北京大學
PEKING UNIVERSITY

Accelerating Pass Order Auto-tuning via Profile-Guided Cost Modeling

Bingyu Gao, Kang Gu, Lu Li, Wei Wei, Ding Li, Yao Guo

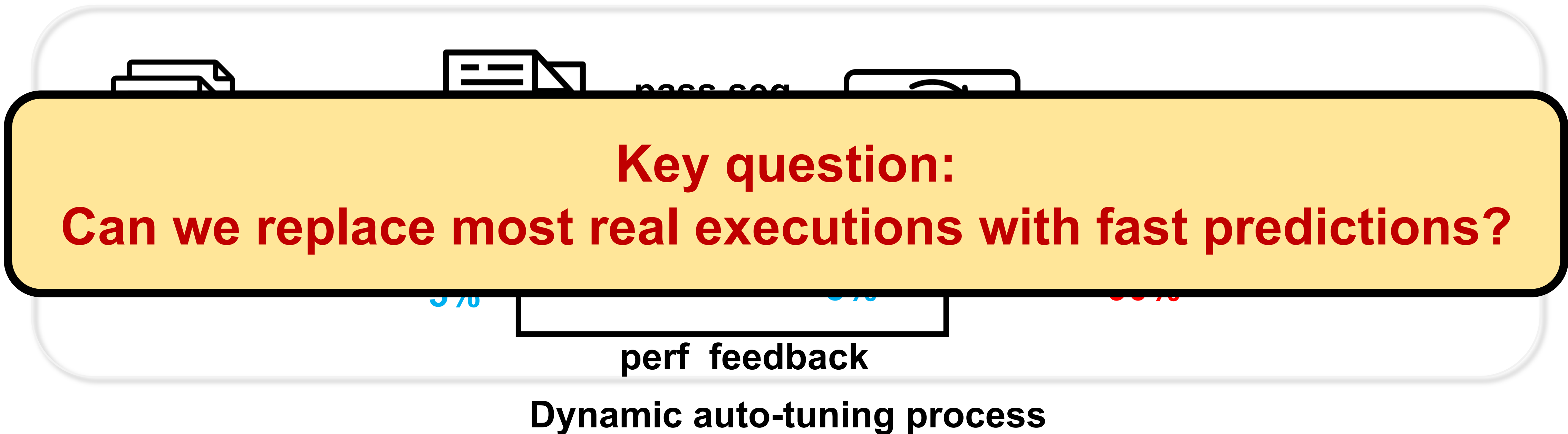
Background

北京大學 PEKING UNIVERSITY

- LLVM pass sequence strongly affects performance
- Search space is too huge
  - LLVM provides 120+ passes
  - 120^n (n is the length of pass sequence)
- The synergistic effect among passes are complicated
- We need dynamic auto-tuning to identify better pass sequences

Dynamic Auto-Tuning Techniques

- Each iteration = generate pass sequence + compile + execute
- A smarter search strategy may converge in fewer iterations, but cannot reduce the per-iteration compile & execute cost



Dynamic auto-tuning process

Key Challenges in Prediction-Based Auto-Tuning

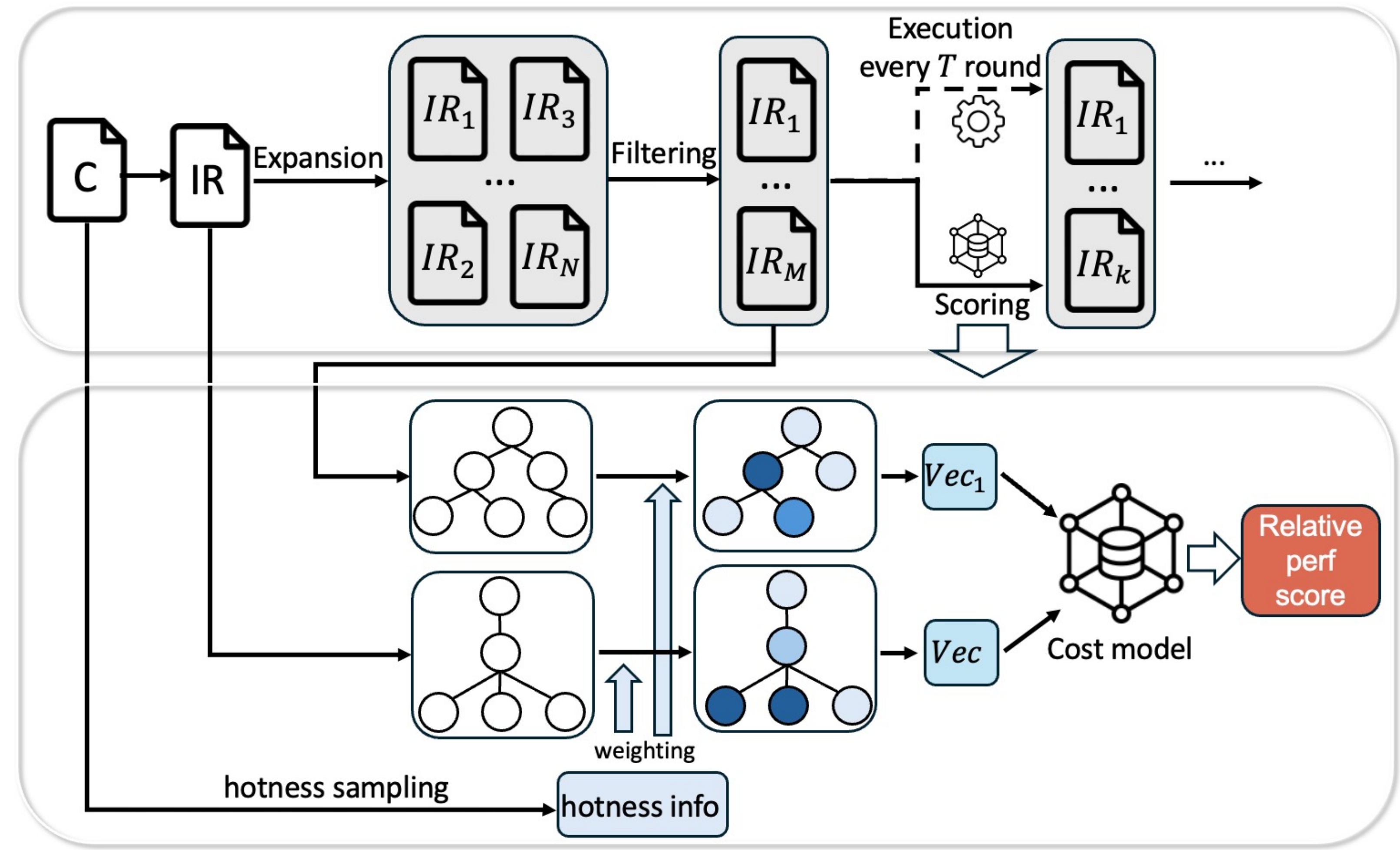
- What to predict?
  - Predicting the absolute performance of a full pass sequence requires modeling all interactions at once
- How to represent programs?
  - Different passes affect different program structures
  - Only a small fraction of code dominates runtime
- How to keep search reliable?
  - Prediction errors accumulate over iterations, search may drift away from promising sequences

Our Solution: Predict Relative Gain, Not Absolute Performance

- What to predict?
  - Predict relative gain instead of final absolute performance
- How to represent programs?
  - Compare only hot and changed IR regions rather than the whole program
- How to keep search reliable?
  - Combine model scoring with periodic real evaluation

Overall Framework: FastTune

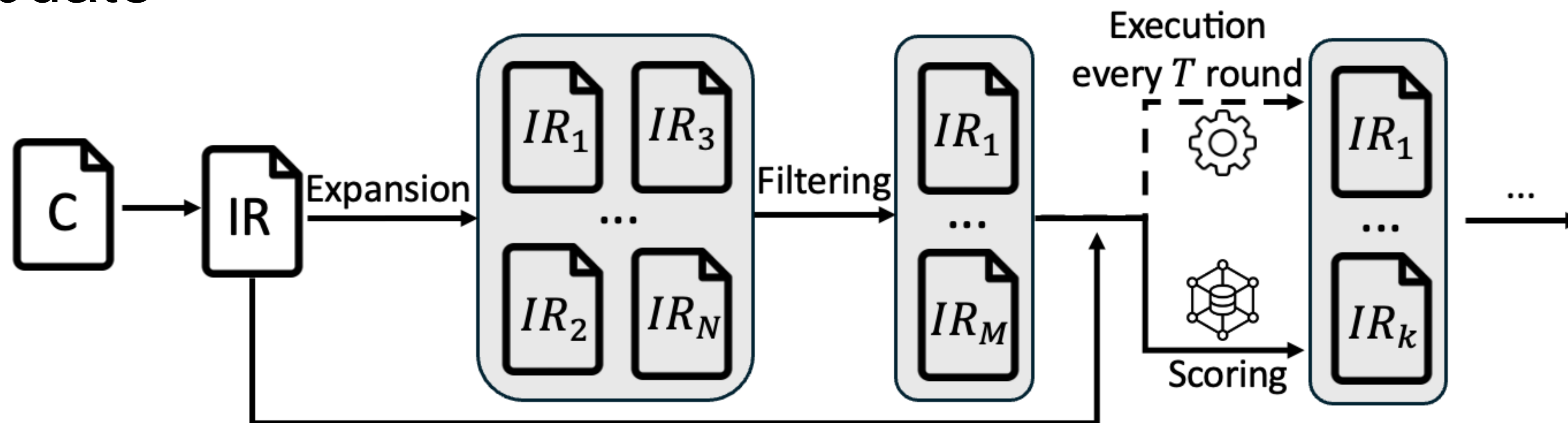
Beam search： Expansion, pruning and periodic real-evaluation



Cost model: Hotness-aware prediction of relative performance gain

Cost-Model-Guided Beam Search

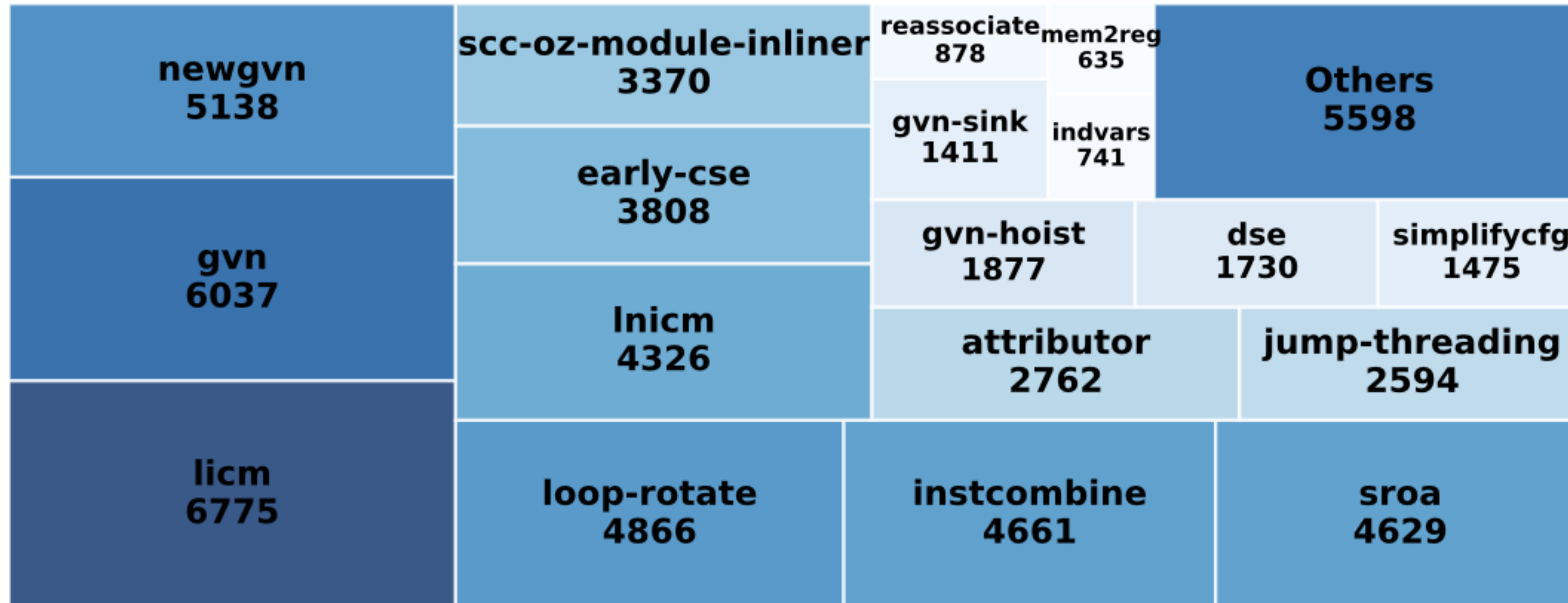
- Candidate Expansion
  - Apply candidate passes to each state in the current beam to generate new IR states
- Candidate Filtering
  - Filter ineffective and duplicate transformations
- Evaluation： Two Paths
  - Default: cost model scores all candidates in milliseconds
  - Every T rounds: real execution on Top-K candidates to correct accumulated errors
- Beam Update



Training Data Construction

北京大学 PEKING UNIVERSITY

- 1183 programs from CodeContests
- Two rounds of expansion from O0 with filtering and dedup
- 106K pairwise training samples, balanced across passes
- ~125 CPU days to construct



Evaluation-Experiment setup

北京大学 PEKING UNIVERSITY

- LLVM version: 20.1.0
- Platform: x86-64, Ubuntu 20.04.6
- Tuning Budget: 5000s
- Candidate Passes: 127
- Benchmarks: 28 programs from cBench and Polybench
- Metric: Speedup over –O3
- Baselines: BOCA, CompTuner, OpenTuner

cBench

ID	Project	File	Lines
C1	automotive_bitcount	bitcnts.c	99
C2	automotive_qsort1	qsort.c	148
C3	network_dijkstra	dijkstra_large.c	199
C4	security_sha	sha.c	210
C5	security_blowfish_d	bf_cfb64.c	127
C6	security_blowfish_e	bf_cfb64.c	127
C7	consumer_jpeg_c	jcphuff.c	829
C8	consumer_lame	psymodel.c	1253
C9	bzip2d	decompress.c	626
C10	bzip2e	blocksort.c	1094
C11	telecom_CRC32	crc_32.c	188

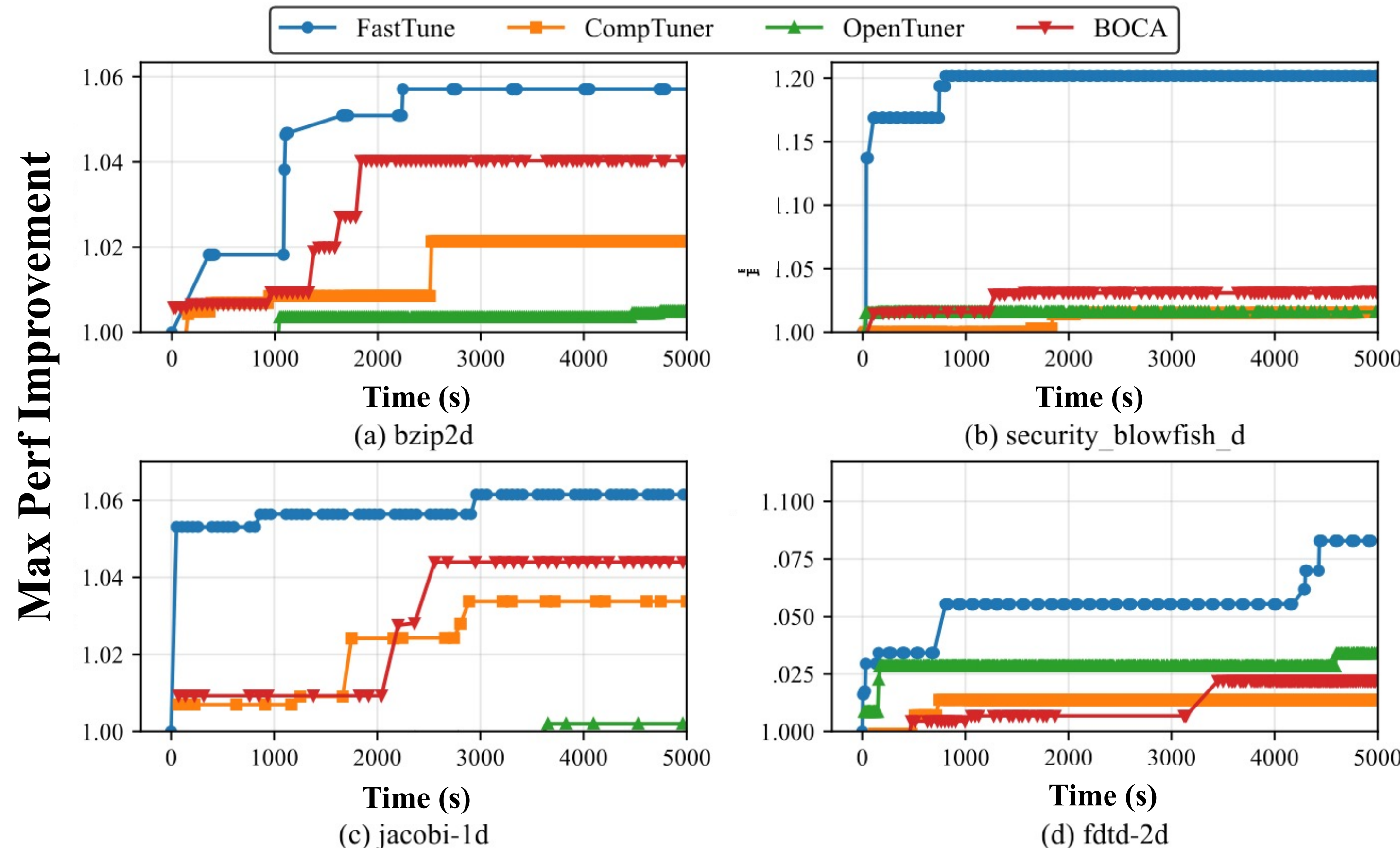
PolyBench

ID	Project	Lines	ID	Project	Lines
P1	durbin	132	P10	heat-3d	211
P2	gesummv	147	P11	adi	168
P3	gemm	232	P12	fdtd-2d	170
P4	ludcmp	184	P13	jacobi-1d	117
P5	syr2k	225	P14	nussinov	569
P6	syrk	210	P15	bicg	145
P7	2mm	160	P16	mvt	147
P8	doitgen	128	P17	symm	151
P9	jacobi-2d	120			

Evaluation-Performance Improvement

PEKING UNIVERSITY

- FastTune converges faster and consistently stays ahead of all baselines
- 500s: +9.8% average speedup over –O3
- 2000s: +12.3% average speedup over –O3
- Best baseline at 5000 s: only about +3.2%
- FastTune achieves both faster convergence and a higher performance ceiling



Evaluation-Cost Model Components

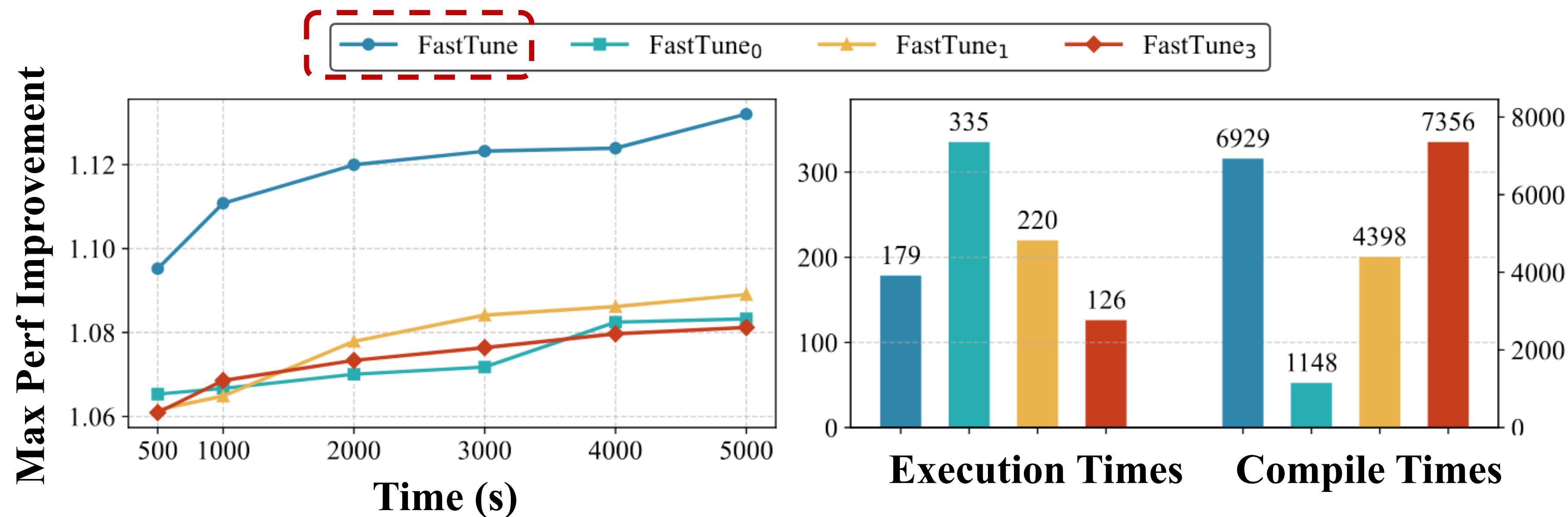
- Graph structure is the most critical: passes like code sinking only change topology, not instruction content
- Hotness weighting provides +15% Top-3 accuracy, without it, cold node dilutes the signal

Configuration	Top-1	Top-3	Top-5	NDCG
ALL	**44.46%**	**58.98%**	**77.53%**	**0.8124**
w/o Hotness weighting	34.46%	43.19%	65.59%	0.7223
w/o Graph Structrue	30.48%	39.79%	61.33%	0.6918

Evaluation-Real Measurement Frequency

北京大學 PEKING UNIVERSITY

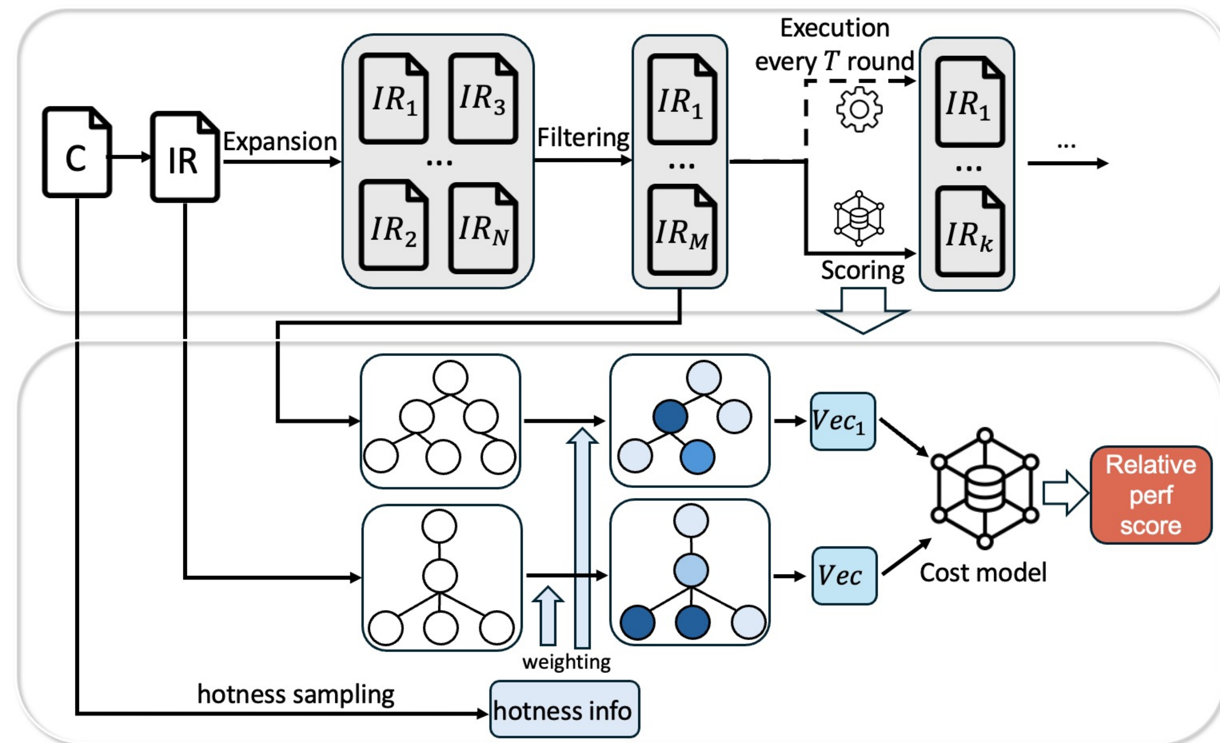
- Too frequent: waste time budget on execution, limits search depth
- Too rare: prediction errors accumulate, search drifts
- T=2 balances exploration breadth with correction accuracy



Conclusion

北京大學 PEKING UNIVERSITY

- FastTune improves LLVM pass ordering by predicting relative gain rather than absolute final performance or execute program
- A hotness-aware cost model and cost-model-guided beam search make tuning both efficient reliable
- Experiments show that FastTune converges faster and achieves clearly better performance than existing baselines



Thanks for Listening

Q&A

bingyugao@stu.pku.edu.cn