Automating Reinforcement Learning Architecture Design *for* Code Optimization

Huanting Wang

Zhanyong Tang
Cheng Zhang
Jiaqi Zhao

NORTHWEST UNIVERSITY

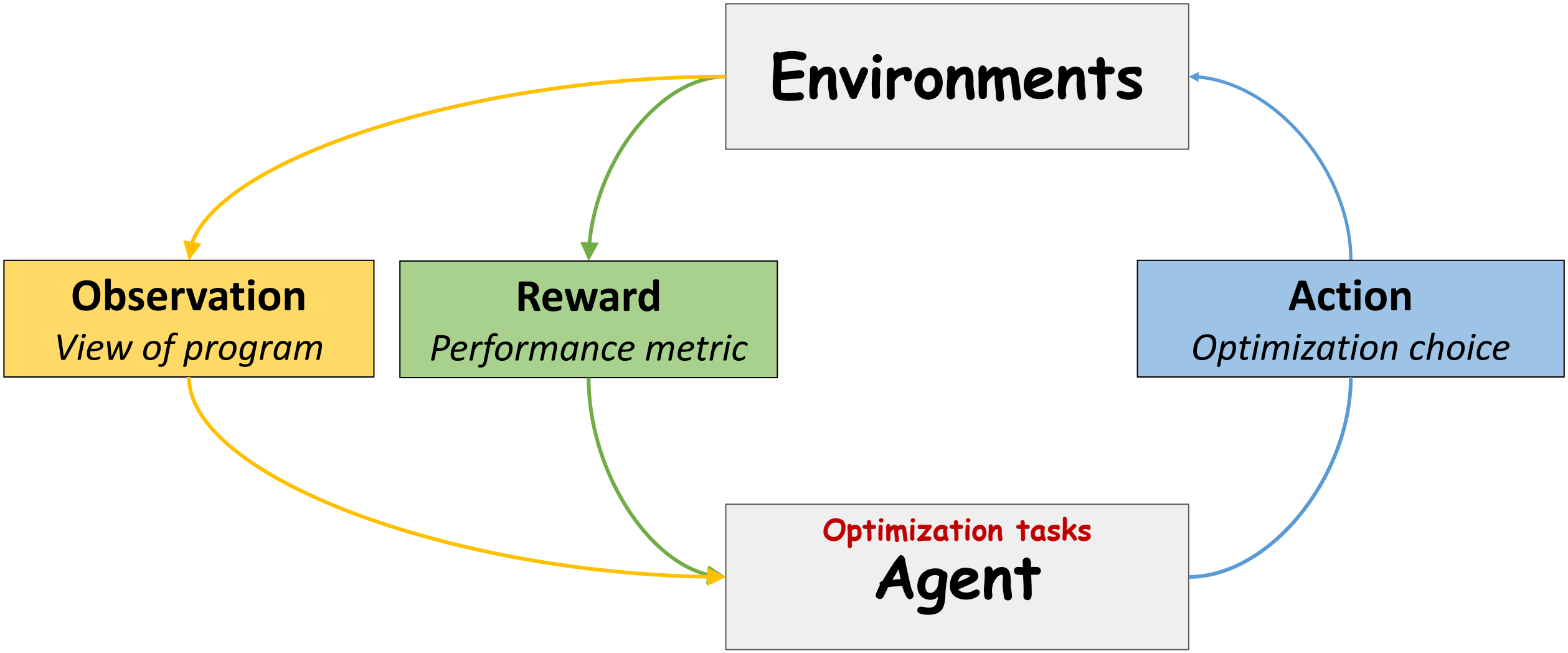
Chris Cummins
Hugh Leather

Meta

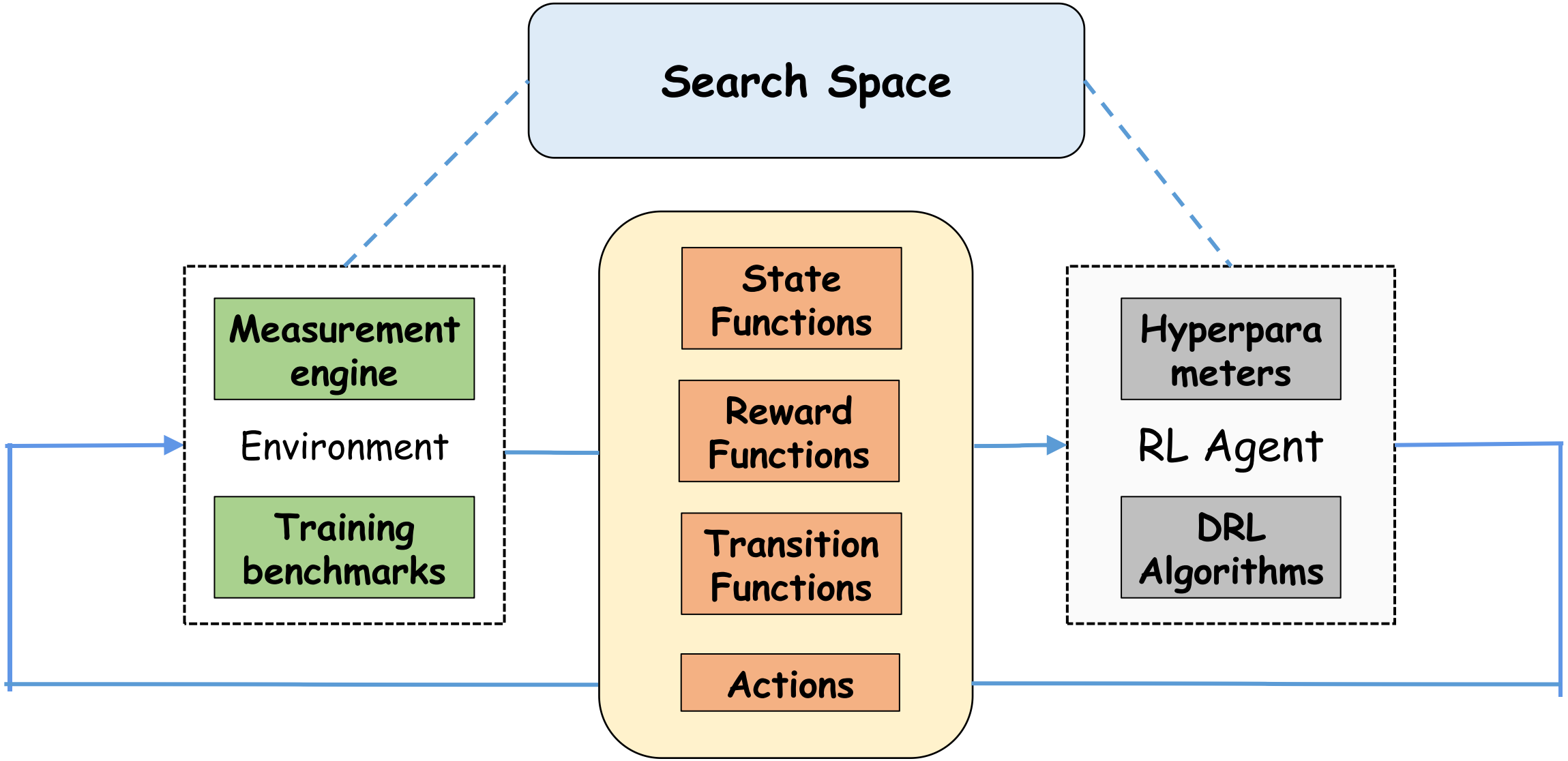
Zheng Wang

UNIVERSITY OF LEEDS

Reinforcement Learning in Code Optimization

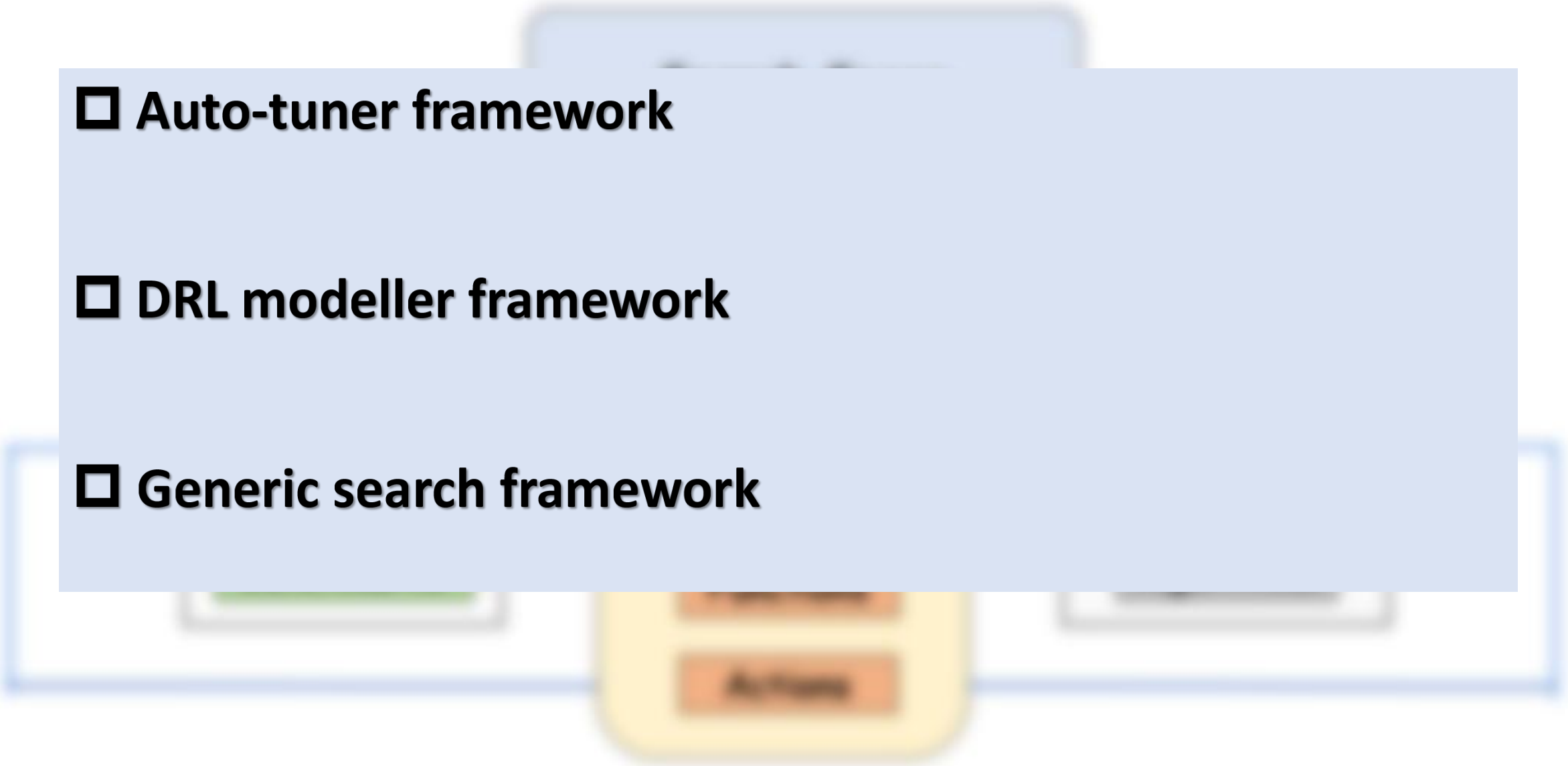


Pipeline in Integrating RL into Compilers

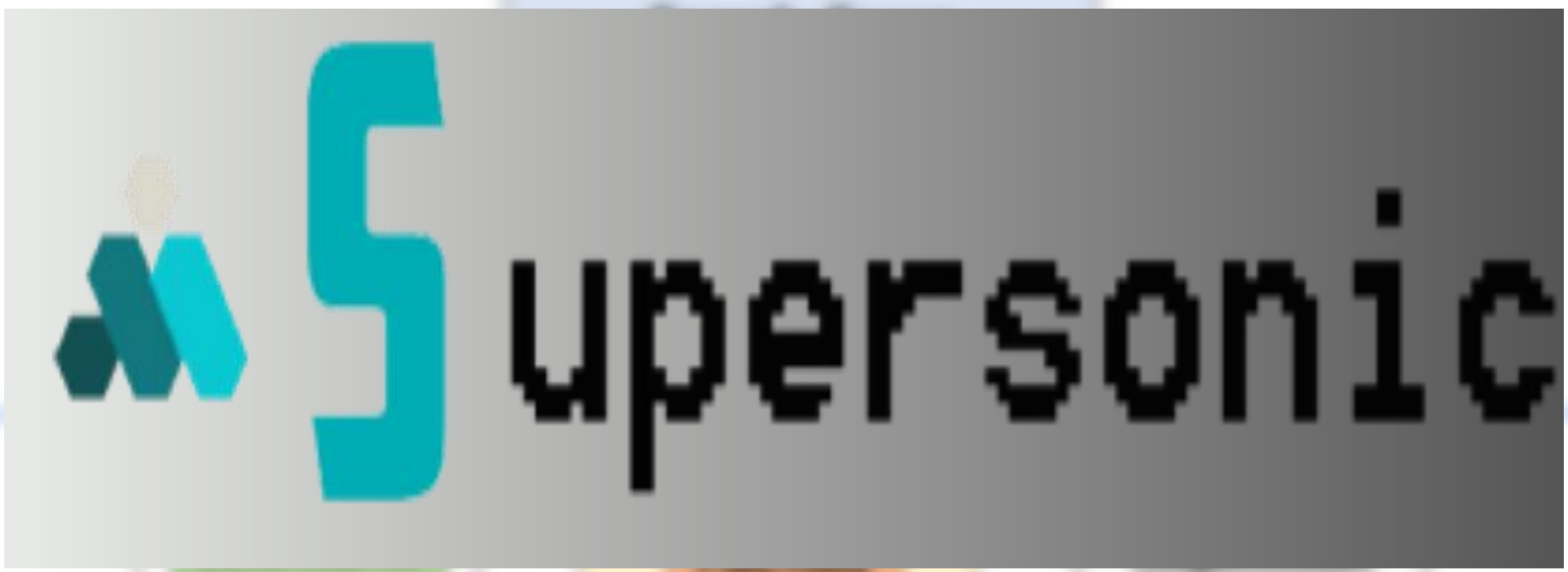


Integrating RL into Compilers

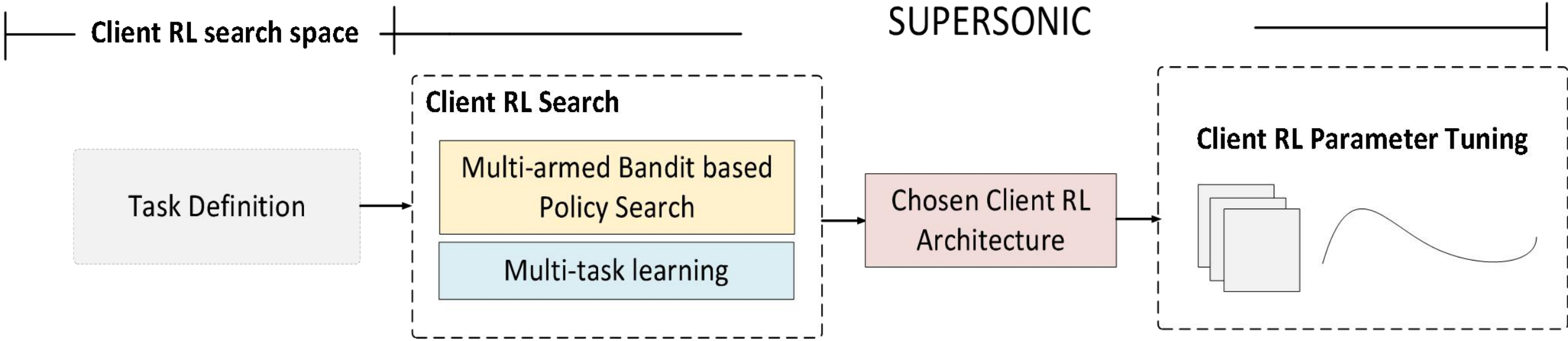
- ☐ Auto-tuner framework
- ☐ DRL modeller framework
- ☐ Generic search framework



Integrating RL into Compilers

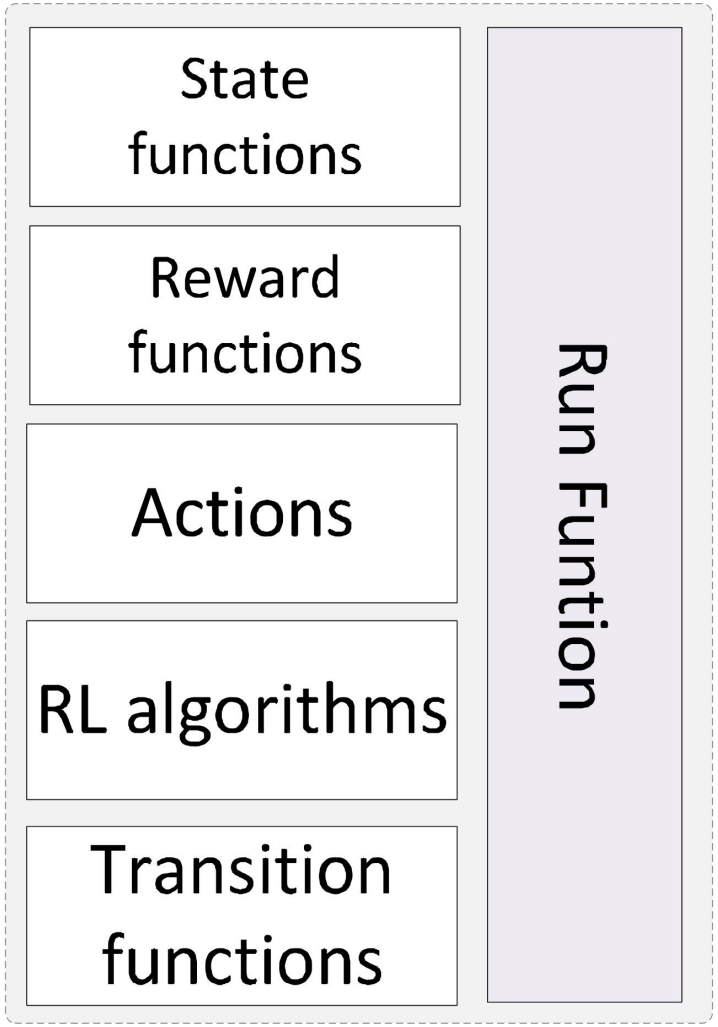


Our Approach



Offline Task defining and Client RL Search; **Done once**

Our Approach: task definition



Client RL search space

Provides:

- A list of state functions

-A list of reward functions

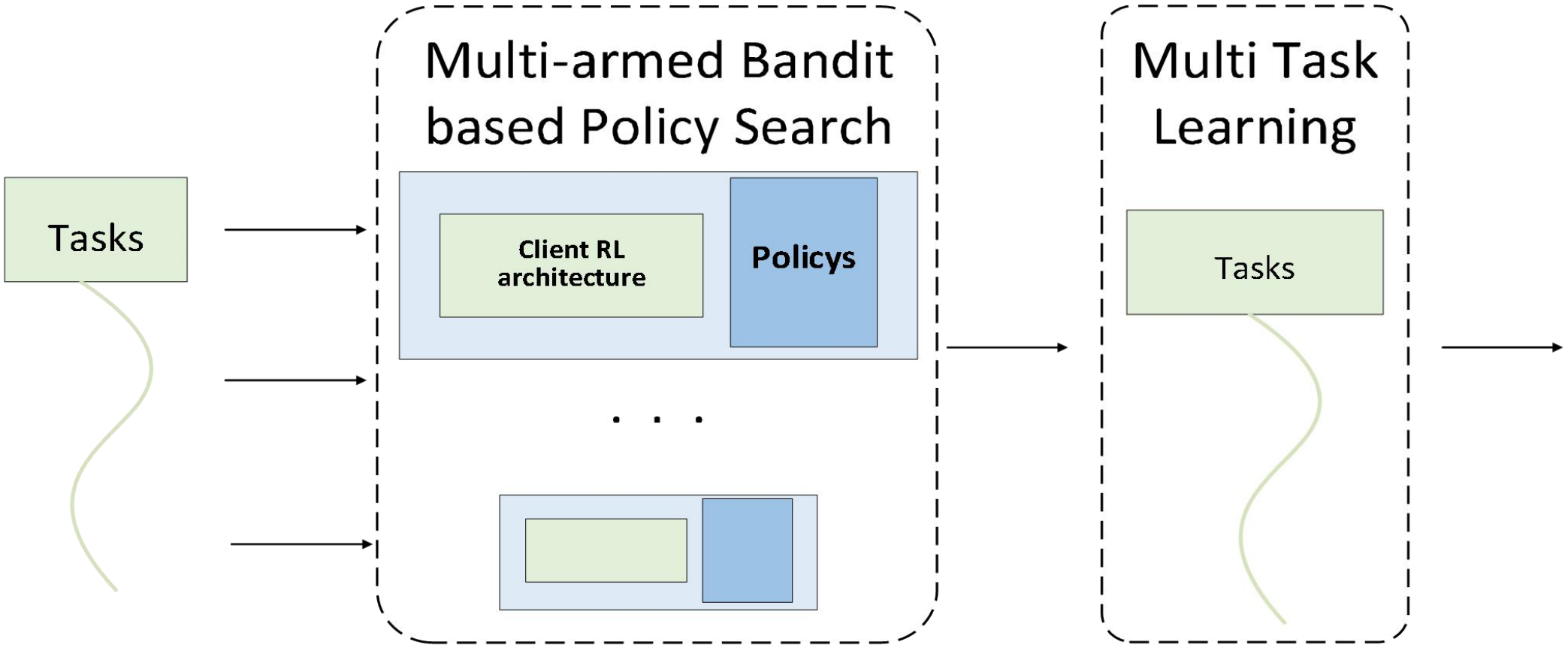
-A list of actions to take

-A list of RL algorithms to take

-Transition functions used by Rl algorithms

-Run function

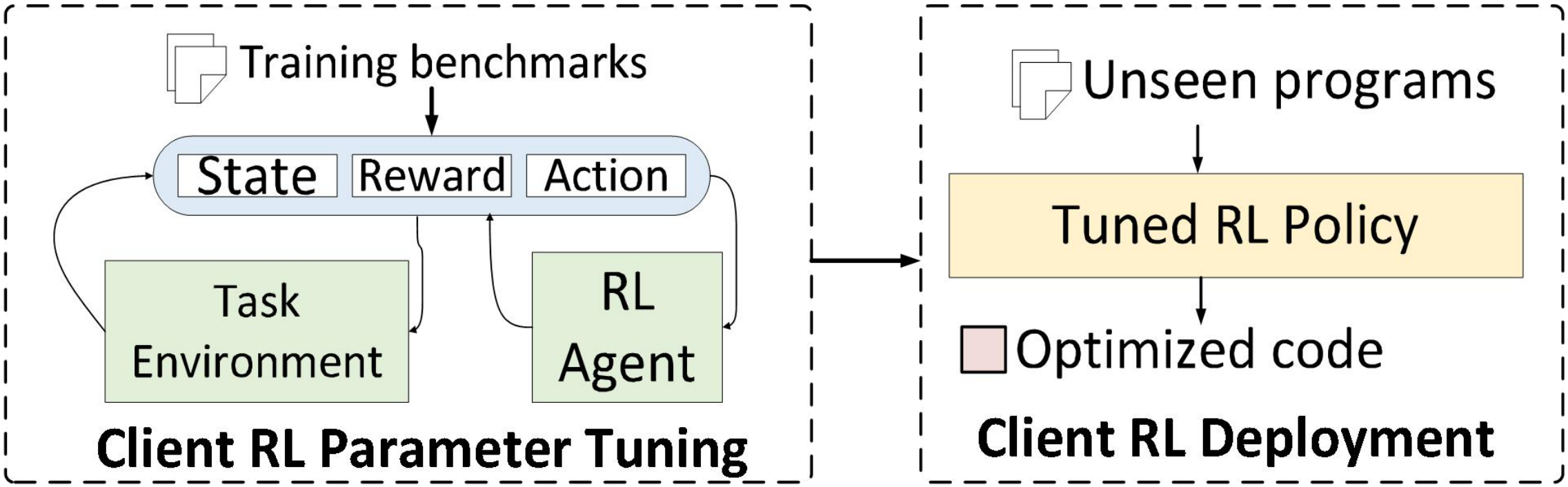
Our Approach: client RL search



Automatically search for the best policy

Parallel learning which policy is the best

Our Approach: parameter tuning and deployment

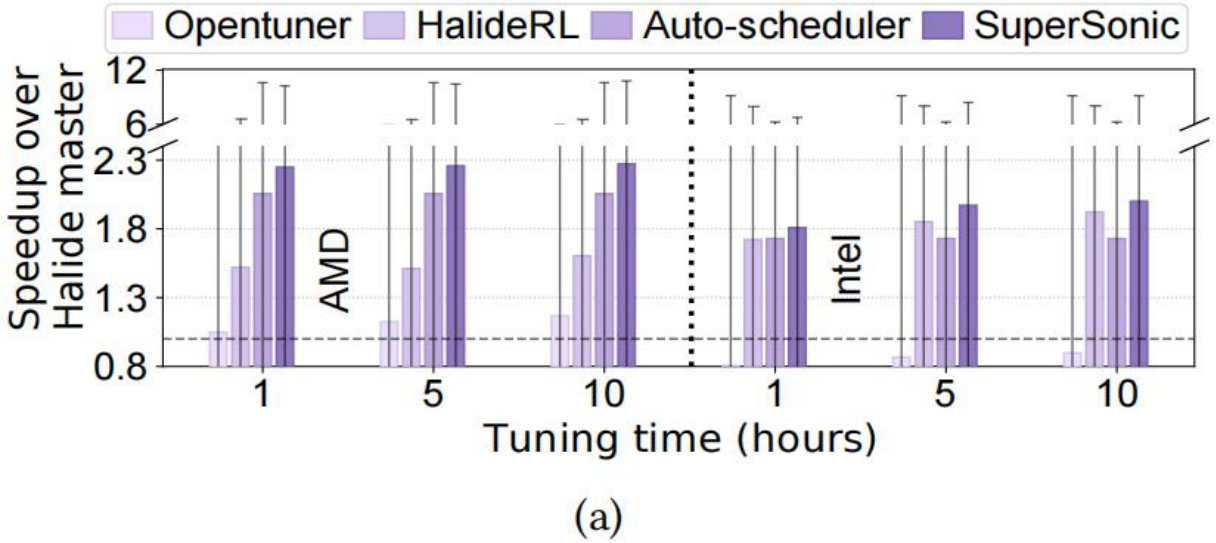


Case Studies

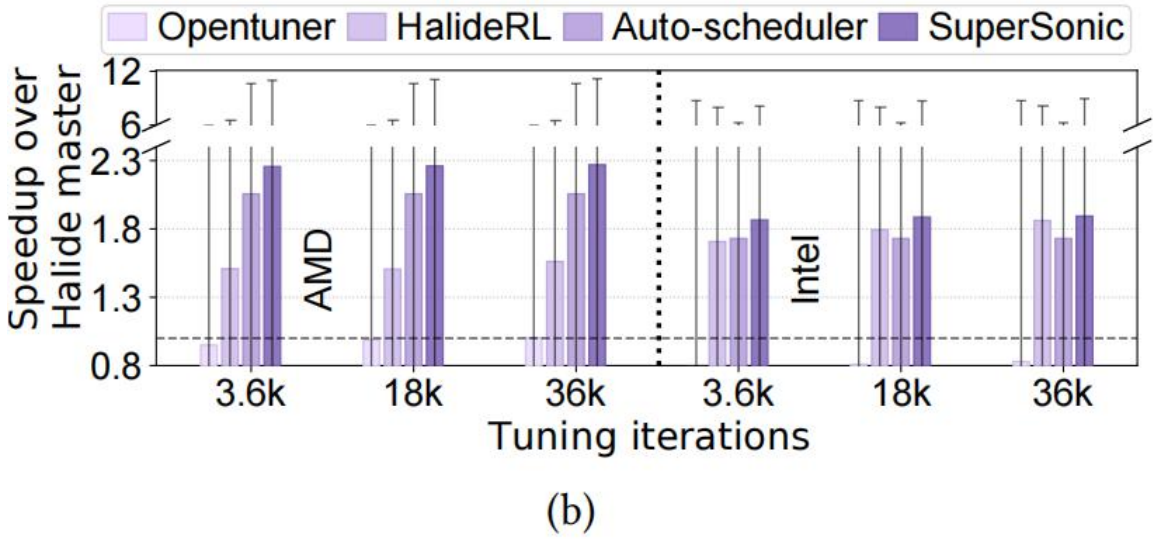
- **Optimizing Image Pipelines**
  - Ten benchmarks
  - Two platforms
  - four SOTAs
- **Neural Network Code Generation**
  - Five benchmarks
  - Two platforms
  - Seven SOTAs
- **Code Size Reduction**
  - 43 benchmarks
  - Four SOTAs
- **Superoptimization**
  - 40 benchmarks
  - Two platforms
  - Three SOTAs

Cross Validation

Optimizing Image Pipelines



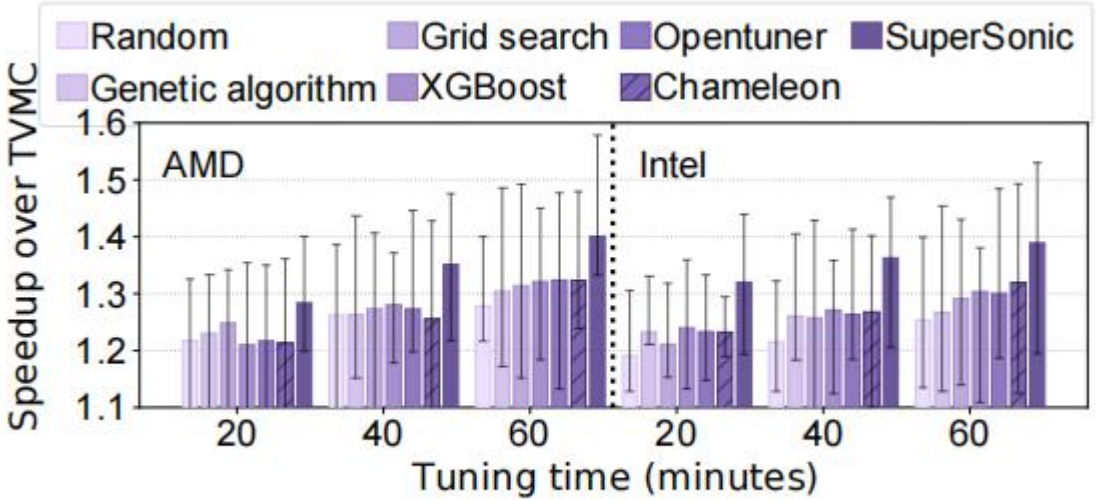
(a)



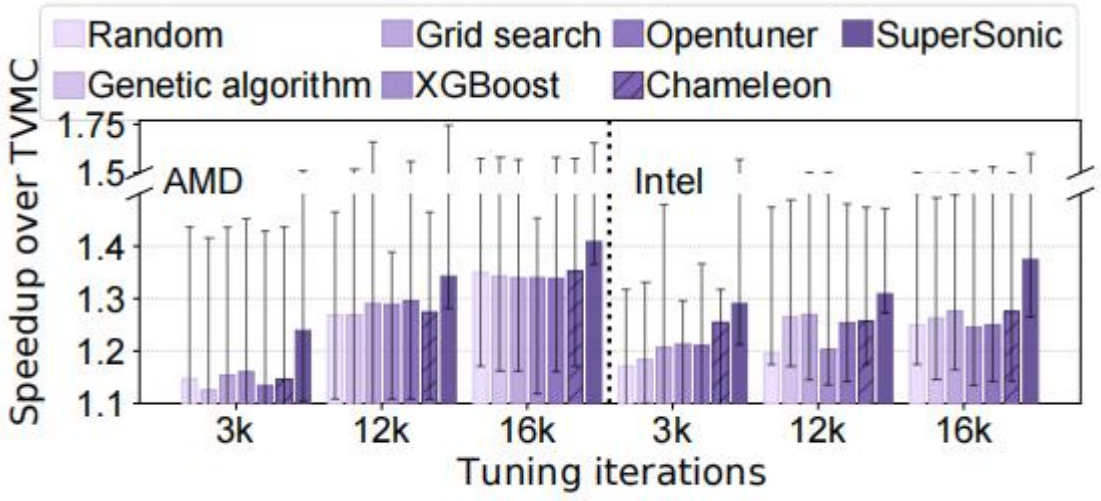
(b)

Our approach gives the **11x speedup** on both platforms

Neural Network Code Generation



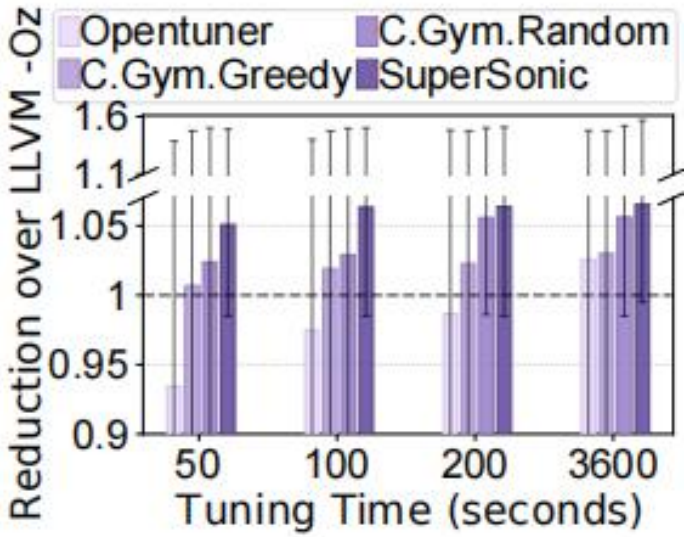
(a)



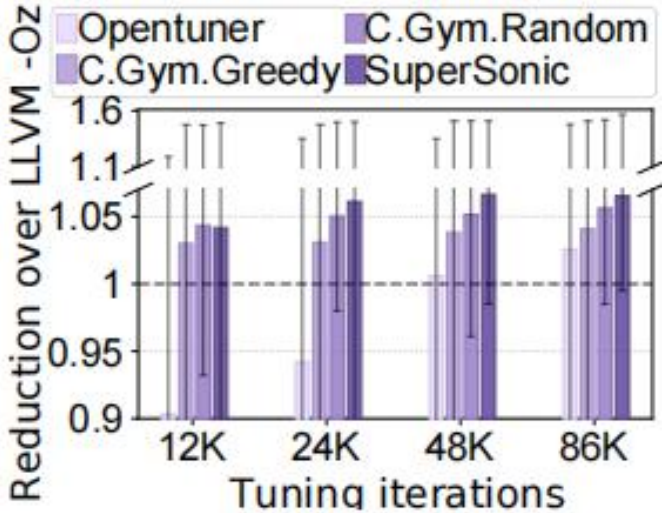
(b)

Our approach can improve the default schedules by up to 1.74x

Code Size Reduction



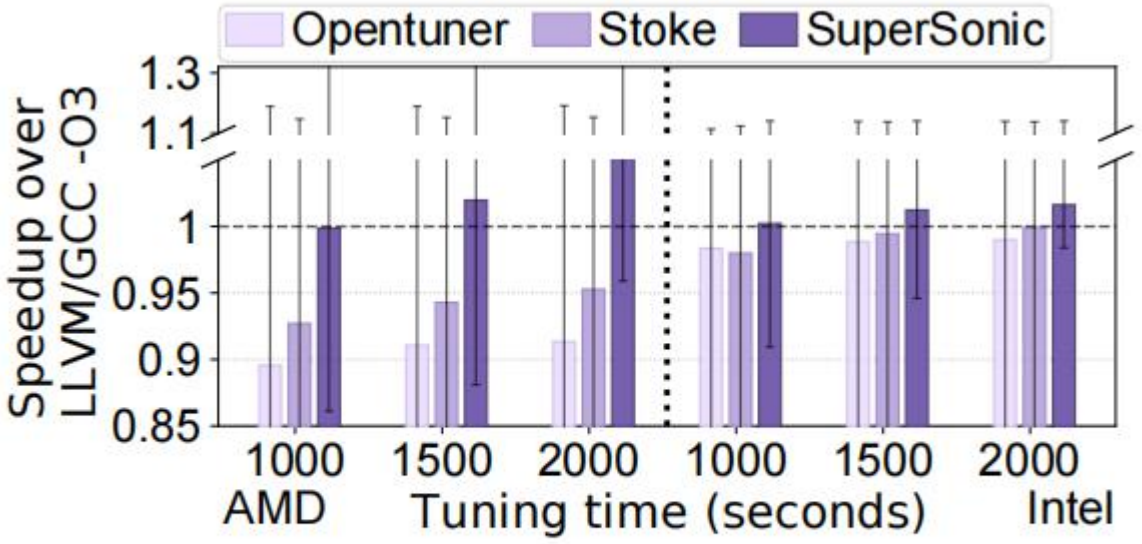
(a)



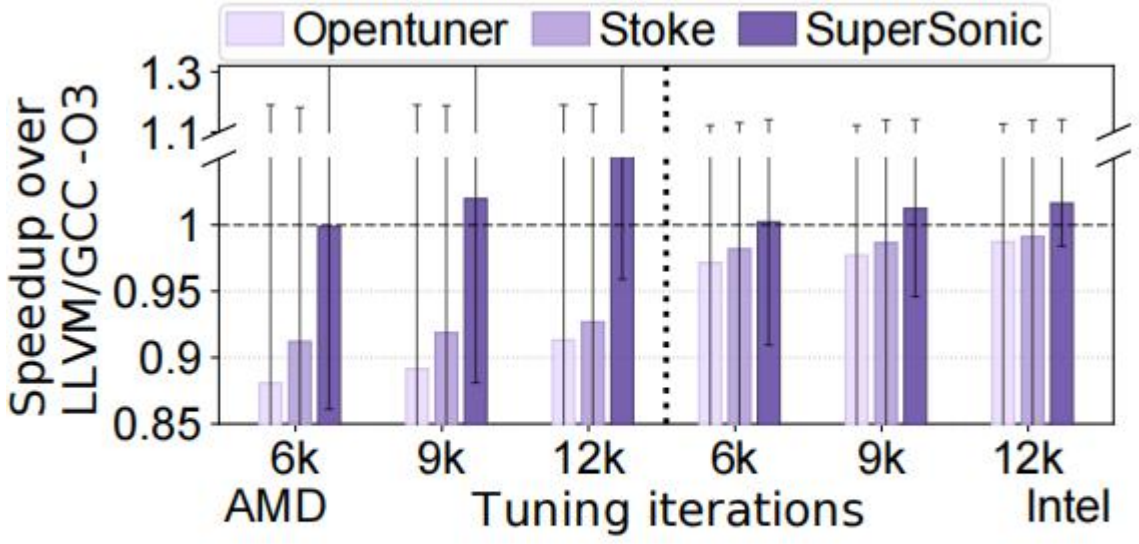
(b)

Supersonic gives the *highest* code size reduction by 1.57x

Superoptimization



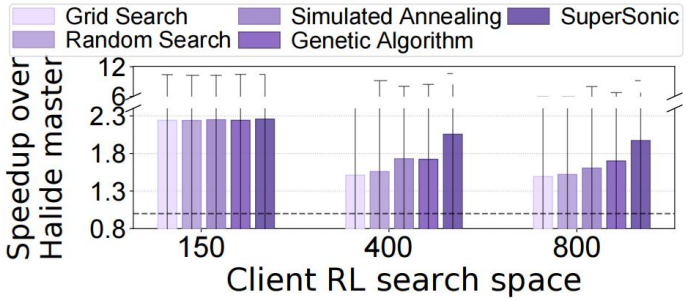
(a)



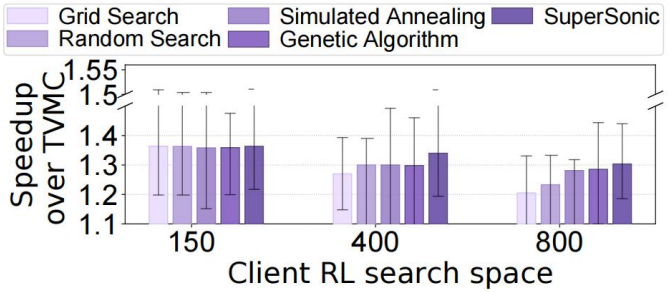
(b)

Supersonic can deliver the *highest* improvement by better exploring the optimization space (up to 1.34x)

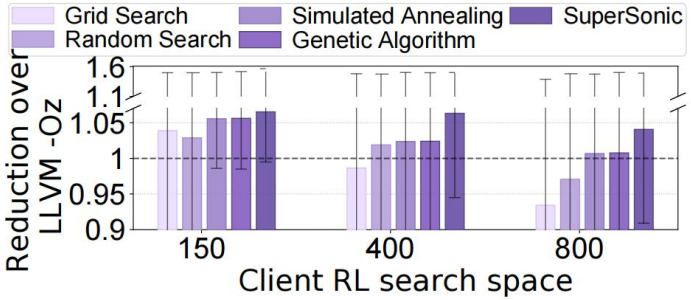
Compare to Other Search Strategies



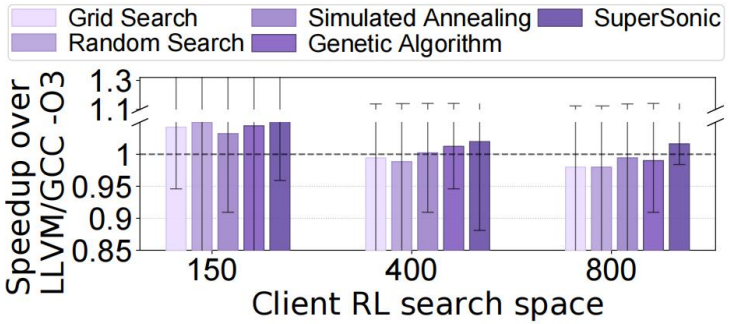
(a) Optimizing Image Pipelines



(b) Neural Network Code Generation



(c) Code size reduction



(d) Superoptimization

All client RL give an **average improvement** over other baseline approaches

Search Overhead

Use cases	MAX	Geomean	Min
Case study 1	28 X	2.5 X	1.4 X
Case study 2	**100 X**	2.5 X	1.4 X
Case study 3	67 X	3.2 X	1.6 X
Case study 4	91 X	3.0X	2.4 X

SuperSonic gives ***up to 100x*** **less search time** compared the best-performing tuning algorithm

Summary

- A generic framework to automatically choose and tune a suitable RL architecture for code optimization tasks
- Using deep RL as a meta-optimizer to support the integration of RL into performance tuners
- Achieving better performance over SOTAs across code optimization tasks

Fork me on Github

https://github.com/HuantWang/SUPERSONIC