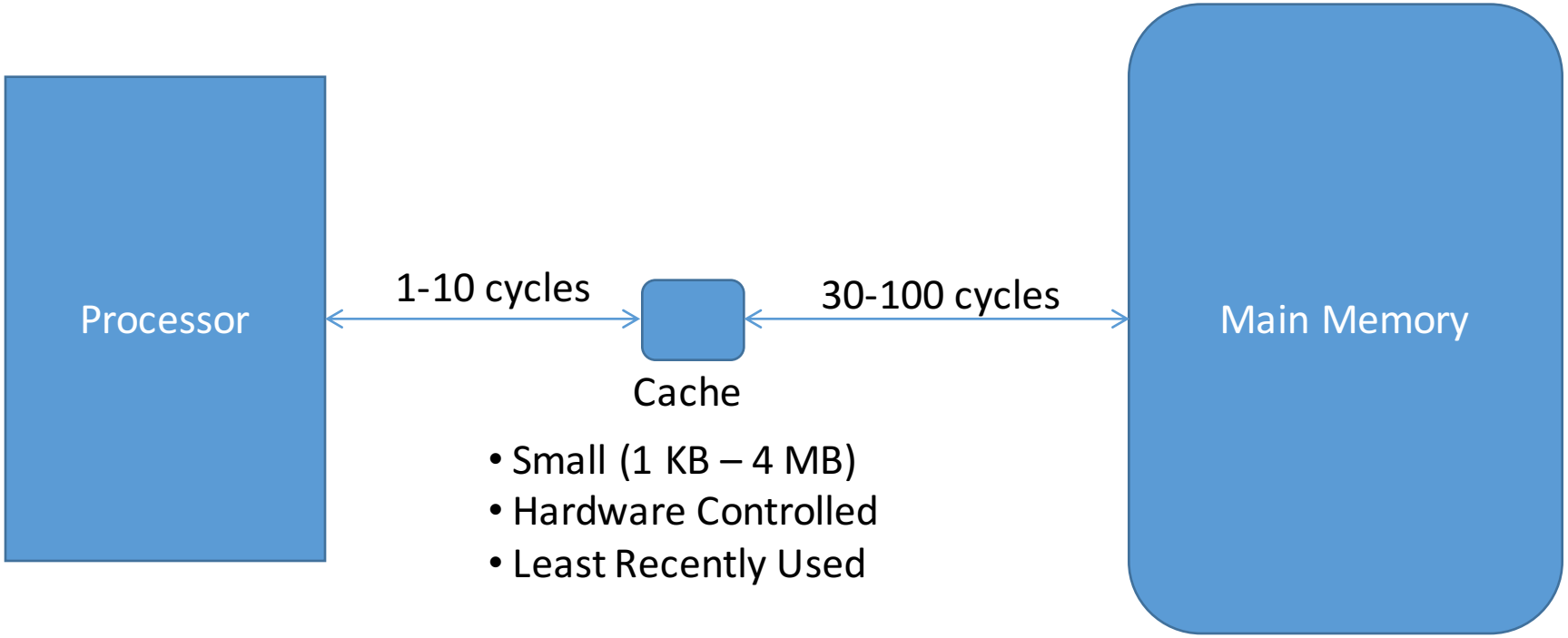


Uses of WCET:

- Primarily in Real time systems, to prove all deadlines are always met.
- Also used in finding performance bugs

WCET Problem : To determine W such that
W ≥ Actual Execution Time of P on A

General Architectural Model



Cache Analysis

Purpose – To statically predict cache behavior which can be safely used for WCET estimation

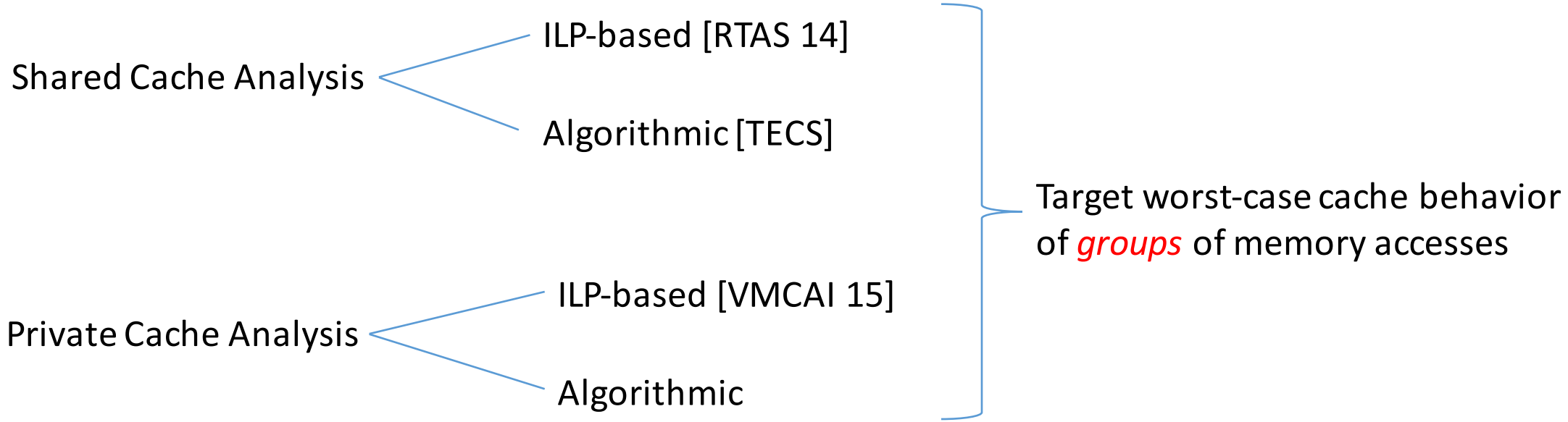
Importance - Has a huge impact on the precision of the WCET estimate

Standard Approach – To find worst-case cache behavior of every memory access *individually* across all execution instances

Issues

- Highly imprecise for shared caches in multi-core architectures
  - Almost impossible to statically guarantee individual cache hits due to interfering accesses from other cores
- Cannot capture complex cache behavior of groups of memory accesses in private caches
  - Quite common in real-world programs

Our Contributions



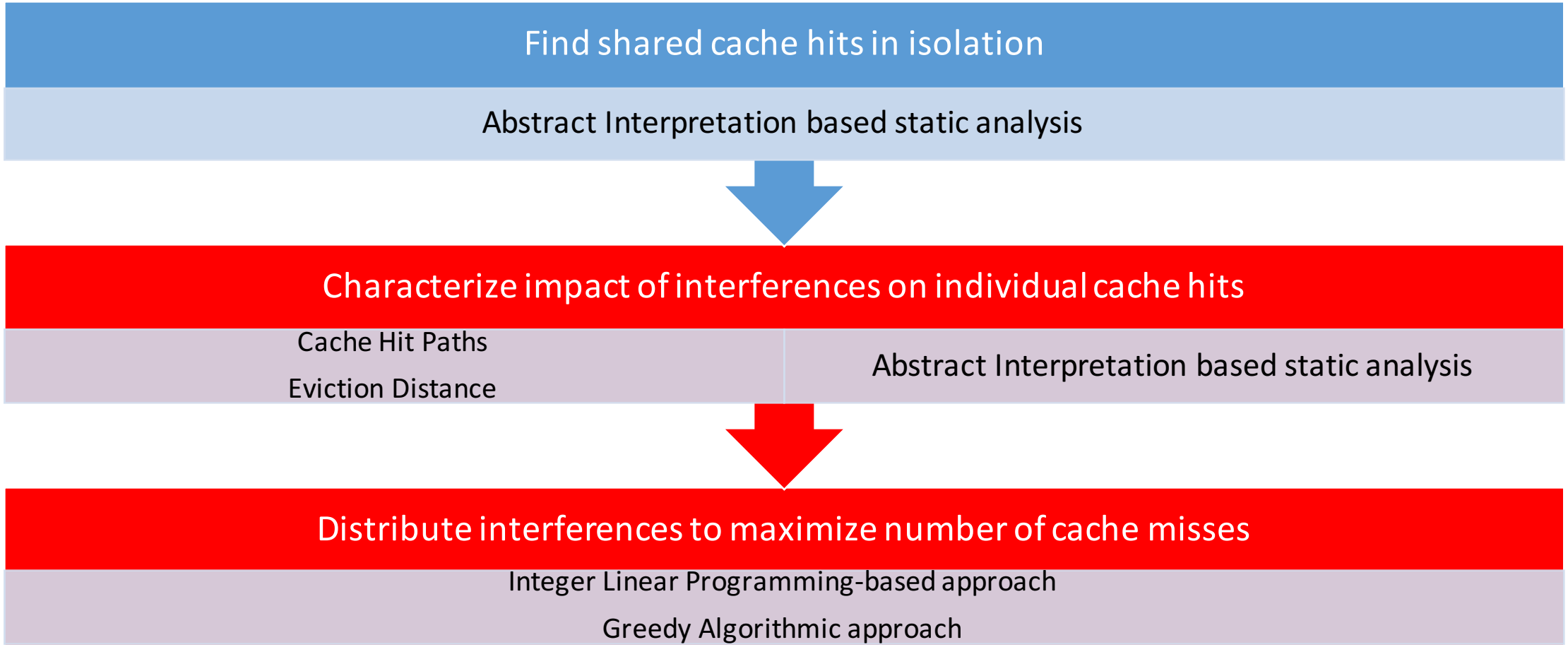
Shared Cache Analysis Problem

- Given assignment of programs to cores, find the shared cache behavior of each program
- Primary Issue : Shared cache accesses made by other cores can evict cache blocks of program under analysis and cause extra cache misses.
- Due to interfering accesses, it is almost impossible to guarantee that an access will always hit the cache.

Our Approach – Worst Case Interference Placement

- Instead of finding *which* accesses are guaranteed to be cache hits, we find *how many* accesses are guaranteed to hit the cache.
- Shared cache analysis as an optimization problem:
  - Distribute interfering accesses across a program to cause the maximum number of misses.

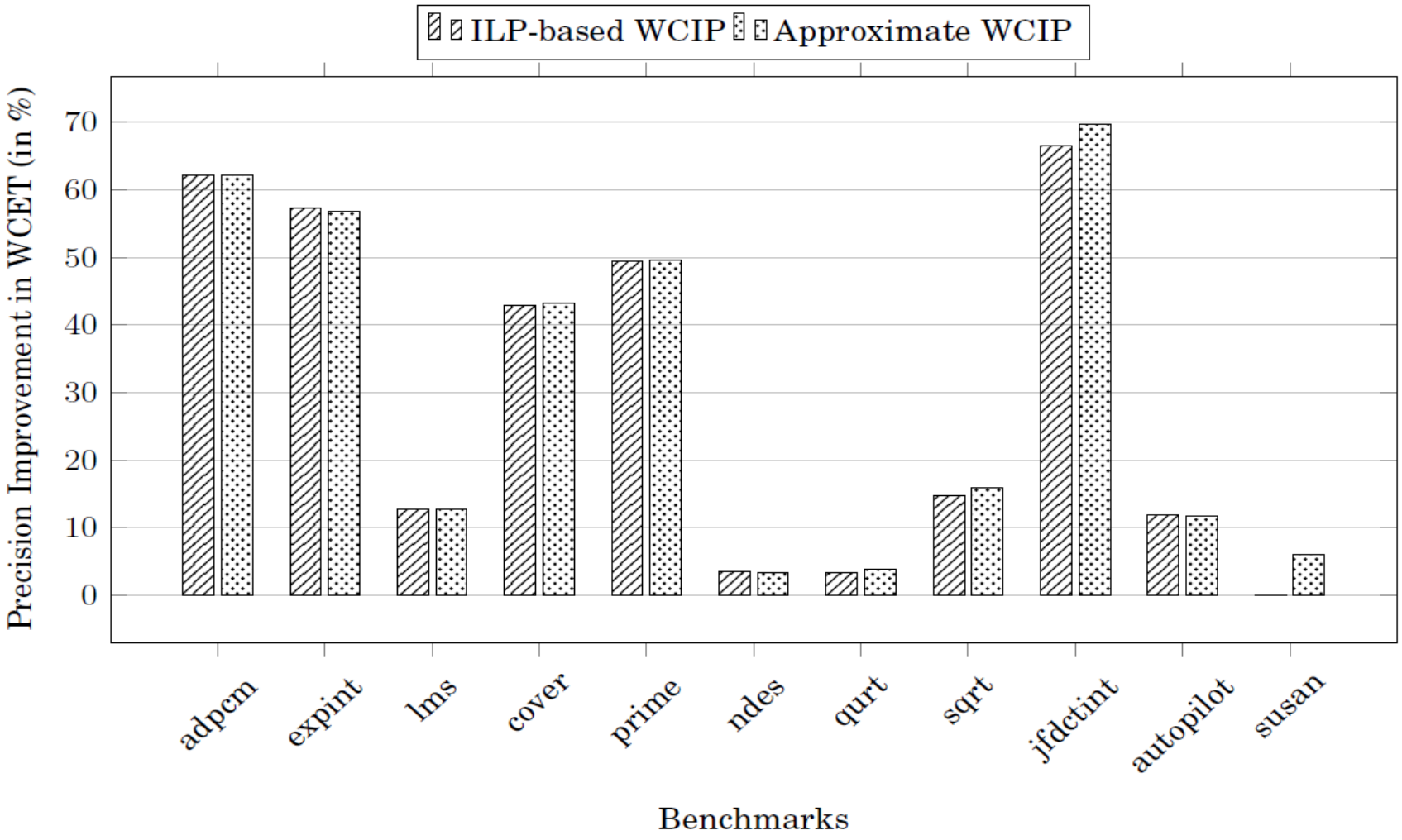
Overview of our approach



Some properties of our approach

- Our approach guarantees that the increase in WCET due to shared cache interference would be *linear* in the amount of interference.
- We also show that shared cache analysis is a computationally hard problem.
  - Finding the worst-case path in the program becomes an NP-Hard problem due to shared cache interference.
- We propose an approximate polynomial-time algorithmic approach which does not lose precision for low amount of interference.

Results

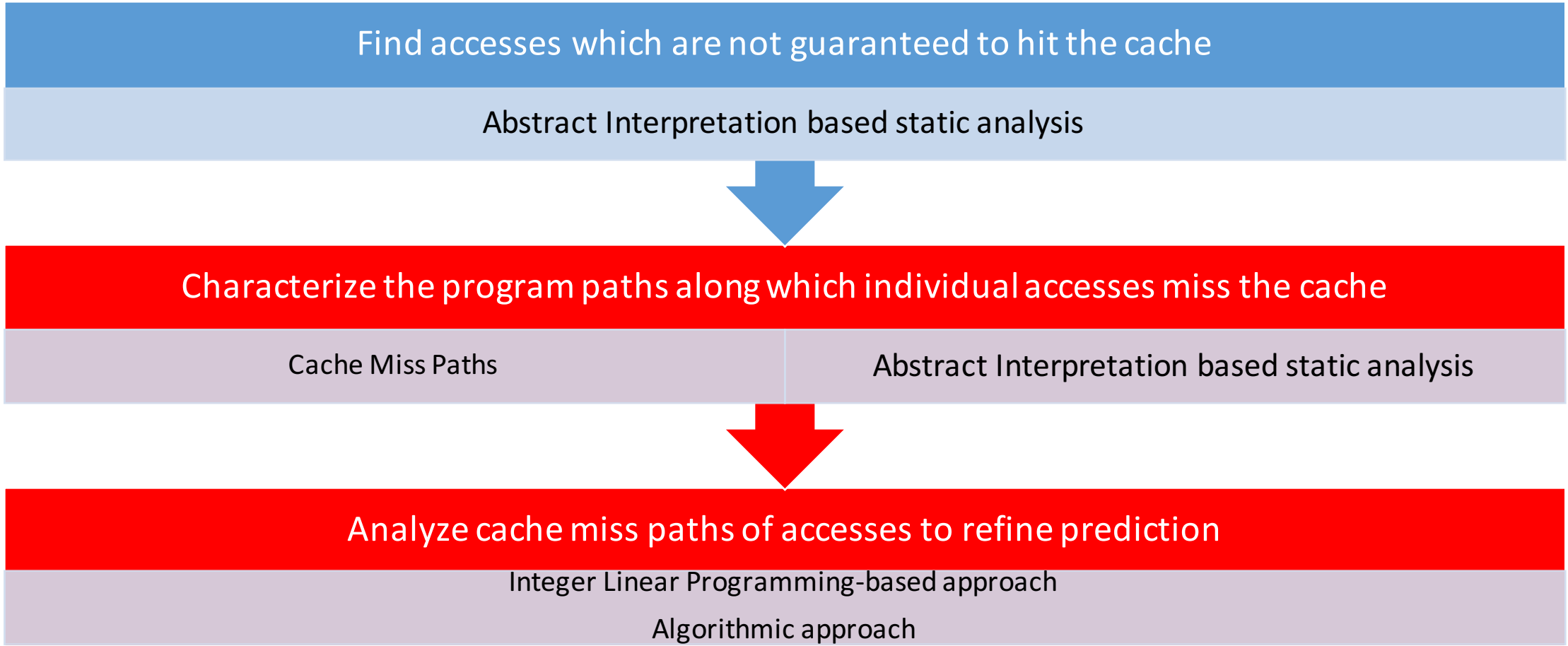


Average Precision Improvement = 26%

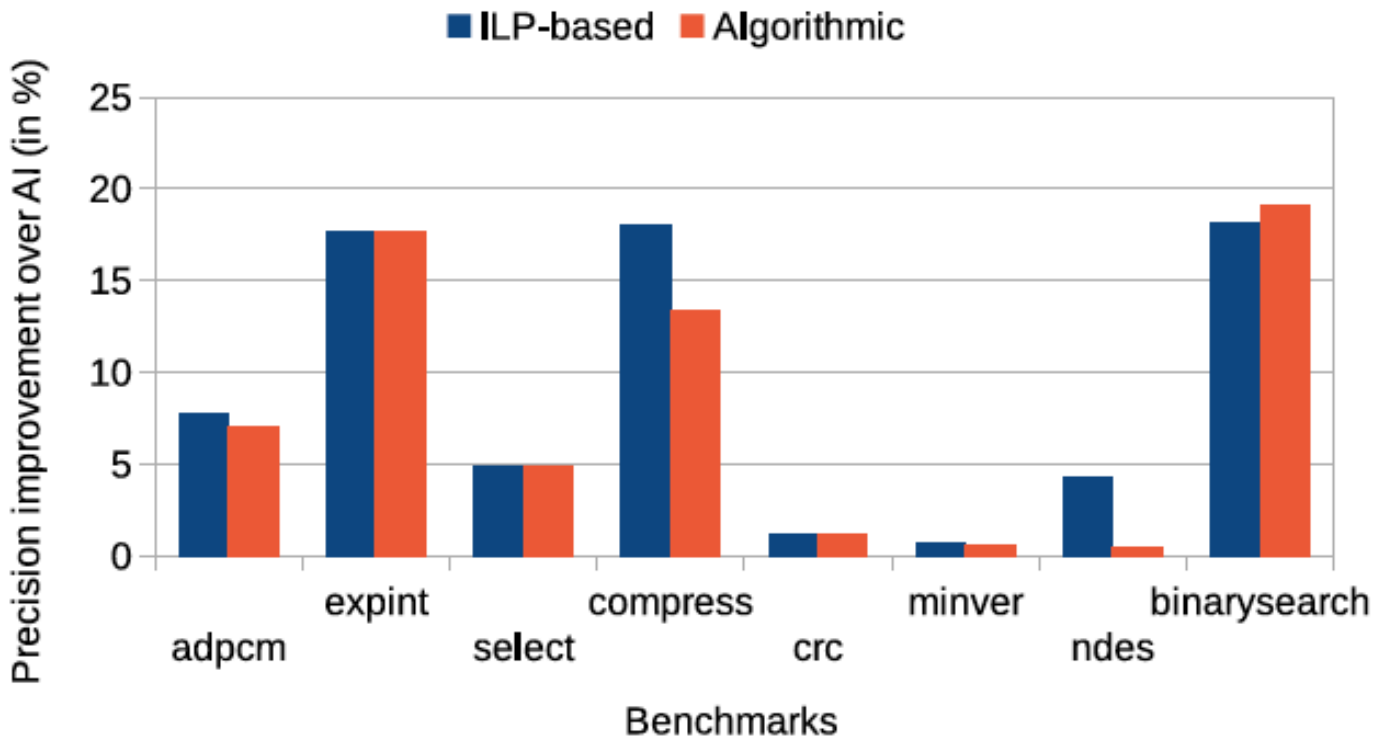
Private Cache Analysis Problem

- No interferences, due to multiple program paths, an access may hit or miss the cache in different execution instances.
- The state-of-the-art approach classifies an access as a cache hit only if it is guaranteed to hit the cache across all execution instances.
- Cache hit-miss prediction can be refined in several ways
  1. Two accesses may never miss the cache together in the same execution instance.
  2. An access inside a loop may not miss the cache in all iterations.
  3. An access may not miss the cache in the worst-case execution instance.

Overview of our approach



Experimental Results



Average Precision Improvement = 8%

Algorithmic approach matches precision improvement of ILP-based approach

Conclusion

- In this thesis, we have proposed precise, scalable approaches to cache analysis aimed towards tighter estimation of WCET.
- Shared cache analysis in multi-cores
  - Our approach, called Worst Case Interference Placement, is significantly precise than previous approaches.
- Private cache analysis
  - Reasonable precision improvement over previous approaches with a moderate increase in analysis time.

Publications based on the thesis

1. Precise shared cache analysis using optimal interference placement. Kartik Nagar and Y.N. Srikant. 20th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2014.
2. Path sensitive cache analysis using cache miss paths. Kartik Nagar and Y.N. Srikant. 16th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI), 2015.
3. Fast and Precise Worst Case Interference Placement for Shared Cache Analysis. Kartik Nagar and Y.N. Srikant. Accepted in ACM Transactions on Embedded Computing Systems (TECS), 2015.