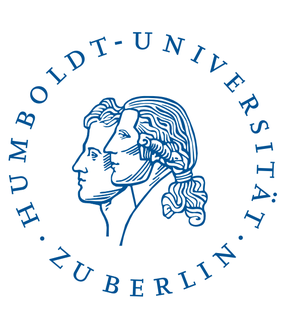
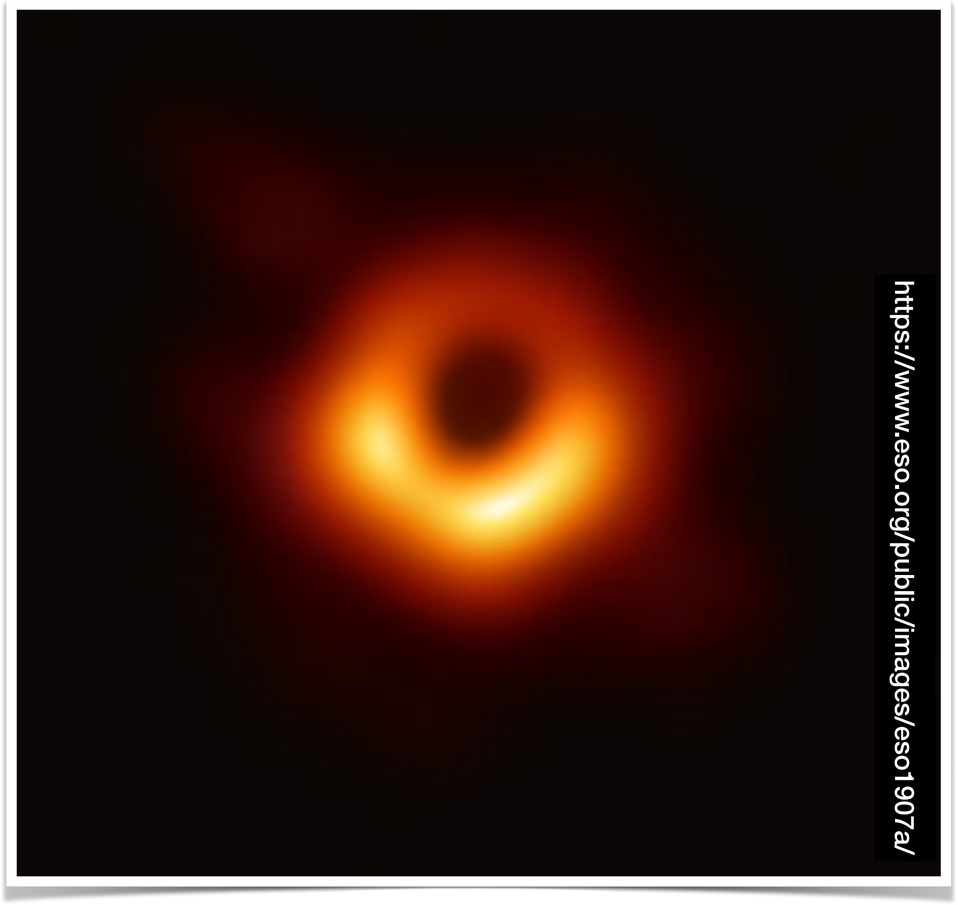


Predictive Resource Management for Scientific Workflows

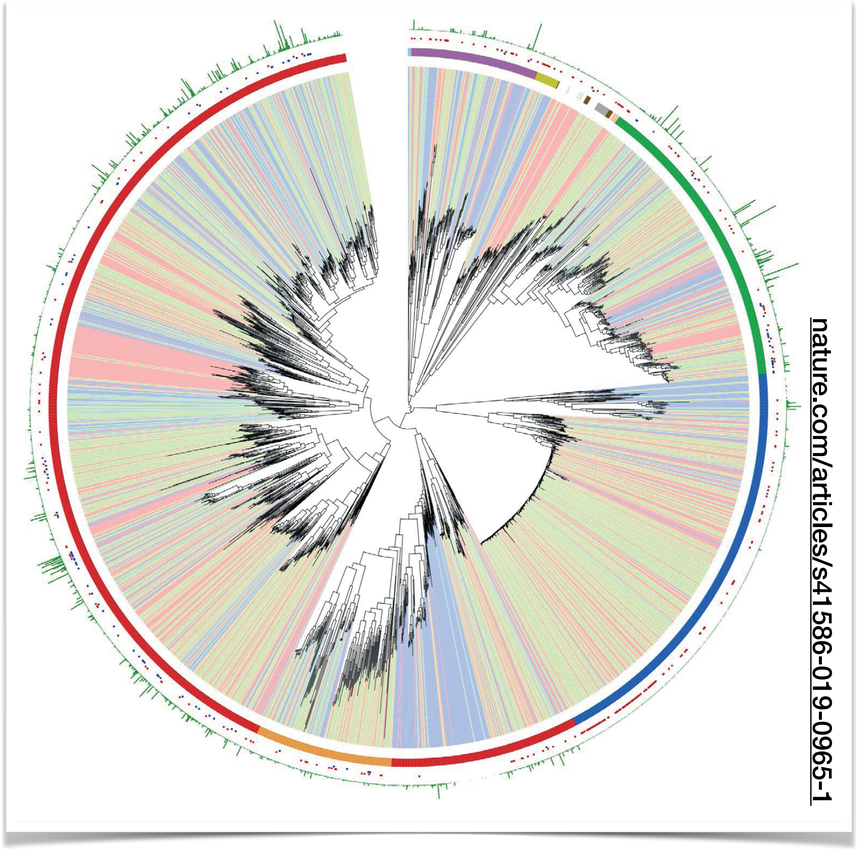
Doctoral Defense

Carl Witt

Science Through Large-Scale Data Analysis



4.5 petabytes input data



10^{12} base pairs input data

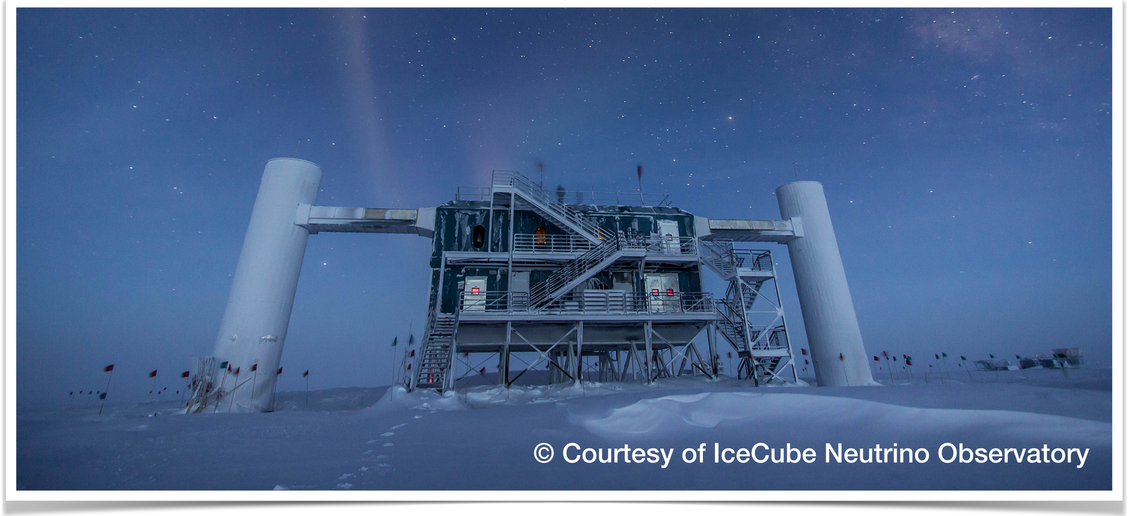
IceCube Neutrino Observatory

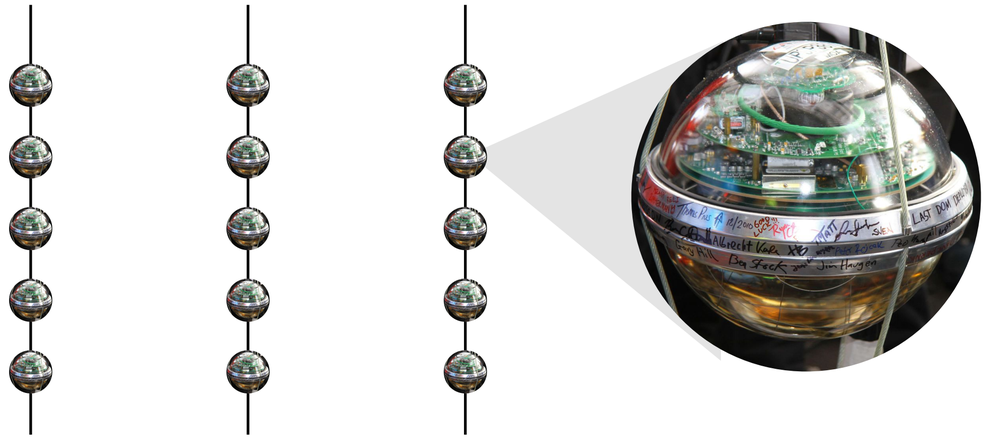
IceCube Experiment

- 5160 optical sensors
- 1 km^3 neutrino telescope

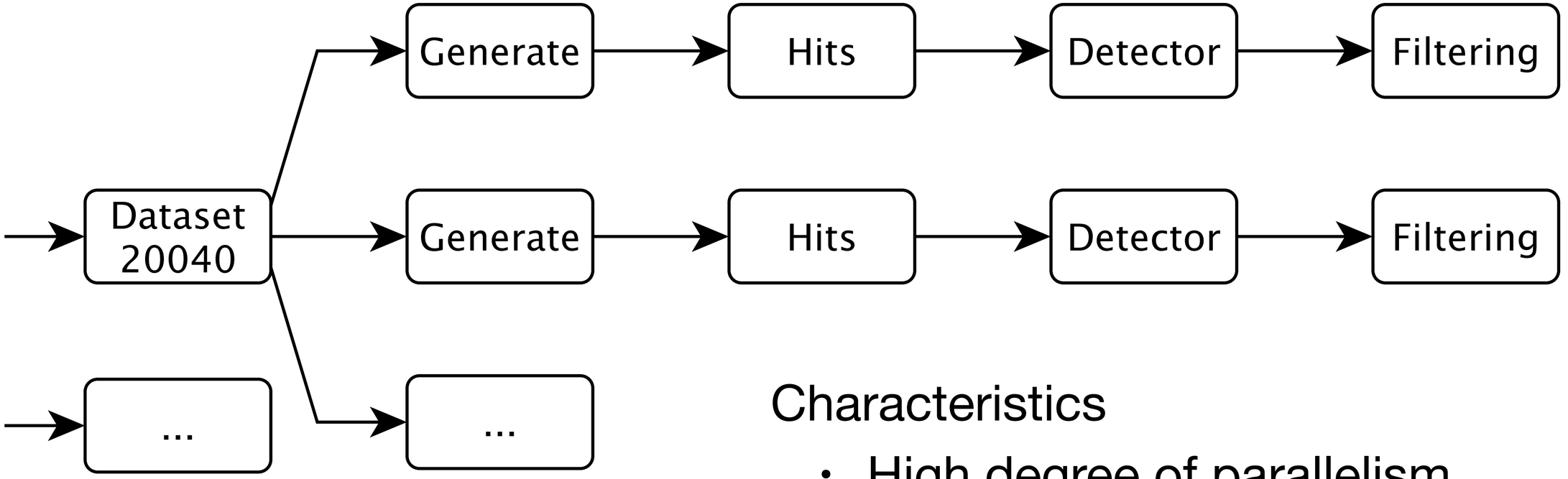
Scientific Workflow

- Simulate response to physical events
- High CPU and main memory demand





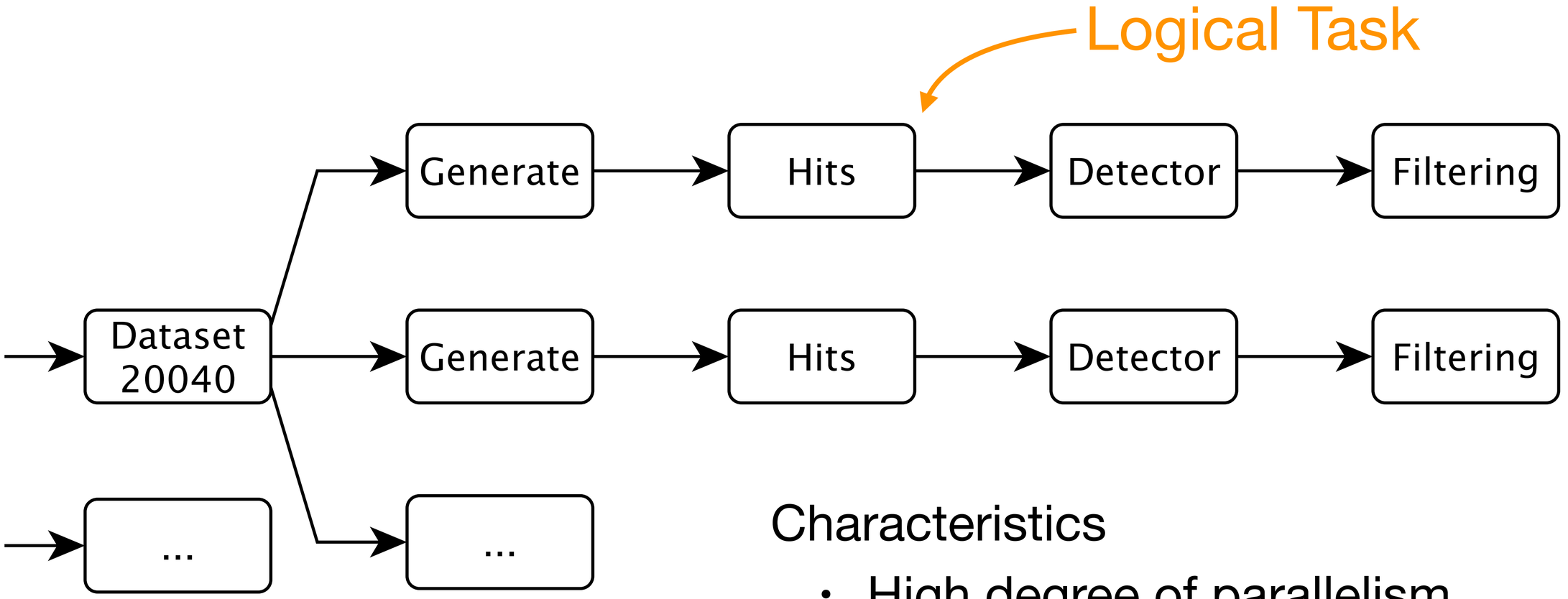
Example Workflow



Characteristics

- High degree of parallelism
- Repetitive tasks

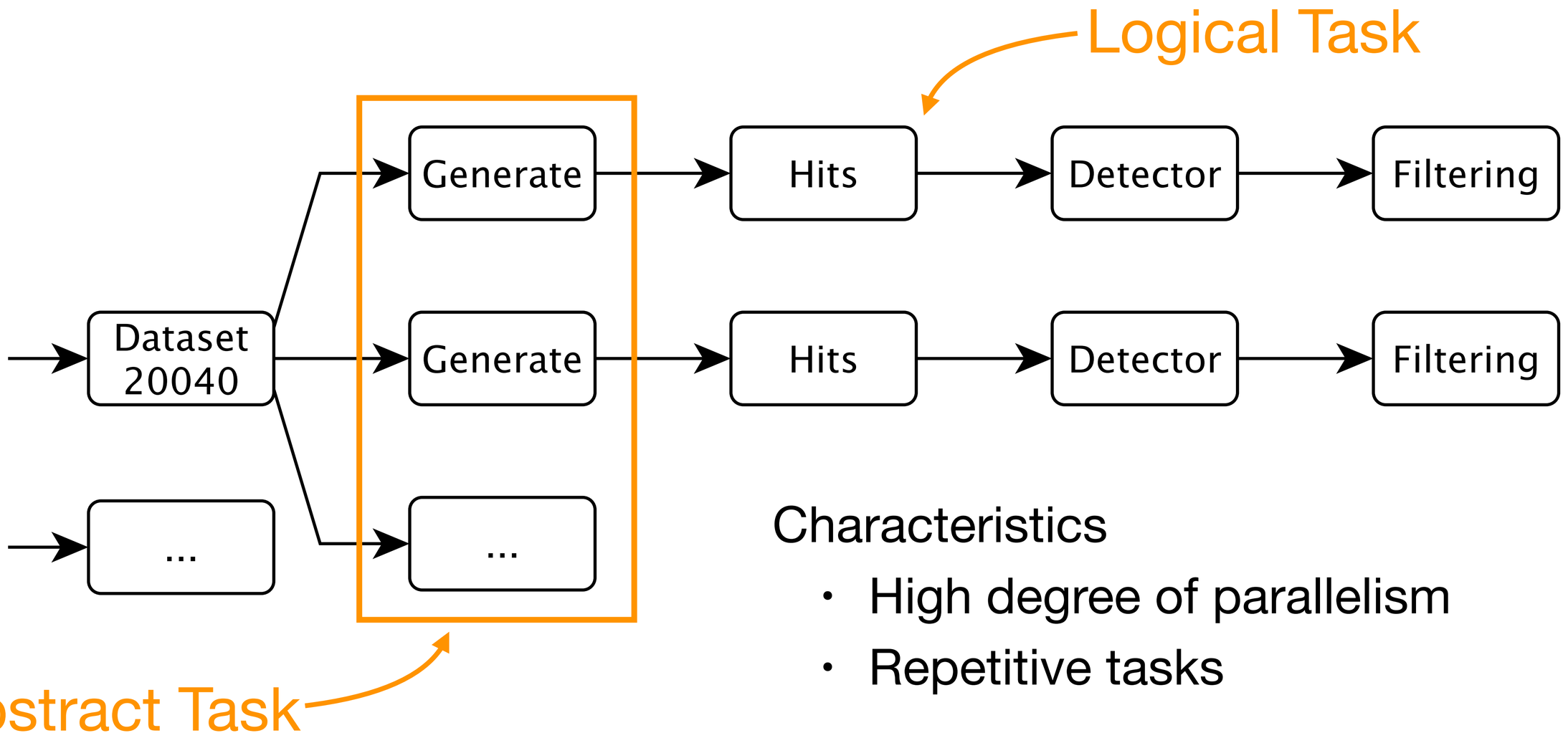
Example Workflow



Characteristics

- High degree of parallelism
- Repetitive tasks

Example Workflow



Characteristics

- High degree of parallelism
- Repetitive tasks

Batch Scheduling

Batch scheduler

- Accepts task descriptions
- Requires resources usage estimates
- Terminates tasks if they exceed reservation

Resource usage estimates

- Delegated to user
- Learned from historical data

Batch Scheduling

Batch scheduler

- Accepts task descriptions
- Requires resources usage estimates
- Terminates tasks if they exceed reservation

Resource usage estimates

- Delegated to user
- Learned from historical data

Predictive Resource Allocation

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Feedback-Based Scheduling of Scientific Workflows

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic

Highlights

[WWL18b] **Carl Witt**, Dennis Wagner, Ulf Leser: “POS: Online Learning for Memory-Aware Scheduling of Scientific Workflows.“ *IEEE 14th International Conference on e-Science*: 399-400 (2018)
[WWL19] **Carl Witt**, Dennis Wagner, and Ulf Leser. “Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows.“ *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

Feedback-Based Scheduling of Scientific Workflows

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic

Highlights

- Online learning

[WWL18b] **Carl Witt**, Dennis Wagner, Ulf Leser: “POS: Online Learning for Memory-Aware Scheduling of Scientific Workflows.“ *IEEE 14th International Conference on e-Science*: 399-400 (2018)
[WWL19] **Carl Witt**, Dennis Wagner, and Ulf Leser. “Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows.“ *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

Feedback-Based Scheduling of Scientific Workflows

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic

Highlights

- Online learning
- Peak memory prediction from scratch

[WWL18b] **Carl Witt**, Dennis Wagner, Ulf Leser: “POS: Online Learning for Memory-Aware Scheduling of Scientific Workflows.“ *IEEE 14th International Conference on e-Science*: 399-400 (2018)
[WWL19] **Carl Witt**, Dennis Wagner, and Ulf Leser. “Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows.“ *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

Feedback-Based Scheduling of Scientific Workflows

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic

Highlights

- Online learning
- Peak memory prediction from scratch
- Comparison to static predictions / user estimates

[WWL18b] **Carl Witt**, Dennis Wagner, Ulf Leser: “POS: Online Learning for Memory-Aware Scheduling of Scientific Workflows.“ *IEEE 14th International Conference on e-Science*: 399-400 (2018)
[WWL19] **Carl Witt**, Dennis Wagner, and Ulf Leser. “Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows.“ *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

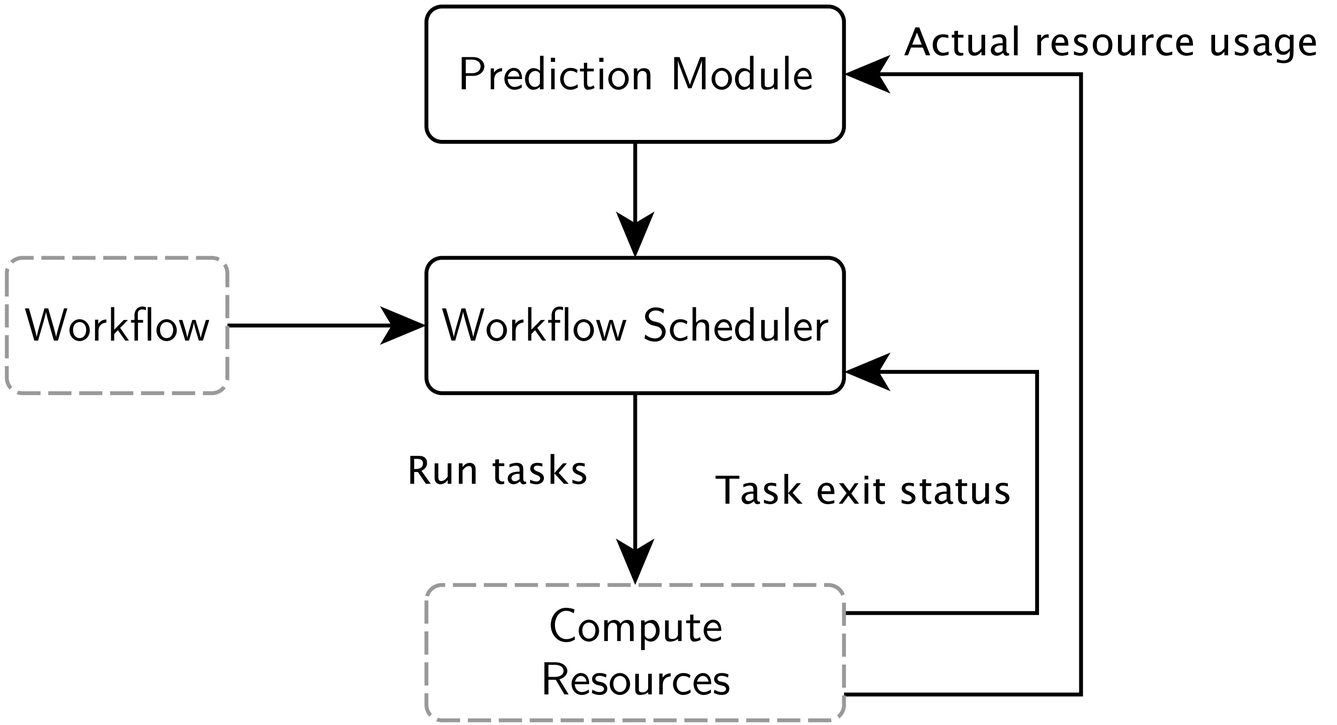
Simulation Overview 1/2

Performance Metrics

- Prediction accuracy
- Execution time

Simulation Parameters

- Workflow
- Scheduling heuristic
- Compute resources
- Prediction model



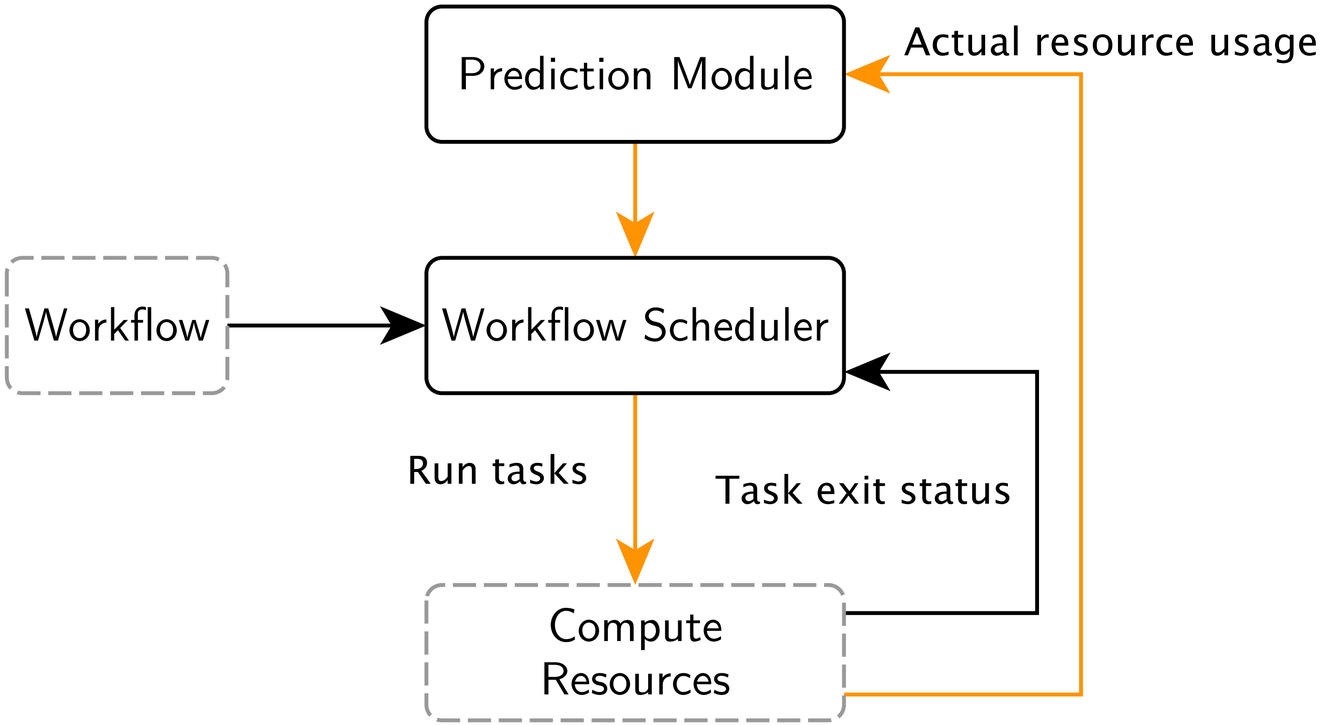
Simulation Overview 1/2

Performance Metrics

- Prediction accuracy
- Execution time

Simulation Parameters

- Workflow
- Scheduling heuristic
- Compute resources
- Prediction model

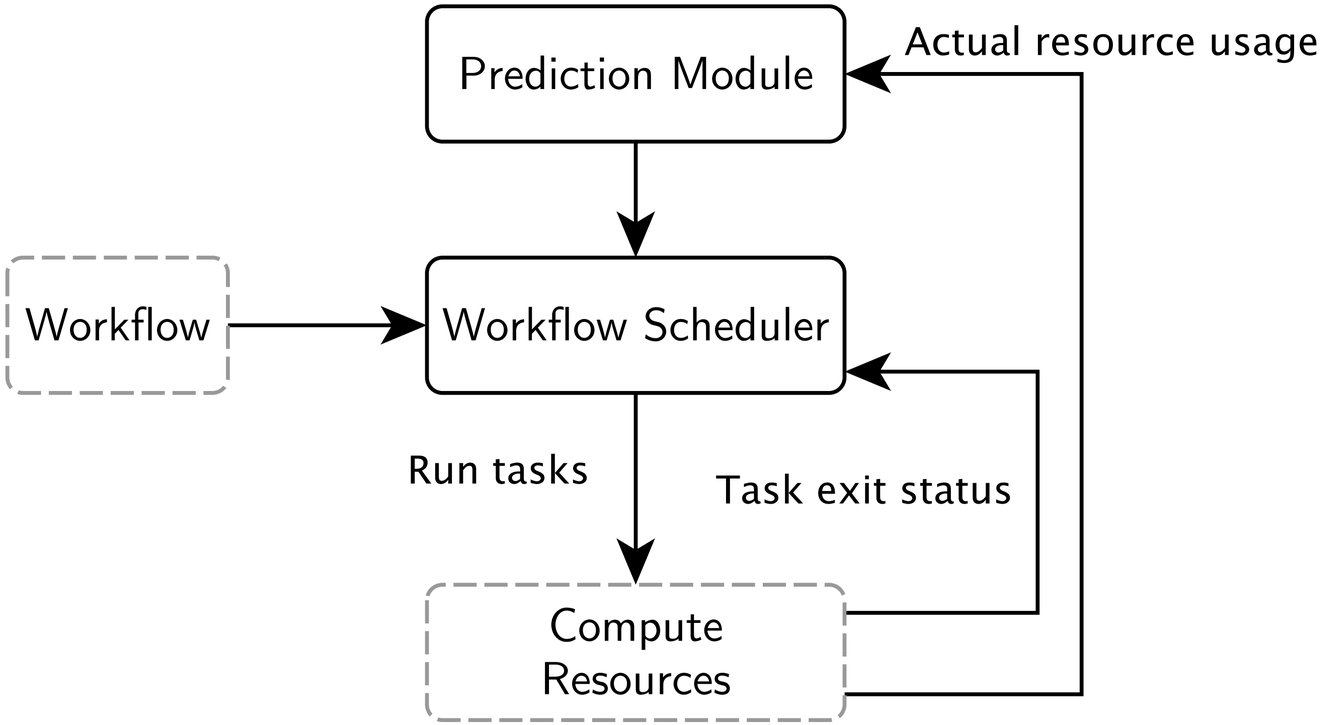


Feedback-Based Scheduling

Simulation Overview 2/2

Workflow execution process

1. Tasks become eligible
   - Predict peak memory usage
   - While resources are available: start tasks
2. Asynchronous task execution
3. Success: update predictions
4. Failure: goto 1 with double allocation



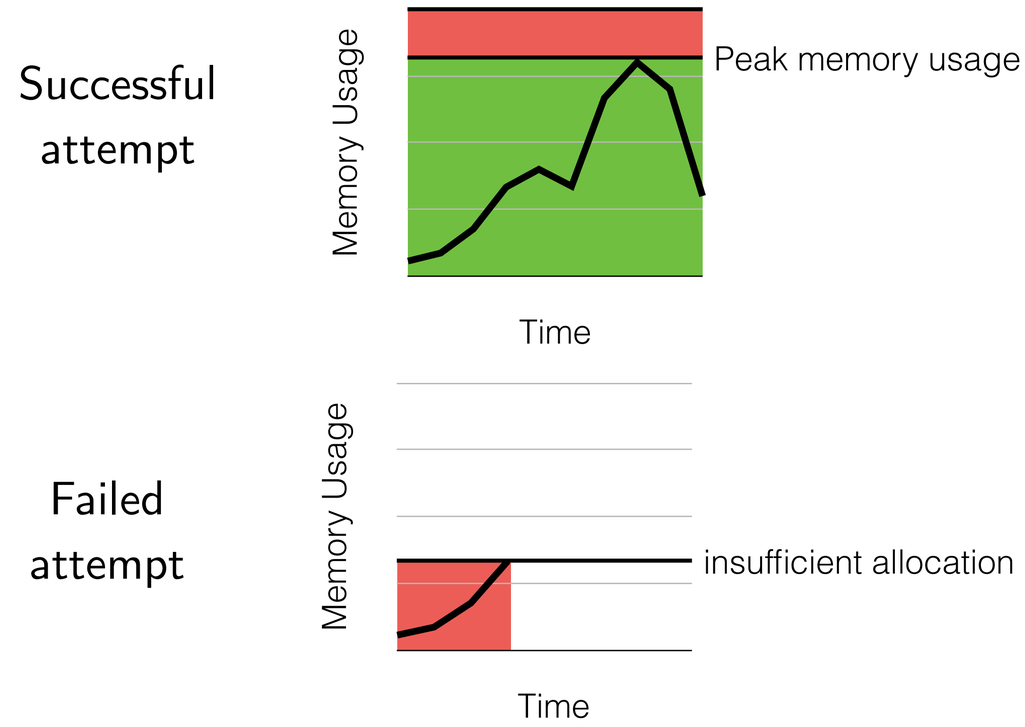
Performance Metrics

Memory Allocation Quality (MAQ)

- Prediction accuracy
- Used allocations: accommodates peak
- Wasted allocations: insufficient and excess allocations

Makespan Ratio (MSR)

- Elapsed real time from start to end



$$\text{MAQ} = \frac{\text{used allocation}}{\text{used allocation} + \text{wasted allocation}}$$

$$\text{MSR} = \frac{\text{wall-clock time}}{\text{lower bound on wall-clock time}}$$

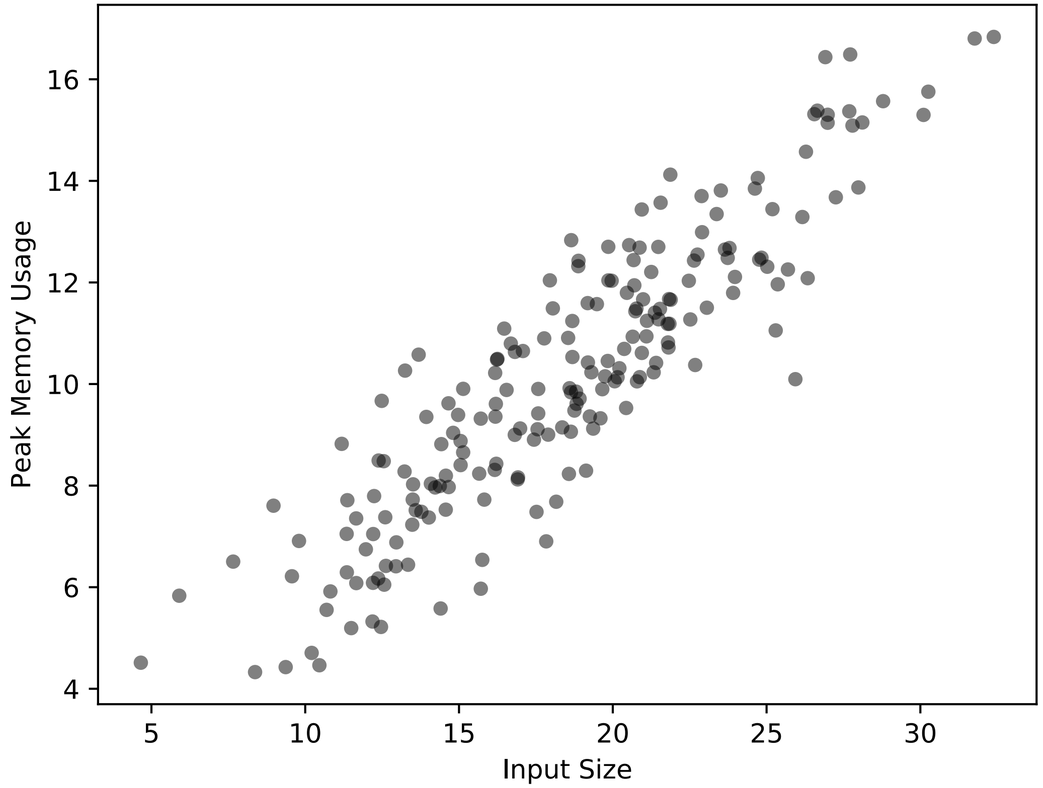
Prediction Models

Hypothetical User Estimate {Power 2, Q99}

- Power 2: Round maximum peak usage to nearest power of 2 GB

Conservative Regression {LR 50…LR 100}

- compute OLS
- add q-th residual percentile to intercept
- q ~ degree of risk aversion



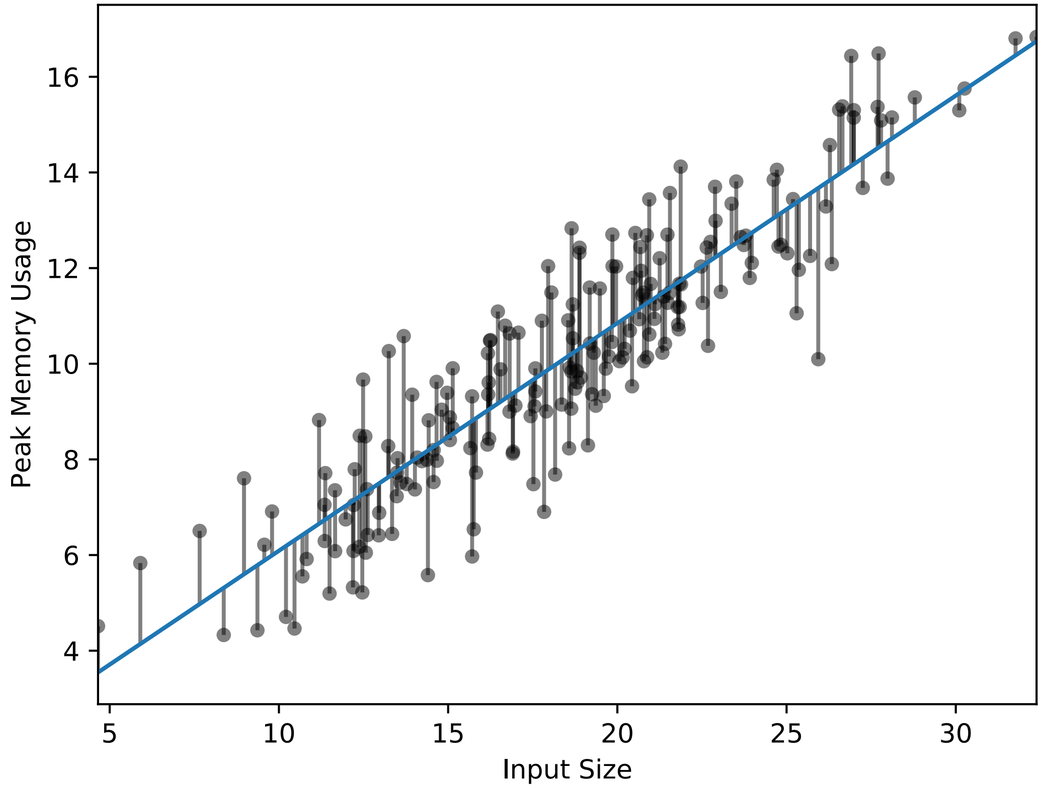
Prediction Models

Hypothetical User Estimate {Power 2, Q99}

- Power 2: Round maximum peak usage to nearest power of 2 GB

Conservative Regression {LR 50…LR 100}

- compute OLS
- add q-th residual percentile to intercept
- q ~ degree of risk aversion



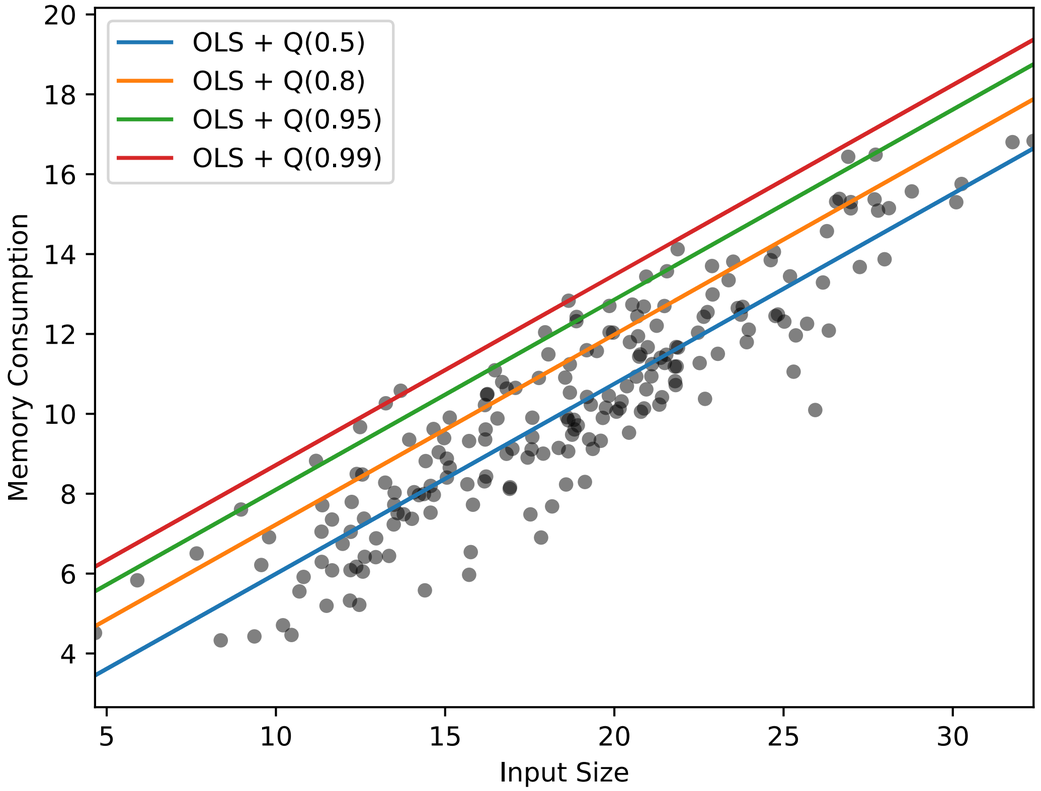
Prediction Models

Hypothetical User Estimate {Power 2, Q99}

- Power 2: Round maximum peak usage to nearest power of 2 GB

Conservative Regression {LR 50…LR 100}

- compute OLS
- add q-th residual percentile to intercept
- q ~ degree of risk aversion



Benchmark Workflows

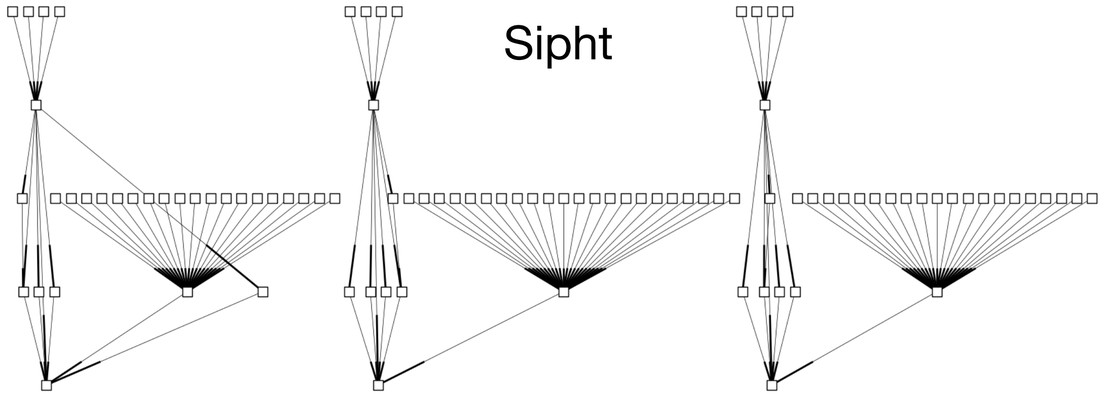
Topology {Montage, Ligo, Sipht, Genome, Cybershake} [JCD+13]

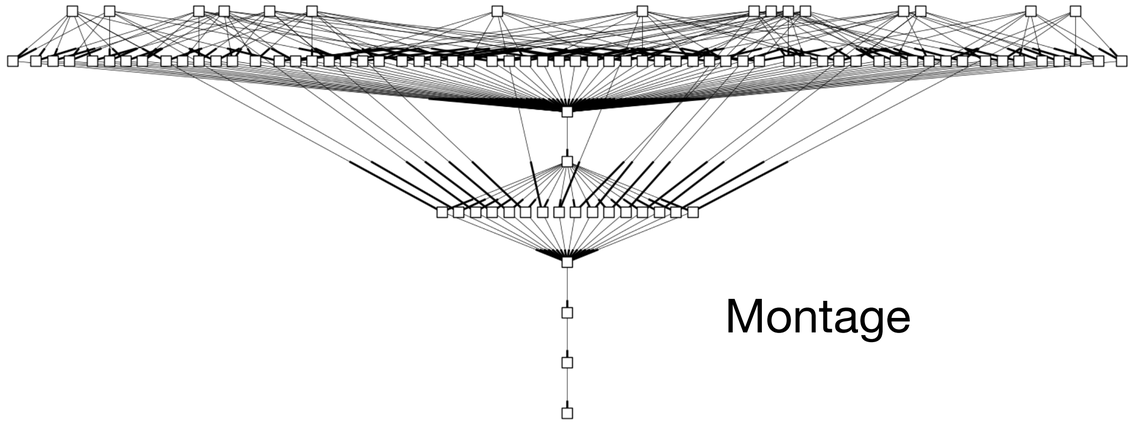
1k tasks per workflow

Resource usage {instance 1…100}

Time to failure {0.1, 0.5, 0.9}

- task execution time modifier





Benchmark Workflows

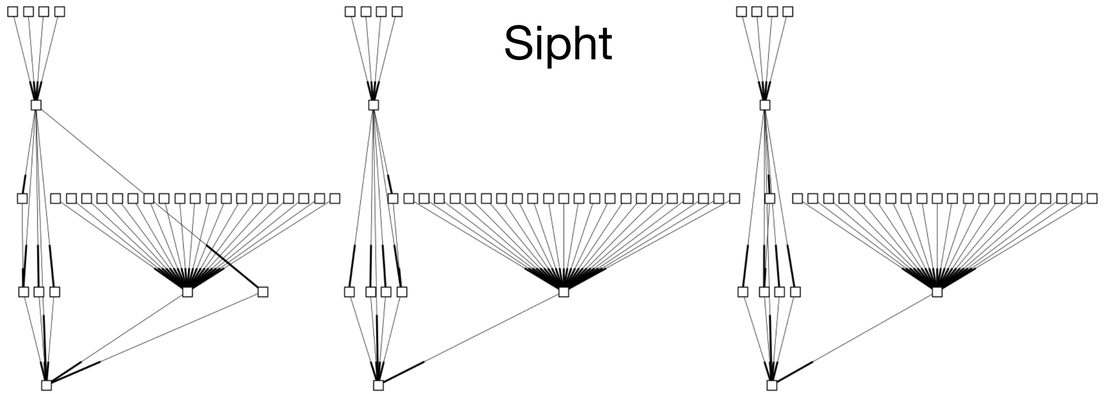
Topology {Montage, Ligo, Sipht, Genome, Cybershake} [JCD+13]

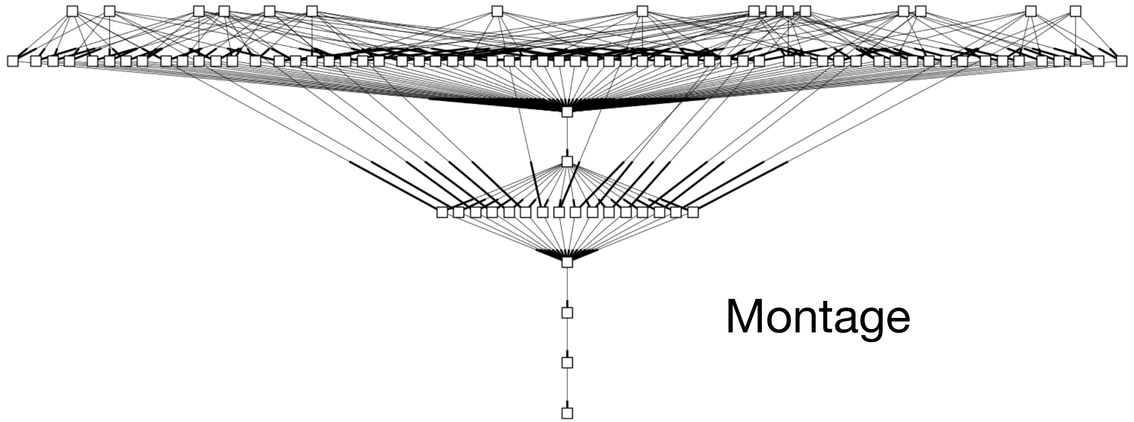
1k tasks per workflow

Resource usage {instance 1…100}

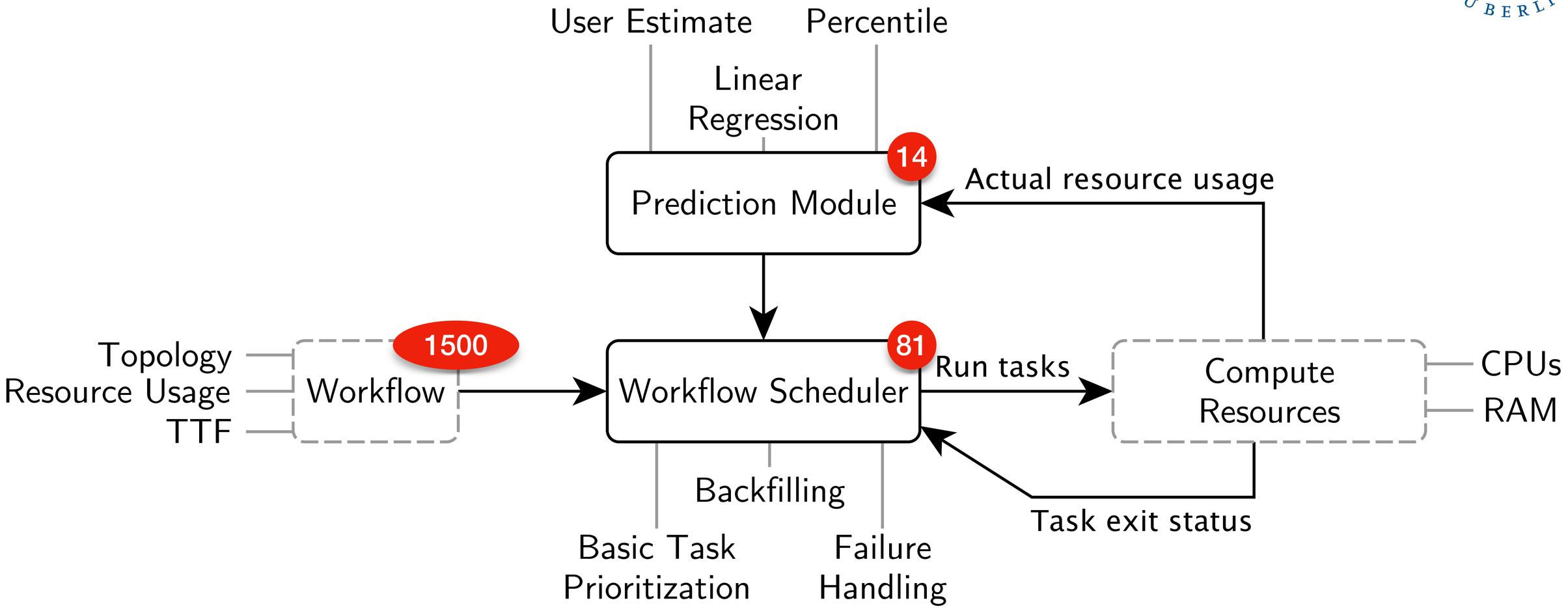
Time to failure {0.1, 0.5, 0.9}

- task execution time modifier



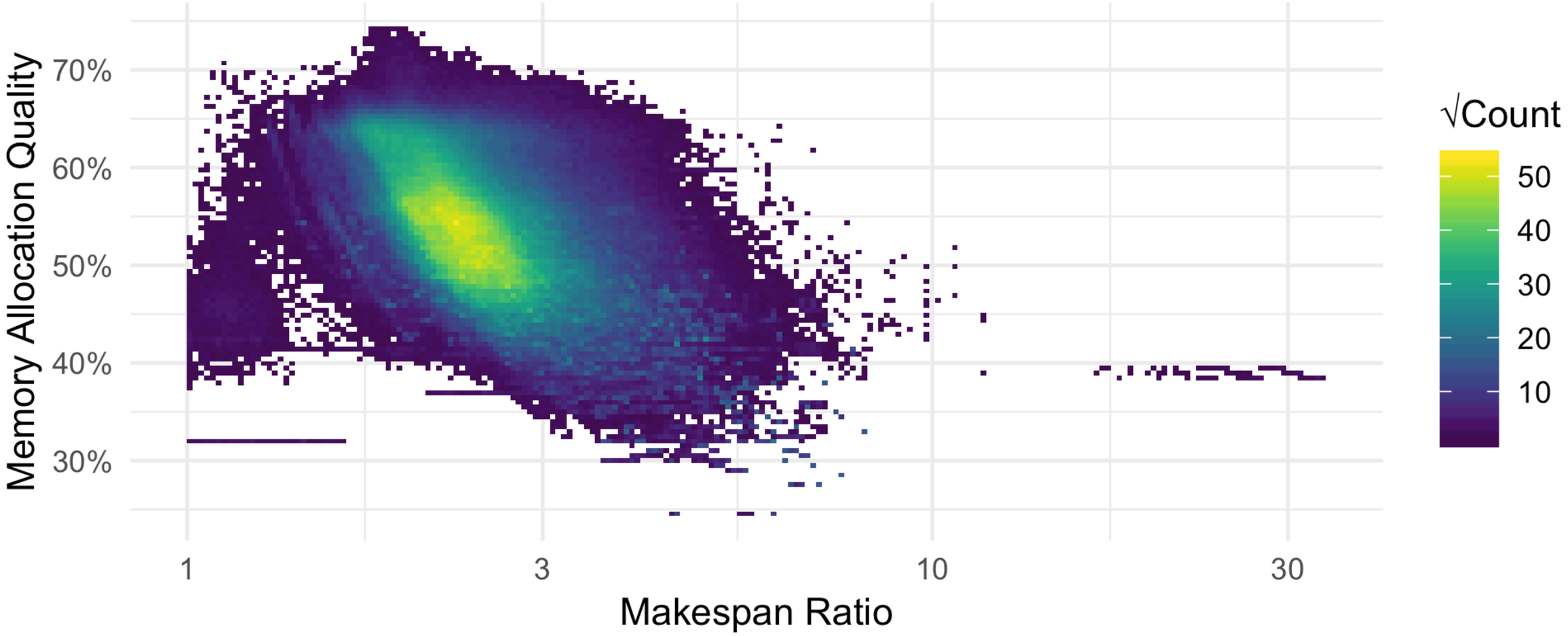


Experimental Setup

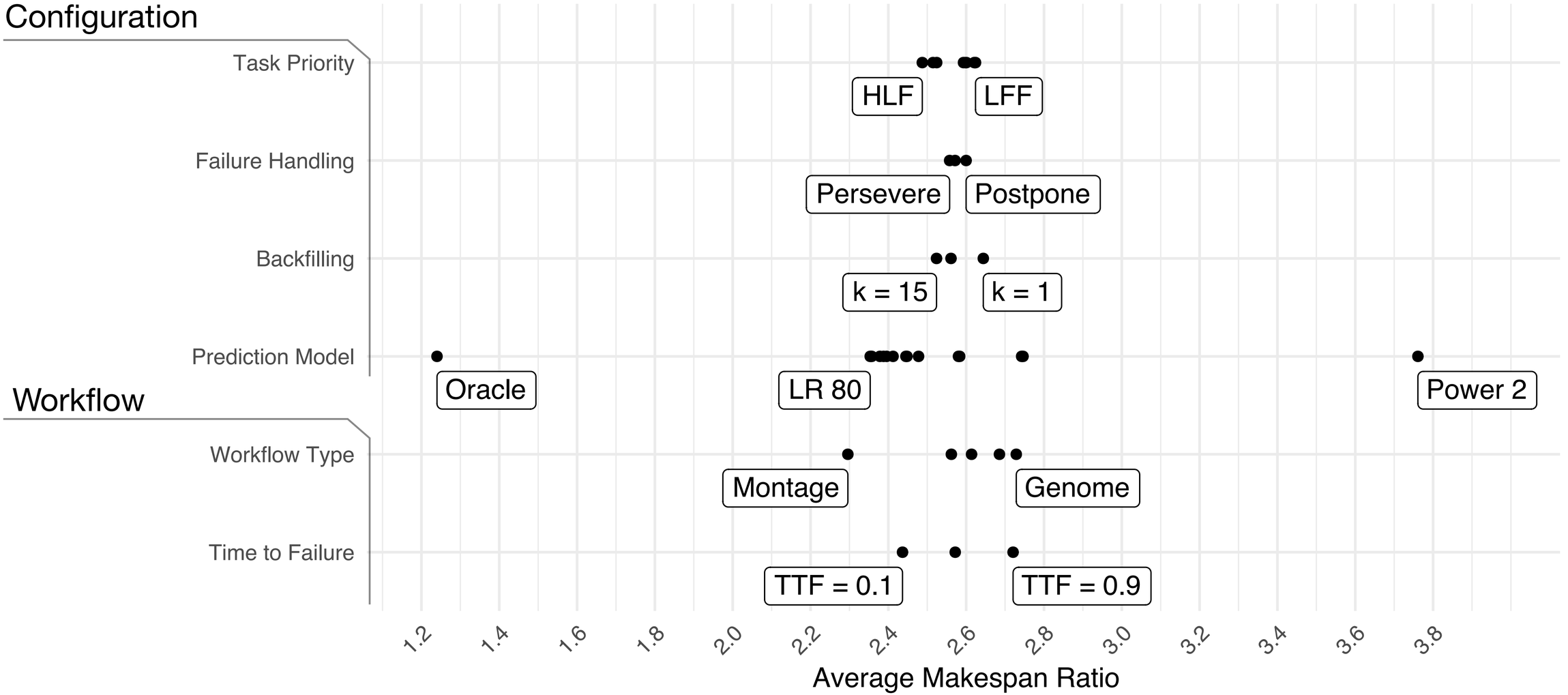


=1.7M simulations (1,500 workflows x 1,134 configurations)

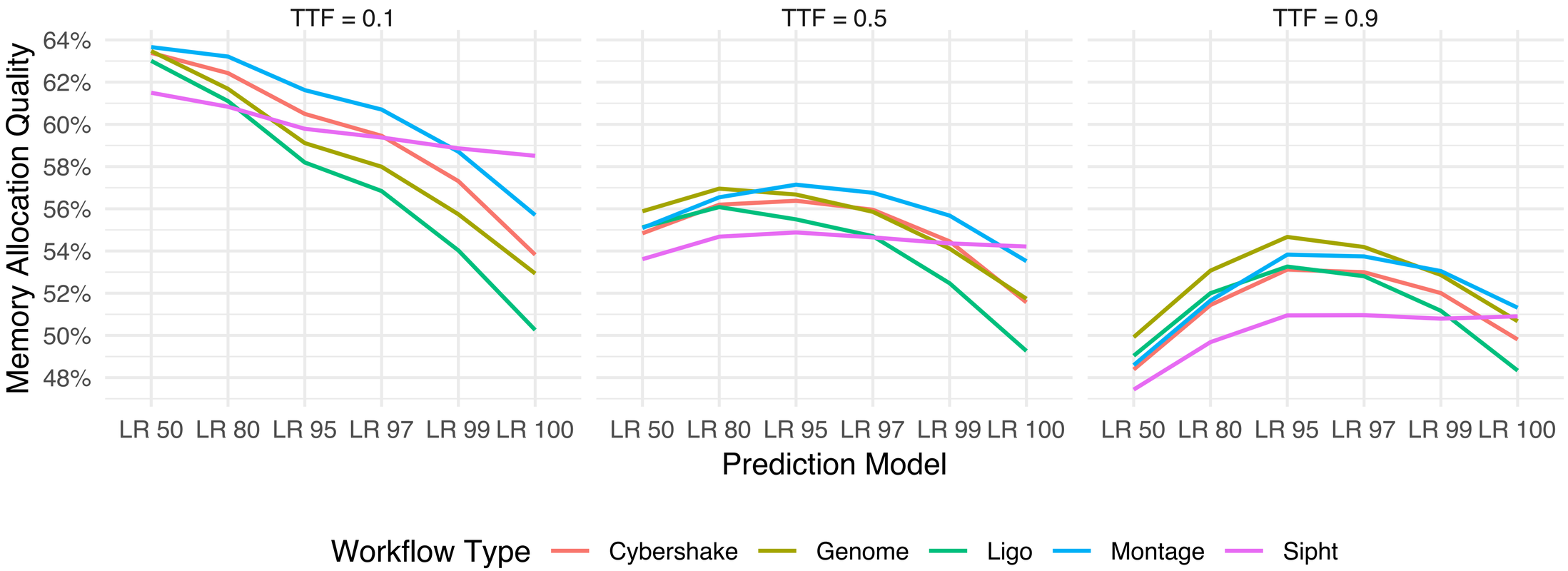
Joint Performance Metric Distribution



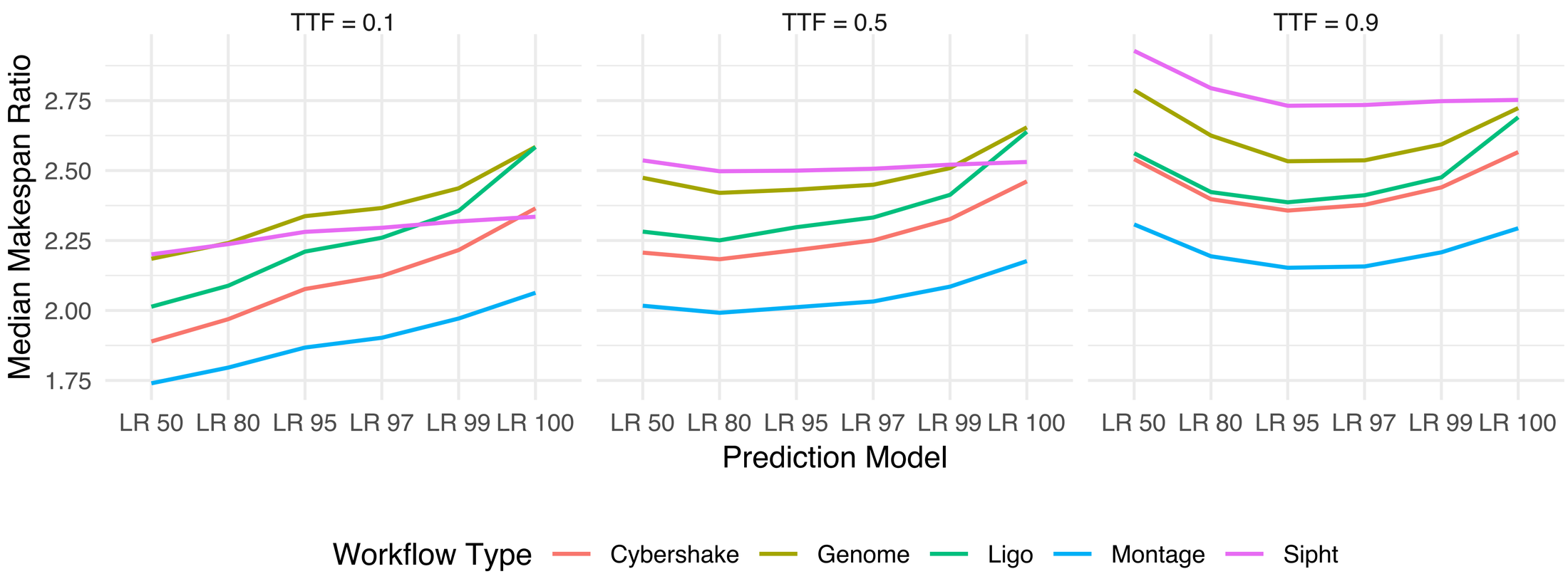
Parameter Impact on Average MSR



Optimal Risk Aversion



Optimal Risk Aversion



Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Thesis Overview

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Low-Wastage Regression for Peak Memory Prediction

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Highlights

[WSL19] **Carl Witt**, Jakob von Santen, and Ulf Leser. “Learning Low-Wastage Memory Allocations for Scientific Workflows at IceCube." *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

Low-Wastage Regression for Peak Memory Prediction

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Highlights

- Custom predictor targeting Memory Allocation Quality

[WSL19] **Carl Witt**, Jakob von Santen, and Ulf Leser. “Learning Low-Wastage Memory Allocations for Scientific Workflows at IceCube." *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

Low-Wastage Regression for Peak Memory Prediction

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Highlights

- Custom predictor targeting Memory Allocation Quality
- Risk aversion considers follow up costs

[WSL19] **Carl Witt**, Jakob von Santen, and Ulf Leser. “Learning Low-Wastage Memory Allocations for Scientific Workflows at IceCube." *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

Low-Wastage Regression for Peak Memory Prediction

Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Highlights

- Custom predictor targeting Memory Allocation Quality
- Risk aversion considers follow up costs
- IceCube evaluation case study

[WSL19] **Carl Witt**, Jakob von Santen, and Ulf Leser. “Learning Low-Wastage Memory Allocations for Scientific Workflows at IceCube." *International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019.

State of the art: Tovar et al. 2018

	Tovar et al. 2018	Witt et al. 2019
Life cycle optimization	✓	✓
Prediction correction	Allocate Fixed Maximum Amount	Exponential Allocation
Model	Constant	Rectified Linear
Task runtimes	N.A.	✓
Time to failure	Optimize for Pessimistic Scenario	Arbitrary

[TSJ+18] Benjamin Tovar, Rafael Ferreira da Silva, Gideon Juve, Ewa Deelman, William Allcock, Douglas Thain, and Miron Livny, “A Job Sizing Strategy for High-Throughput Scientific Workflows.,” *IEEE Trans. Parallel Distrib. Syst.*, 2018.

Problem

Training Data

- Task run time
- Time to failure
- Input size
- Peak memory usage

Objective

- Maximize MAQ und exponential allocation

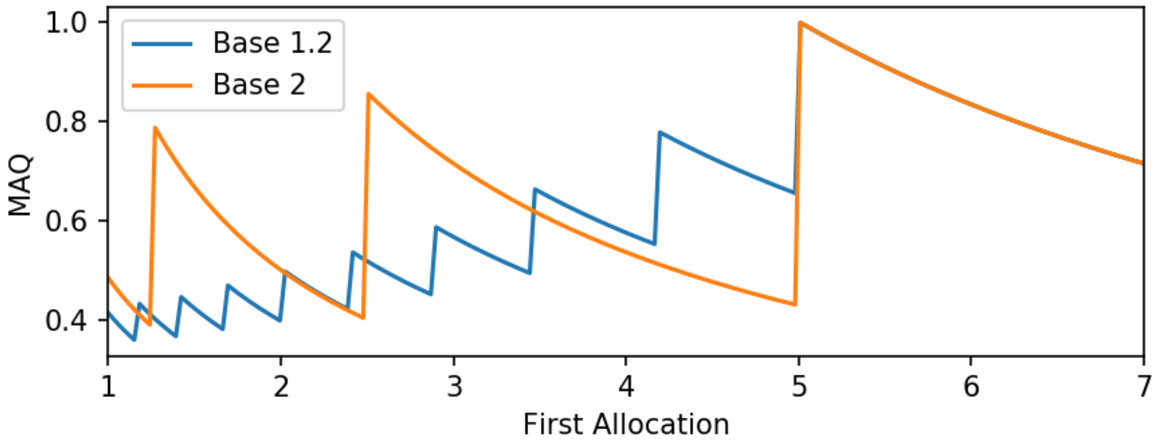
Problem

Training Data

- Task run time
- Time to failure
- Input size
- Peak memory usage

Objective

- Maximize MAQ und exponential allocation



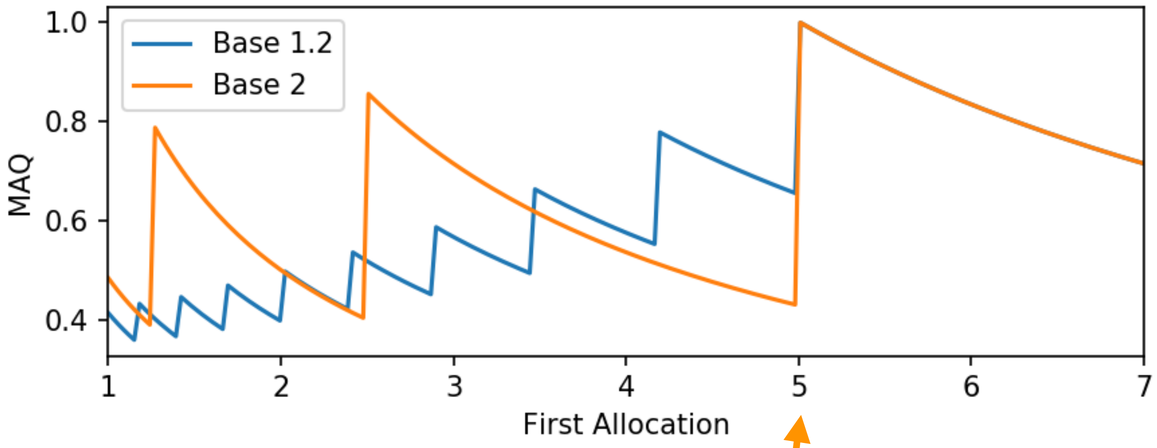
Problem

Training Data

- Task run time
- Time to failure
- Input size
- Peak memory usage

Objective

- Maximize MAQ und exponential allocation



Actual Peak Memory Usage

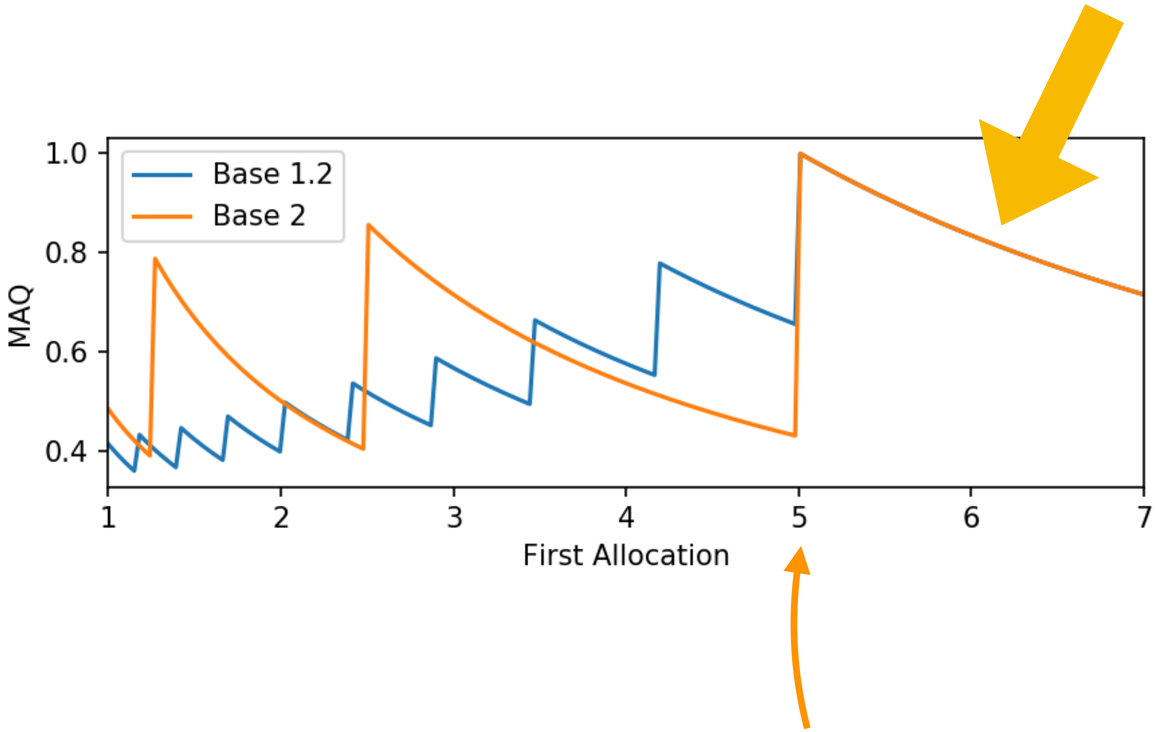
Problem

Training Data

- Task run time
- Time to failure
- Input size
- Peak memory usage

Objective

- Maximize MAQ und exponential allocation



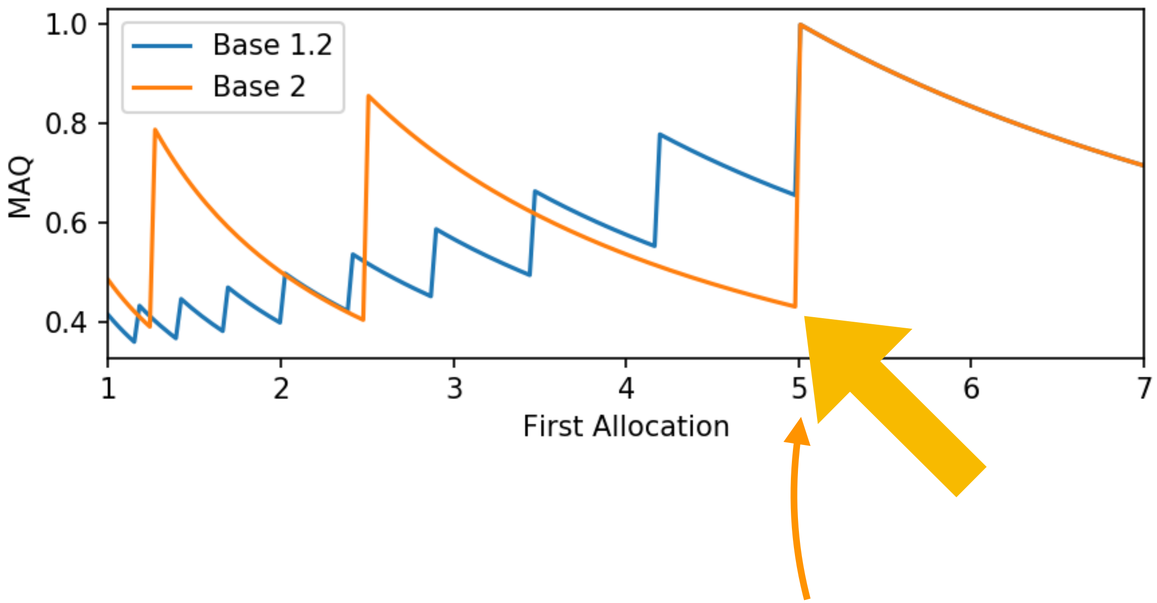
Problem

Training Data

- Task run time
- Time to failure
- Input size
- Peak memory usage

Objective

- Maximize MAQ und exponential allocation



Actual Peak Memory Usage

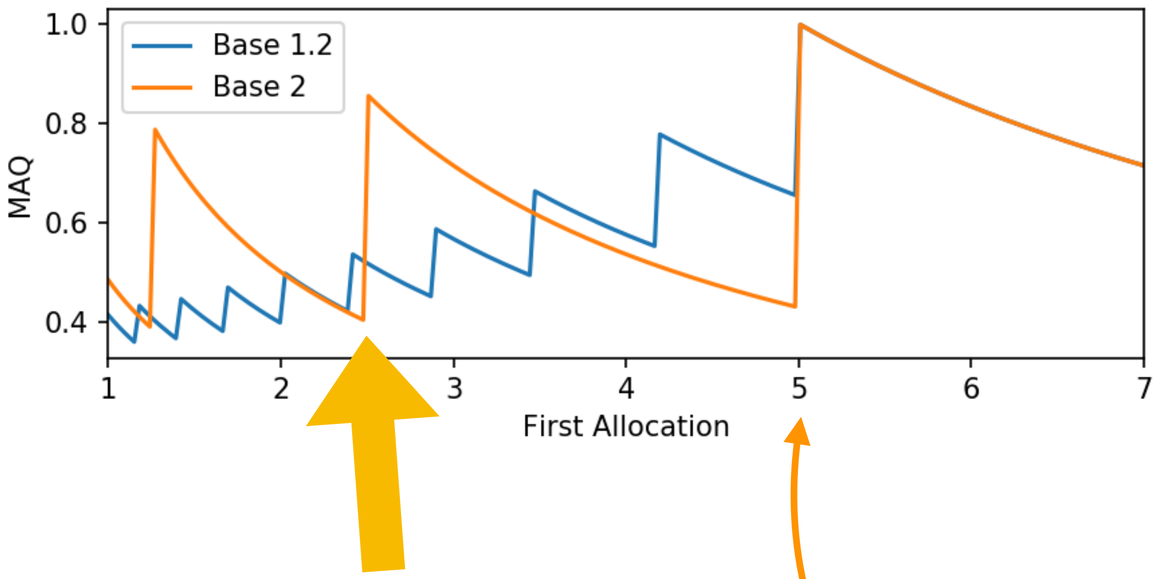
Problem

Training Data

- Task run time
- Time to failure
- Input size
- Peak memory usage

Objective

- Maximize MAQ und exponential allocation



Actual Peak Memory Usage

Approach

- Minimize wastage over all attempts
- First allocation: rectified linear function of task input size
- Wastage resulting from model parameters
- Constrained optimization

Approach

- Minimize wastage over all attempts

$$k_i = 1 + \max\left(0, \left\lceil \log_b \frac{r_i}{a_{i1}} \right\rceil\right)$$

- First allocation: rectified linear function of task input size

- Wastage resulting from model parameters

- Constrained optimization

Approach

- Minimize wastage over all attempts

$$k_i = 1 + \max\left(0, \left\lceil \log_b \frac{r_i}{a_{i1}} \right\rceil\right)$$

- First allocation: rectified linear function of task input size

$$a_{i1} = \max(\theta_1 x_i + \theta_0, a_l)$$

- Wastage resulting from model parameters

- Constrained optimization

Approach

- Minimize wastage over all attempts

$$k_i = 1 + \max\left(0, \left\lceil \log_b \frac{r_i}{a_{i1}} \right\rceil\right)$$

- First allocation: rectified linear function of task input size

$$a_{i1} = \max(\theta_1 x_i + \theta_0, a_l)$$

- Wastage resulting from model parameters

$$W(D, \theta, b) = \sum_{i=1}^{n} \underbrace{\left(a_{i1} b^{k_i - 1} - r_i\right) \tau_i}_{\text{oversizing}} + \underbrace{a_{i1} \frac{b^{k_i - 1} - 1}{b - 1} \tau_i^*}_{\text{undersizing}}$$

- Constrained optimization

Approach

- Minimize wastage over all attempts

$$k_i = 1 + \max\left(0, \left\lceil \log_b \frac{r_i}{a_{i1}} \right\rceil\right)$$

- First allocation: rectified linear function of task input size

$$a_{i1} = \max(\theta_1 x_i + \theta_0, a_l)$$

- Wastage resulting from model parameters

$$W(D, \theta, b) = \sum_{i=1}^{n} \underbrace{\left(a_{i1} b^{k_i - 1} - r_i\right) \tau_i}_{\text{oversizing}} + \underbrace{a_{i1} \frac{b^{k_i - 1} - 1}{b - 1} \tau_i^*}_{\text{undersizing}}$$

- Constrained optimization

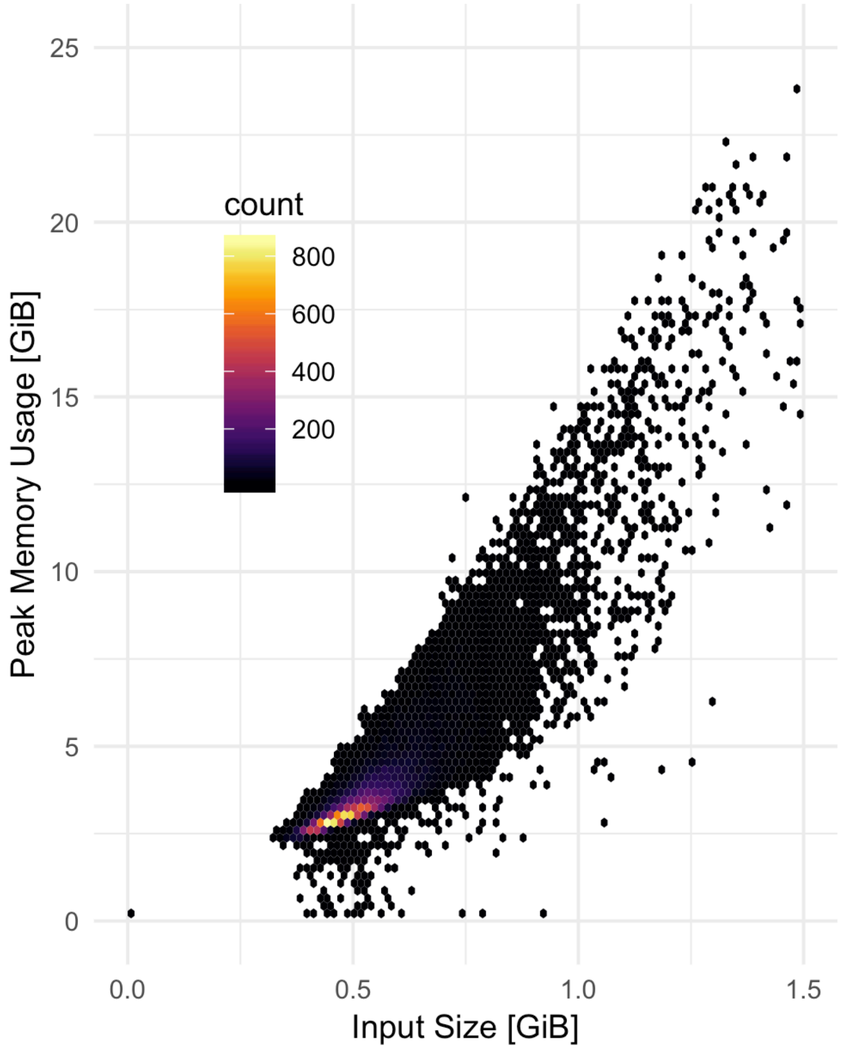
$$\underset{\theta \in \mathbb{R}^2, b \in \mathbb{R}}{\text{minimize}}\ W(D, \theta, b) \text{ s.t. } b > 1$$

Optimizer

- Initial solution: Quantile Regression
- First allocations account for run times
- Quantile regression does not
- Quantiles from median to 99th

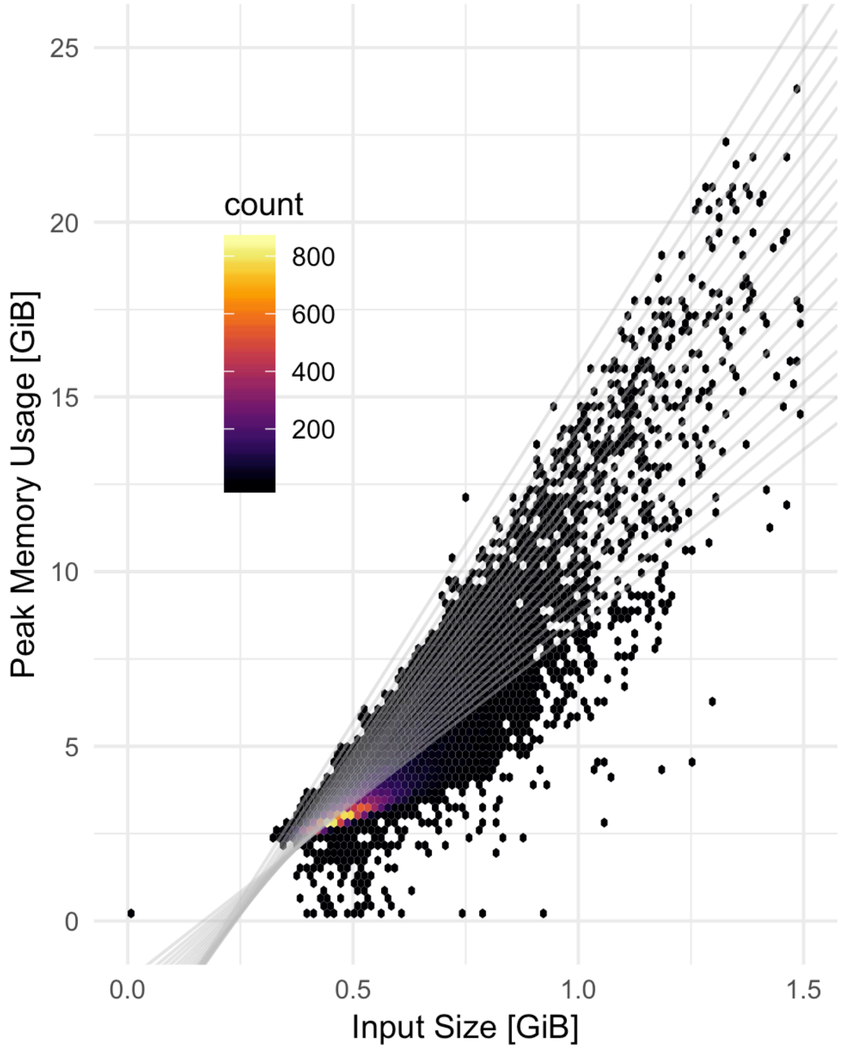
Optimizer

- Initial solution: Quantile Regression
- First allocations account for run times
- Quantile regression does not
- Quantiles from median to 99th

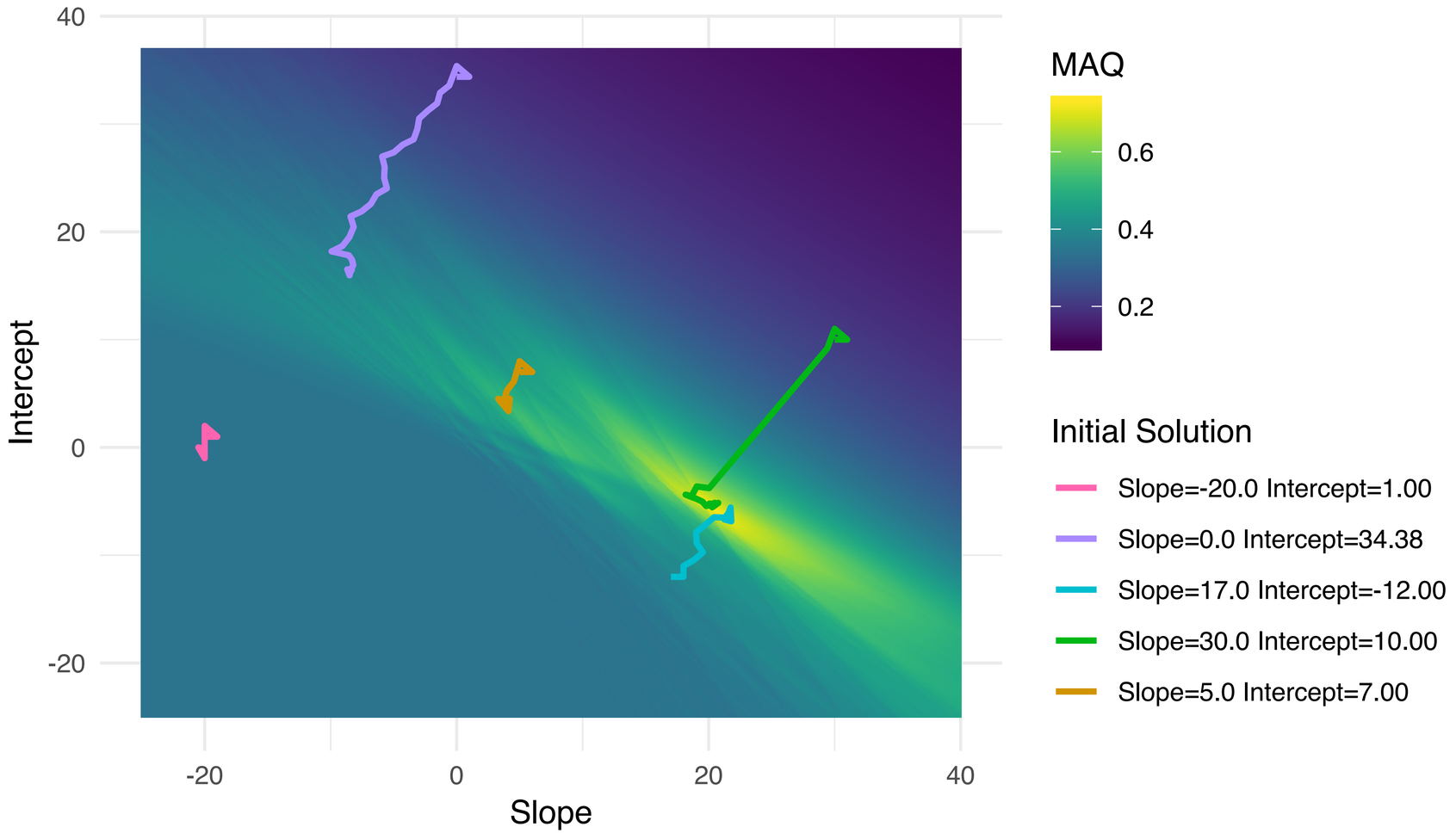


Optimizer

- Initial solution: Quantile Regression
- First allocations account for run times
- Quantile regression does not
- Quantiles from median to 99th



Constrained Optimization by Linear Approximation



Evaluation Setup

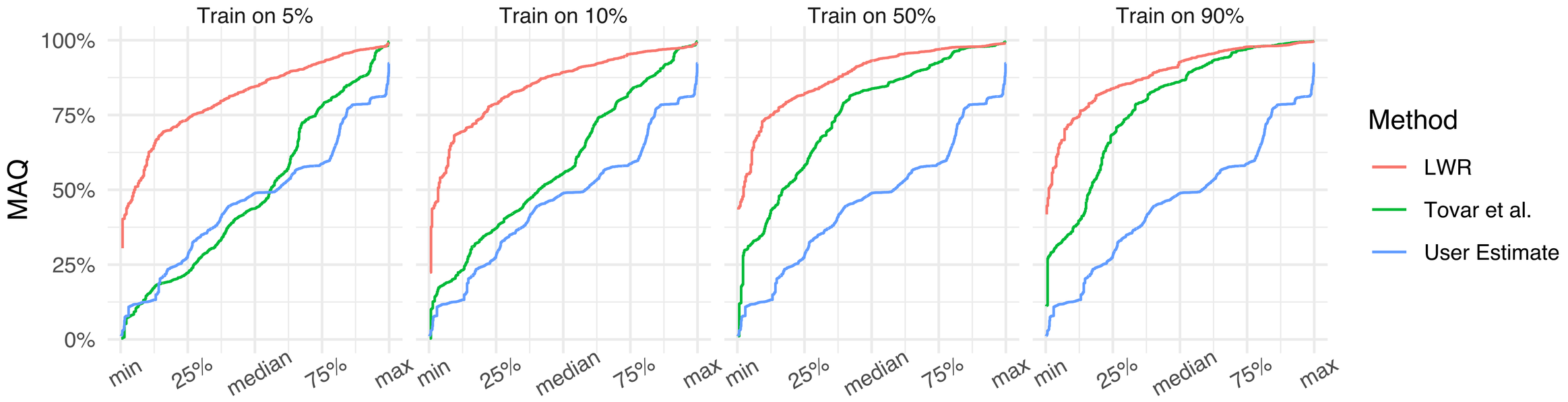
IceCube production logs: 750k tasks

- Task run time
- Time to failure
- Input file size
- Peak memory usage
- User estimates

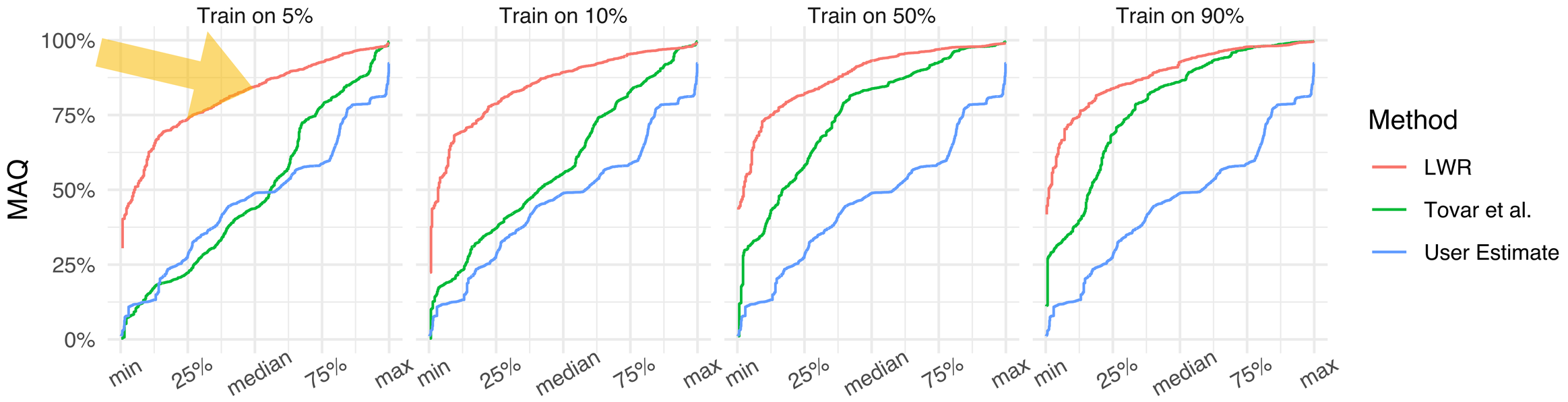
Grouped in 142 abstract tasks

- first k% for training
- rest for validation
- ordered by task finish times

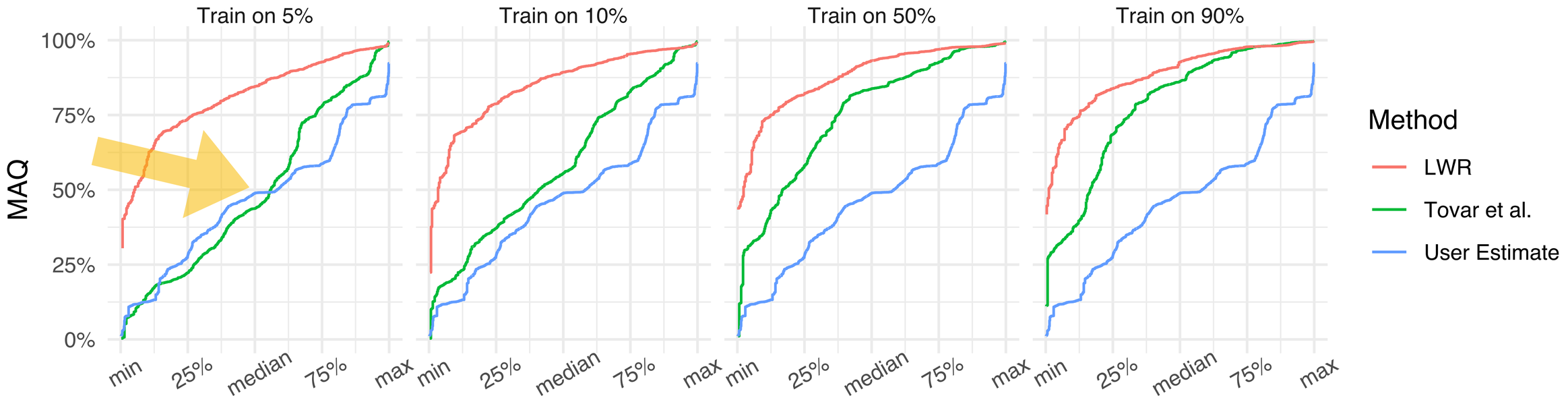
Results


Train on 5%
Train on 10%
Train on 50%
Train on 90%
MAQ
100%
75%
50%
25%
0%
min
25%
median
75%
max
Method
LWR
Tovar et al.
User Estimate

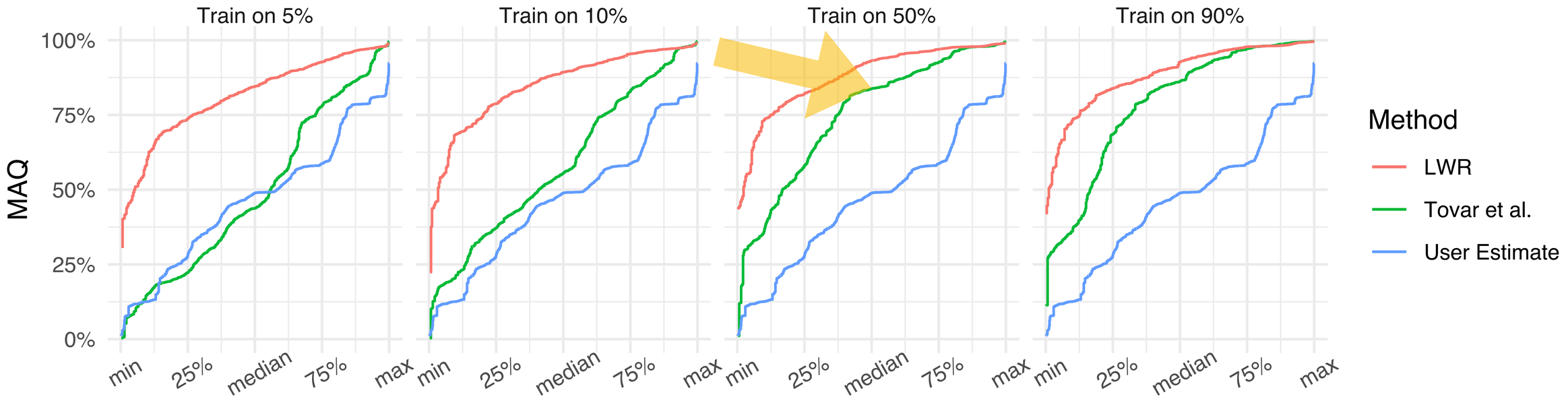
Results


Train on 5%
Train on 10%
Train on 50%
Train on 90%
MAQ
100%
75%
50%
25%
0%
min
25%
median
75%
max
Method
LWR
Tovar et al.
User Estimate

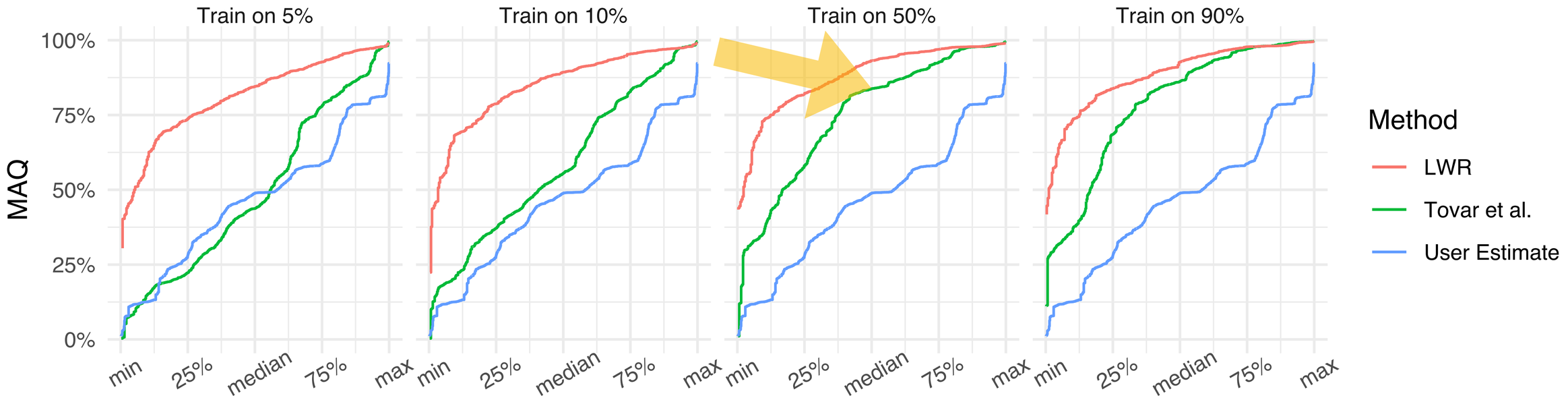
Results


Train on 5%
Train on 10%
Train on 50%
Train on 90%
MAQ
100%
75%
50%
25%
0%
min
25%
median
75%
max
Method
LWR
Tovar et al.
User Estimate

Results


Train on 5%
Train on 10%
Train on 50%
Train on 90%
MAQ
100%
75%
50%
25%
0%
min
25%
median
75%
max
Method
LWR
Tovar et al.
User Estimate

Results



Effective Memory Allocation Quality: 71.2%

- wastage on test set + increased wastage during training
- up to 40% throughput gain

Discussion

- Optimizing Memory Allocation Quality vs. Makespan Ratio
- Determines degree of risk aversion based on task resource usage distribution
- Context-sensitive predictions?
  - Workflow topology
  - Current load

Thank you.

References

[WSL19] **Carl Witt**, Jakob van Santen, Ulf Leser: "Learning Low-Wastage Memory Allocations for Scientific Workflows at IceCube." IEEE International Conference on High Performance Computing & Simulation (2019)

[WWL19] **Carl Witt**, Dennis Wagner, Ulf Leser: "Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows." IEEE International Conference on High Performance Computing & Simulation (2019)

[WBGL19] **Carl Witt**, Marc Bux, Wladislaw Gusew, Ulf Leser: “Predictive performance modeling for distributed batch processing using black box monitoring and machine learning.” Inf. Syst. 82: 33-52 (2019)

[WWL18a] **Carl Witt**, Sam Wheating, Ulf Leser: "LOS: Level Order Sampling for Task Graph Scheduling on Heterogeneous Resources." Workflows in Support of Large-Scale Science (WORKS): 20-30 (2018)

[WWL18b] **Carl Witt**, Dennis Wagner, Ulf Leser: "POS: Online Learning for Memory-Aware Scheduling of Scientific Workflows." IEEE 14th International Conference on e-Science: 399-400 (2018)

[BBW17] Marc Bux, Jörgen Brandt, **Carl Witt**, Jim Dowling, Ulf Leser: "Hi-WAY: Execution of Scientific Workflows on Hadoop YARN." EDBT: 668-679 (2017)

[JCD+13] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi, "Characterizing and profiling scientific workflows." *FGCS*, vol. 29, no. 3, pp. 682–692, 2013.

Thesis Overview

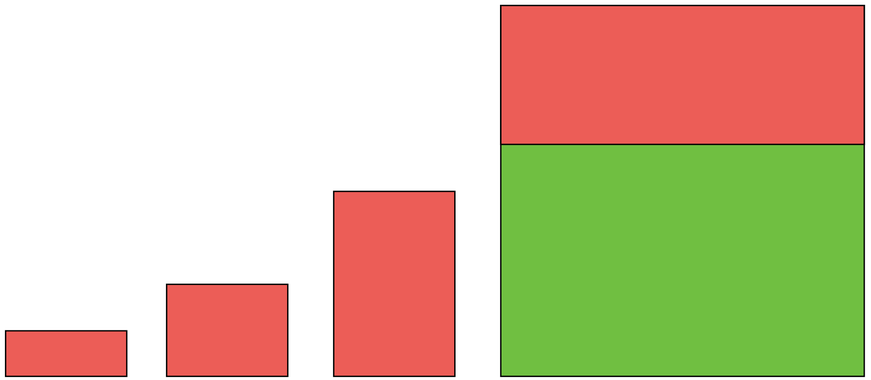
Contribution	References	Execution Environment	Scheduling	Prediction Model	Evaluation Workflows
Chapter 3 Level-Order Sampling for Static Task Graph Scheduling	[WWL18a]	Connected Processors	Static	Oracle	Random
Chapter 4 Feedback-Based Scheduling of Scientific Workflows	[WWL18b, WWL19]	Batch Scheduler	Dynamic	Simple Online	Synthetic
Chapter 5 Low-Wastage Regression for Peak Memory Prediction	[WSL19]	Batch Scheduler	N.A.	Asymmetric Costs	Real-World

Backup Slides

Contributions

Assumption	Chapter 3	Chapter 4	Chapter 5
Execution environment	dedicated	batch scheduler	batch scheduler
Modeled resources	processors	processors, memory	memory
Prediction model	oracle	off-the-shelf, online updates	custom, online updates
User estimates	*n.a.*	hypothetical	real
Scheduling	static	dynamic	*n.a.*
Evaluation criteria	makespan	makespan, utilization	utilization
Workflows	random	synthetic	real

Memory Allocation Quality



Attempt	Outcome	Allocation	Usage	Wastage
1	Fail	1 GBh	0 GBh	1 GBh
2	Fail	2 GBh	0 GBh	2 GBh
3	Fail	4 GBh	0 GBh	4 GBh
4	Success	8 GBh	5 GBh	3 GBh
		15 GBh	5 GBh	10 GBh

$$\mathrm{MAQ} = \frac{\text{used allocation}}{\text{used allocation} + \text{wasted allocation}}$$

Research Overview

Scheduling with predicted memory usage

- Evaluation of scheduling strategies
- Interactions with different prediction models

Static Task Graph Scheduling

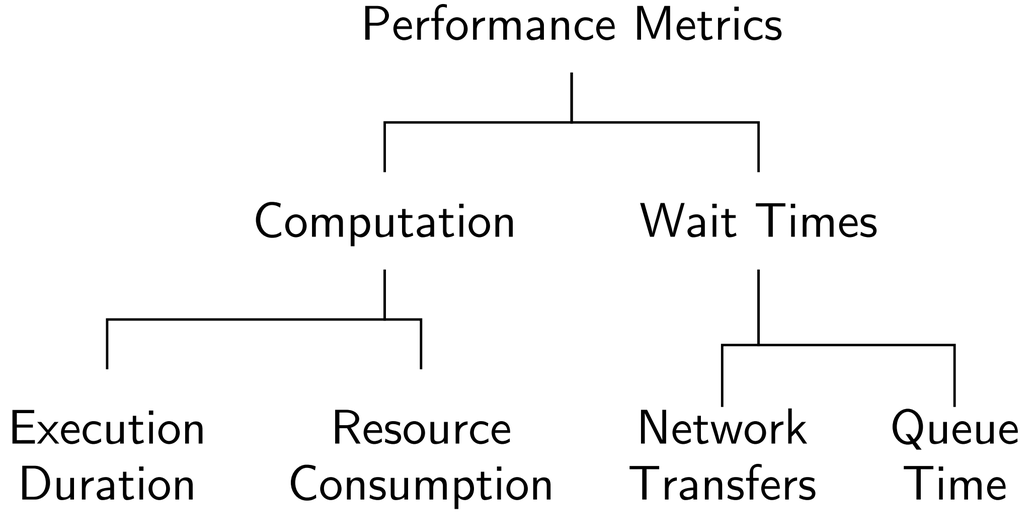
- assigning tasks with known run times to processors
- selective randomization in areas with high expected improvement potential

Adversarial Workflow Evolution

- evolutionary generate difficult-to-schedule workflows

Related Work: Resource Usage Prediction

- Predictive performance modeling



- Statistical and ML techniques
- Black-box, i.e., non-invasive

Information Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Predictive performance modeling for distributed batch processing using black box monitoring and machine learning

Carl Witt , Marc Bux, Wladislaw Gusew, Ulf Leser

Humboldt-Universität zu Berlin, 6, Unter den Linden, Berlin, Germany

h i g h l i g h t s

- Comprehensive review of performance prediction in distributed computing.
- Approaches for runtime, resource consumption, and wait time prediction.
- Focus on data-driven methods based on non intrusive black box monitoring.

a r t i c l e i n f o

Article history:
Received 28 March 2018
Received in revised form 28 December 2018
Accepted 12 January 2019
Available online xxxx
Recommended by Khuzaima Daudjee

Keywords:
Distributed computing
Resource management
Machine learning
Black box monitoring

a b s t r a c t

In many domains, the previous decade was characterized by increasing data volumes and growing complexity of data analyses, creating new demands for batch processing on distributed systems. Effective operation of these systems is challenging when facing uncertainties about the performance of jobs and tasks under varying resource configurations, e. g., for scheduling and resource allocation. We survey predictive performance modeling (PPM) approaches to estimate performance metrics such as execution duration, required memory or wait times of future jobs and tasks based on past performance observations. We focus on non-intrusive methods, i. e., methods that can be applied to any workload without modification, since the workload is usually a black box from the perspective of the systems managing the computational infrastructure. We classify and compare sources of performance variation, predicted performance metrics, limitations and challenges, required training data, use cases, and the underlying prediction techniques. We conclude by identifying several open problems and pressing research needs in the field.

© 2019 Elsevier Ltd. All rights reserved.

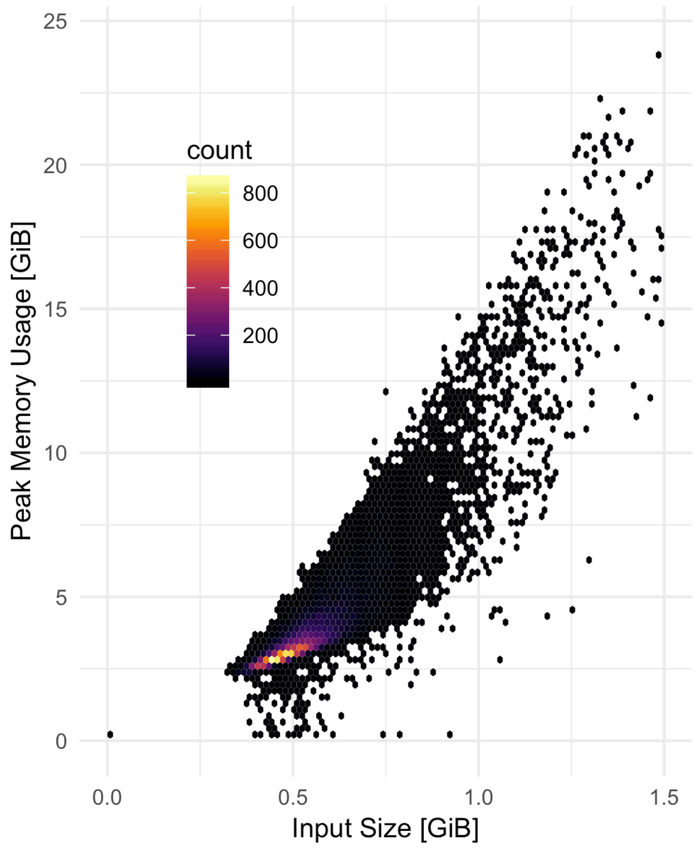
Contents

Corresponding author.
E-mail addresses: wittcarl@informatik.hu-berlin.de (C. Witt), buxmarcn@informatik.hu-berlin.de (M. Bux), gusewwly@informatik.hu-berlin.de (W. Gusew), leser@informatik.hu-berlin.de (U. Leser).

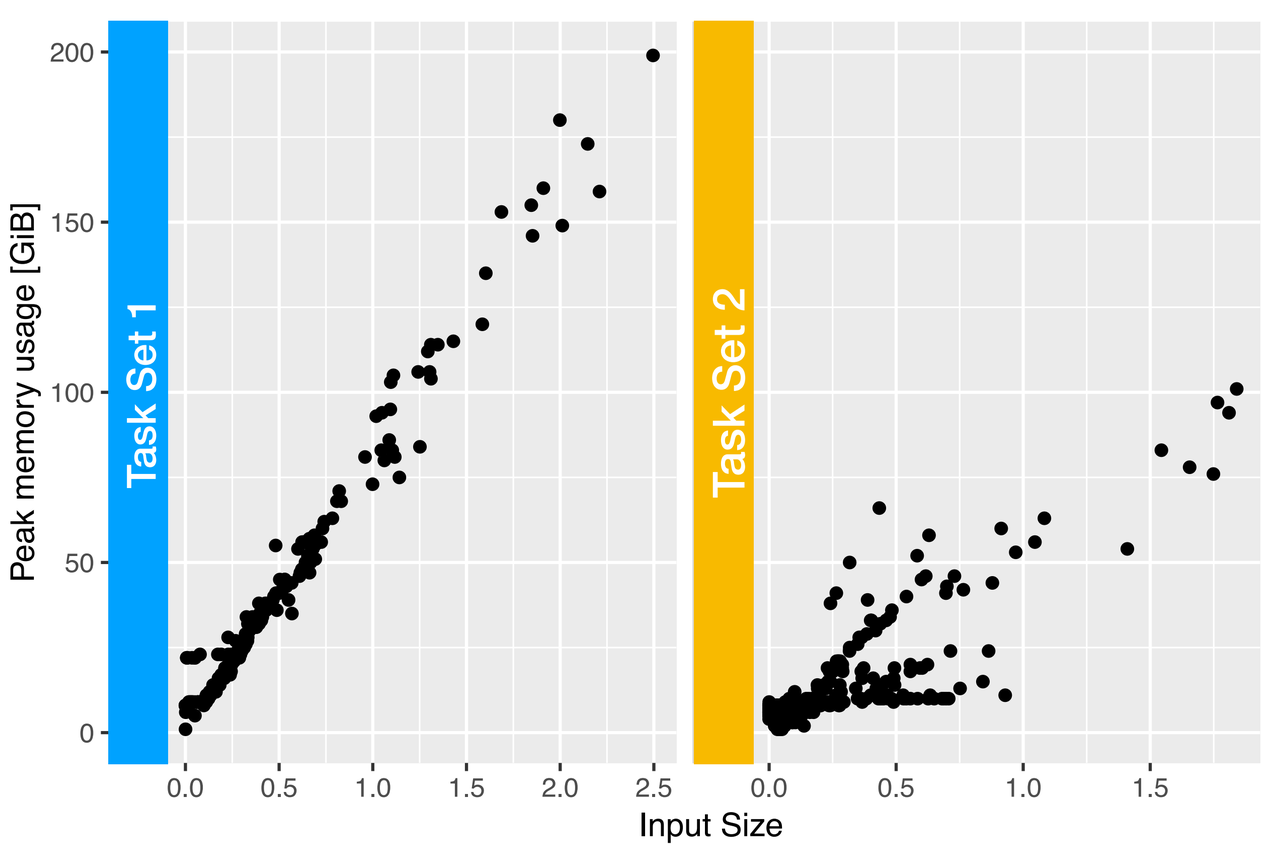
https://doi.org/10.1016/j.is.2019.01.006
0306-4379/

Carl Witt, Marc Bux, Wladislaw Gusew, and Ulf Leser. “Predictive Performance Modeling for Distributed Batch Processing using Black-Box Monitoring and Machine Learning." *Information Systems, Volume 82, pp. 33-52, 2019*

Input Size as Predictor

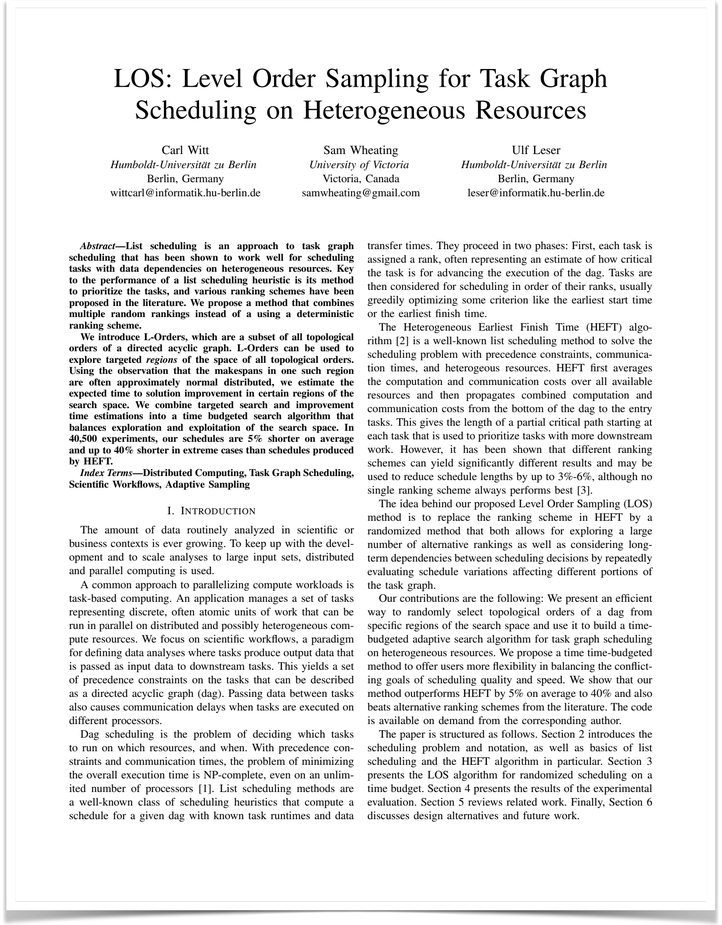


EMBL-EBI



LOS: Level-Order Sampling for Task Graph Scheduling on Heterogeneous Resources

- Method for static scheduling
- Randomized search
- Adaptive exploration of search space



LOS: Level Order Sampling for Task Graph Scheduling on Heterogeneous Resources

Carl Witt
Humboldt-Universität zu Berlin
Berlin, Germany
wittcarl@informatik.hu-berlin.de

Sam Wheating
University of Victoria
Victoria, Canada
samwheating@gmail.com

Ulf Leser
Humboldt-Universität zu Berlin
Berlin, Germany
leser@informatik.hu-berlin.de

***Abstract*—List scheduling is an approach to task graph scheduling that has been shown to work well for scheduling tasks with data dependencies on heterogeneous resources. Key to the performance of a list scheduling heuristic is its method to prioritize the tasks, and various ranking schemes have been proposed in the literature. We propose a method that combines multiple random rankings instead of a using a deterministic ranking scheme.**

We introduce L-Orders, which are a subset of all topological orders of a directed acyclic graph. L-Orders can be used to explore targeted *regions* of the space of all topological orders. Using the observation that the makespans in one such region are often approximately normal distributed, we estimate the expected time to solution improvement in certain regions of the search space. We combine targeted search and improvement time estimations into a time budgeted search algorithm that balances exploration and exploitation of the search space. In 40,500 experiments, our schedules are 5% shorter on average and up to 40% shorter in extreme cases than schedules produced by HEFT.

***Index Terms*—Distributed Computing, Task Graph Scheduling, Scientific Workflows, Adaptive Sampling**

I. Introduction

The amount of data routinely analyzed in scientific or business contexts is ever growing. To keep up with the development and to scale analyses to large input sets, distributed and parallel computing is used.

A common approach to parallelizing compute workloads is task-based computing. An application manages a set of tasks representing discrete, often atomic units of work that can be run in parallel on distributed and possibly heterogeneous compute resources. We focus on scientific workflows, a paradigm for defining data analyses where tasks produce output data that is passed as input data to downstream tasks. This yields a set of precedence constraints on the tasks that can be described as a directed acyclic graph (dag). Passing data between tasks also causes communication delays when tasks are executed on different processors.

Dag scheduling is the problem of deciding which tasks to run on which resources, and when. With precedence constraints and communication times, the problem of minimizing the overall execution time is NP-complete, even on an unlimited number of processors [1]. List scheduling methods are a well-known class of scheduling heuristics that compute a schedule for a given dag with known task runtimes and data transfer times. They proceed in two phases: First, each task is assigned a rank, often representing an estimate of how critical the task is for advancing the execution of the dag. Tasks are then considered for scheduling in order of their ranks, usually greedily optimizing some criterion like the earliest start time or the earliest finish time.

The Heterogeneous Earliest Finish Time (HEFT) algorithm [2] is a well-known list scheduling method to solve the scheduling problem with precedence constraints, communication times, and heterogeous resources. HEFT first averages the computation and communication costs over all available resources and then propagates combined computation and communication costs from the bottom of the dag to the entry tasks. This gives the length of a partial critical path starting at each task that is used to prioritize tasks with more downstream work. However, it has been shown that different ranking schemes can yield significantly different results and may be used to reduce schedule lengths by up to 3%-6%, although no single ranking scheme always performs best [3].

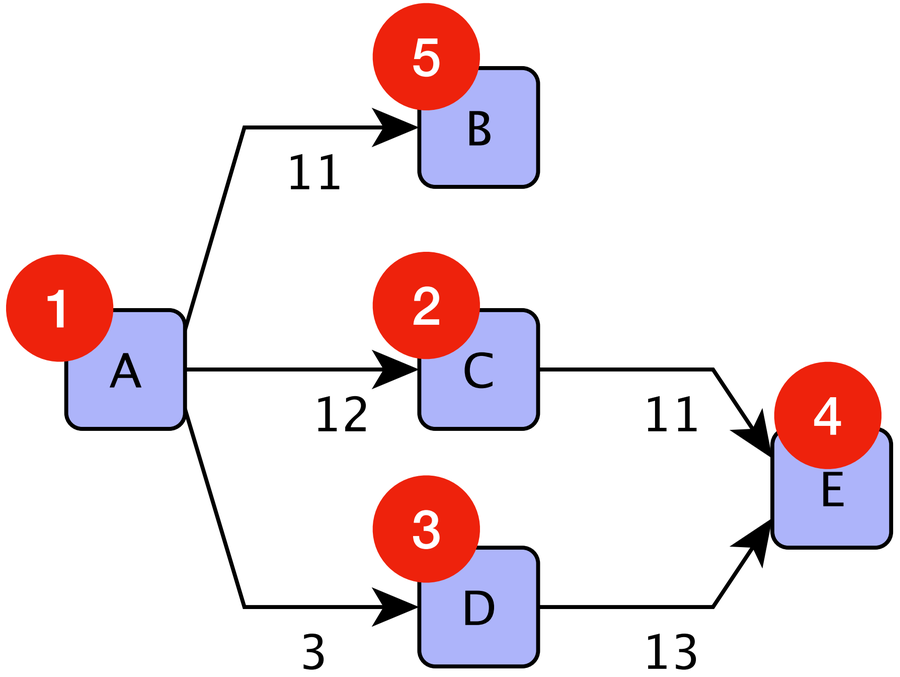
The idea behind our proposed Level Order Sampling (LOS) method is to replace the ranking scheme in HEFT by a randomized method that both allows for exploring a large number of alternative rankings as well as considering long-term dependencies between scheduling decisions by repeatedly evaluating schedule variations affecting different portions of the task graph.

Our contributions are the following: We present an efficient way to randomly select topological orders of a dag from specific regions of the search space and use it to build a time-budgeted adaptive search algorithm for task graph scheduling on heterogeneous resources. We propose a time time-budgeted method to offer users more flexibility in balancing the conflicting goals of scheduling quality and speed. We show that our method outperforms HEFT by 5% on average to 40% and also beats alternative ranking schemes from the literature. The code is available on demand from the corresponding author.

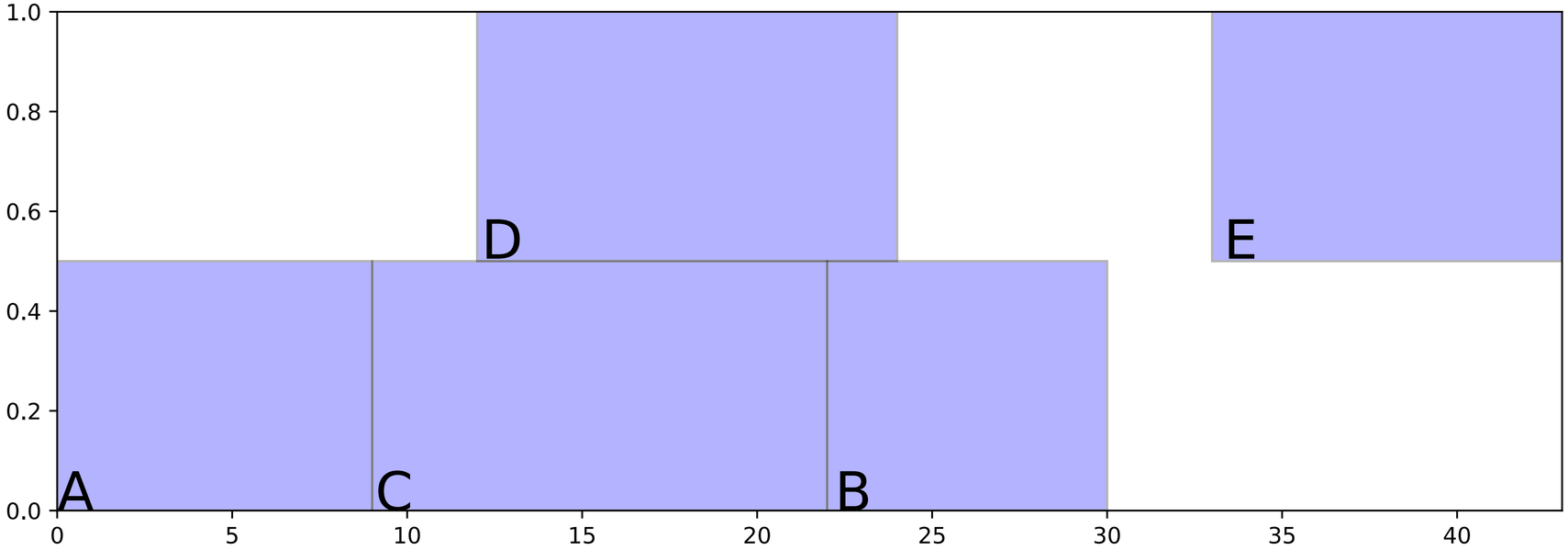
The paper is structured as follows. Section 2 introduces the scheduling problem and notation, as well as basics of list scheduling and the HEFT algorithm in particular. Section 3 presents the LOS algorithm for randomized scheduling on a time budget. Section 4 presents the results of the experimental evaluation. Section 5 reviews related work. Finally, Section 6 discusses design alternatives and future work.

Carl Witt, Sam Wheating, and Ulf Leser. "LOS: Level Order Sampling for Task Graph Scheduling on Heterogeneous Resources." *2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE, 2018.

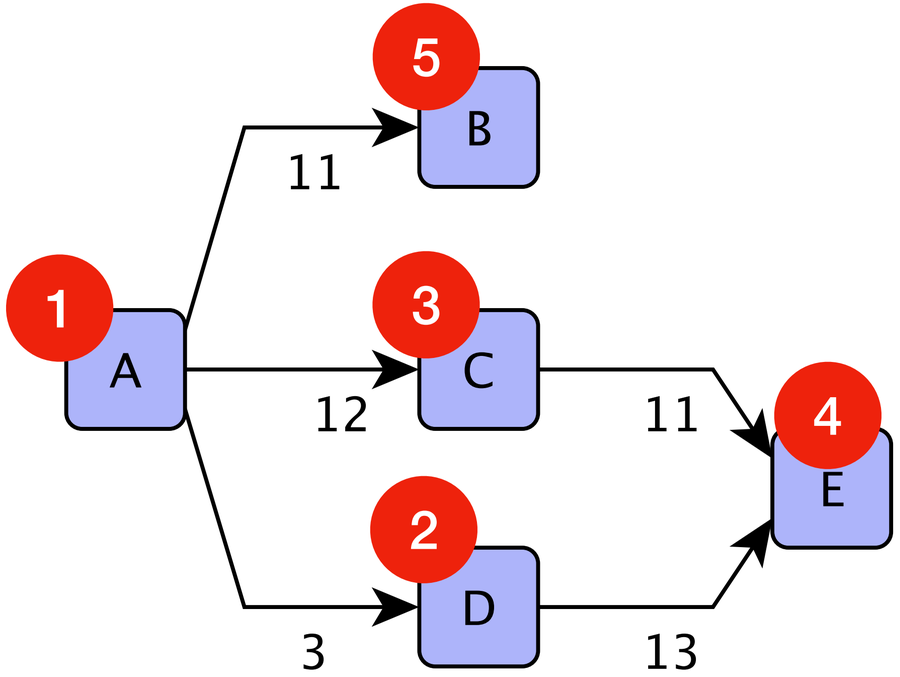
Current Ranking



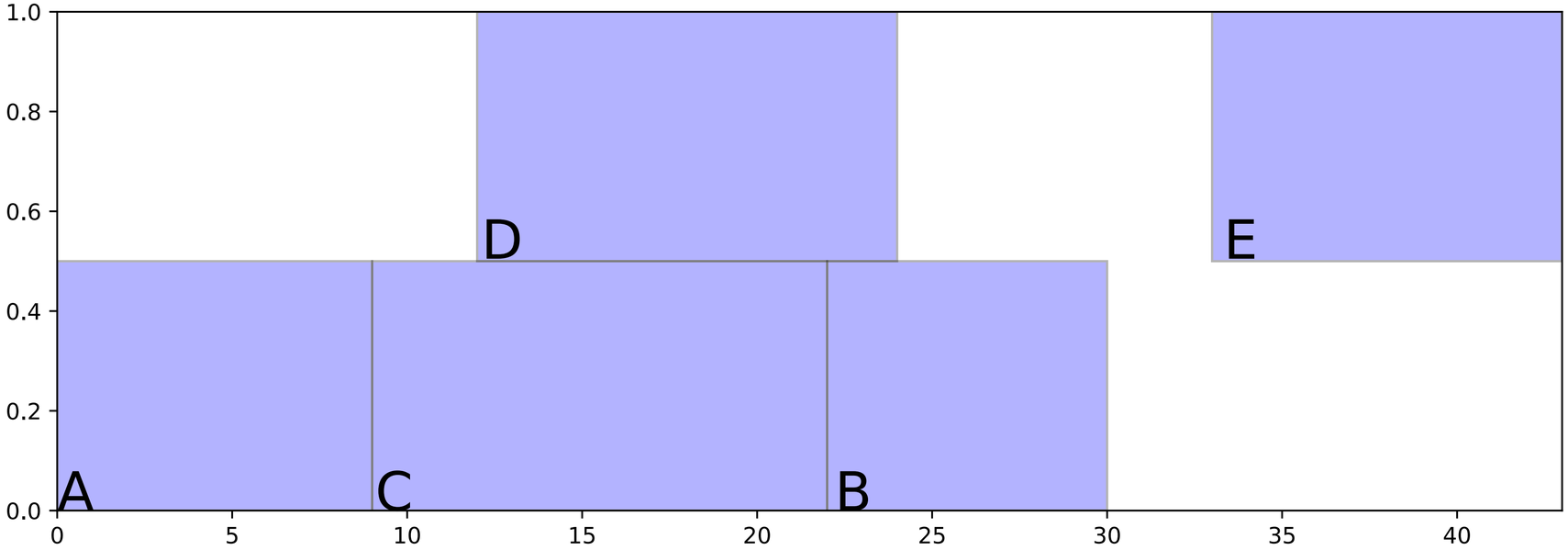
Schedule



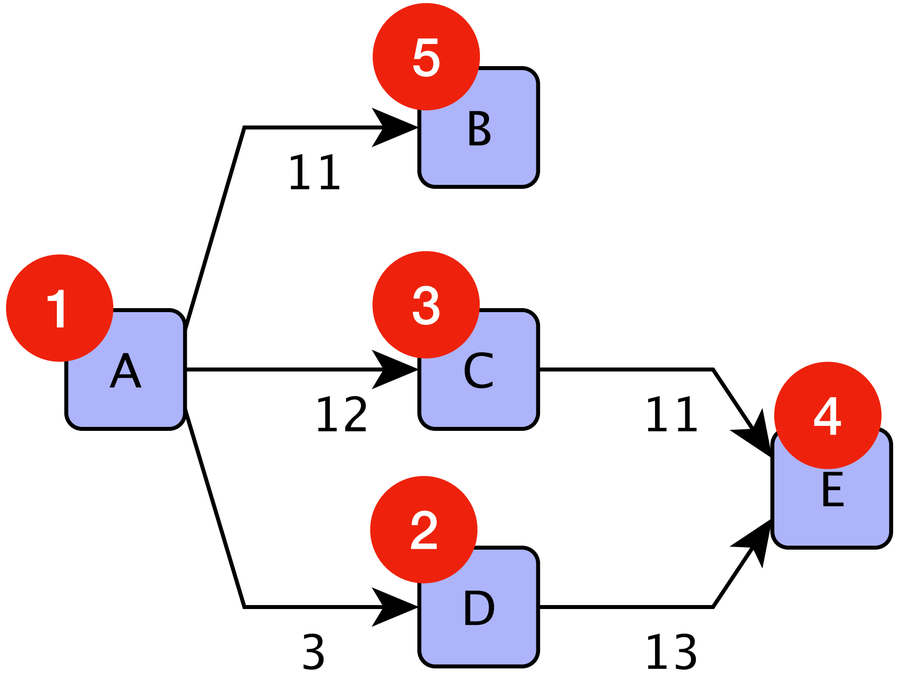
Random Variation



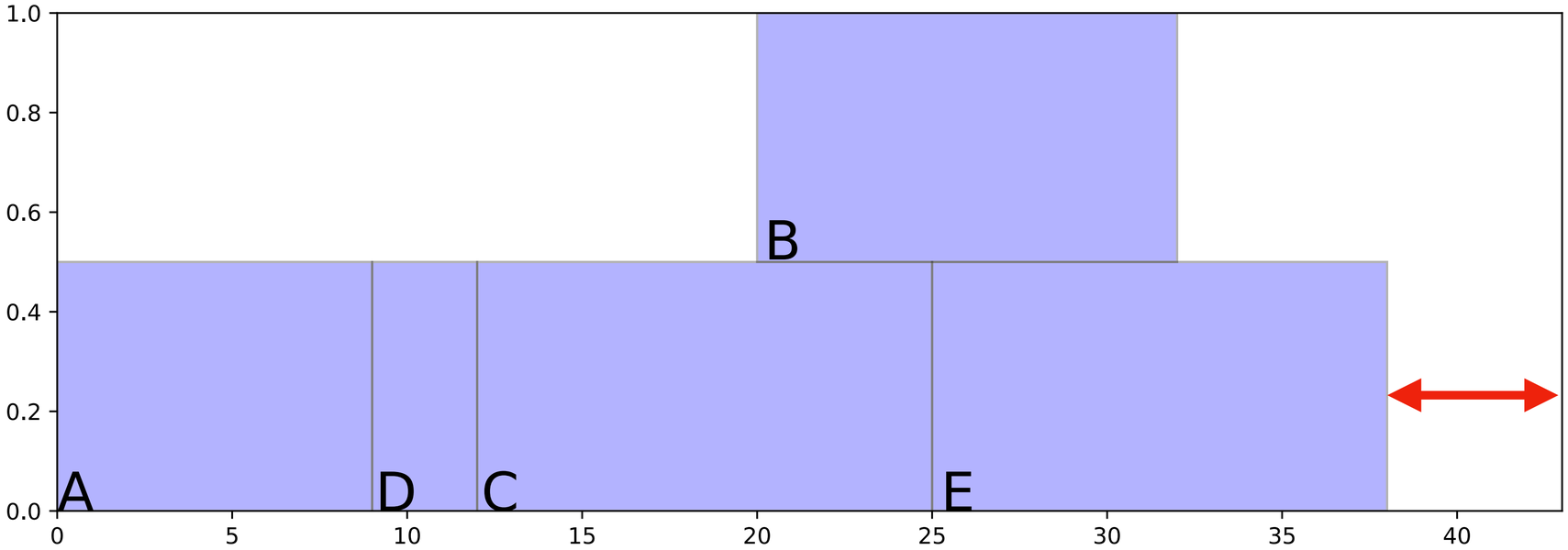
Schedule



Random Variation

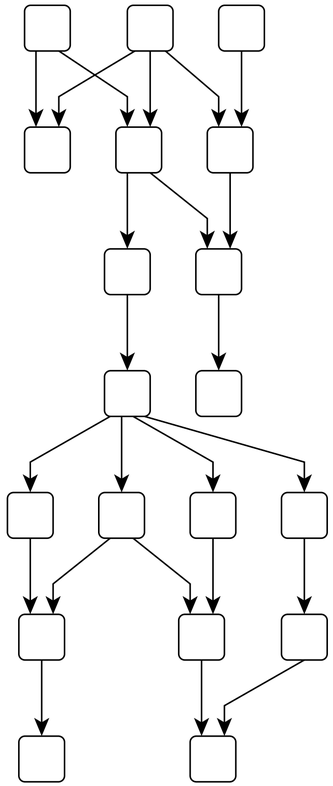


Schedule

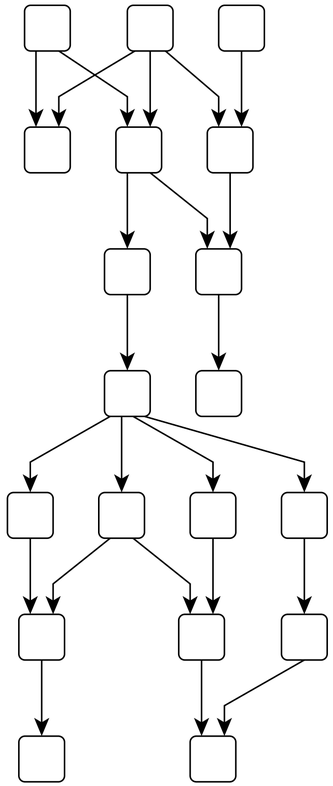


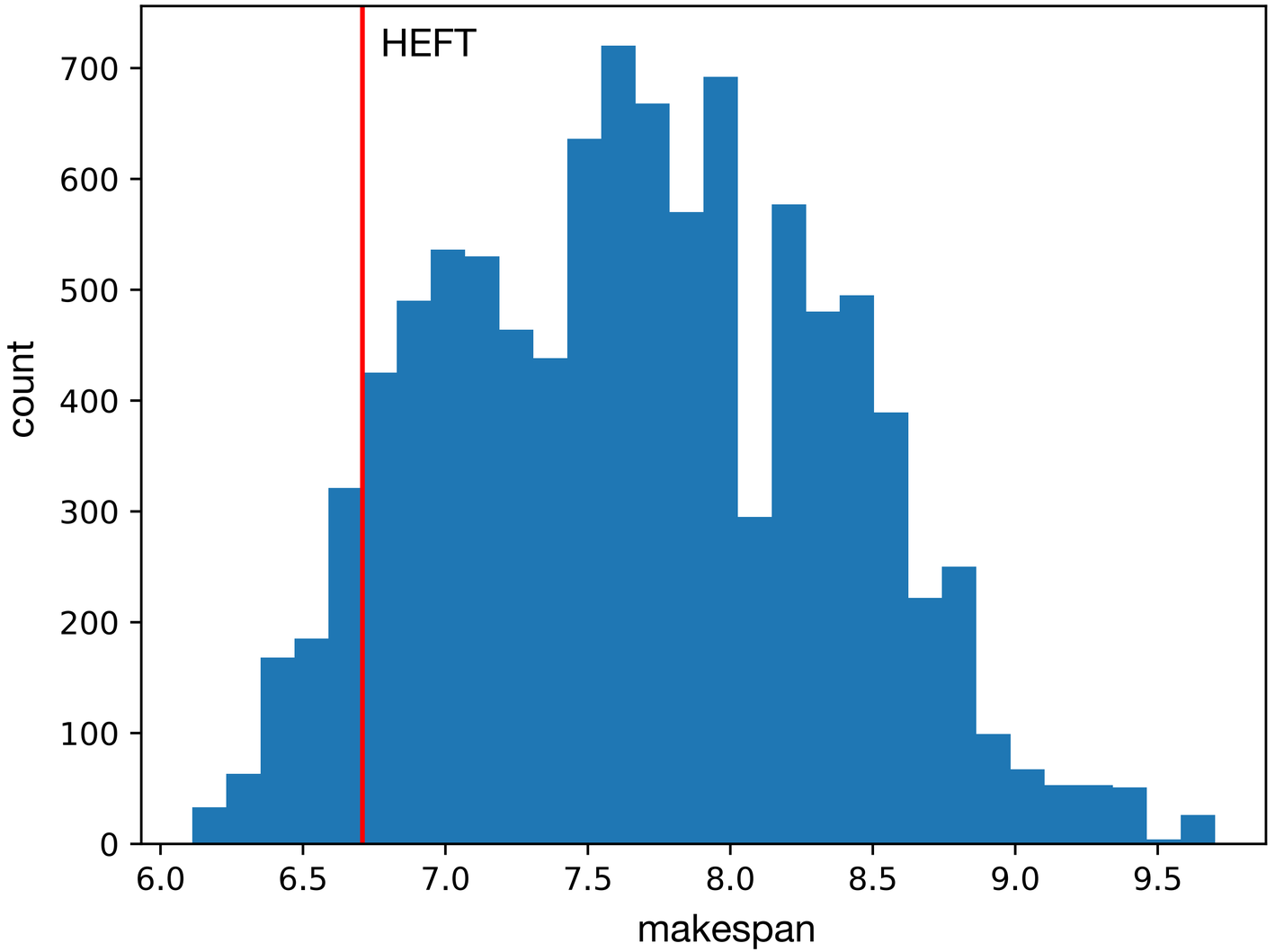
Makespan reduction

Impact of Different Rankings

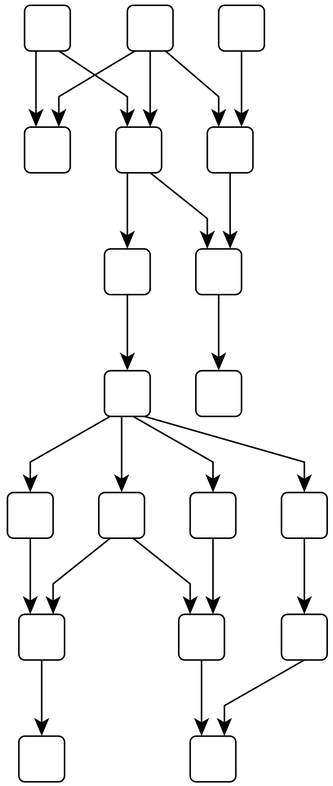


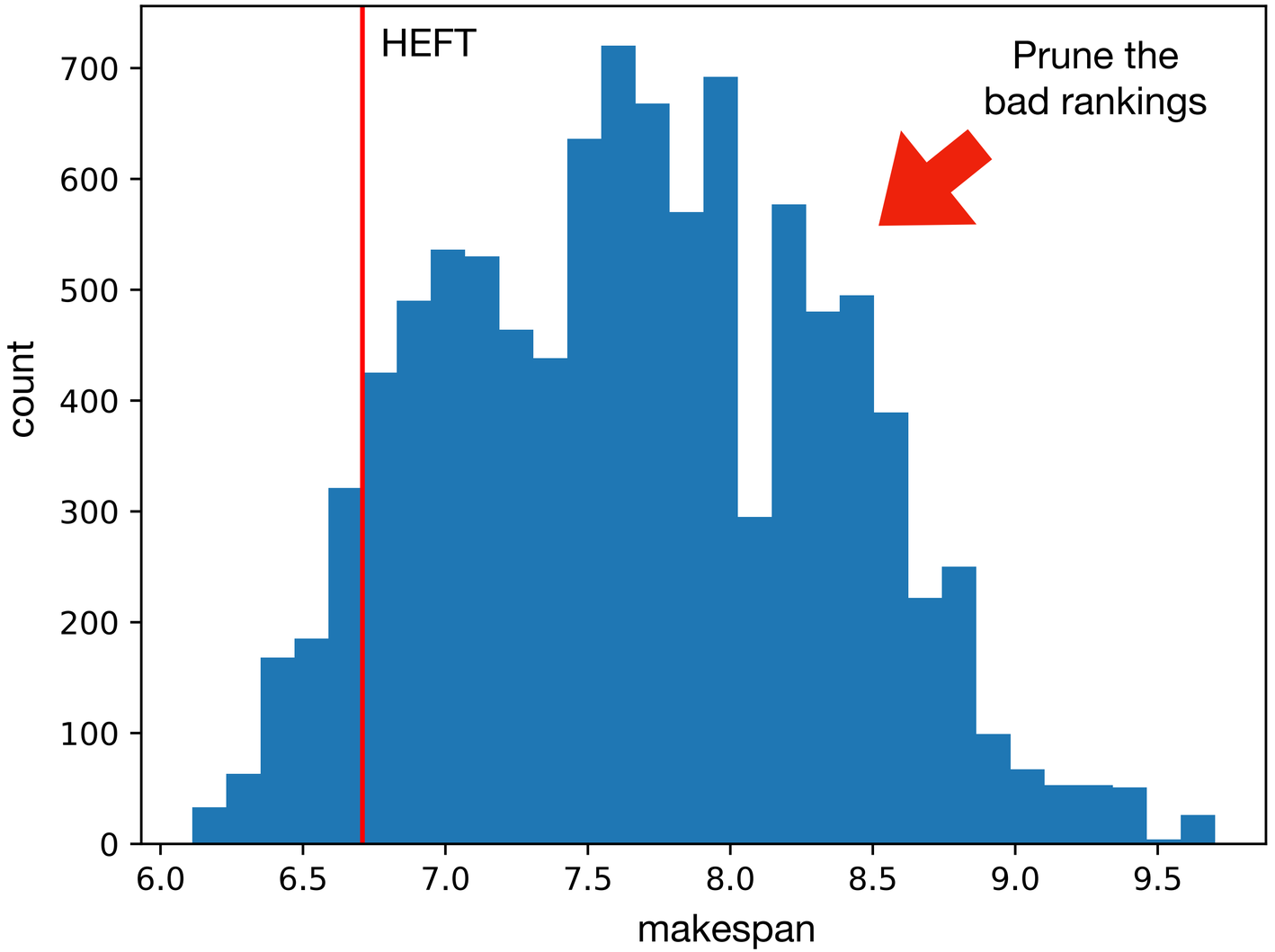
Impact of Different Rankings



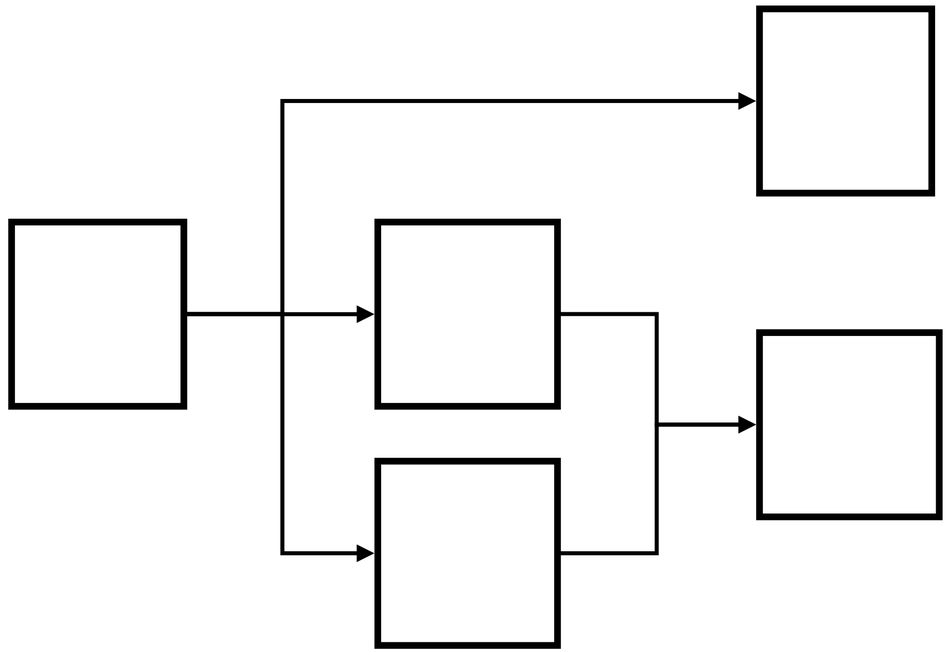


Impact of Different Rankings



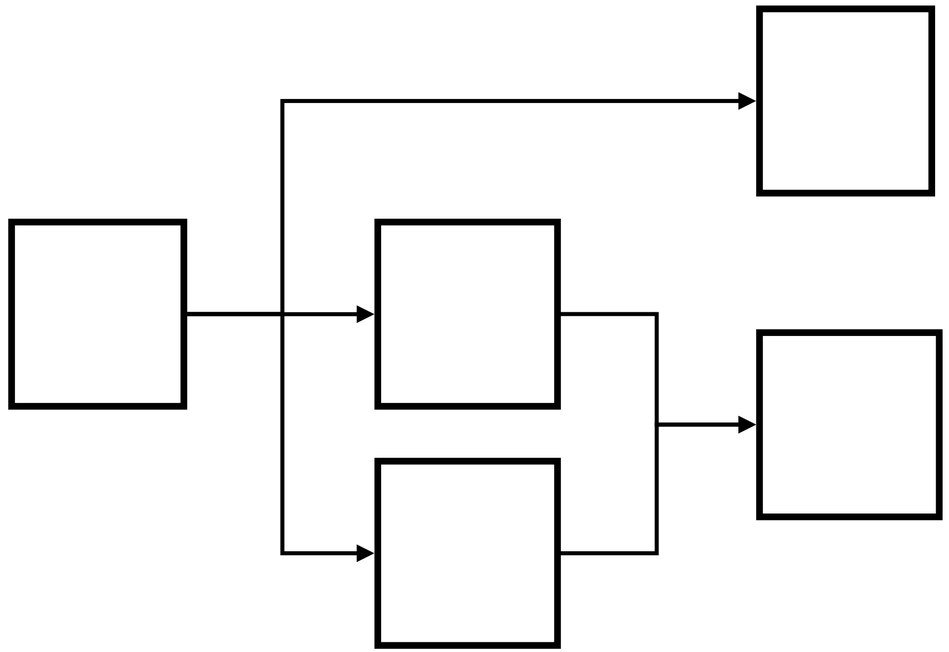


Levels for Random Rankings



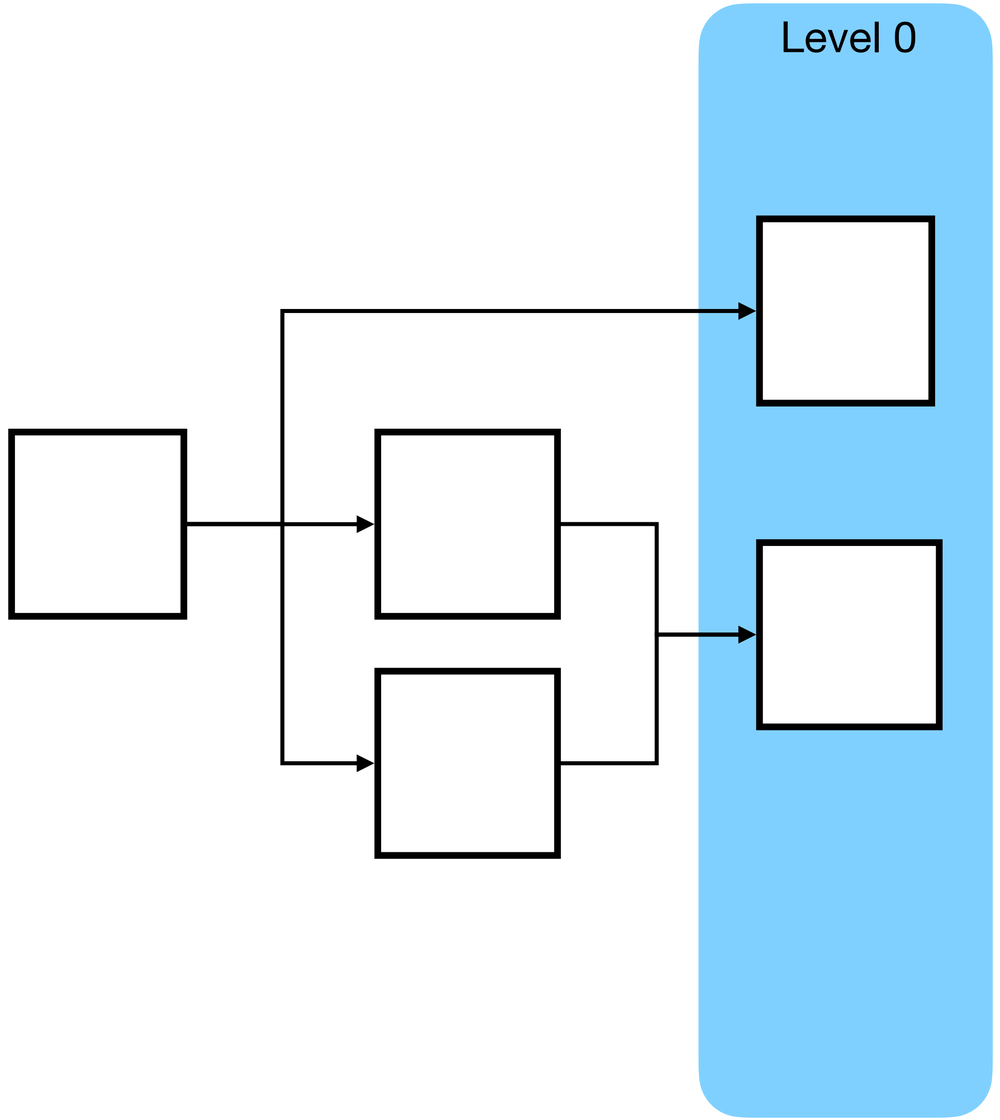
Levels for Random Rankings

Level: longest path to leaf



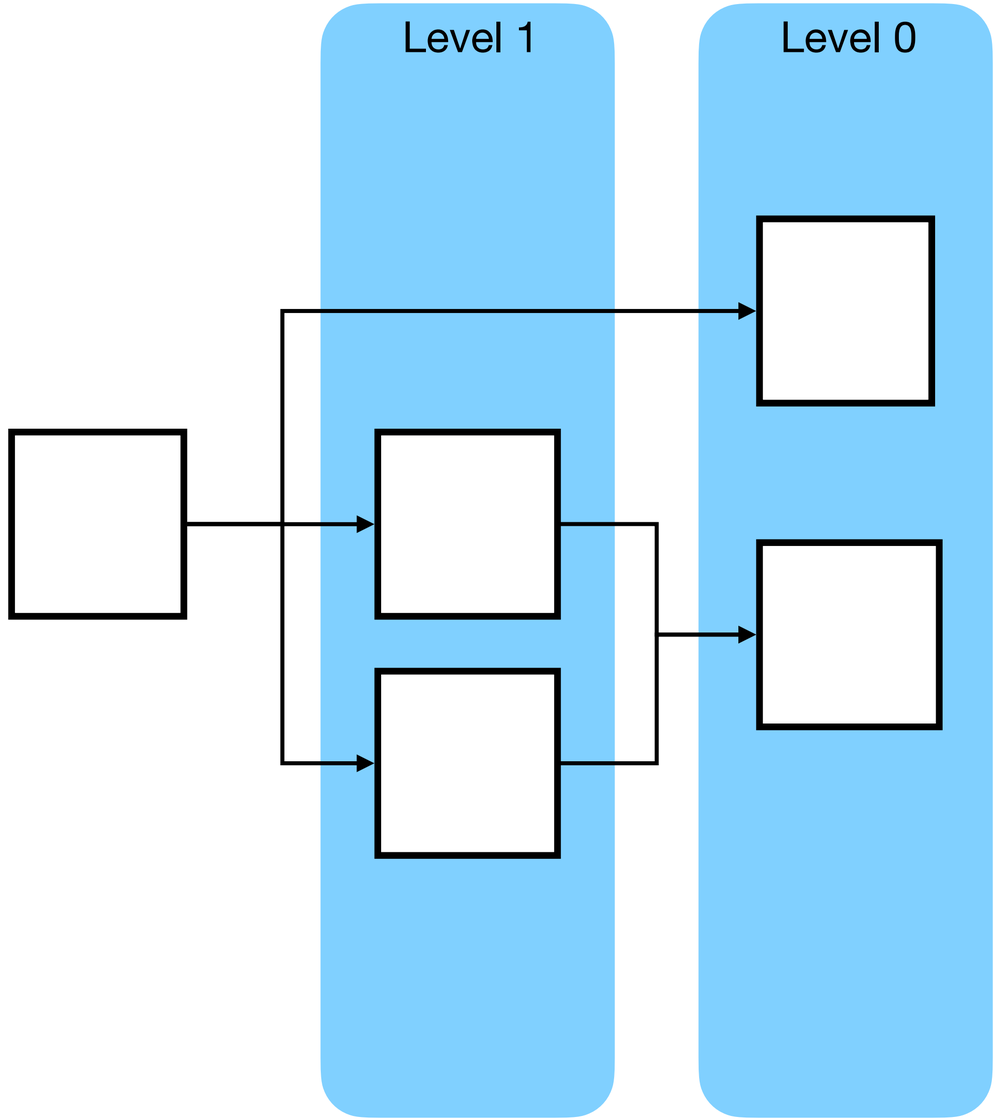
Levels for Random Rankings

Level: longest path to leaf



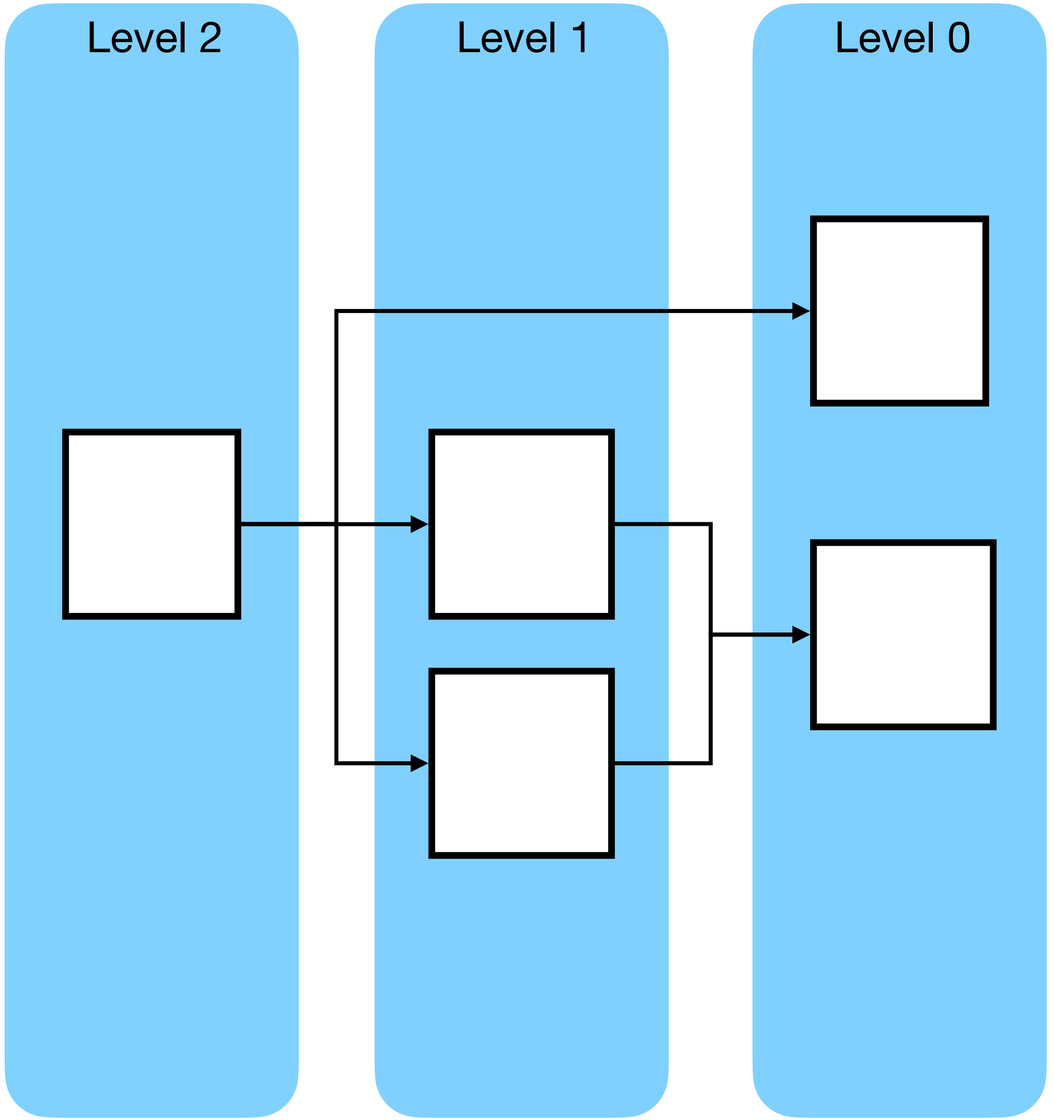
Levels for Random Rankings

Level: longest path to leaf



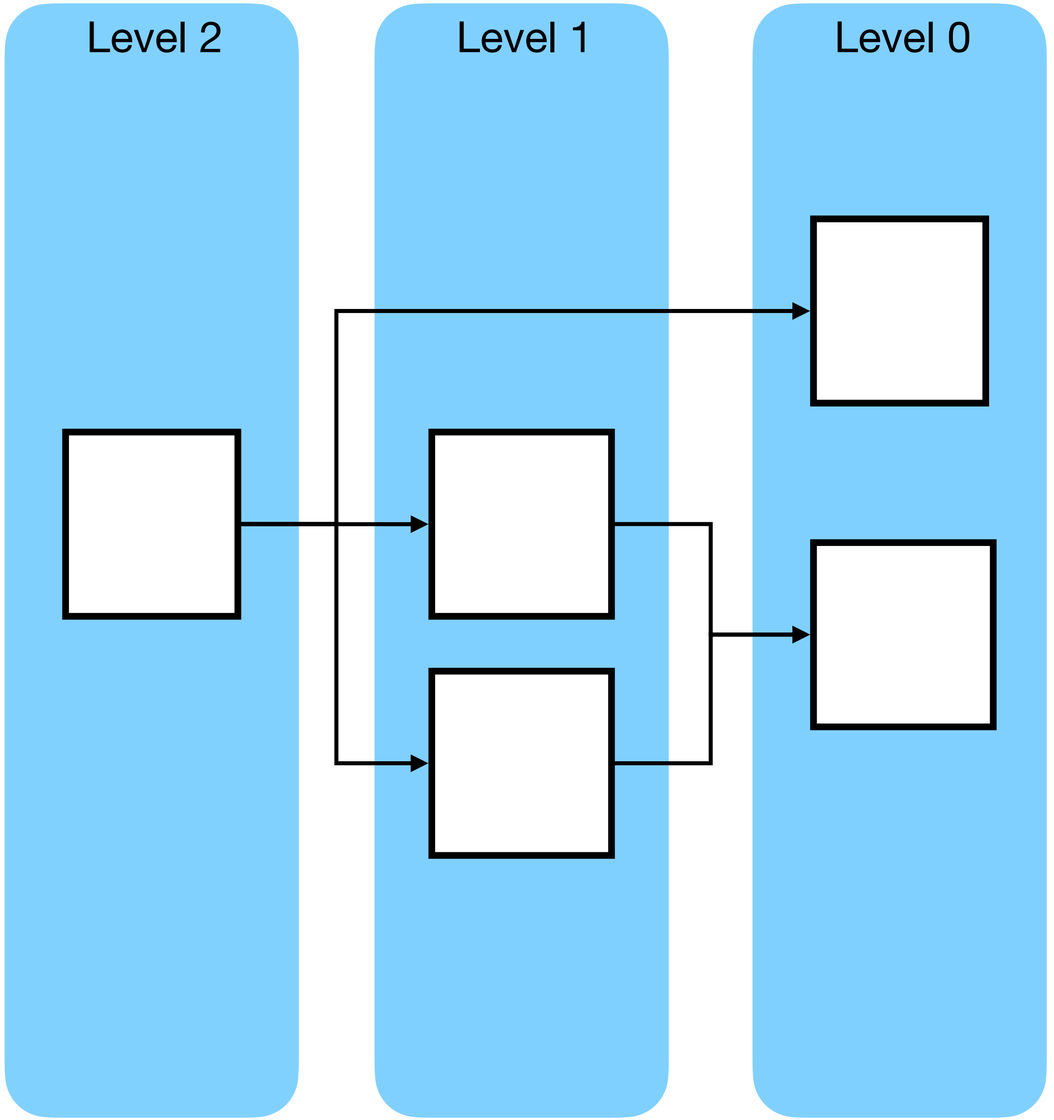
Levels for Random Rankings

Level: longest path to leaf



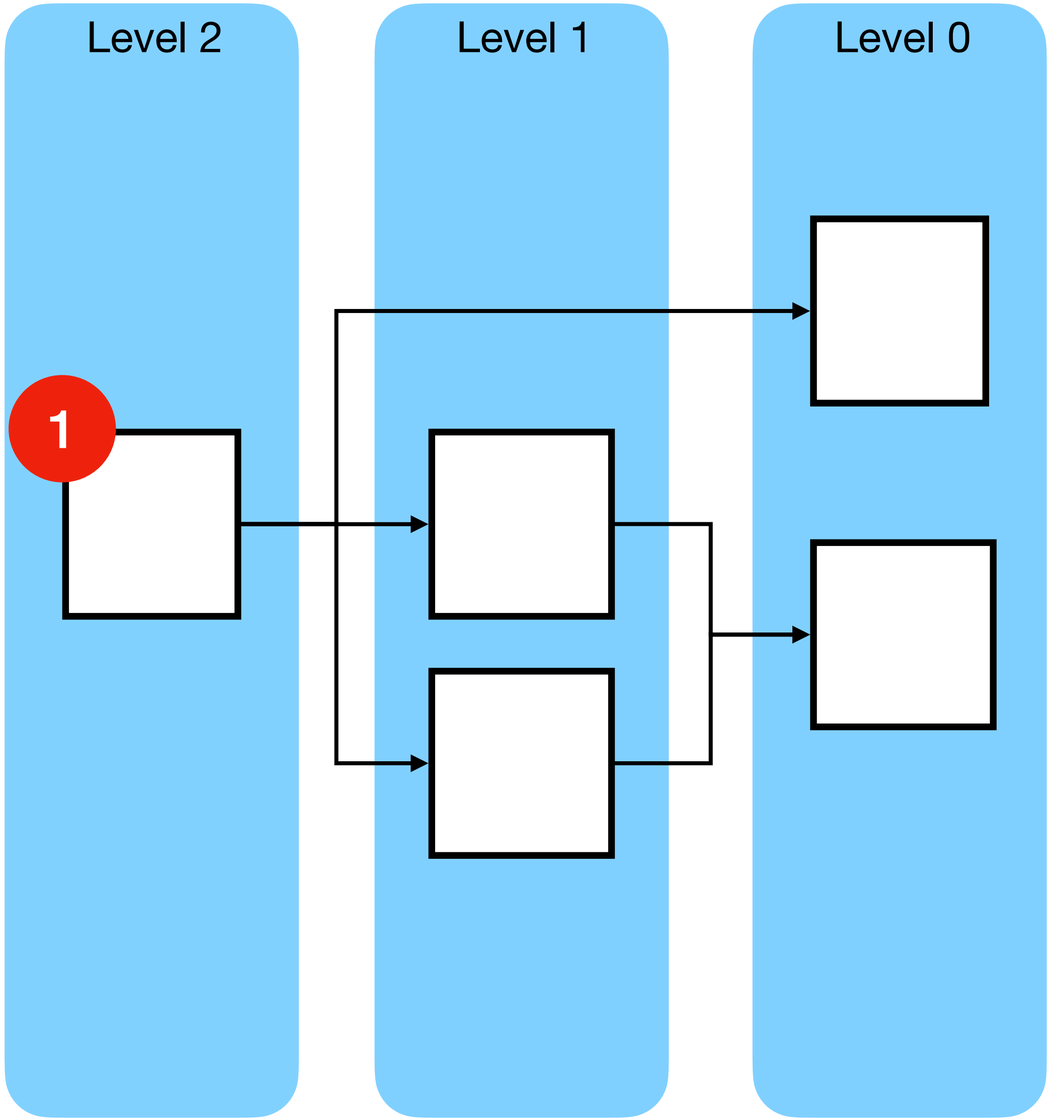
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level



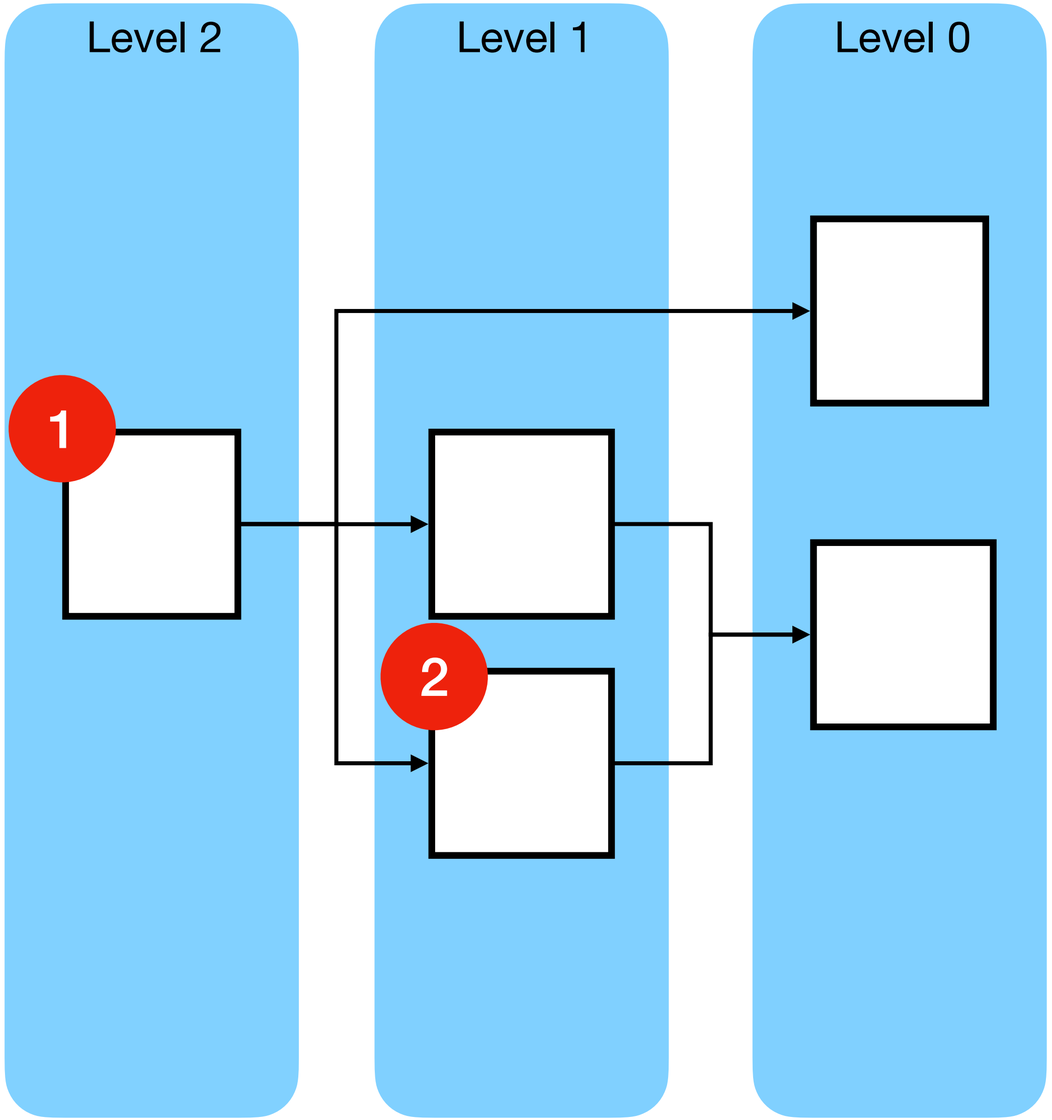
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level



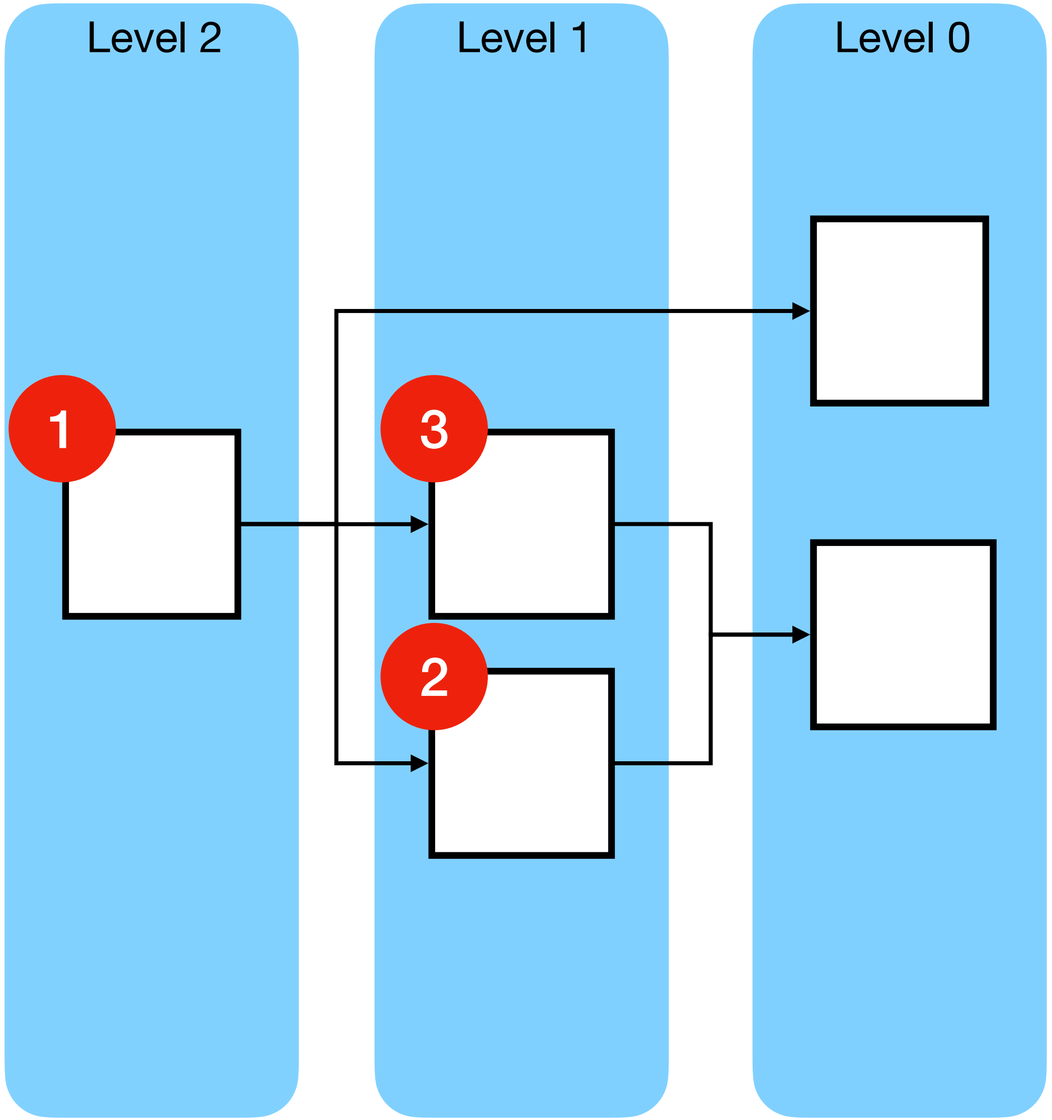
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level



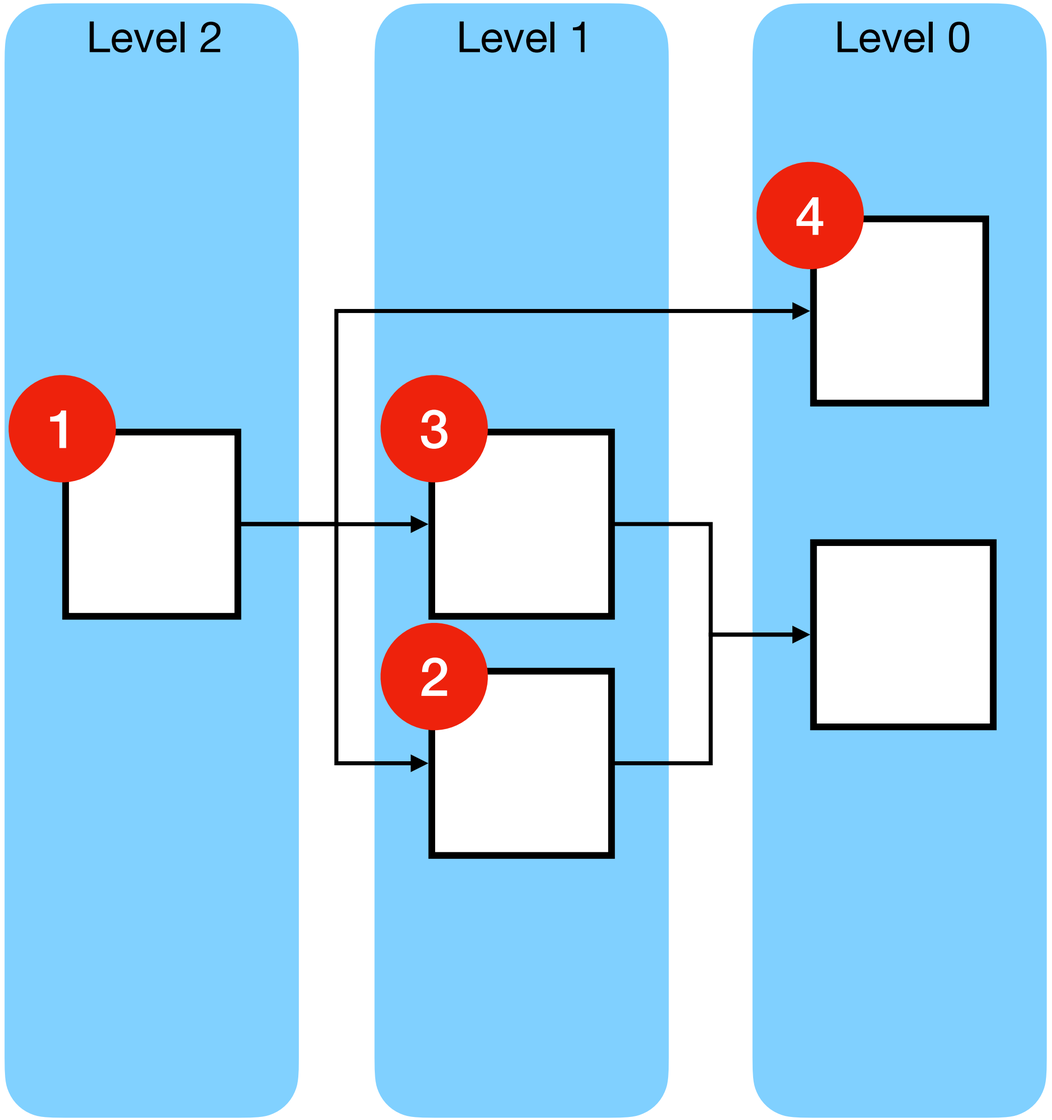
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level



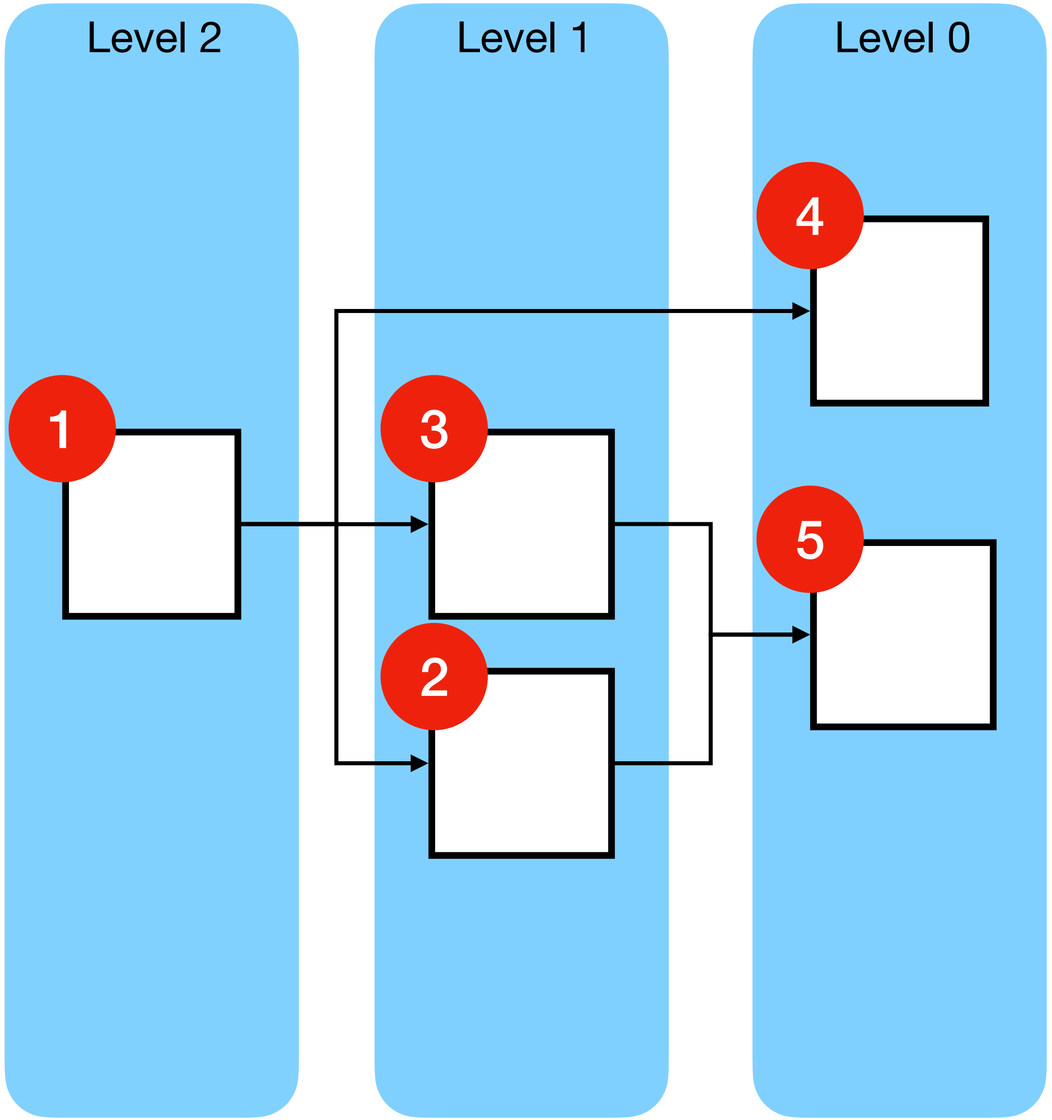
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level



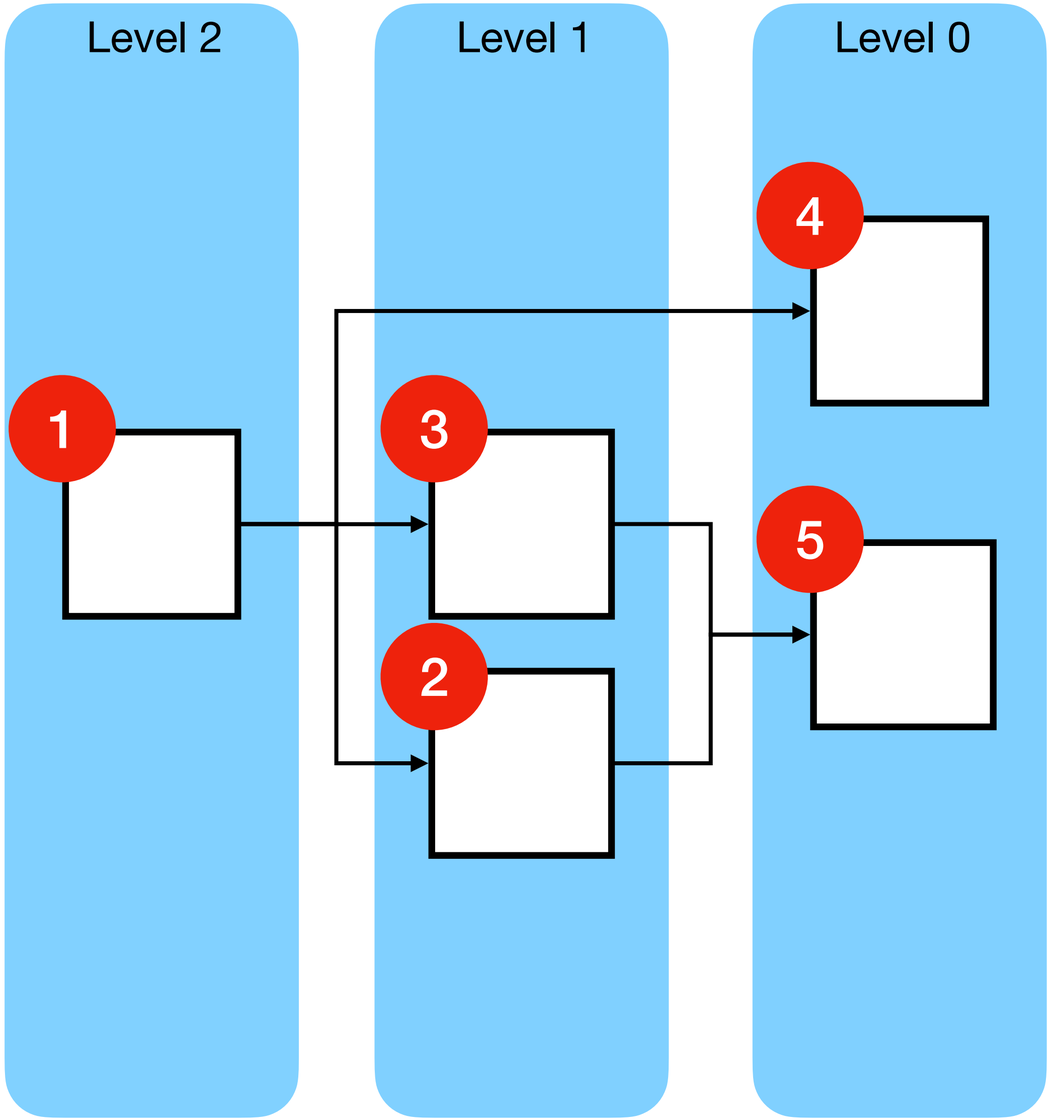
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level



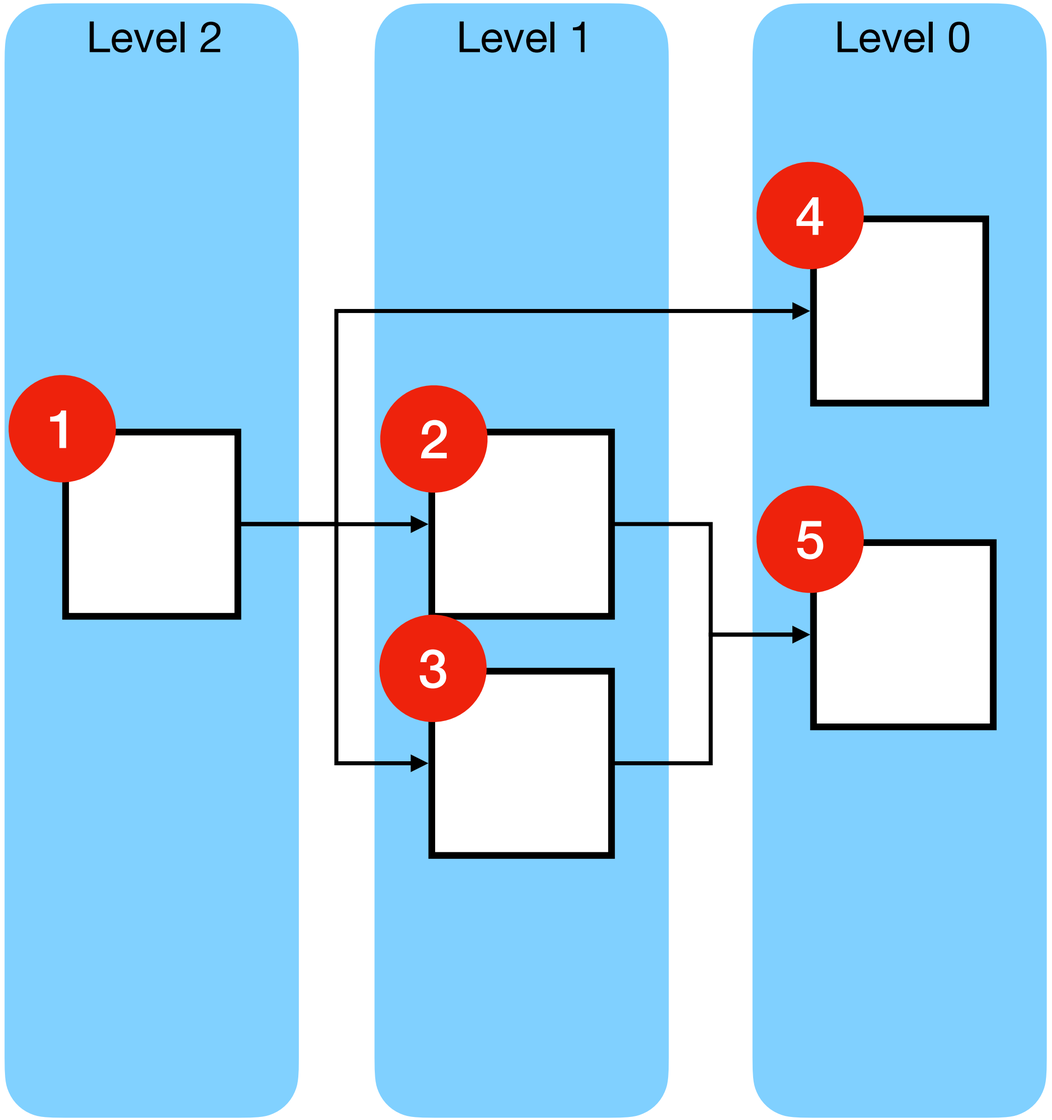
Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level
Shuffle: random permutation of tasks in one level

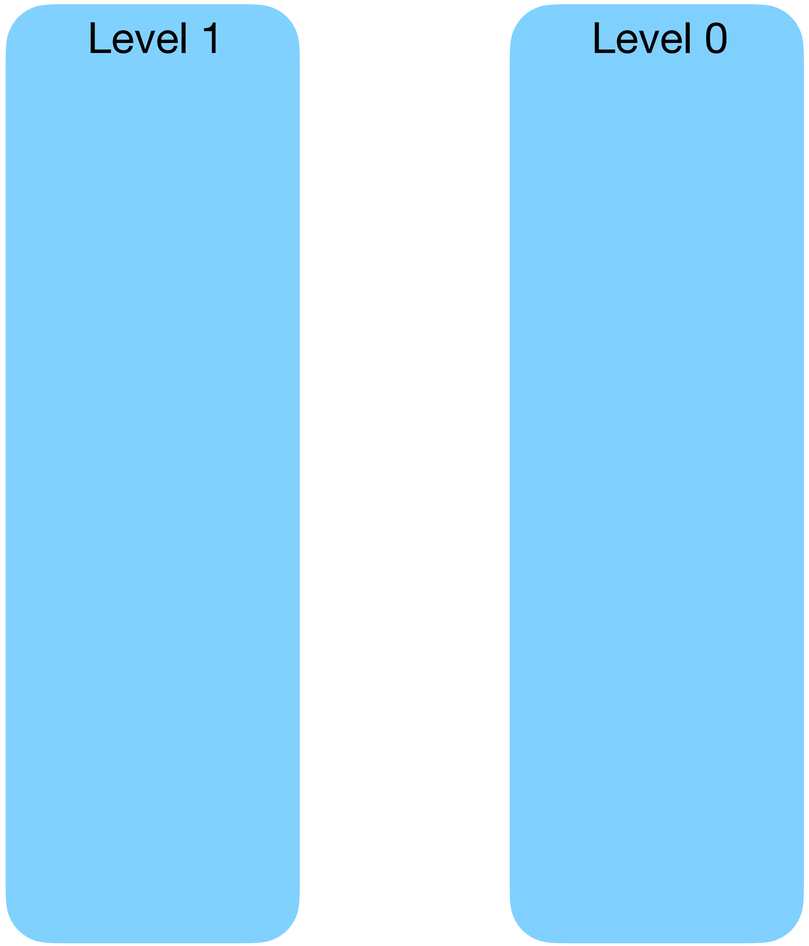


Levels for Random Rankings

Level: longest path to leaf
Ranking: concatenate permutations for each level
Shuffle: random permutation of tasks in one level

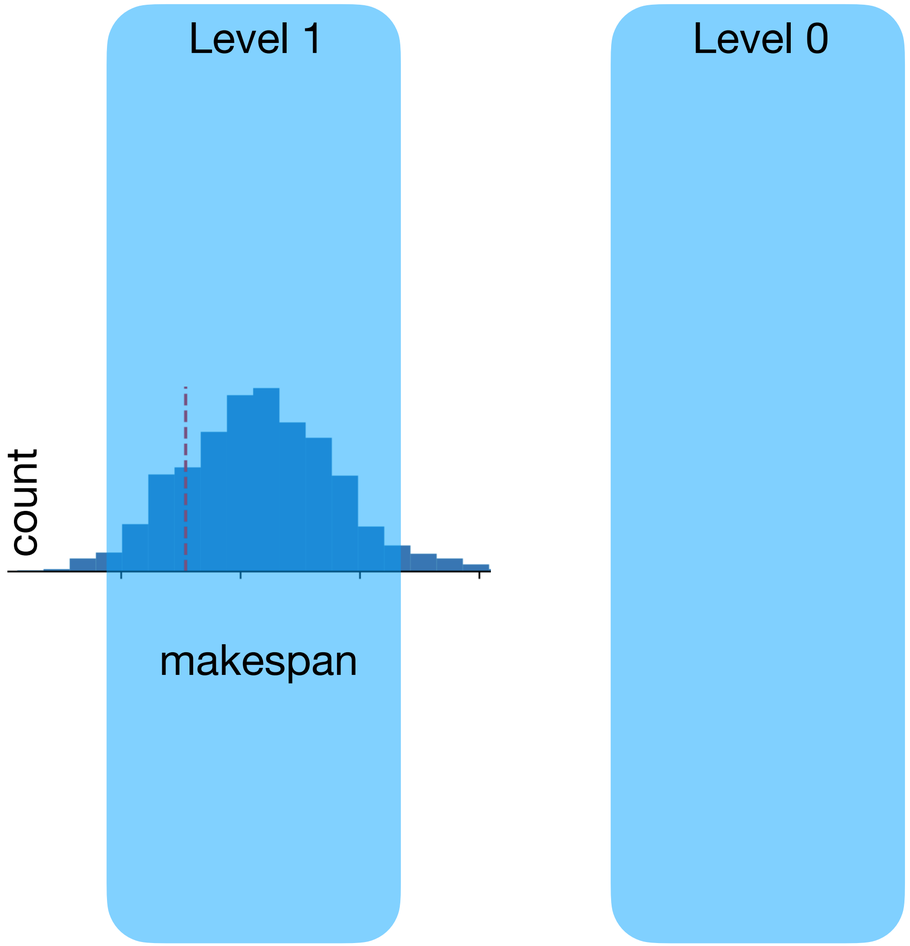


Estimating Improvement Probabilities



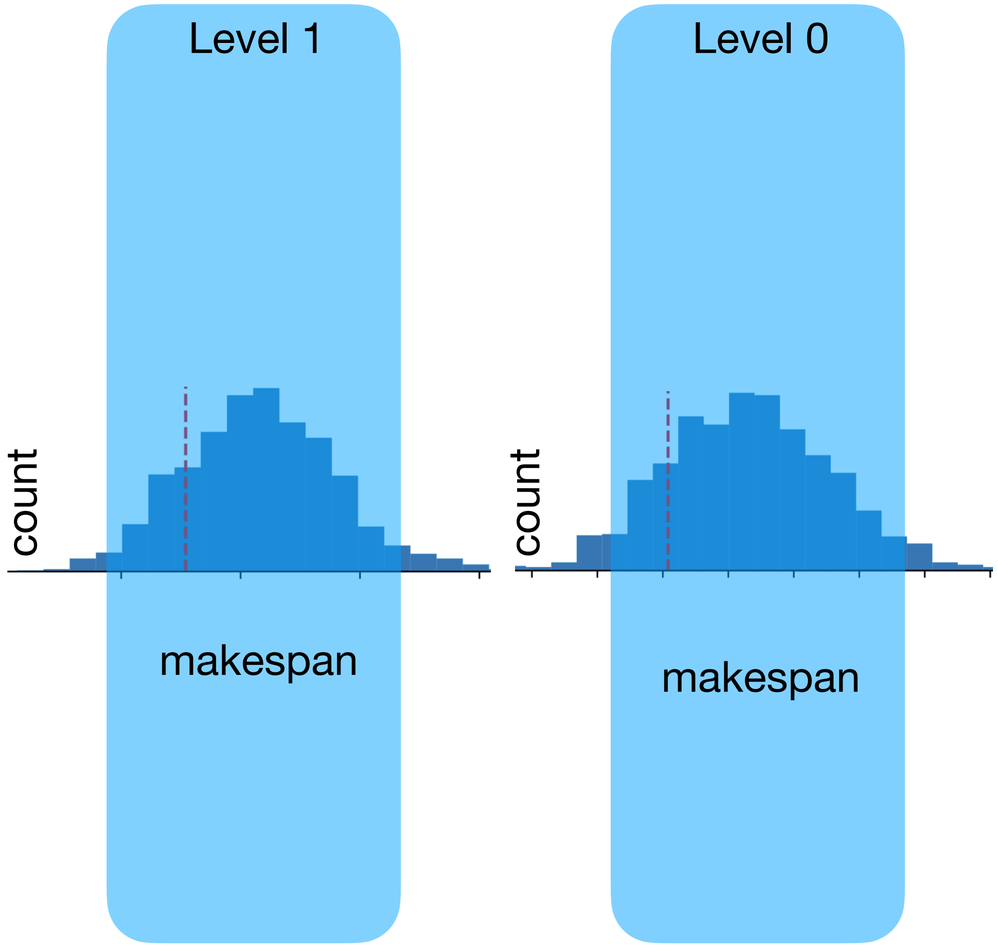
Level selection probability:
share of total improvement probabilities

Estimating Improvement Probabilities



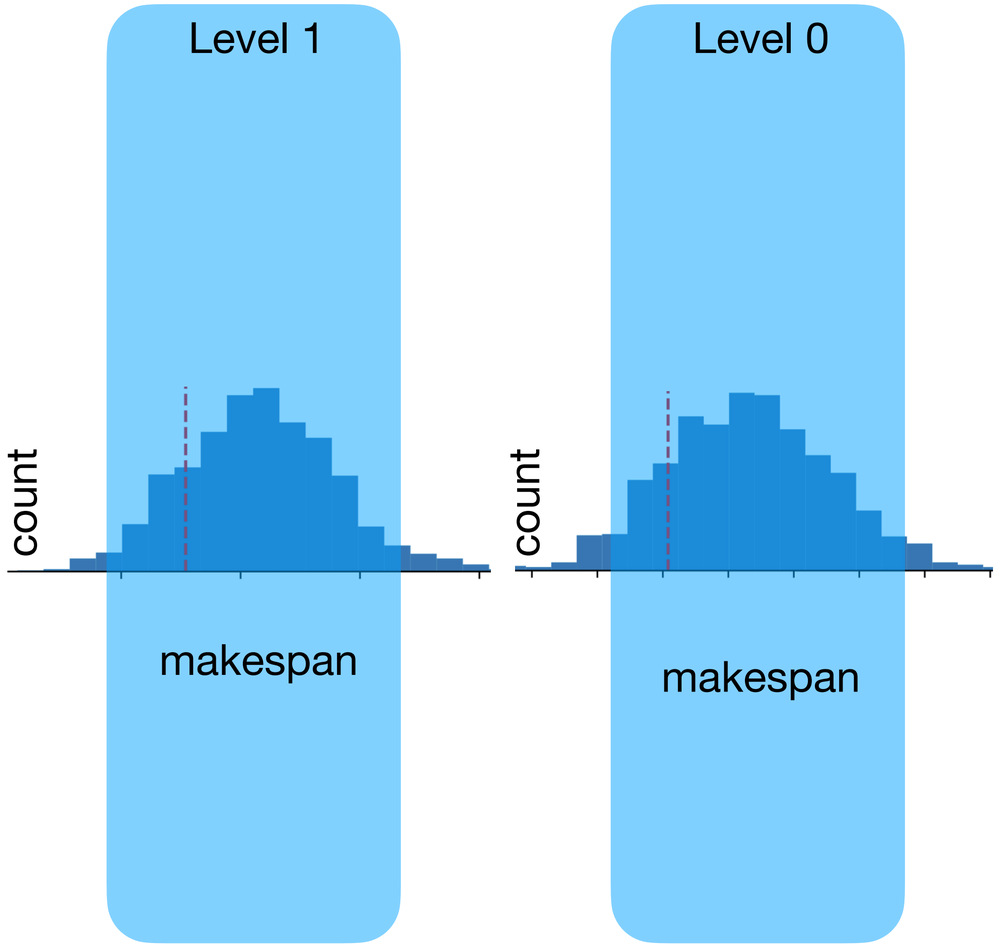
Level selection probability:
share of total improvement probabilities

Estimating Improvement Probabilities

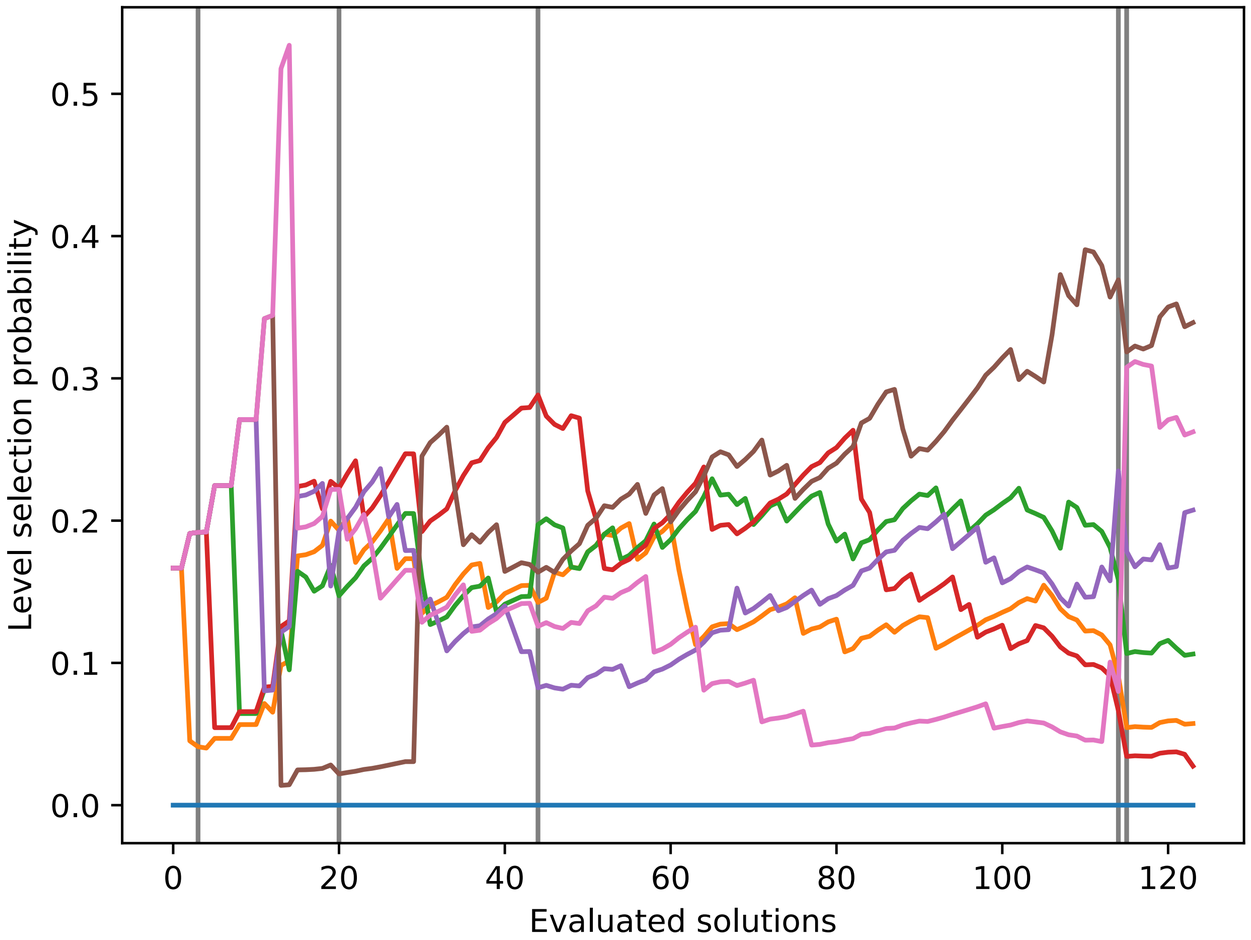


Level selection probability:
share of total improvement probabilities

Estimating Improvement Probabilities

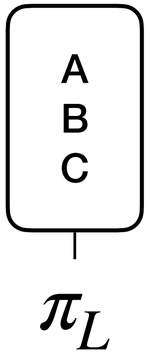


Level selection probability:
share of total improvement probabilities



Exploitation and Exploration

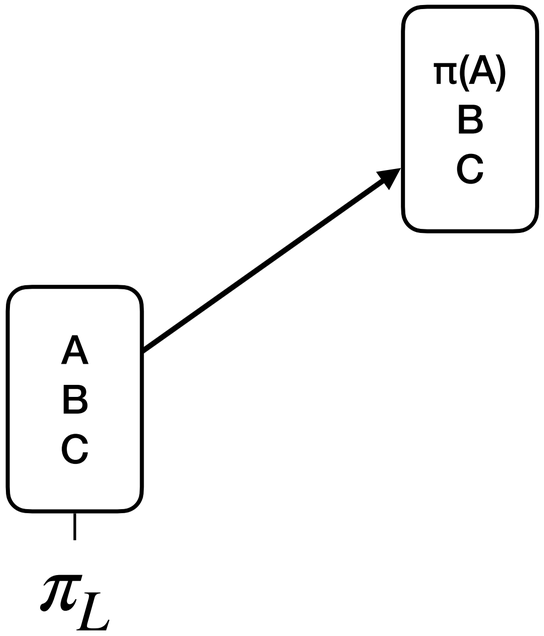
Time Budget
60 seconds



Exploit

...

Exploitation and Exploration



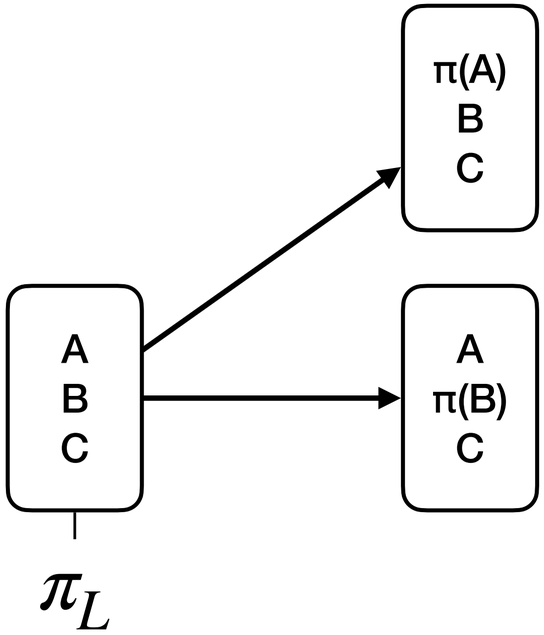
Time Budget
60 seconds

Exploit

...

Exploitation and Exploration

Time Budget
60 seconds

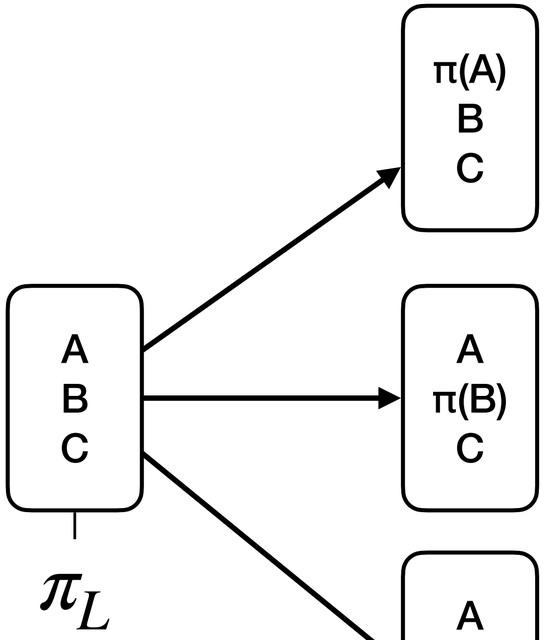


Exploit

...

Exploitation and Exploration

Time Budget
60 seconds

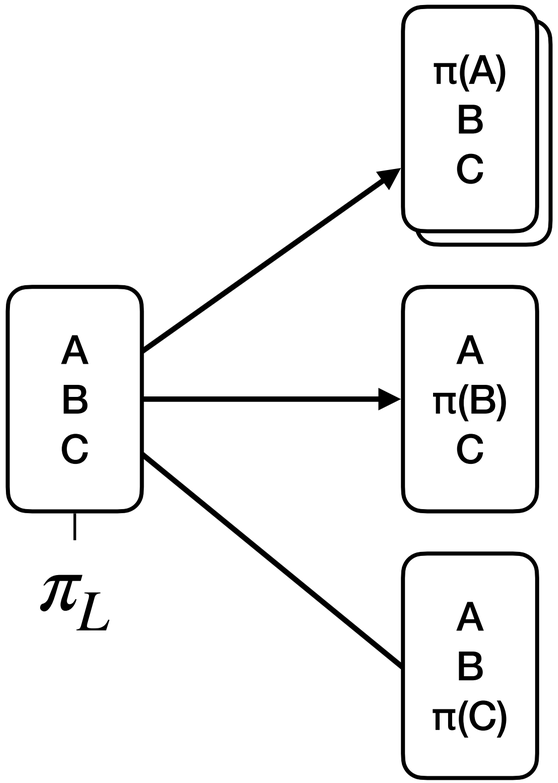


Exploit

...

Exploitation and Exploration

Time Budget
60 seconds

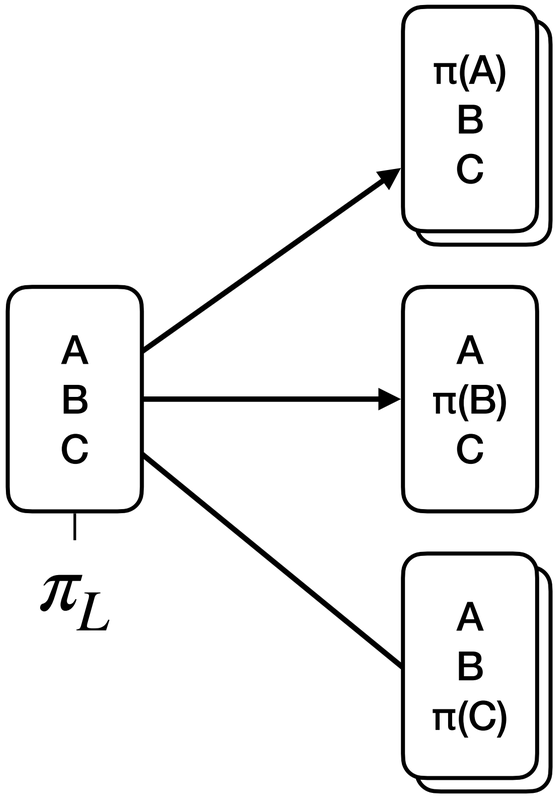


Exploit

...

Exploitation and Exploration

Time Budget
60 seconds

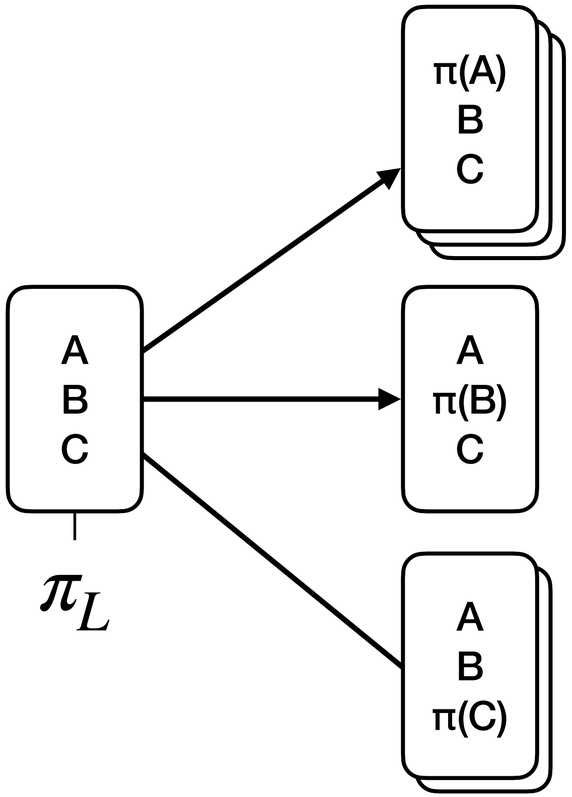


Exploit

...

Exploitation and Exploration

Time Budget
60 seconds

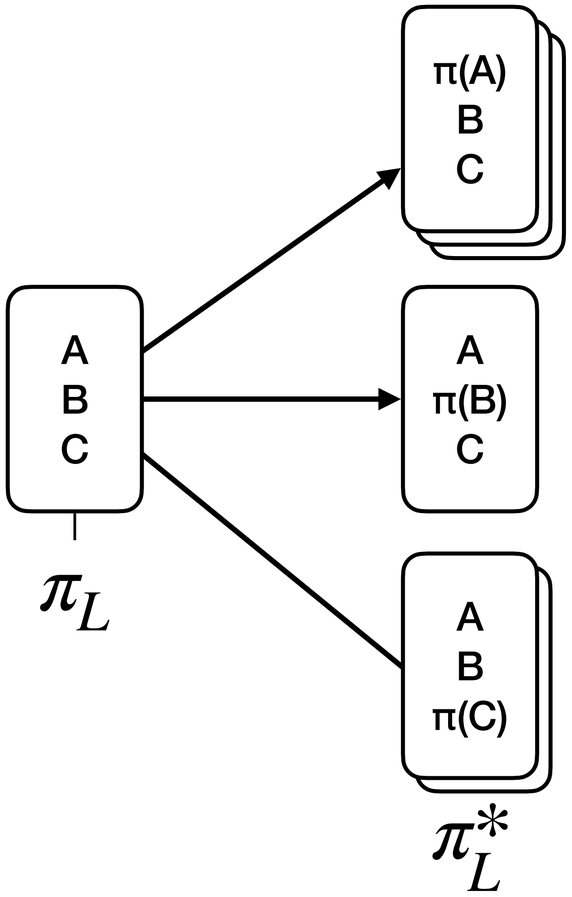


Exploit

...

Exploitation and Exploration

Time Budget
60 seconds

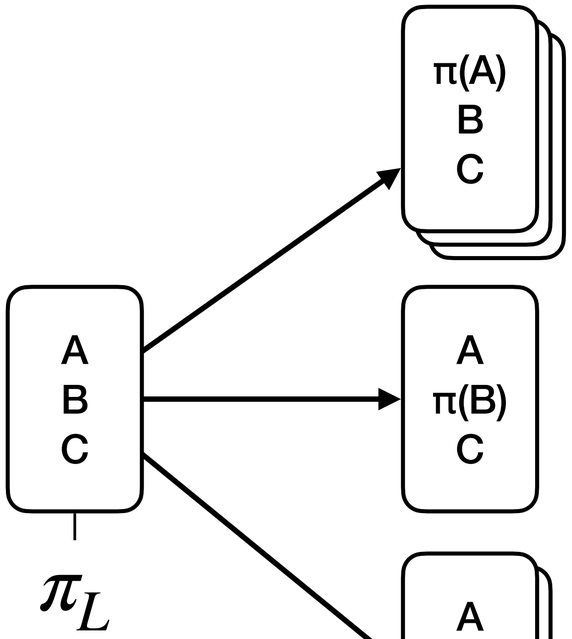


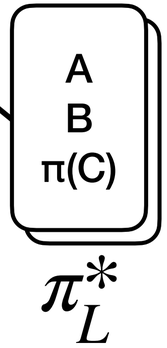
Exploit

...

Exploitation and Exploration

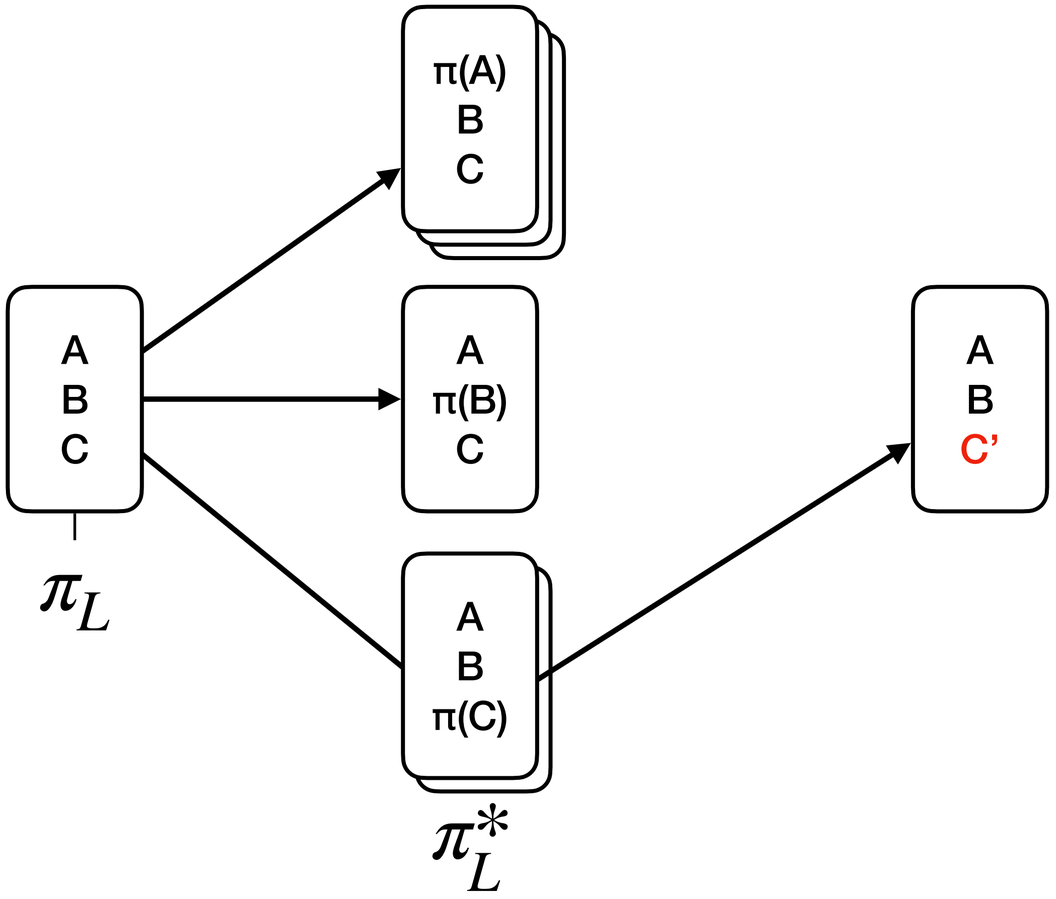
Time Budget
60 seconds





Exploitation and Exploration

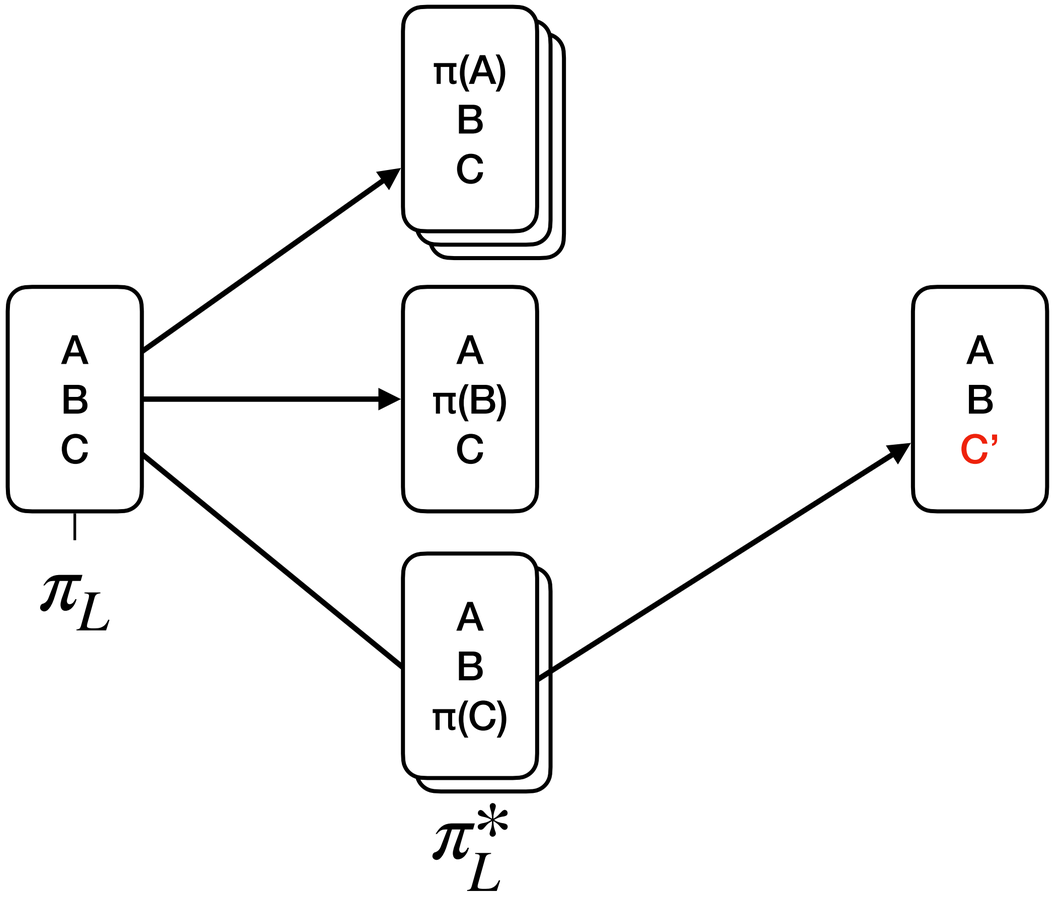
Time Budget
60 seconds



Exploit | Explore | ...

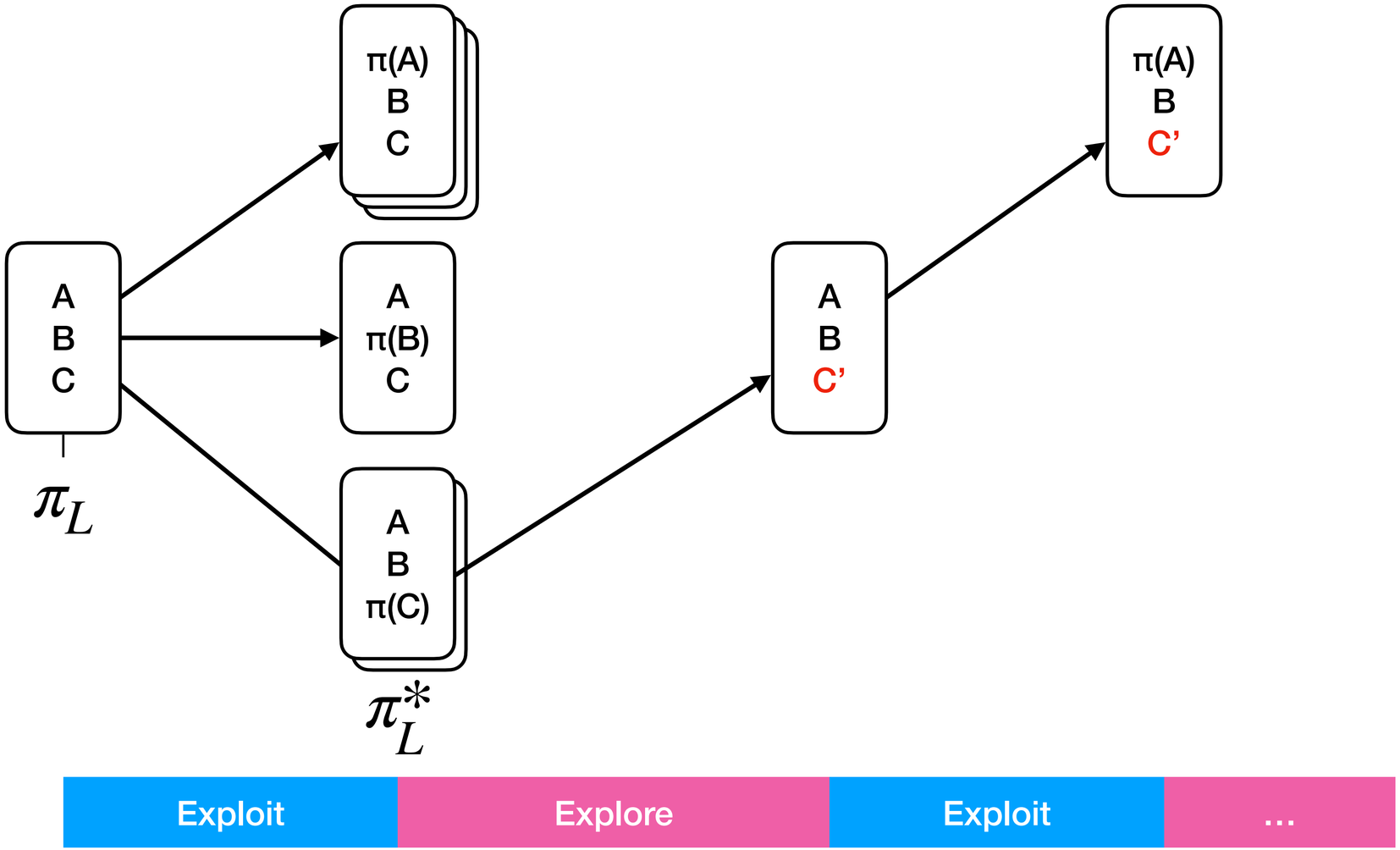
Exploitation and Exploration

Time Budget
60 seconds



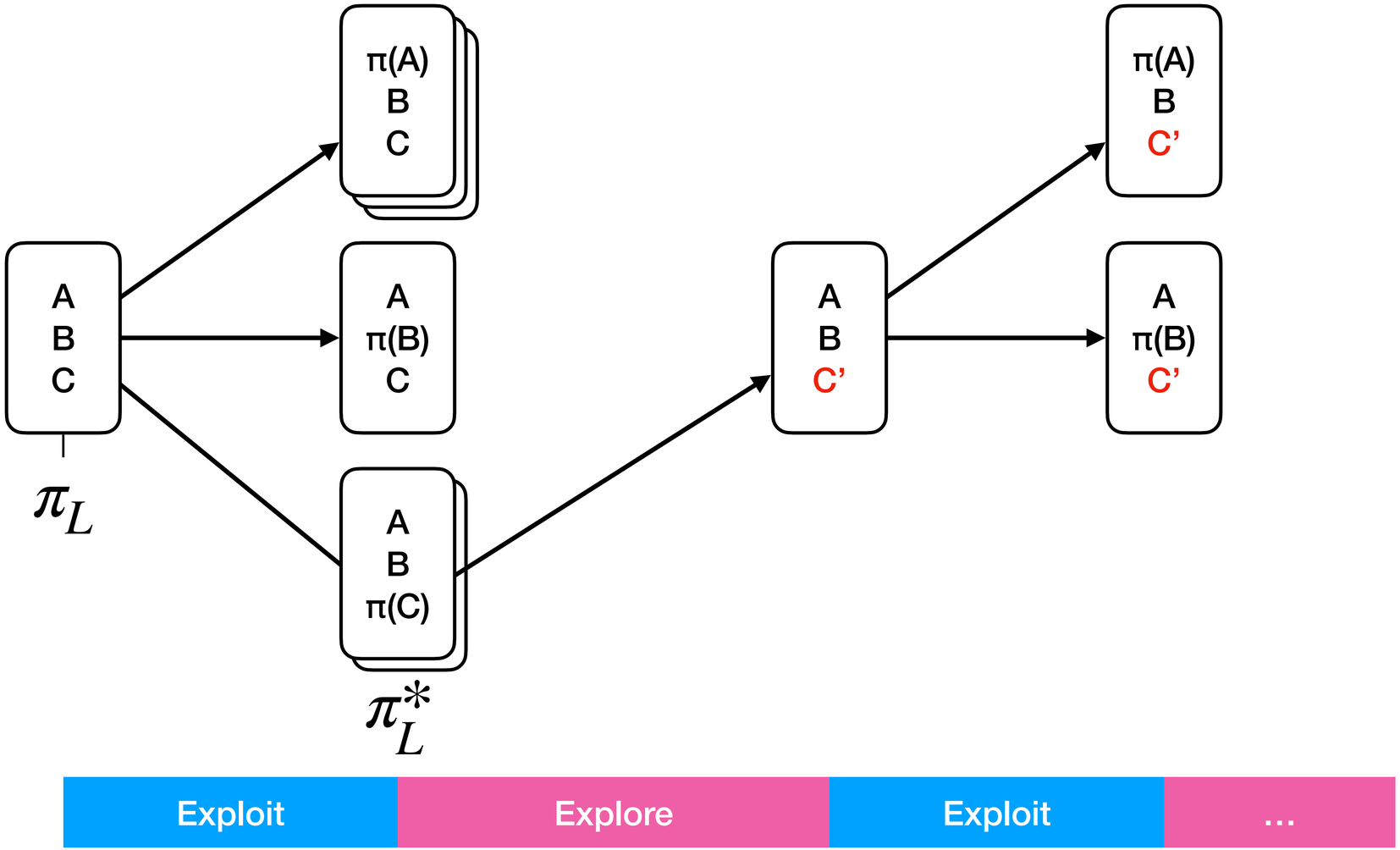
Exploitation and Exploration

Time Budget
60 seconds



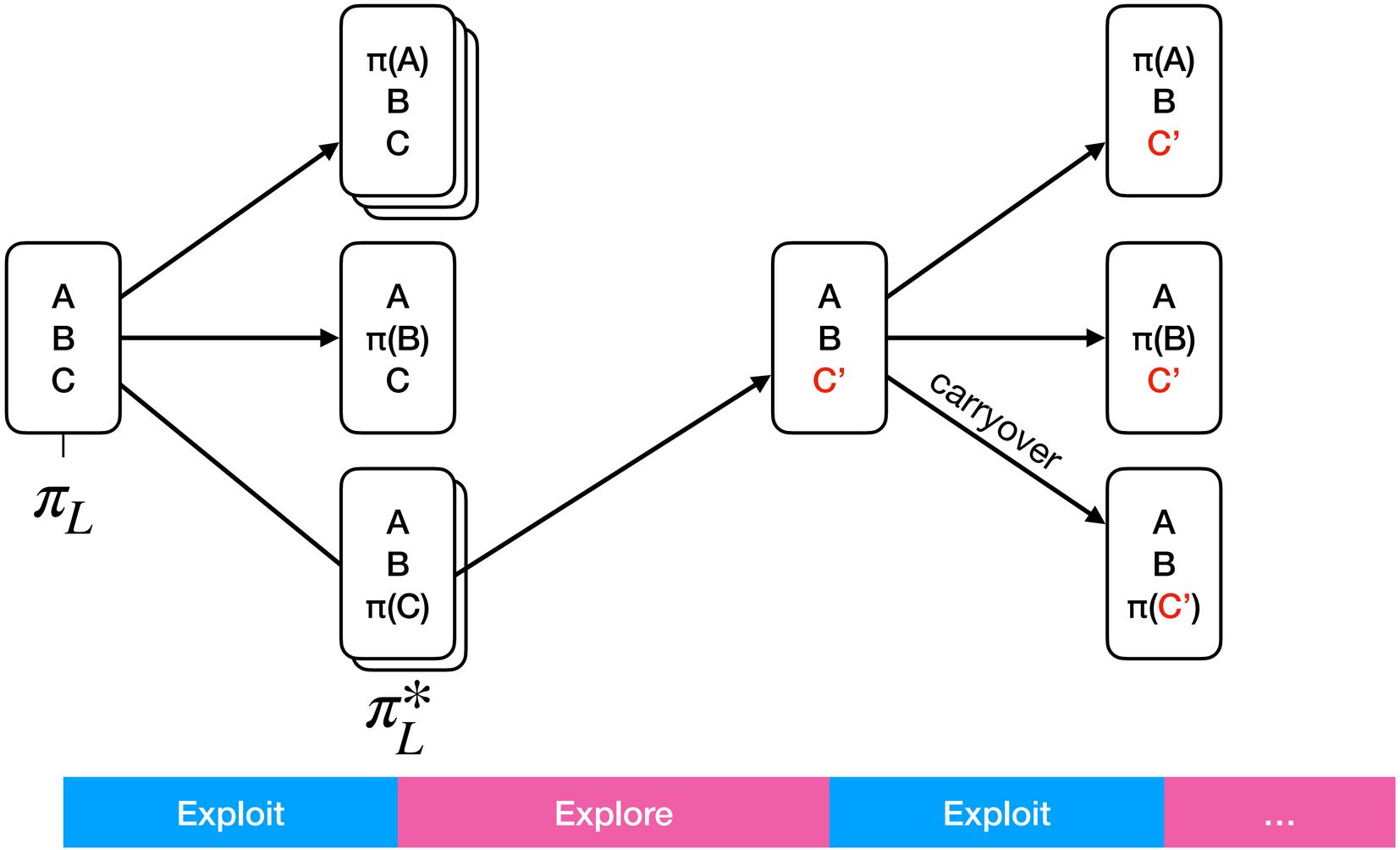
Exploitation and Exploration

Time Budget
60 seconds



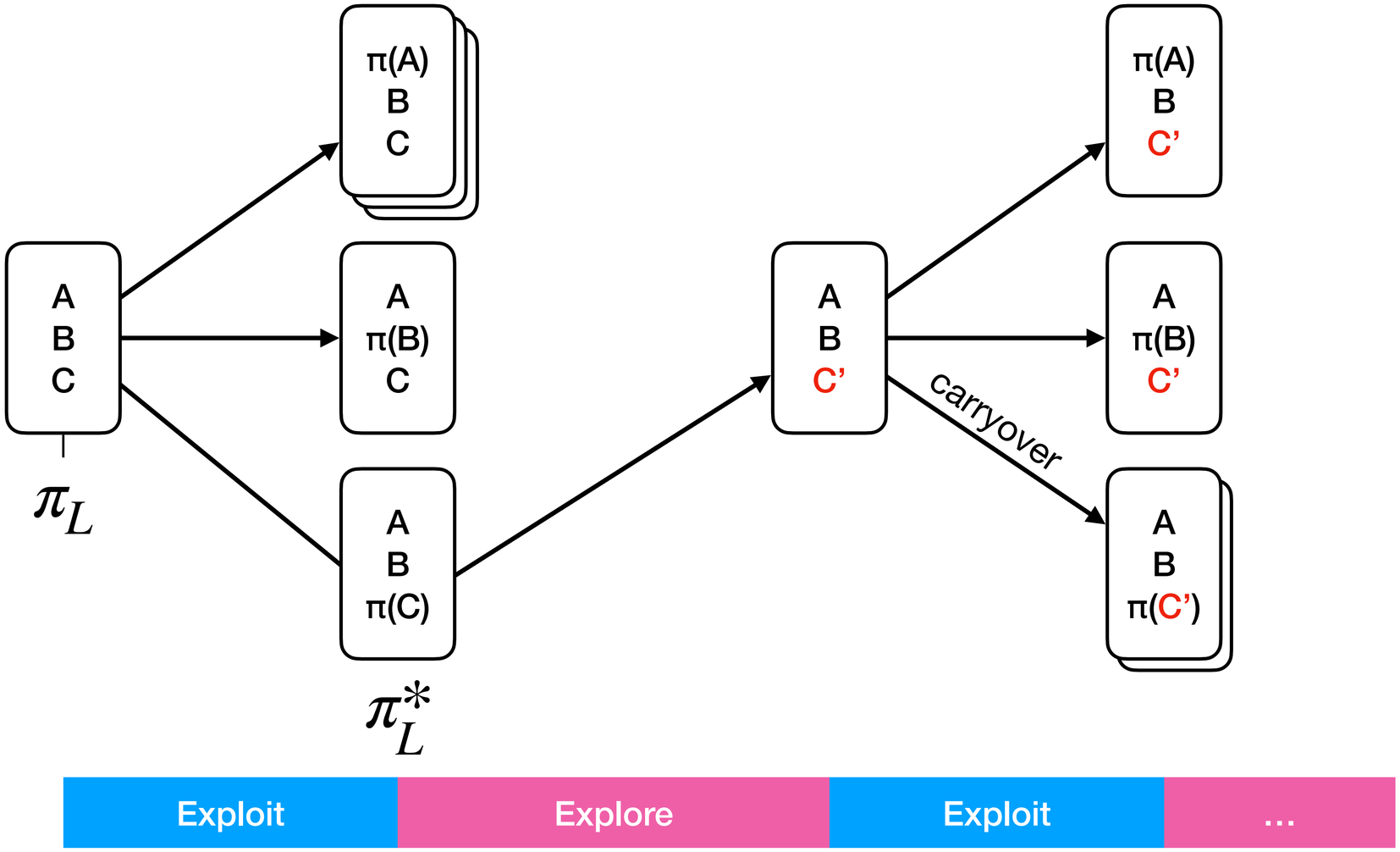
Exploitation and Exploration

Time Budget
60 seconds



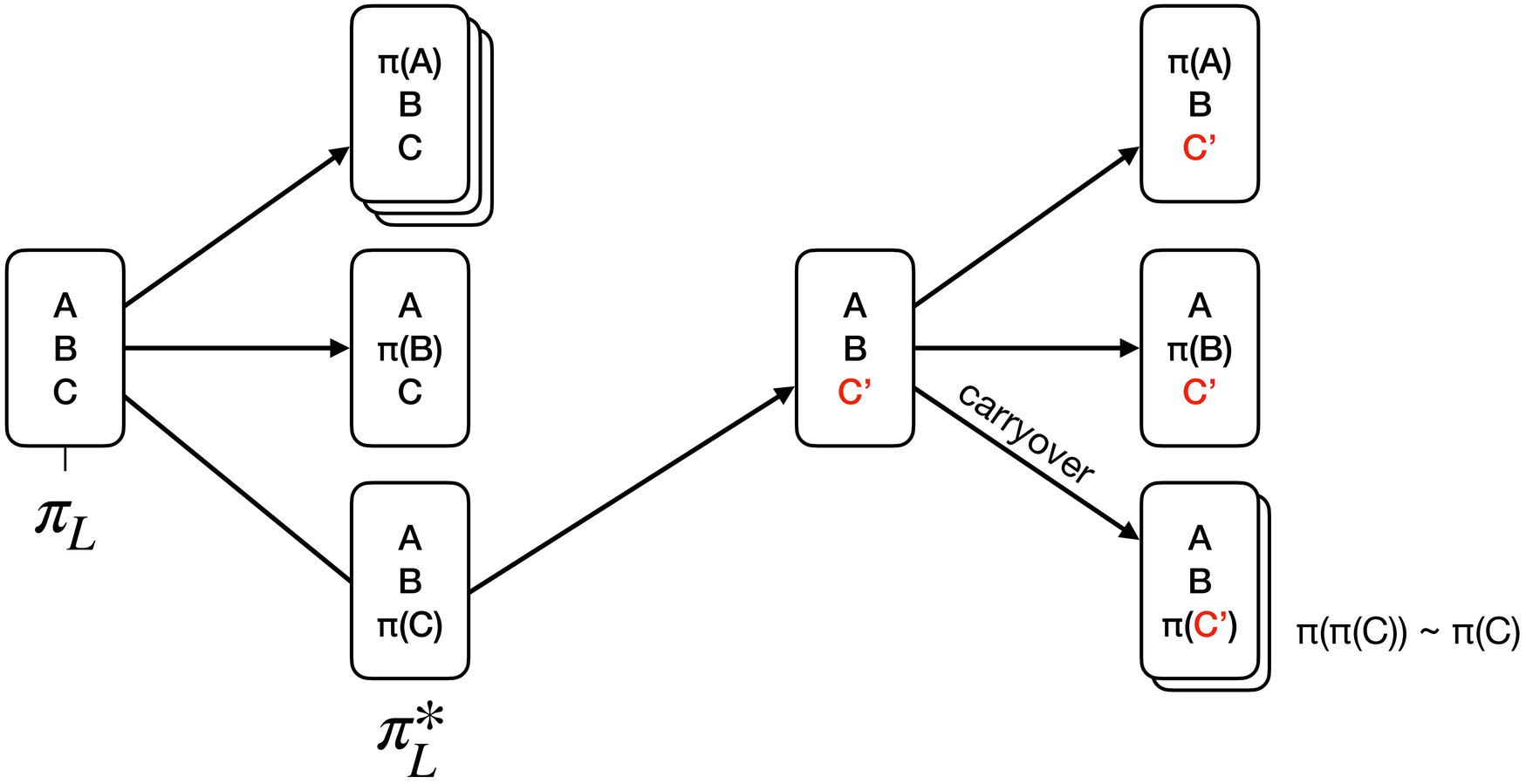
Exploitation and Exploration

Time Budget
60 seconds

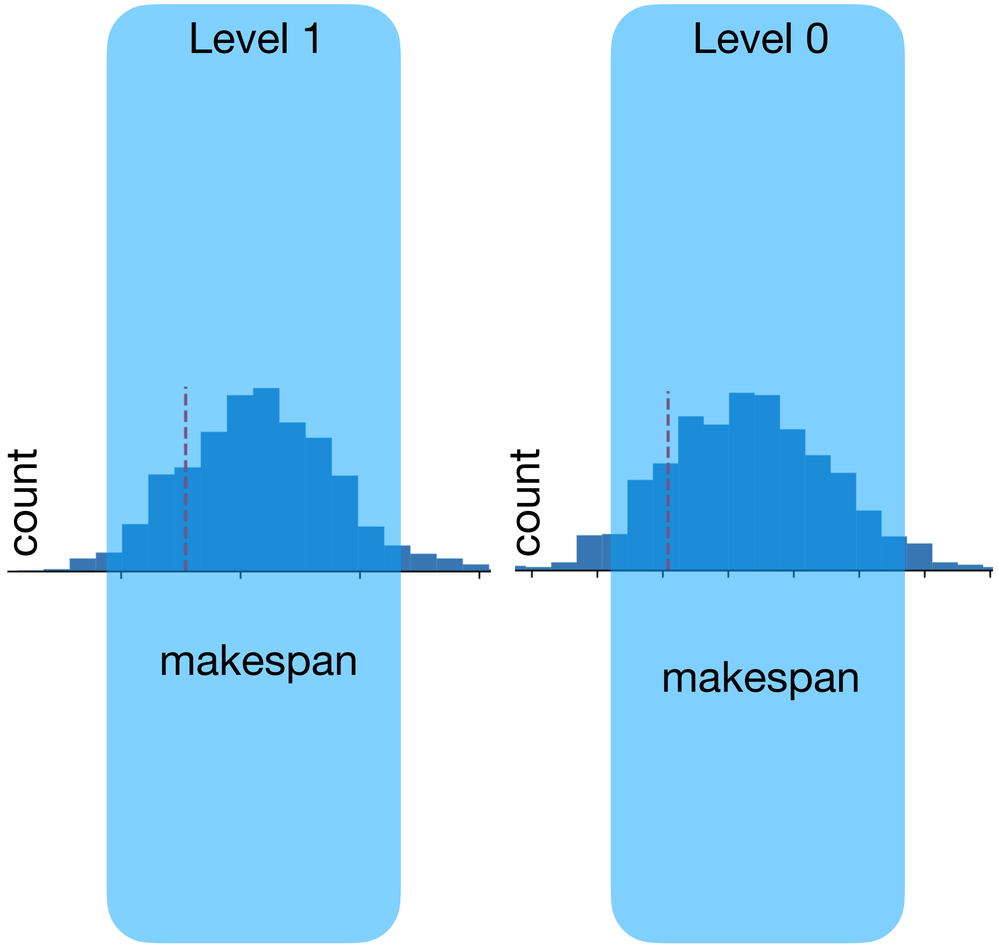


Exploitation and Exploration

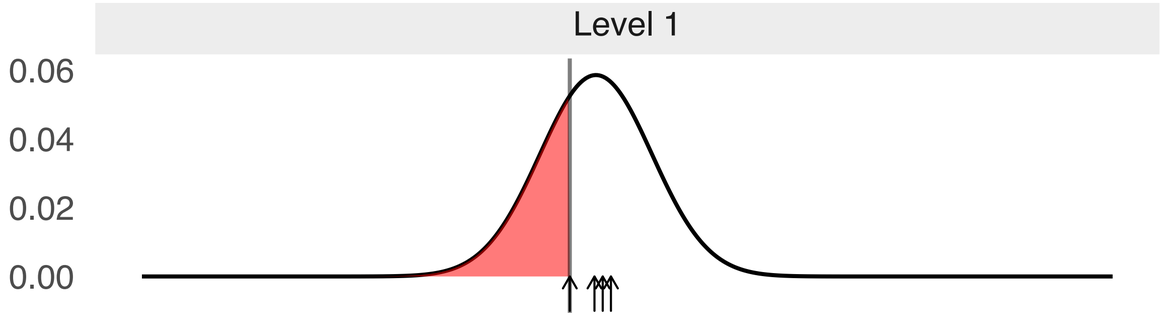
Time Budget
60 seconds



Estimating Improvement Probabilities

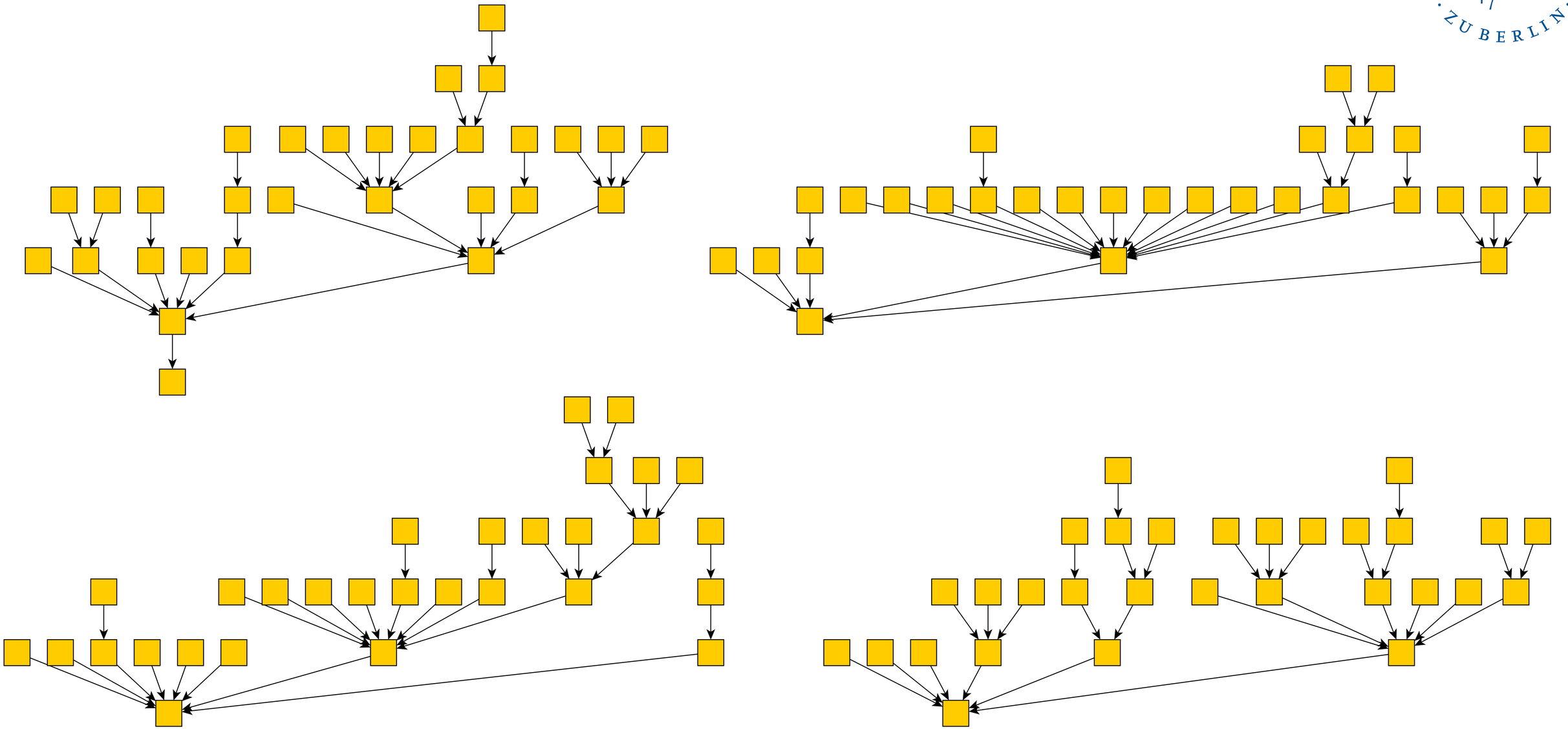


Level selection probability:
share of total improvement probabilities

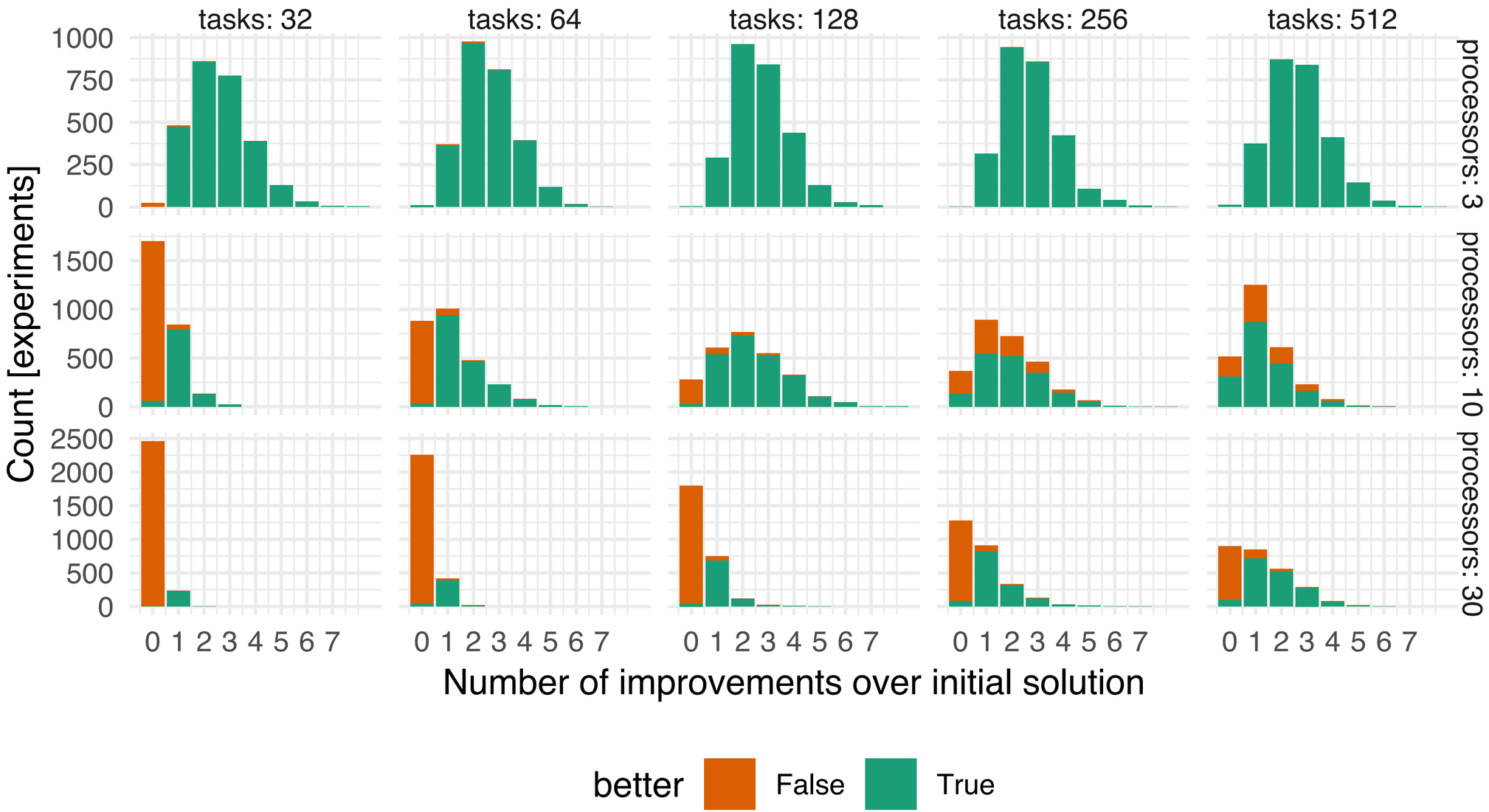


$$p_{\mathcal{N}}(m \leq r) = \frac{1}{2} F_{\mathcal{N}}(r; \mu, \sigma^2_{95\%})$$

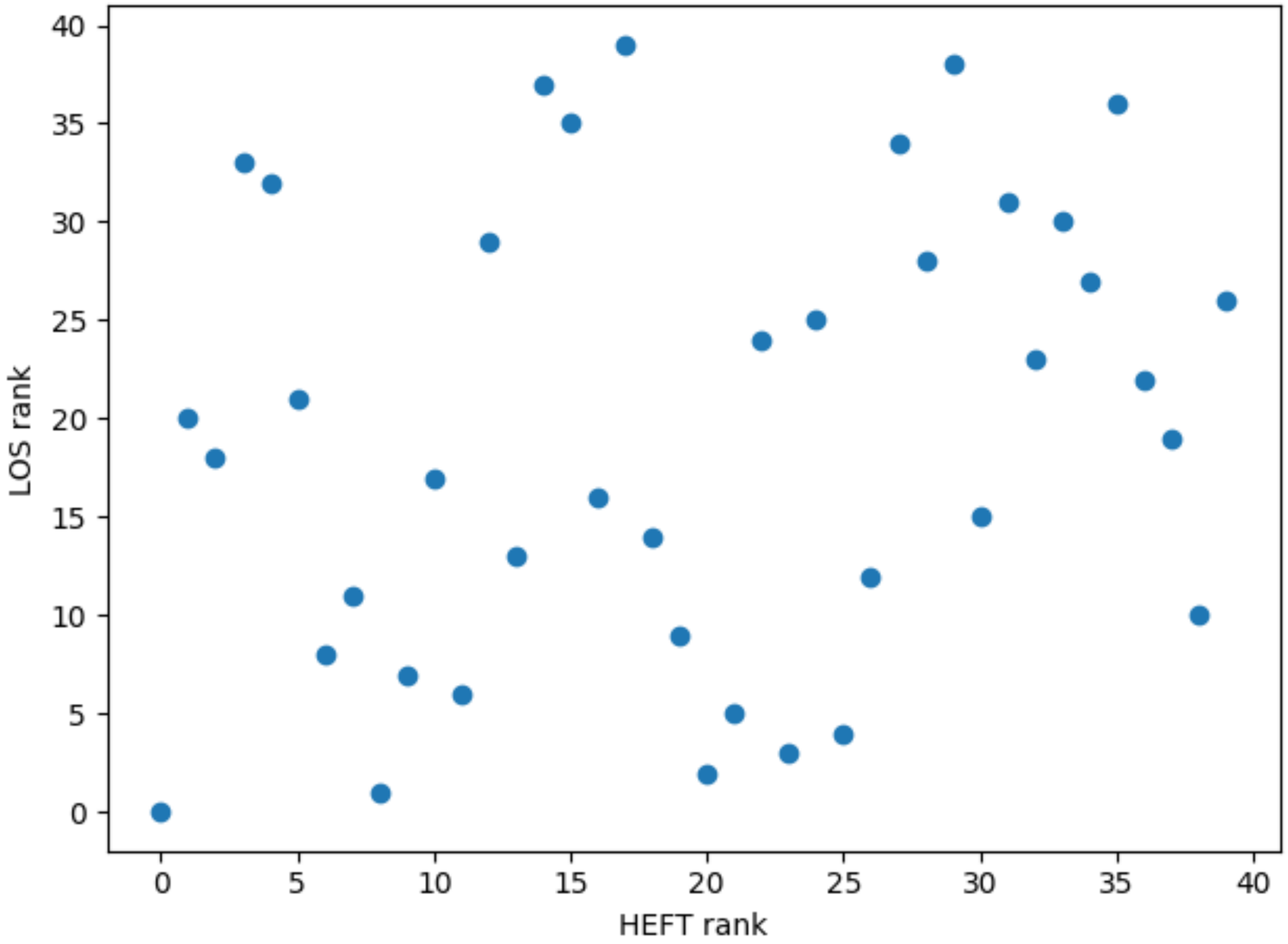
Krapivsky Random Graphs



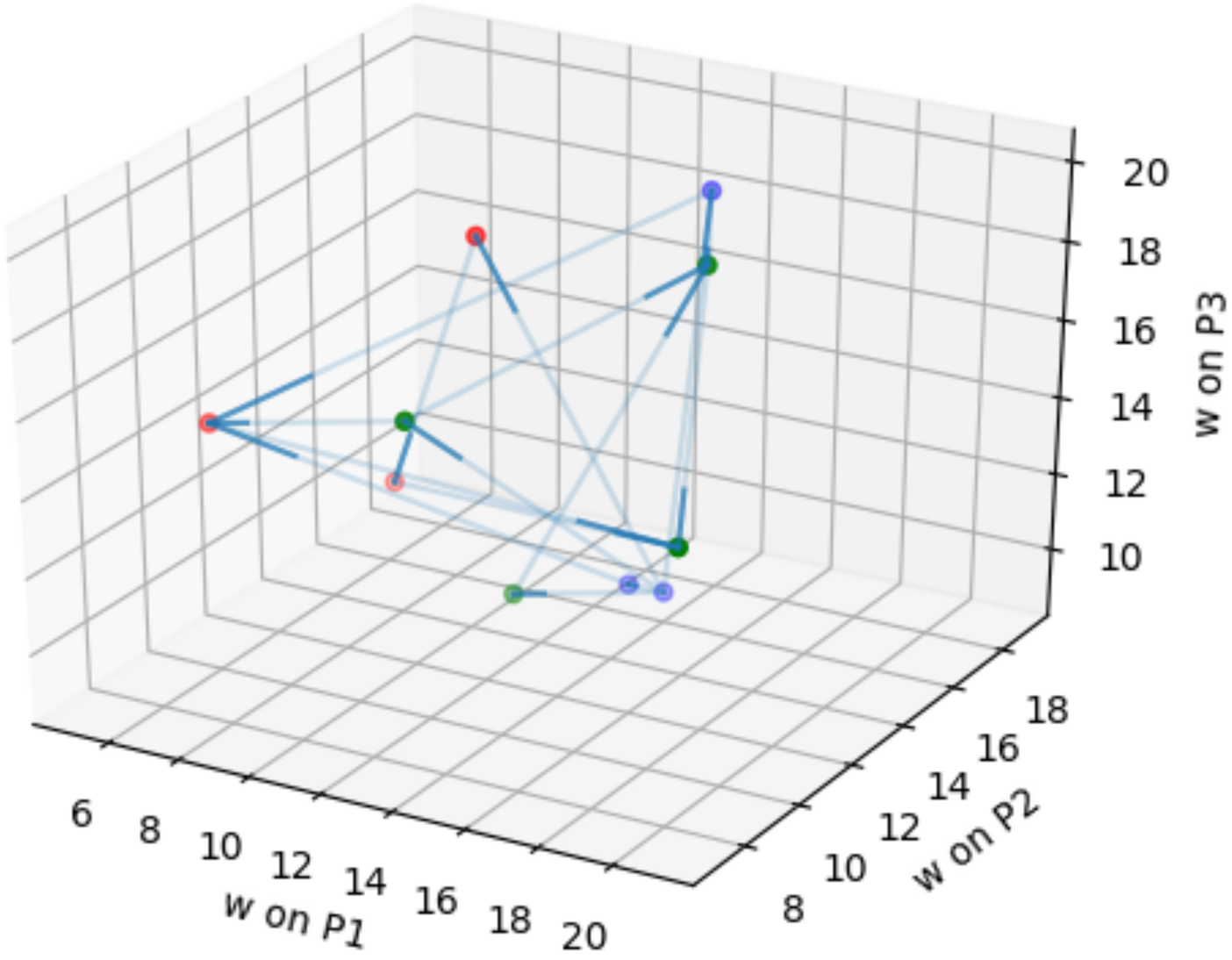
Results: Progress and Convergence



Sample Ranking Comparison



Schedule in Runtime Space



Experimental Evaluation

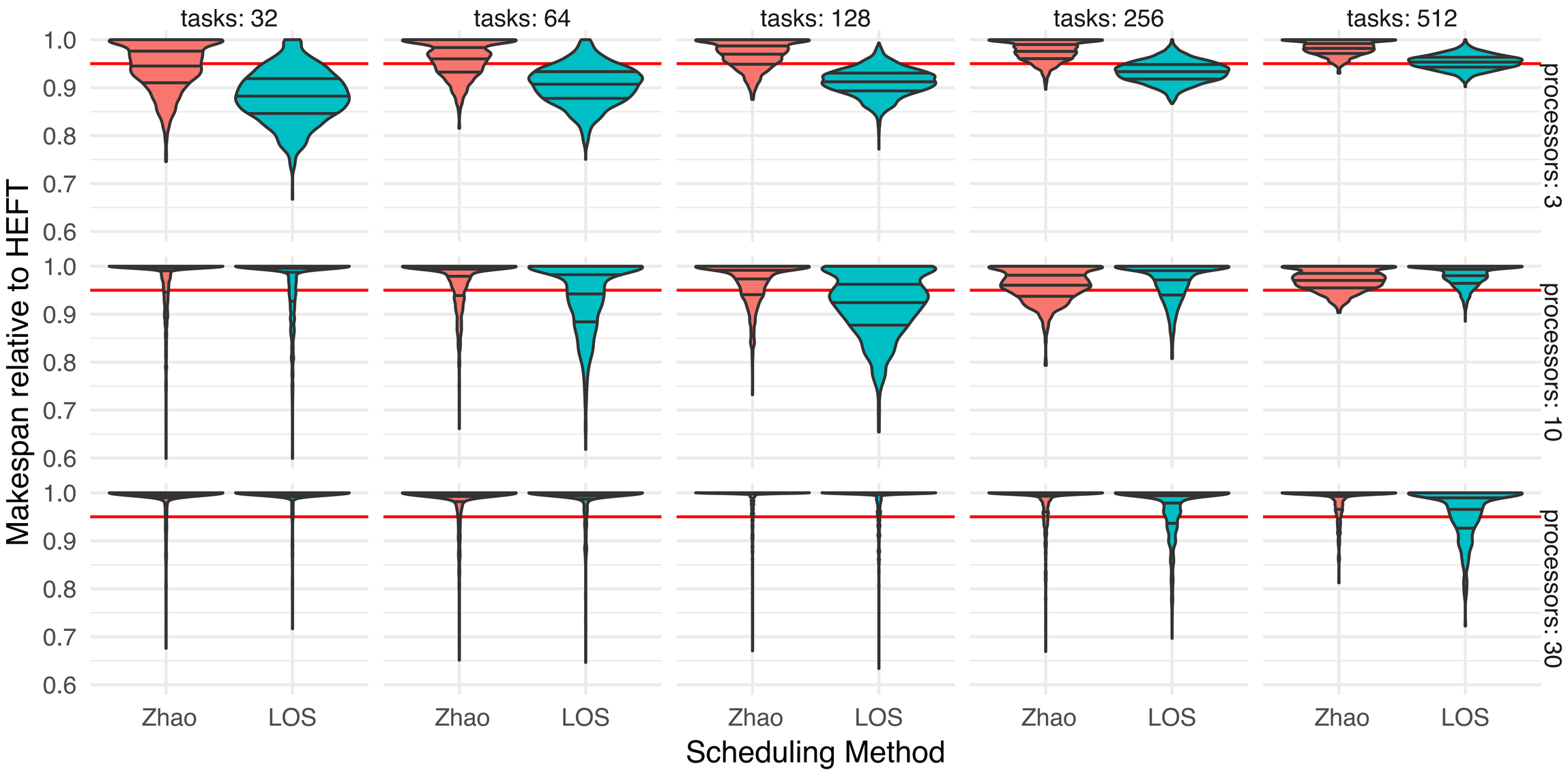
Parameters

- Number of tasks
  $N \in \{2^k \mid 5 \leq k \leq 9\}$
- Number of processors
  $P \in \{3,10,30\}$
- Workflows: 4500
- Time budget: 30s/5m
- Repetitions: 3
- Baselines
  - HEFT
  - Zhao et al.

	N	P	Seed	dag id	Sched	Makespan
1	32	3	1	1	LOS	1,171
2	32	3	2	1	LOS	1,081
3	32	3	3	1	LOS	1,048
4	32	3	-	1	Zhao	1,022
5	32	3	-	1	HEFT	1,180
...						
40.5K	512	30	-	4500	HEFT	1,011

Paul L. Krapivsky and Sidney Redner: "Organization of growing random networks." *Physical Review E* 63.6 (2001): 066123.

Results: Schedule Quality



Summary

LOS

- time-budgeted static scheduler
- randomization of task ranks
- adaptively explores search space
- 5%-40% shorter schedules than HEFT

Further directions

- granularity
- revisiting promising regions
- adaptive search vs. genetic algorithms

POS

Related Work for Chapter 4

Tsafrir: simple averaging models

Gaussier: no dependencies between tasks

Mitzenmacher: no parallelism, no dependencies

Thamsen: CPU utilization control loop, white box data

Ilyushkin: autoscaling in the cloud

Scheduling Heuristics

Scheduling Heuristics

Task Priority {HLF, LFF, FIFO, LIFO, SIL, SIF, SMF, SML, Random}

- Highest Level First (HLF): Prioritize tasks with many downstream tasks
- Least Finished First (LFF): Prioritize tasks with few training data points

Scheduling Heuristics

Task Priority {HLF, LFF, FIFO, LIFO, SIL, SIF, SMF, SML, Random}

- Highest Level First (HLF): Prioritize tasks with many downstream tasks
- Least Finished First (LFF): Prioritize tasks with few training data points

Failure Handling {Persevere, Postpone, None}

- Increase/decrease task priority after out-of-memory failure?

Scheduling Heuristics

Task Priority {HLF, LFF, FIFO, LIFO, SIL, SIF, SMF, SML, Random}

- Highest Level First (HLF): Prioritize tasks with many downstream tasks
- Least Finished First (LFF): Prioritize tasks with few training data points

Failure Handling {Persevere, Postpone, None}

- Increase/decrease task priority after out-of-memory failure?

Backfilling {1, 5, 15}

- Out-of-order execution of tasks

= 81 scheduling heuristics

Scheduling Heuristics

Dynamic scheduling

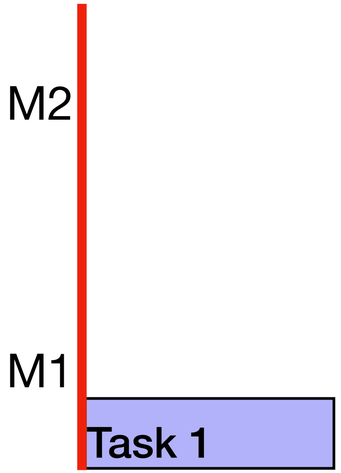
- Just in time decision making

Ready Task	Memory Estimate
Task 1	5GB
Task 2	11 GB
Task 3	7 GB
Task 4	15 GB
Task 5	15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

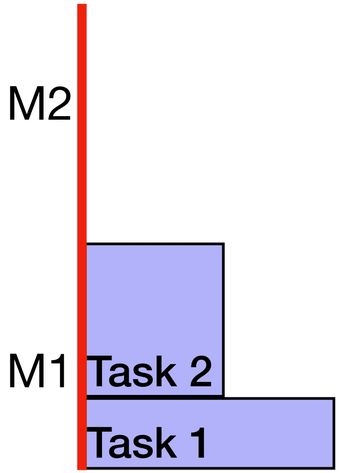


Ready Task	Memory Estimate
Task 1	5GB
Task 2	11 GB
Task 3	7 GB
Task 4	15 GB
Task 5	15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

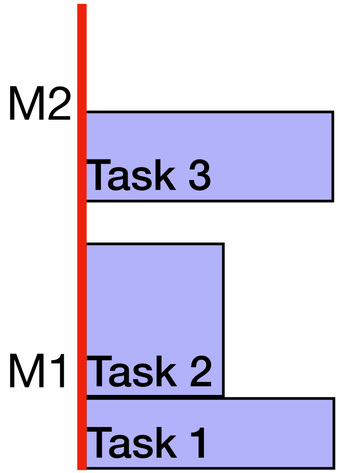


Ready Task	Memory Estimate
Task 1	5GB
Task 2	11 GB
Task 3	7 GB
Task 4	15 GB
Task 5	15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

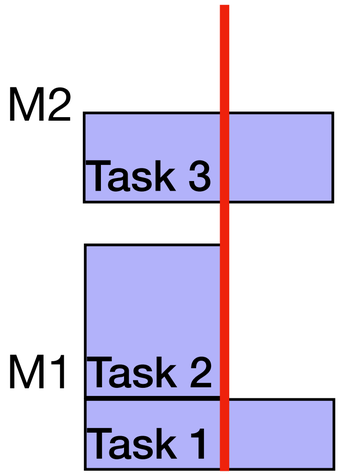


Ready Task	Memory Estimate
Task 1	5GB
Task 2	11 GB
Task 3	7 GB
Task 4	15 GB
Task 5	15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

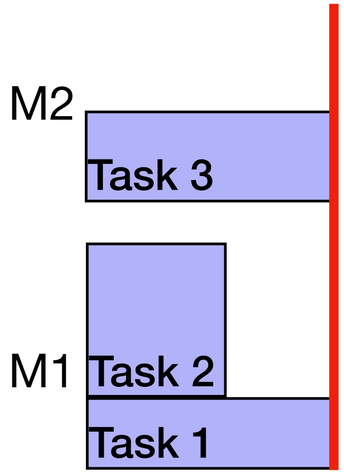


Ready Task		Memory Estimate
Task 1	⏱️	5GB
Task 2	✔️	11 GB
Task 3	⏱️	7 GB
Task 4		15 GB
Task 5		15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

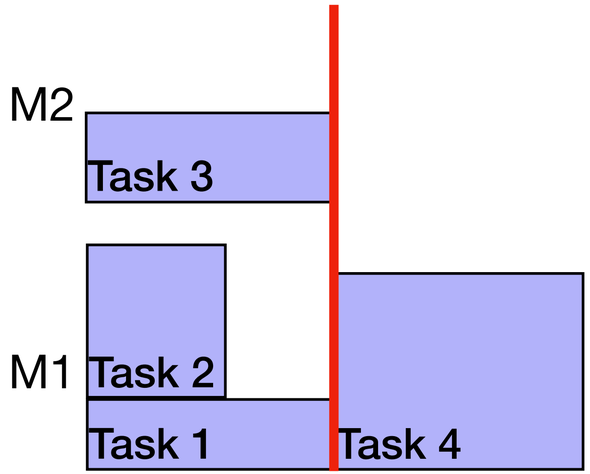


Ready Task		Memory Estimate
Task 1	✔	5GB
Task 2	✔	11 GB
Task 3	✔	7 GB
Task 4		15 GB
Task 5		15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

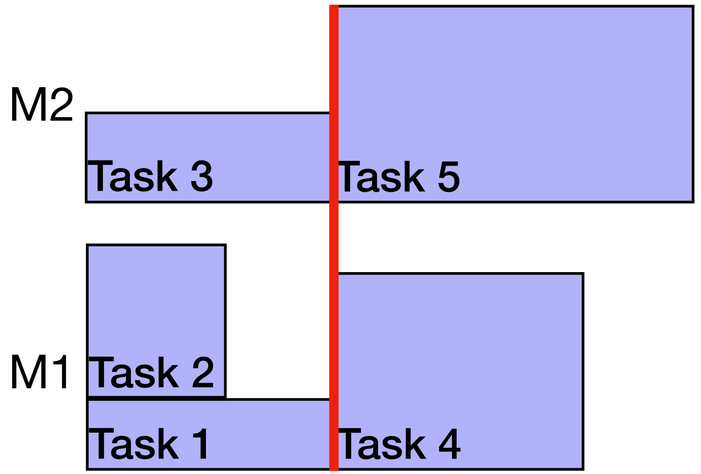


Ready Task		Memory Estimate
Task 1	✔	5GB
Task 2	✔	11 GB
Task 3	✔	7 GB
Task 4		15 GB
Task 5		15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making

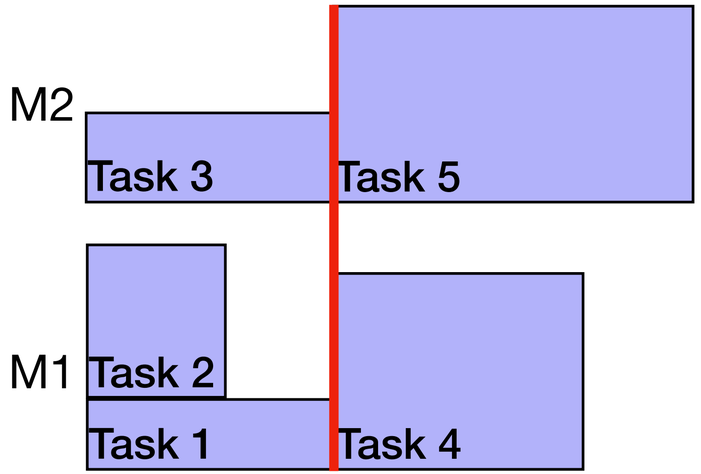


Ready Task		Memory Estimate
Task 1	✔	5GB
Task 2	✔	11 GB
Task 3	✔	7 GB
Task 4		15 GB
Task 5		15 GB

Scheduling Heuristics

Dynamic scheduling

- Just in time decision making
- Task priorities

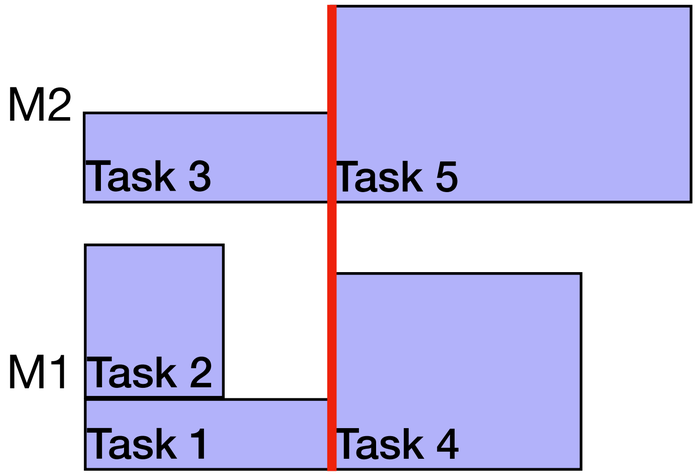


Ready Task		Memory Estimate
Task 1	✔	5GB
Task 2	✔	11 GB
Task 3	✔	7 GB
Task 4		15 GB
Task 5		15 GB

Scheduling Heuristics

Dynamic scheduling

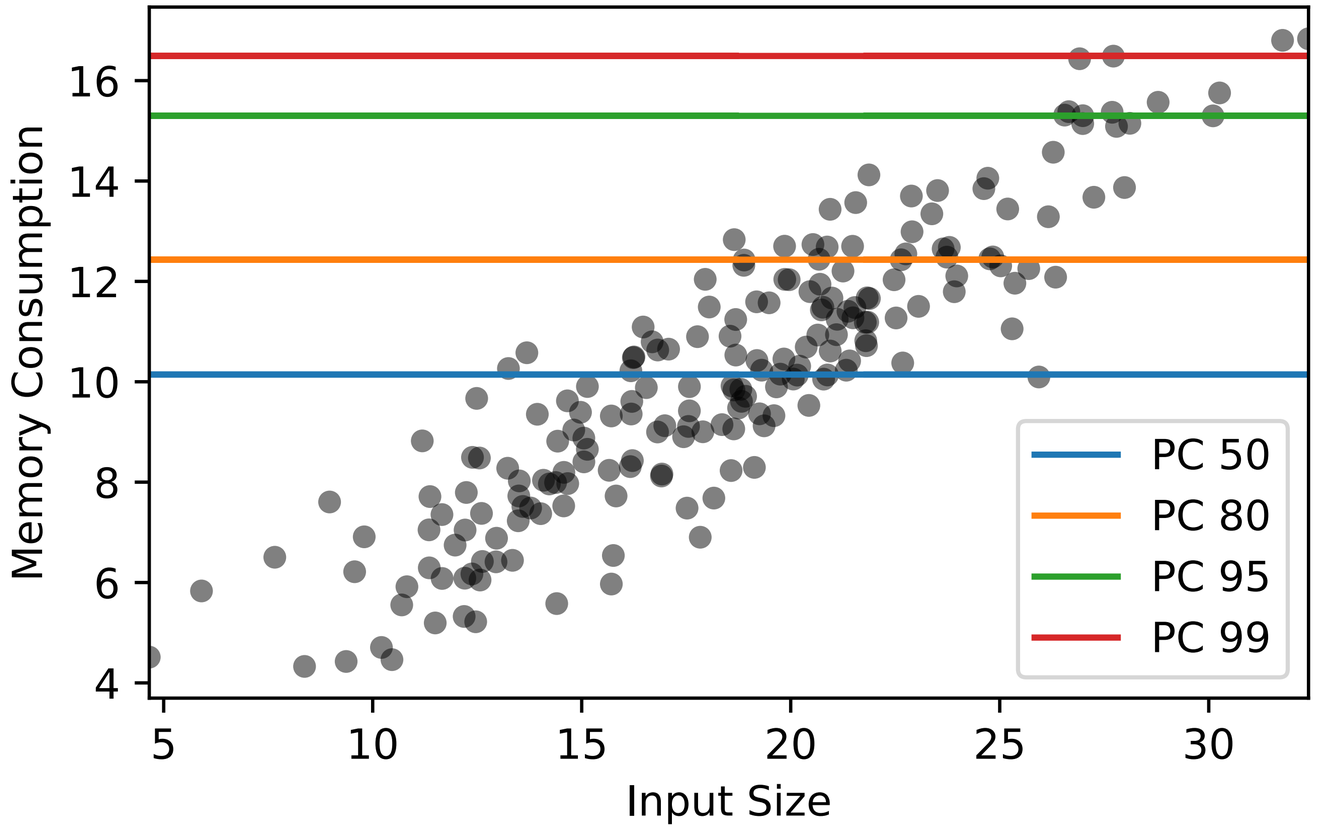
- Just in time decision making
- Task priorities
- Failure handling



Ready Task		Memory Estimate
Task 1	✔	5GB
Task 2	✔	11 GB
Task 3	✔	7 GB
Task 4		15 GB
Task 5		15 GB

Prediction Models

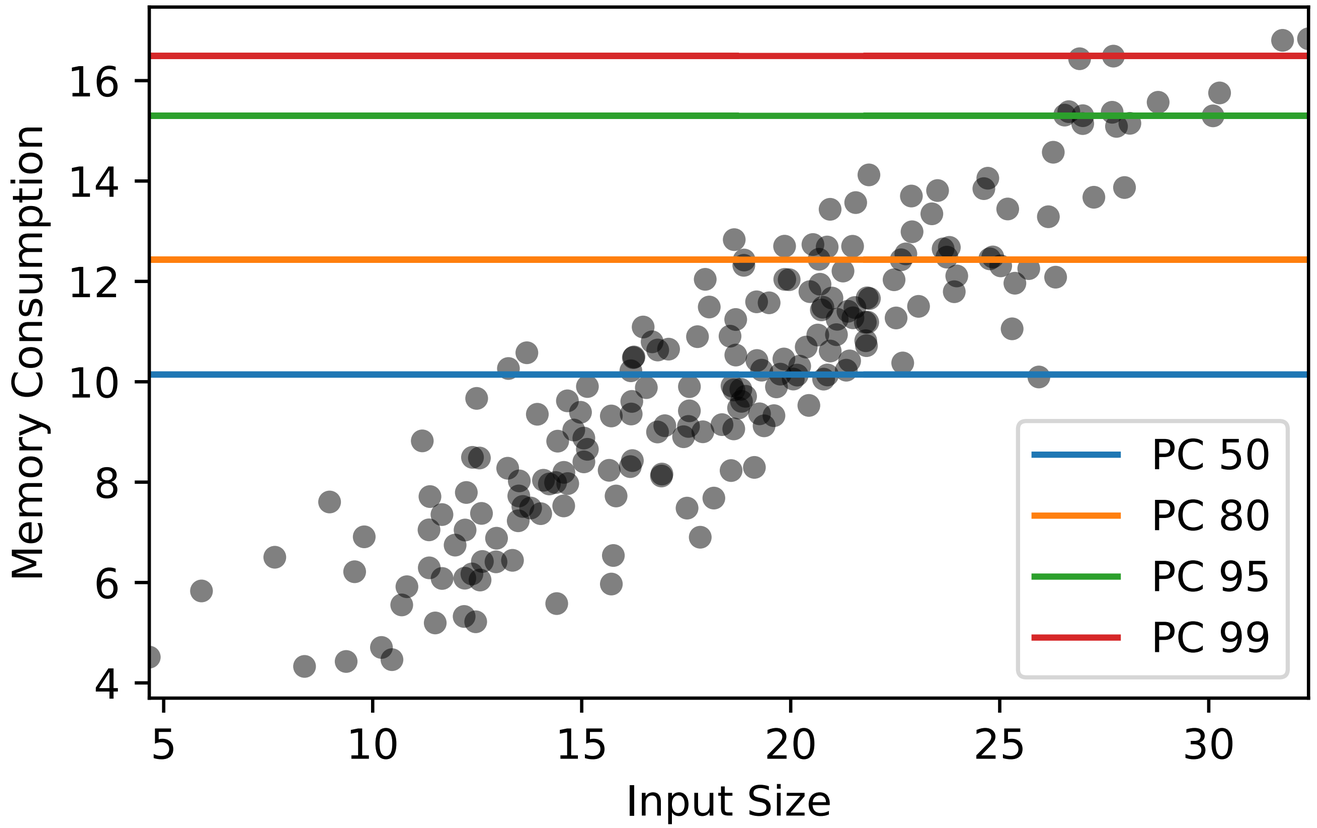
Simple online prediction models



Prediction Models

Simple online prediction models

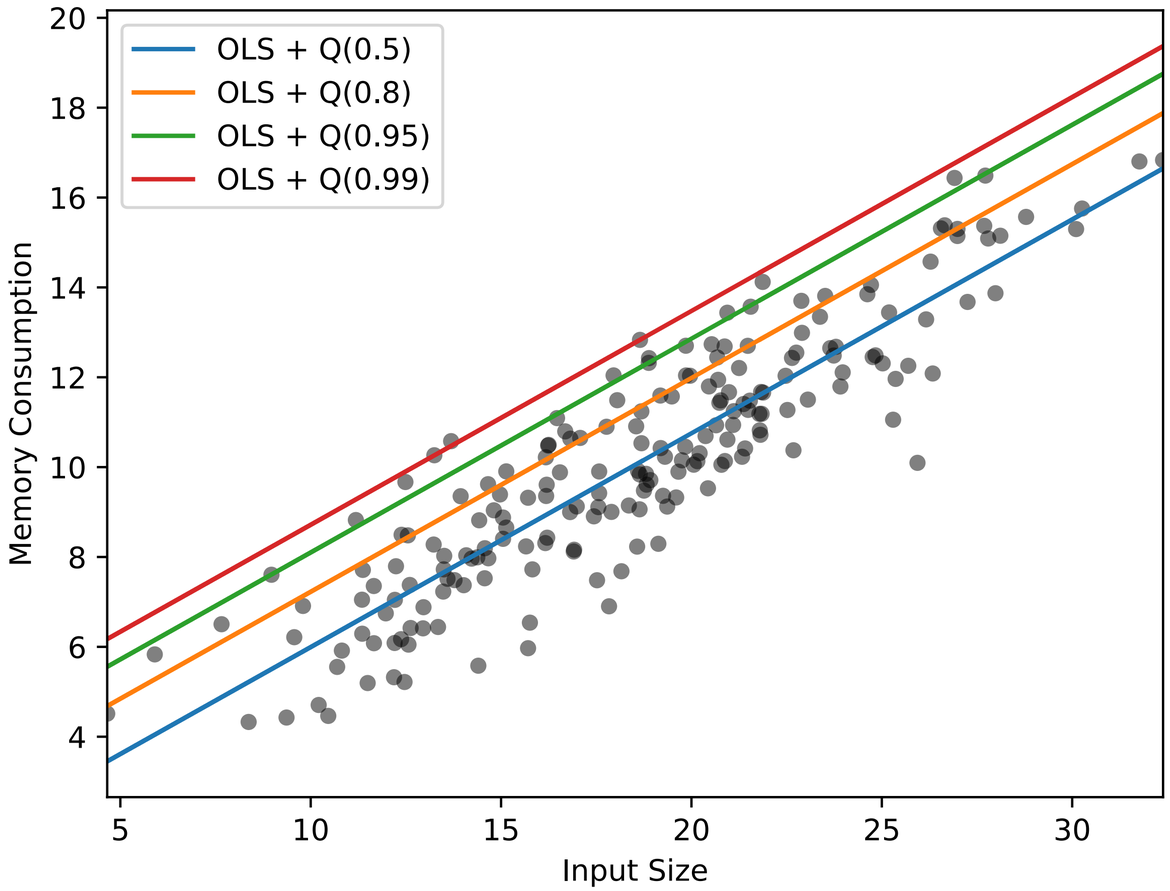
- PC: Percentile of observed usage



Prediction Models

Simple online prediction models

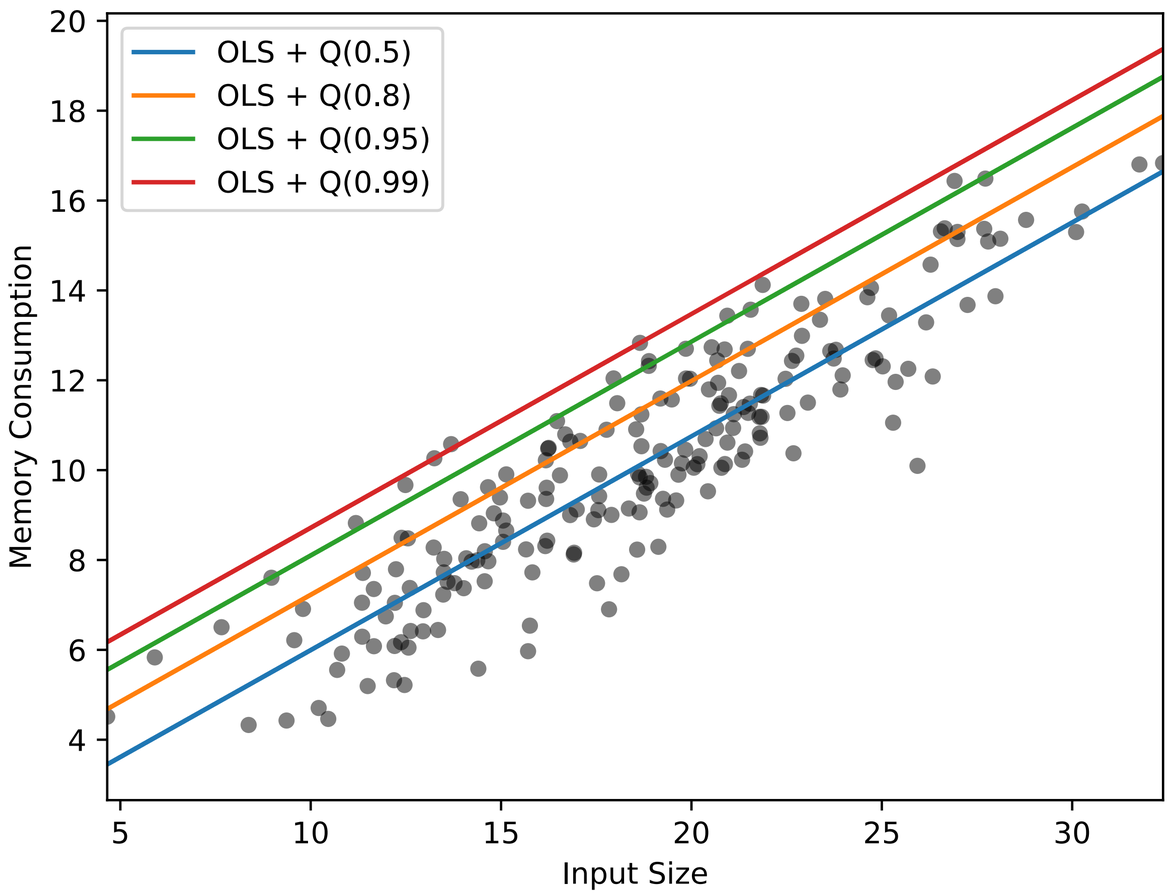
- PC: Percentile of observed usage
- LR: Conservative linear regression



Prediction Models

Simple online prediction models

- PC: Percentile of observed usage
- LR: Conservative linear regression
- Fixed percentiles (0.5, …, 100)

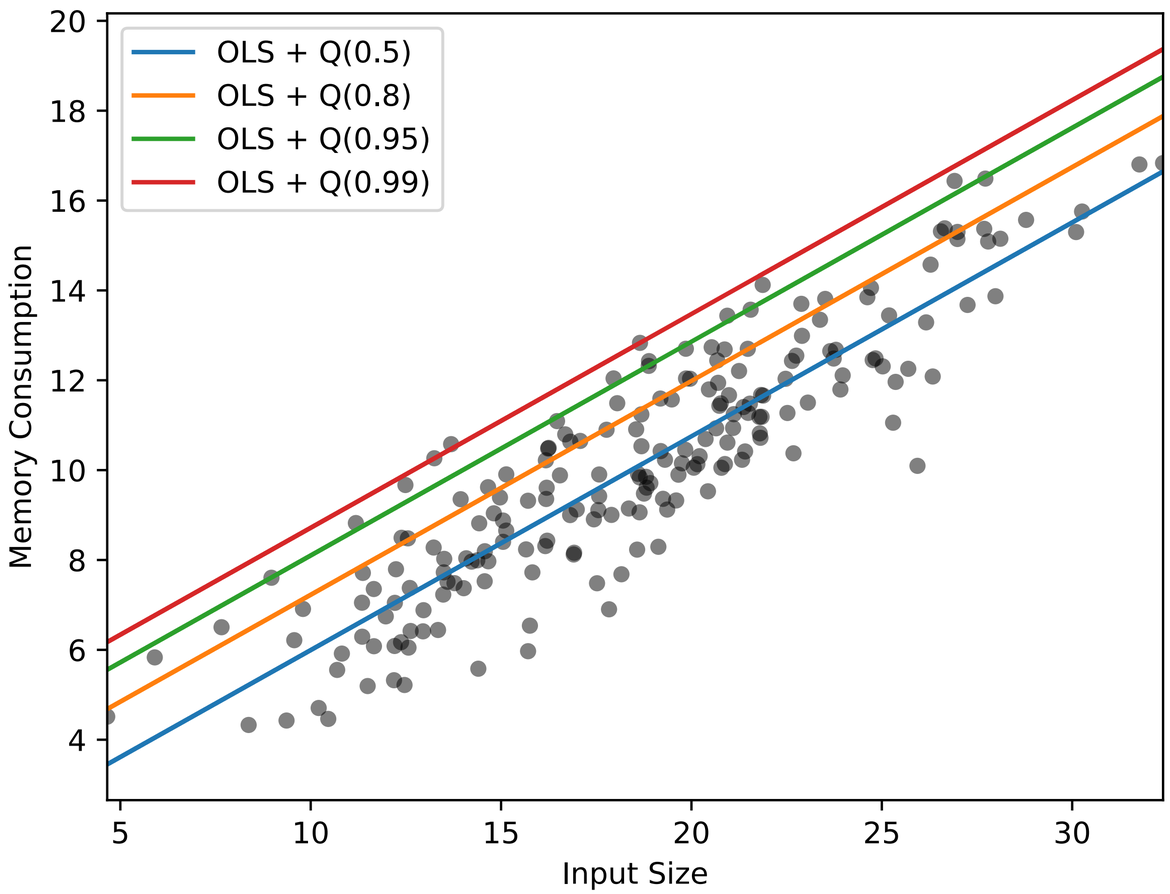


Prediction Models

Simple online prediction models

- PC: Percentile of observed usage
- LR: Conservative linear regression
- Fixed percentiles (0.5, …, 100)

Hypothetical user estimates



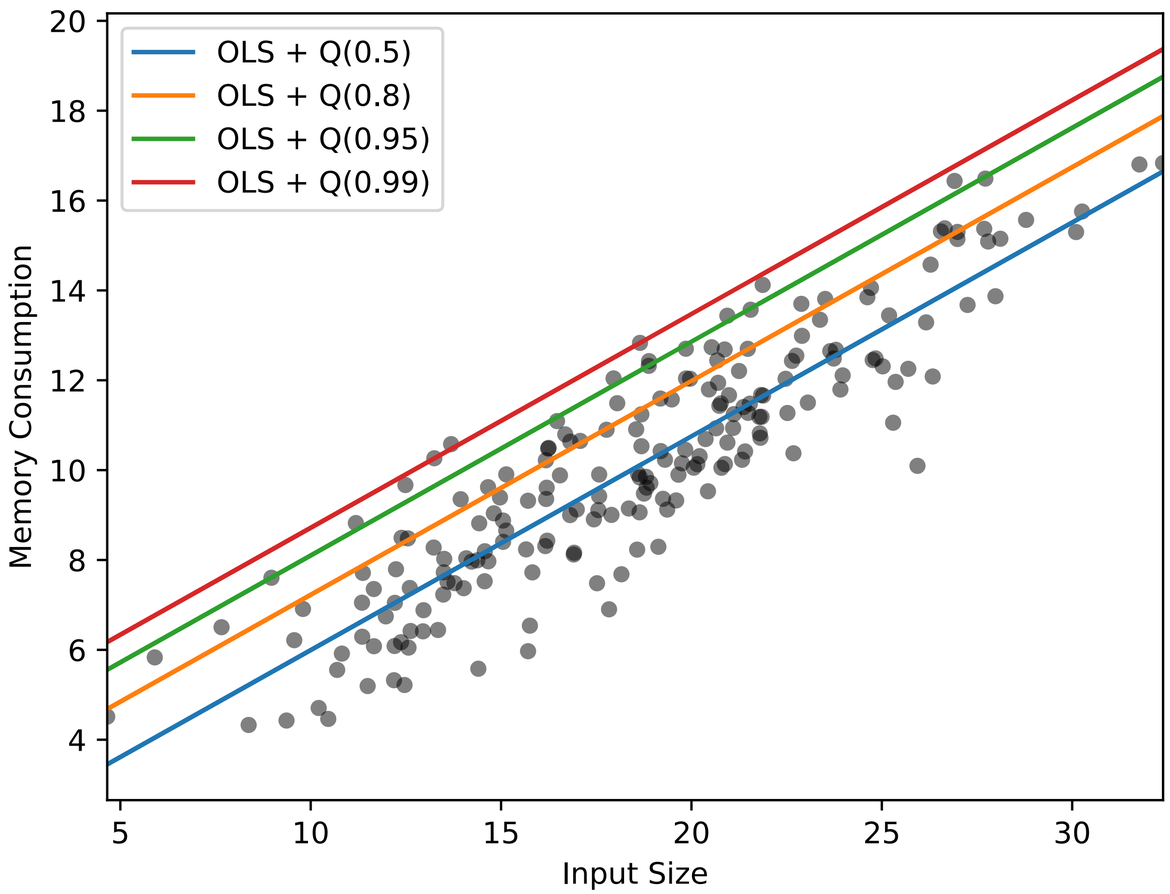
Prediction Models

Simple online prediction models

- PC: Percentile of observed usage
- LR: Conservative linear regression
- Fixed percentiles (0.5, …, 100)

Hypothetical user estimates

- Per abstract task



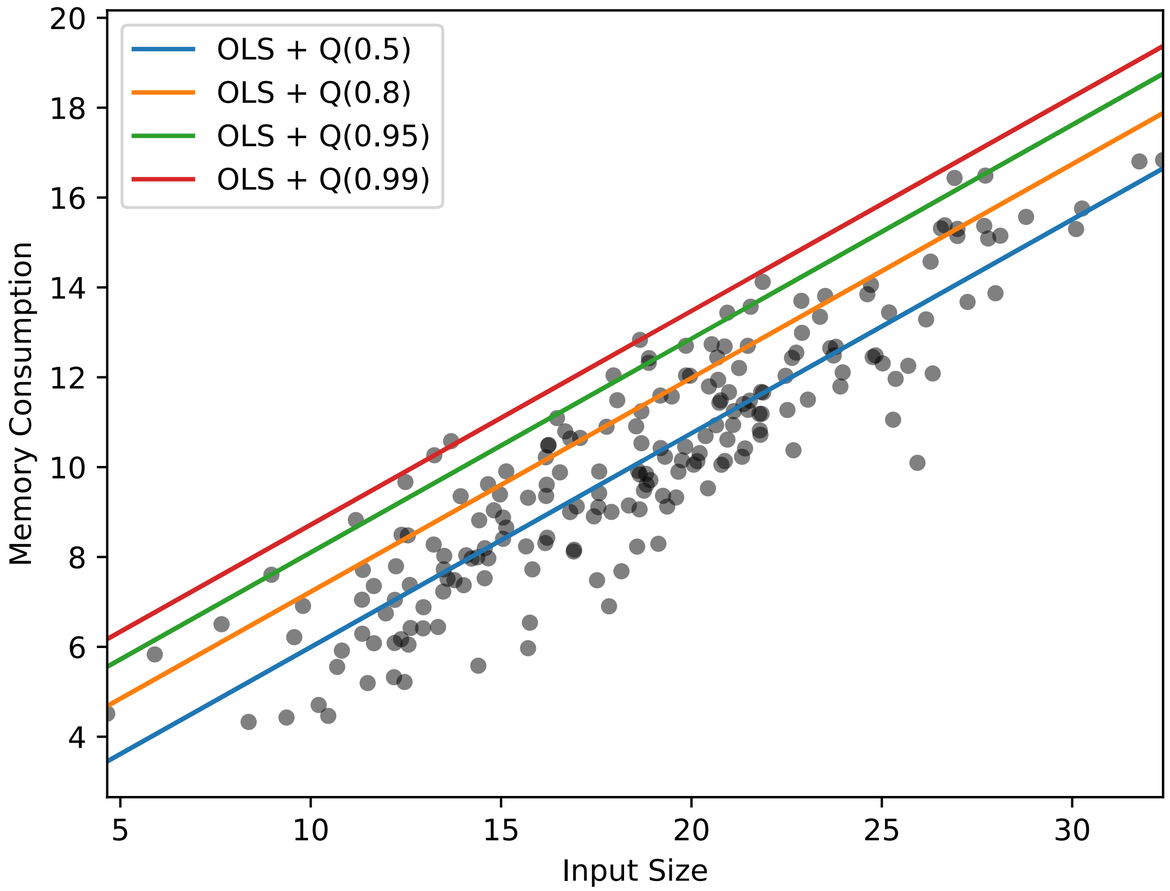
Prediction Models

Simple online prediction models

- PC: Percentile of observed usage
- LR: Conservative linear regression
- Fixed percentiles (0.5, …, 100)

Hypothetical user estimates

- Per abstract task
- Q99: 99th percentile of true memory usages



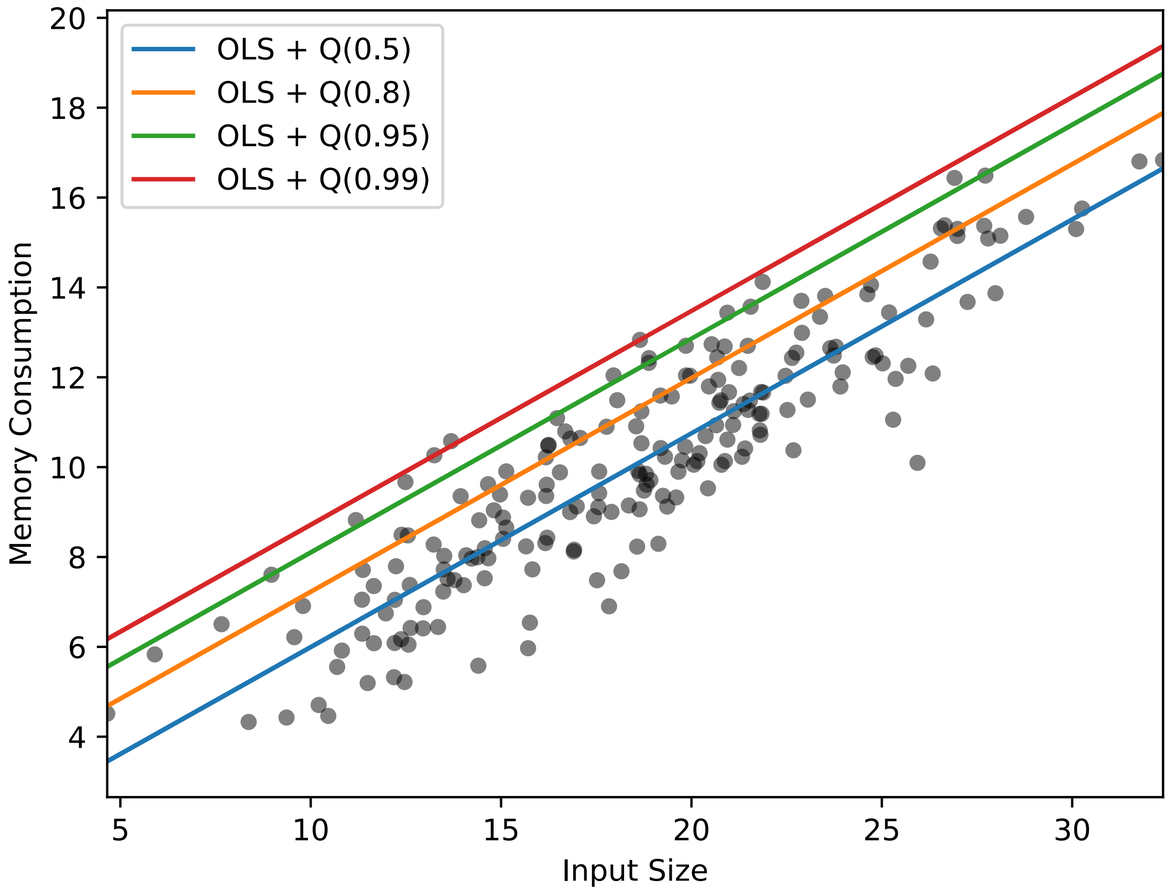
Prediction Models

Simple online prediction models

- PC: Percentile of observed usage
- LR: Conservative linear regression
- Fixed percentiles (0.5, …, 100)

Hypothetical user estimates

- Per abstract task
- Q99: 99th percentile of true memory usages
- Power 2: Round maximum usage to nearest power of 2

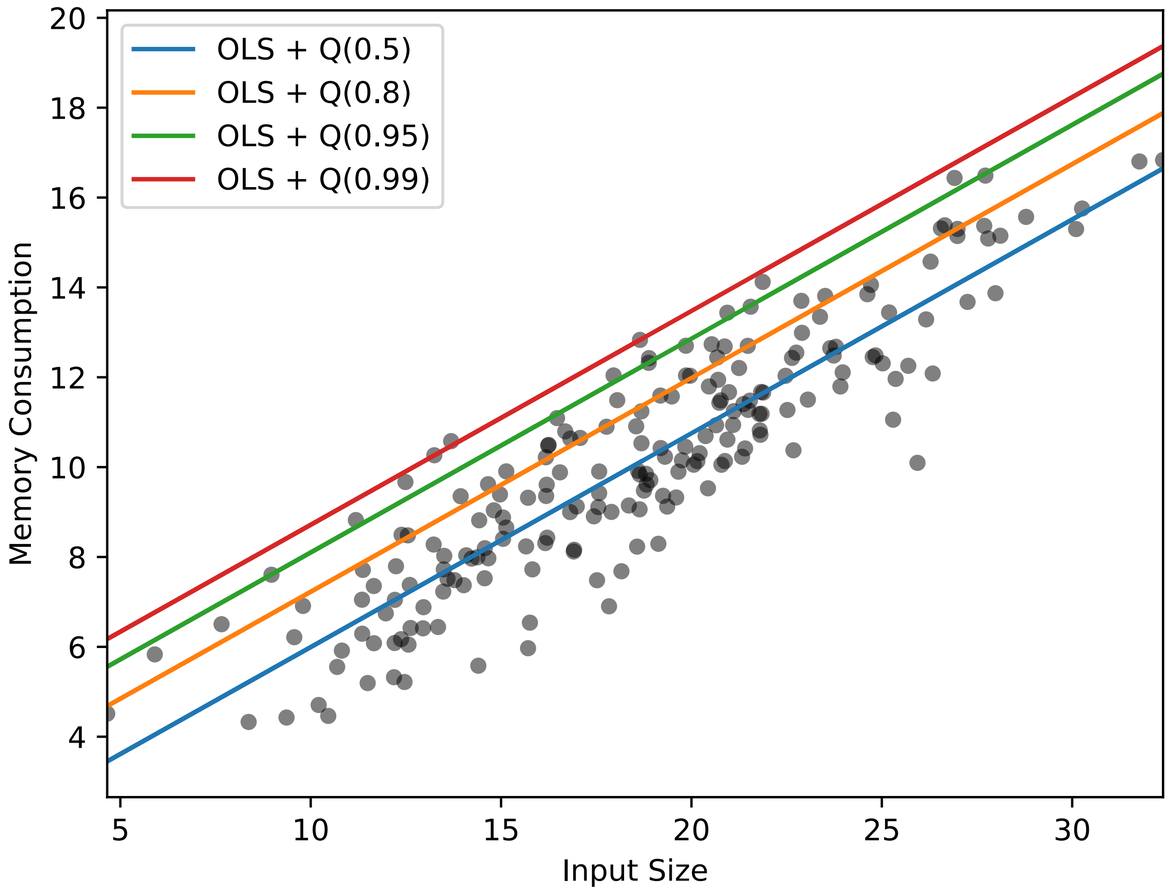


Prediction Models

Simple online prediction models
- PC: Percentile of observed usage
- LR: Conservative linear regression
- Fixed percentiles (0.5, …, 100)

Hypothetical user estimates
- Per abstract task
- Q99: 99th percentile of true memory usages
- Power 2: Round maximum usage to nearest power of 2

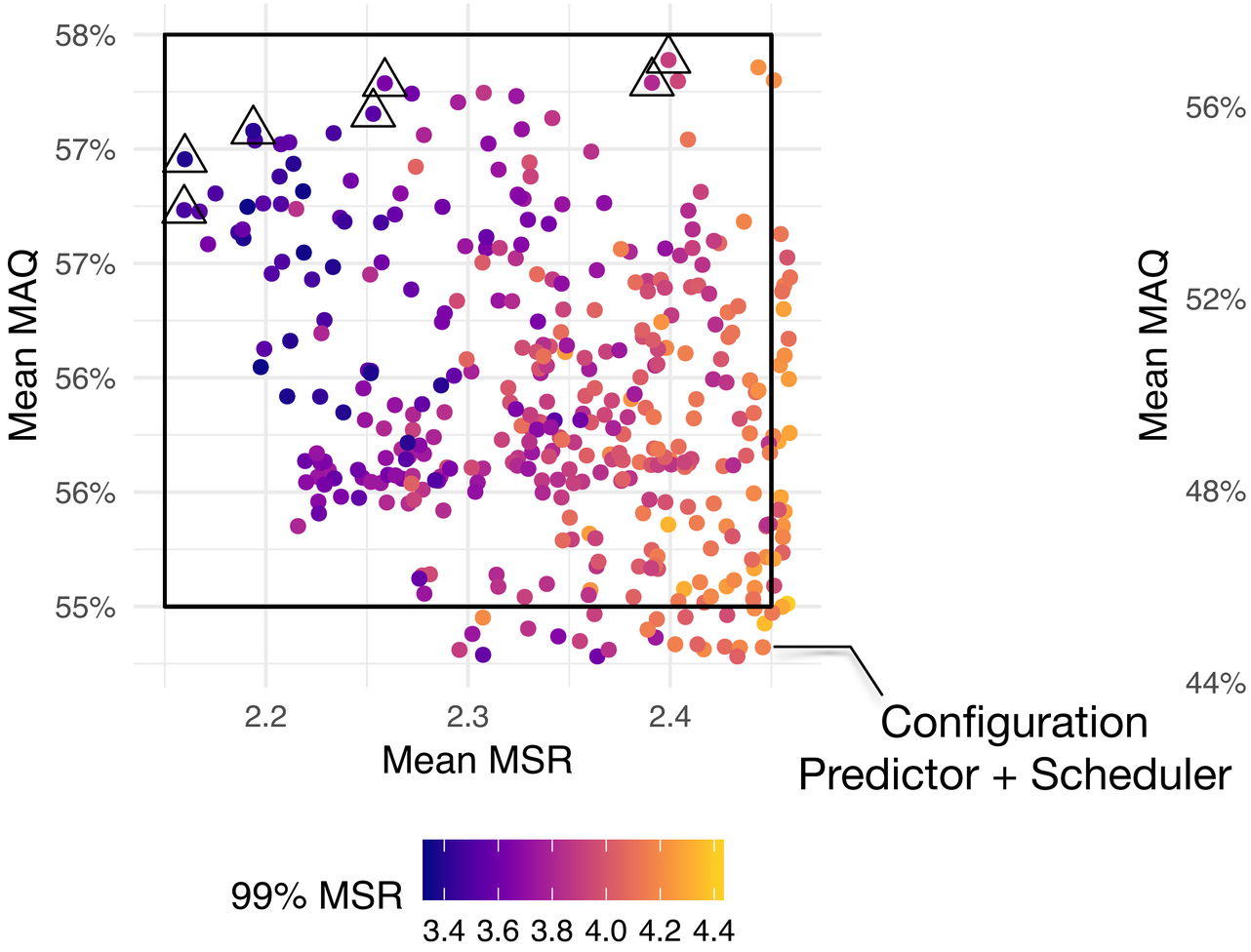
Total: 14 prediction model (2*6+2)

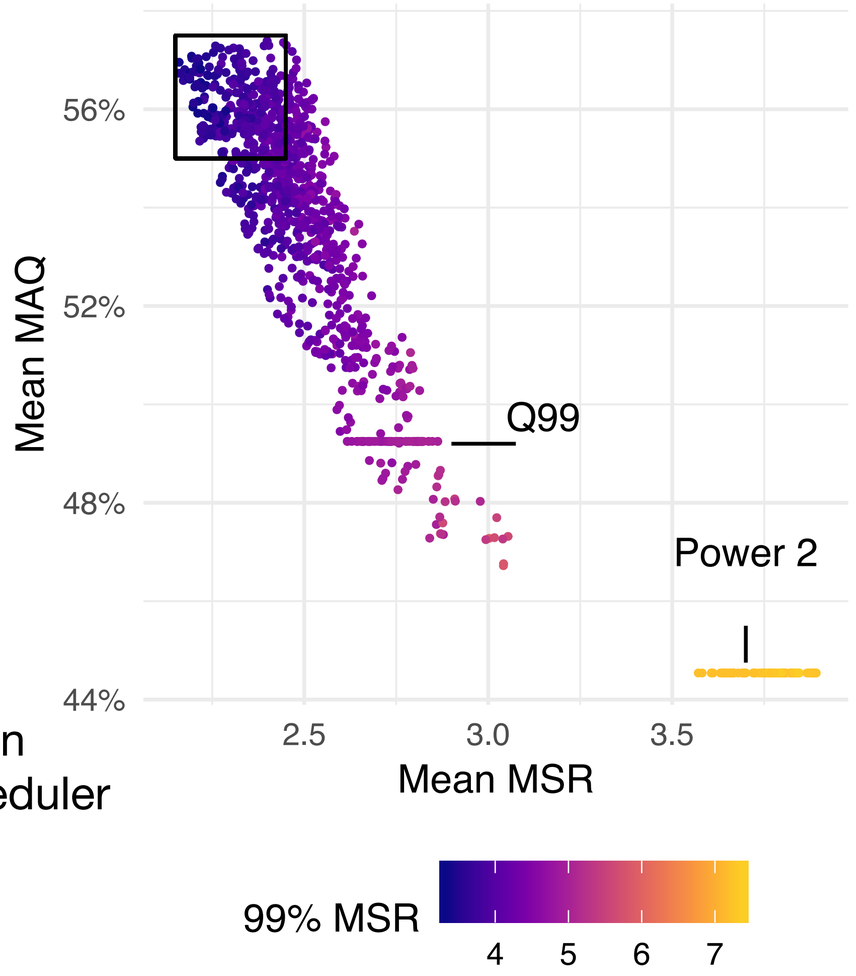


Scheduling Heuristics

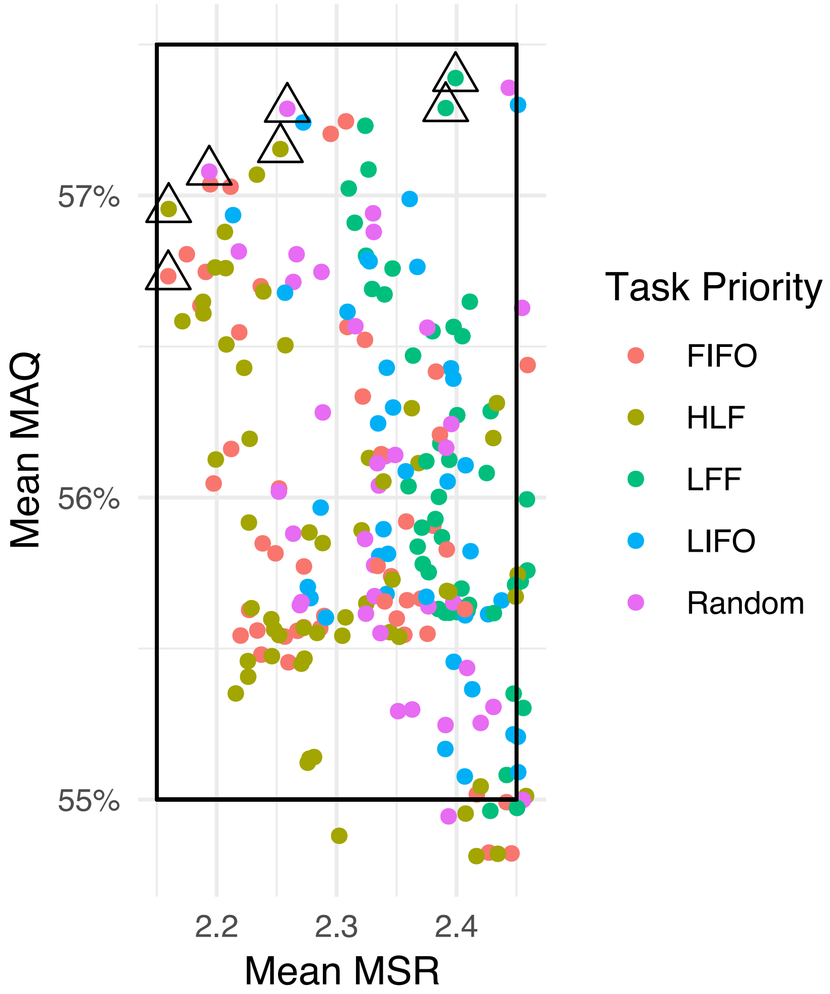
First Priority	Failure Handling	Name
Longest path to exit	• **Most failures first** • **Least failures first**	**TR** **Persevere/Postpone**
Random	• **Extra queue for non-first attempts** • **Single queue**	**Random** **Persevere/None**
Tasks that has been ready for longest time	• Keep first attempt's arrival time • Use current time	FIFO Persevere/Postpone
Task that has been ready for shortest time	None	LIFO
Abstract task with least finished tasks (few training observations)	• Most failures first • Least failures first	LFF Persevere/Postpone
Biggest/smallest predicted memory first	None	BMF/SMF
Biggest/smallest input file size first	None	BFF/SFF

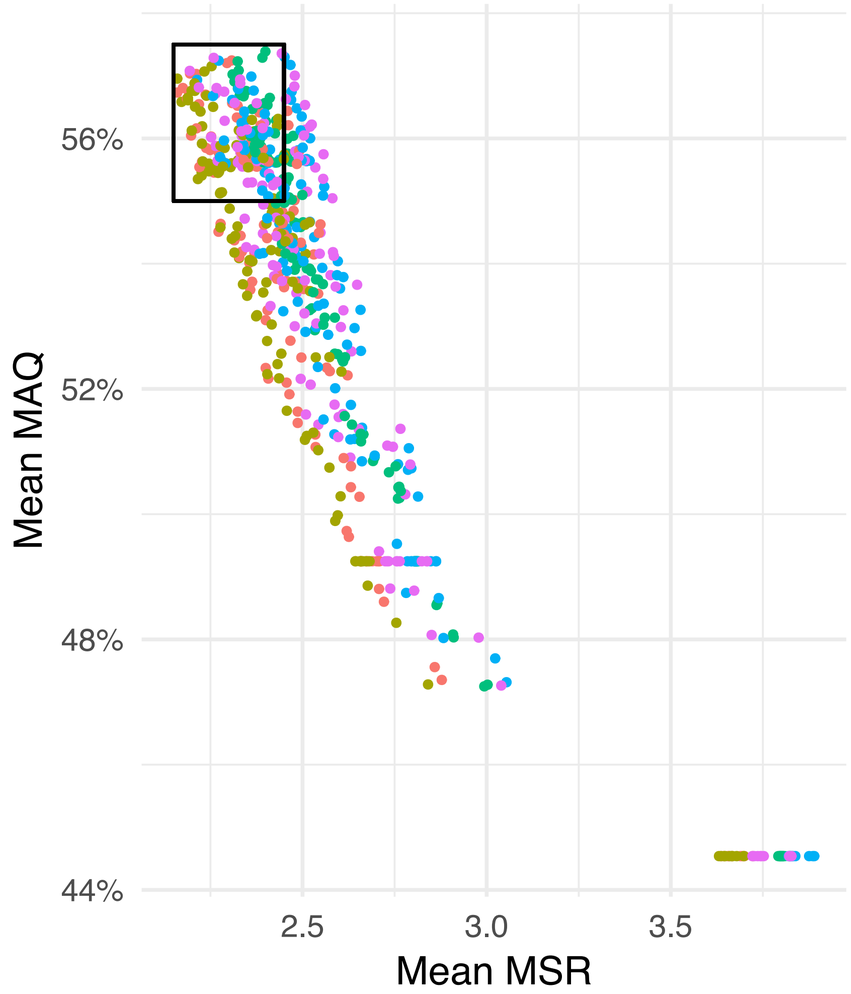
Average Configuration Performance



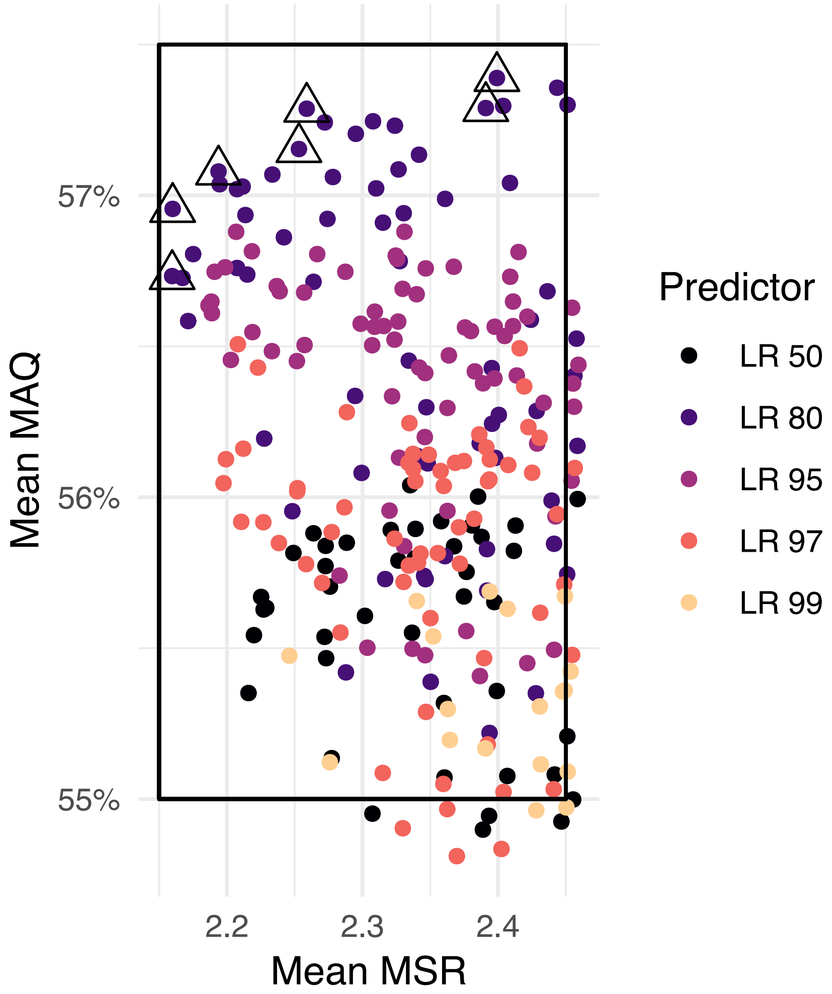


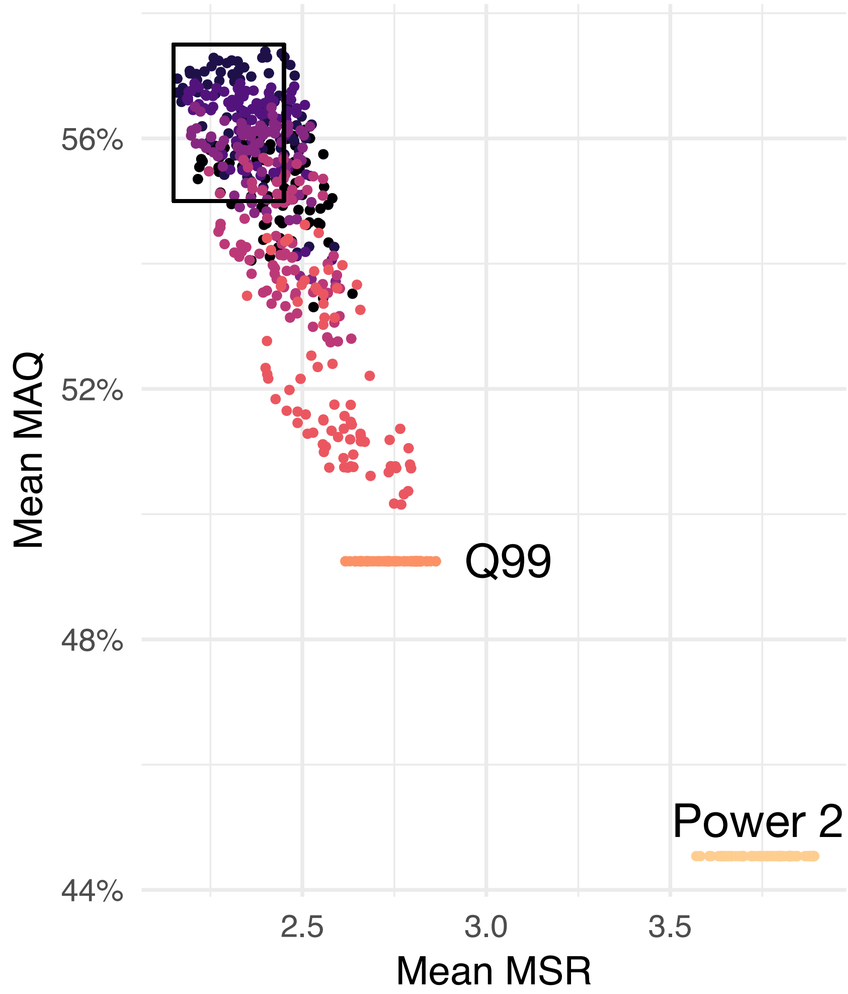
Average Scheduler Performance



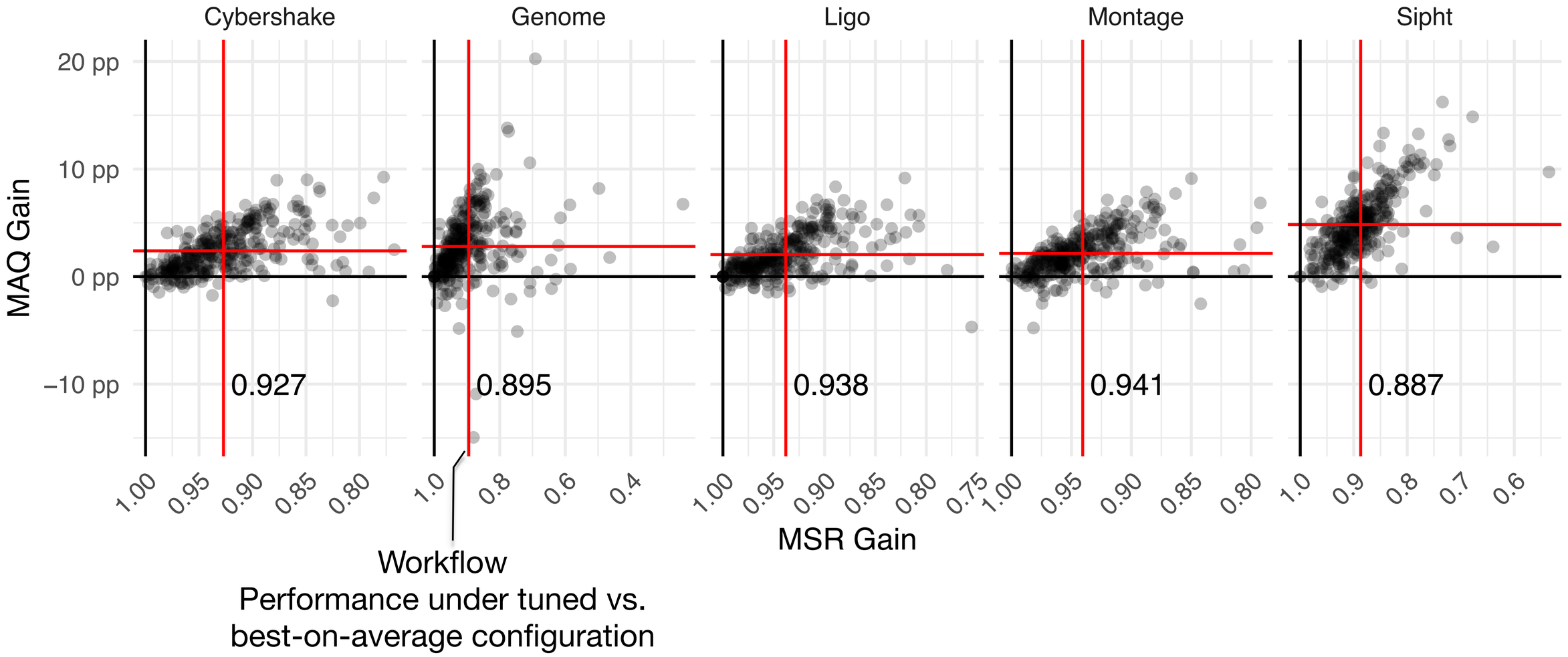


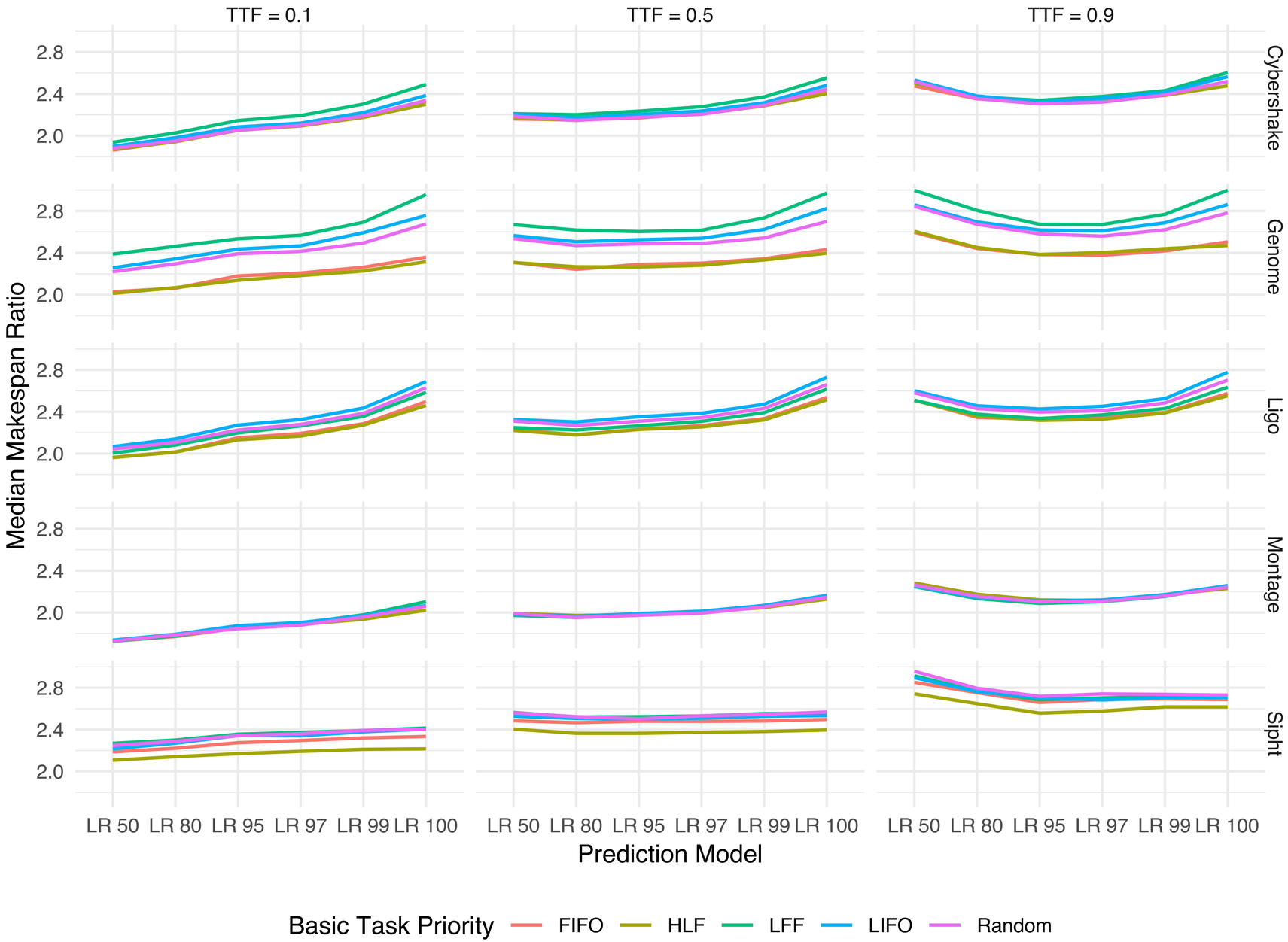
Average Predictor Performance



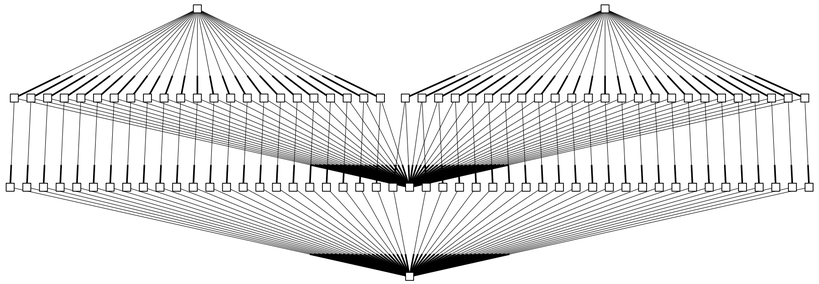


Workflows: Tuning Potential

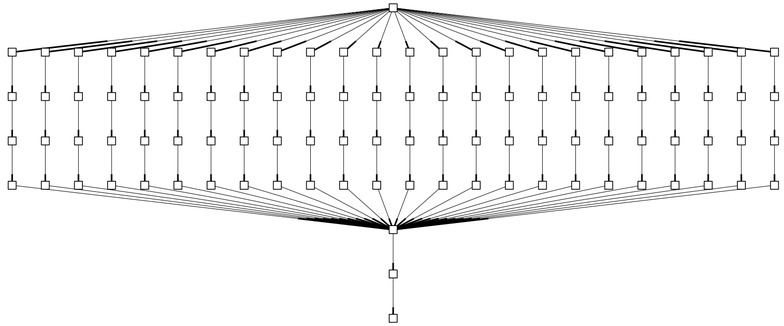



TTF = 0.1
TTF = 0.5
TTF = 0.9
Cybershake
Genome
Ligo
Montage
Sipht
Median Makespan Ratio
2.8
2.4
2.0
LR 50
LR 80
LR 95
LR 97
LR 99
LR 100
Prediction Model
Basic Task Priority
FIFO
HLF
LFF
LIFO
Random

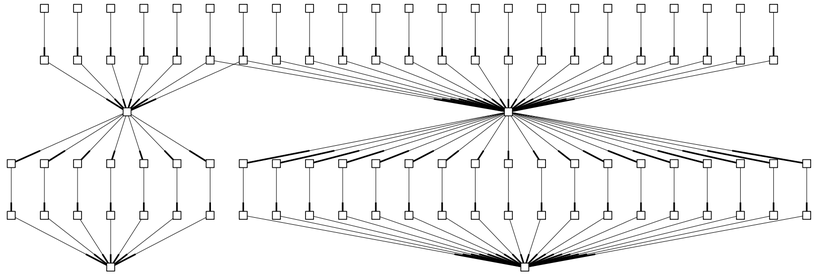
CyberShake



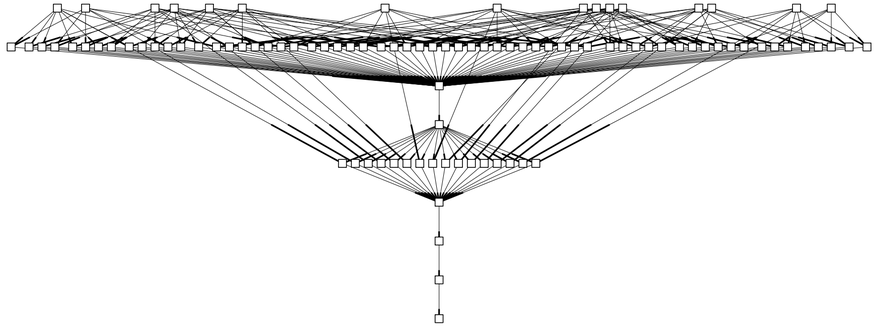
Epigenomics



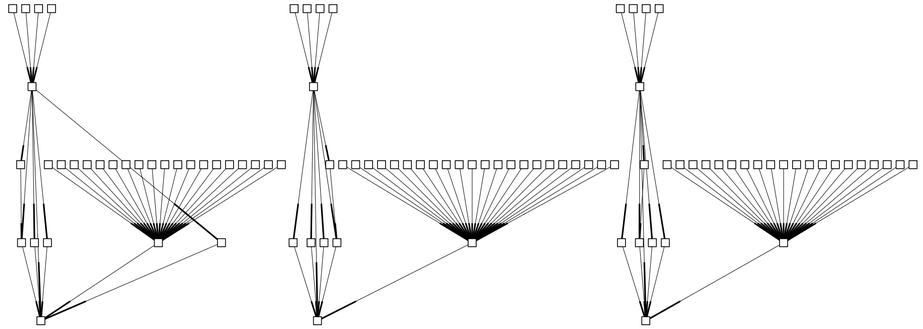
LIGO



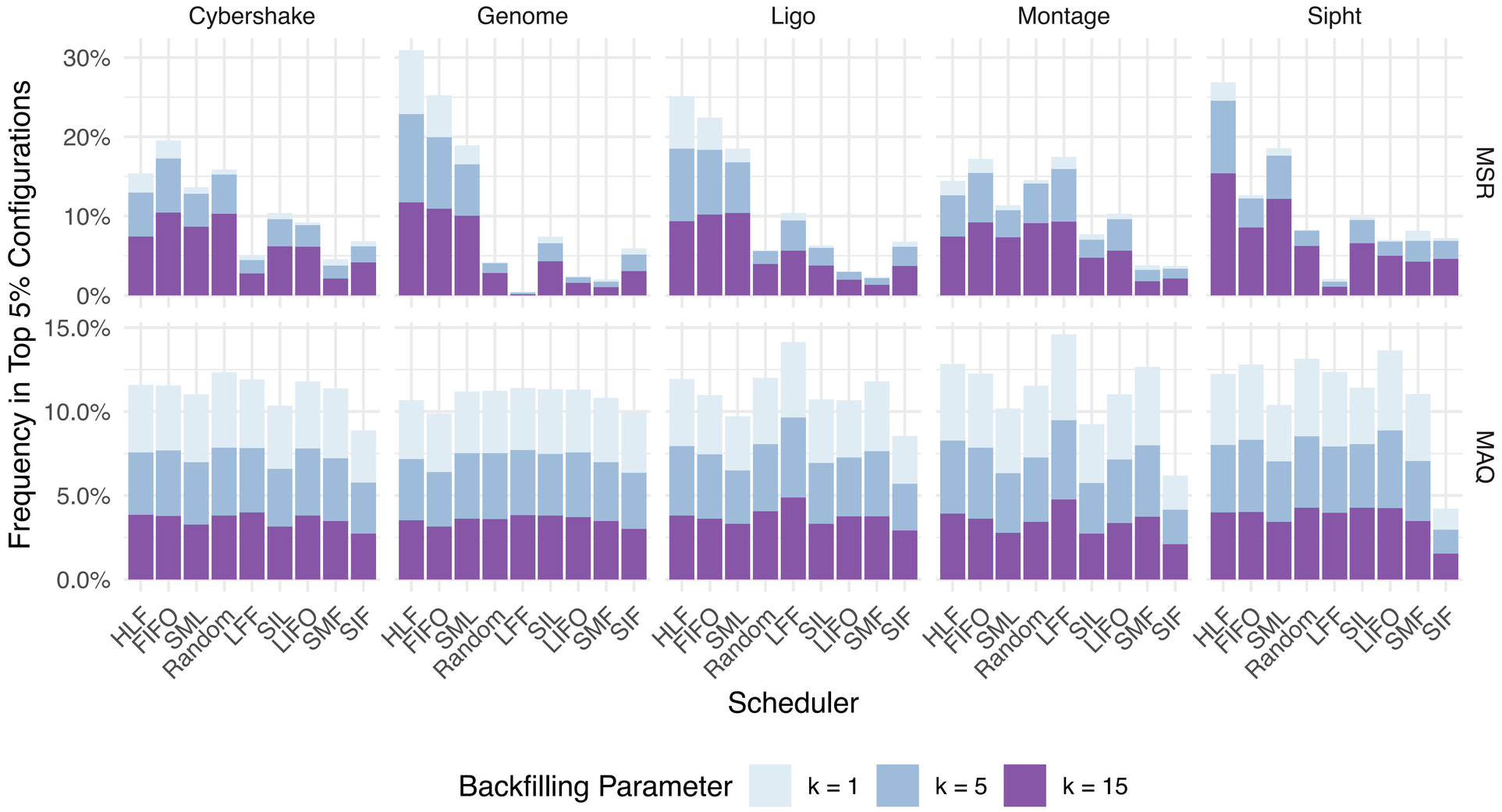
Montage



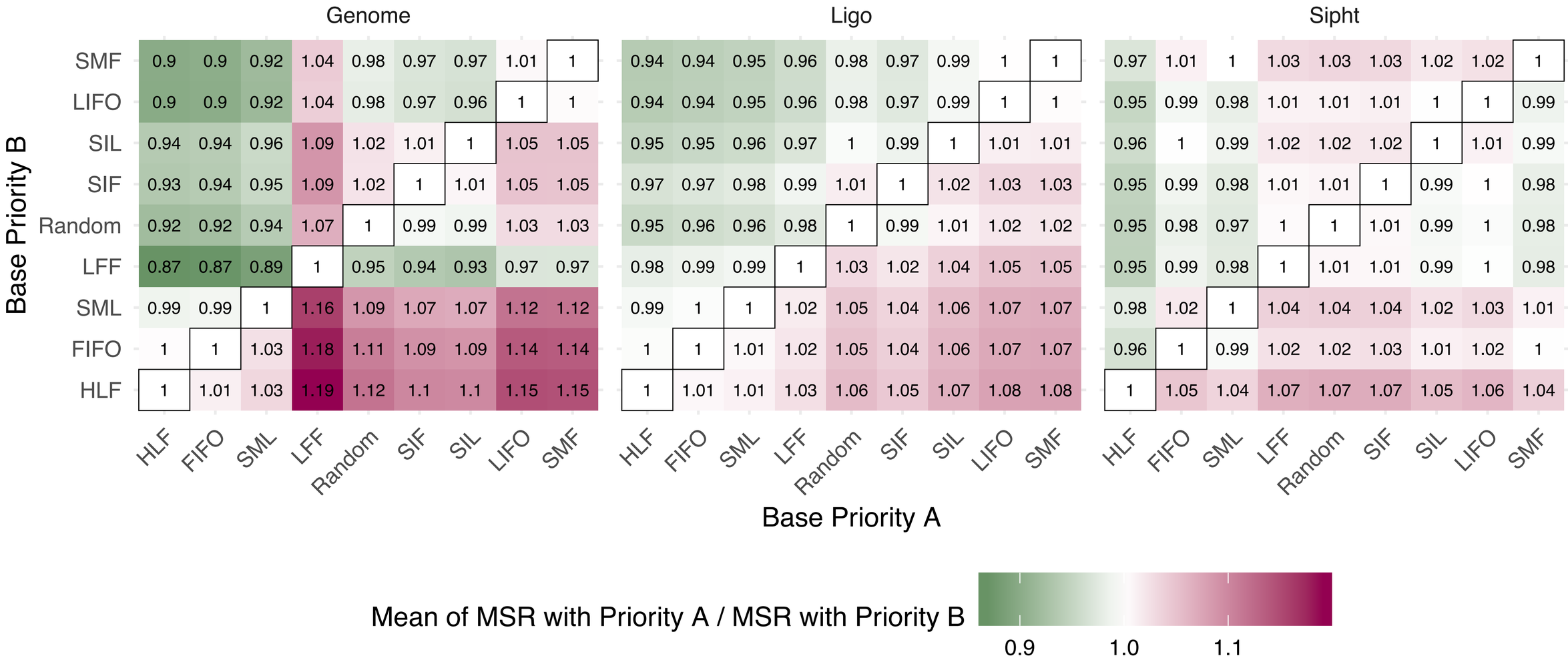
SIPHT



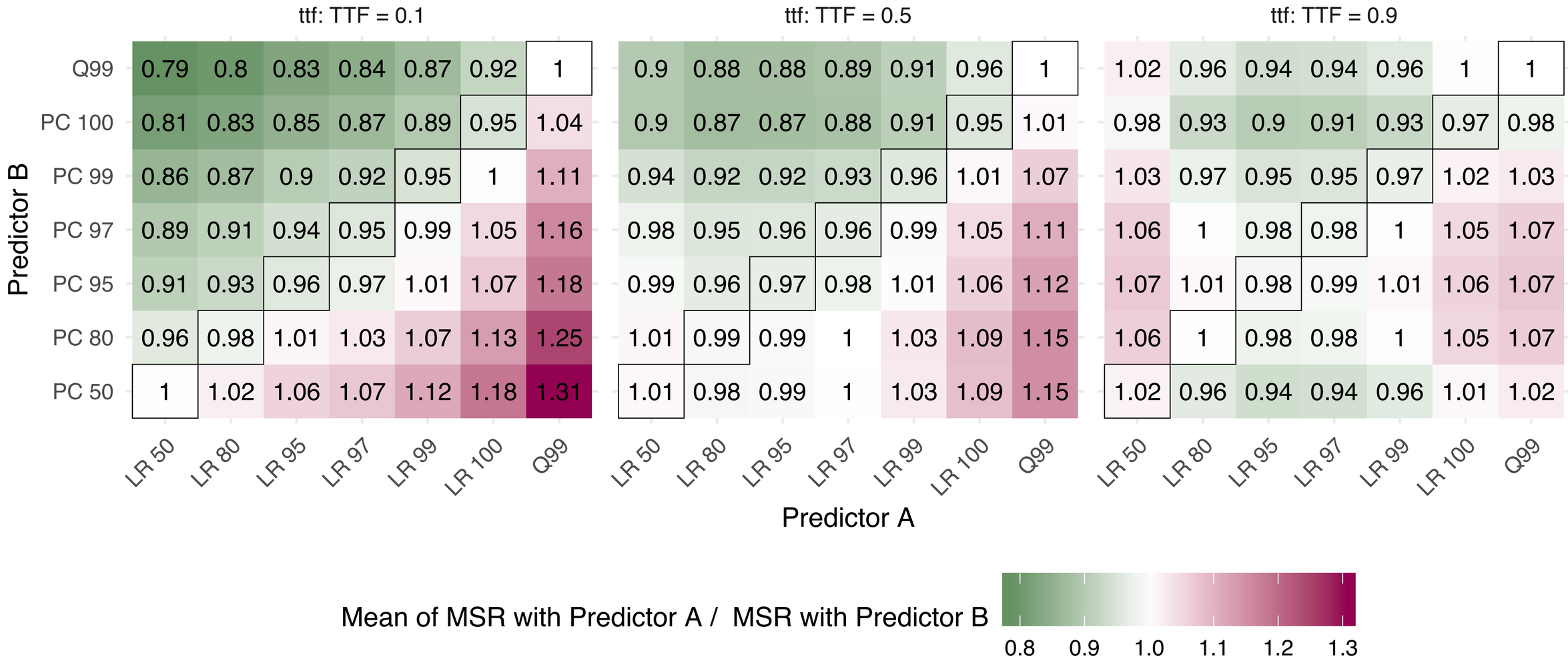
Top-Performing Scheduling Heuristics



Scheduler Comparison



Predictor Comparison



Scheduler Callbacks

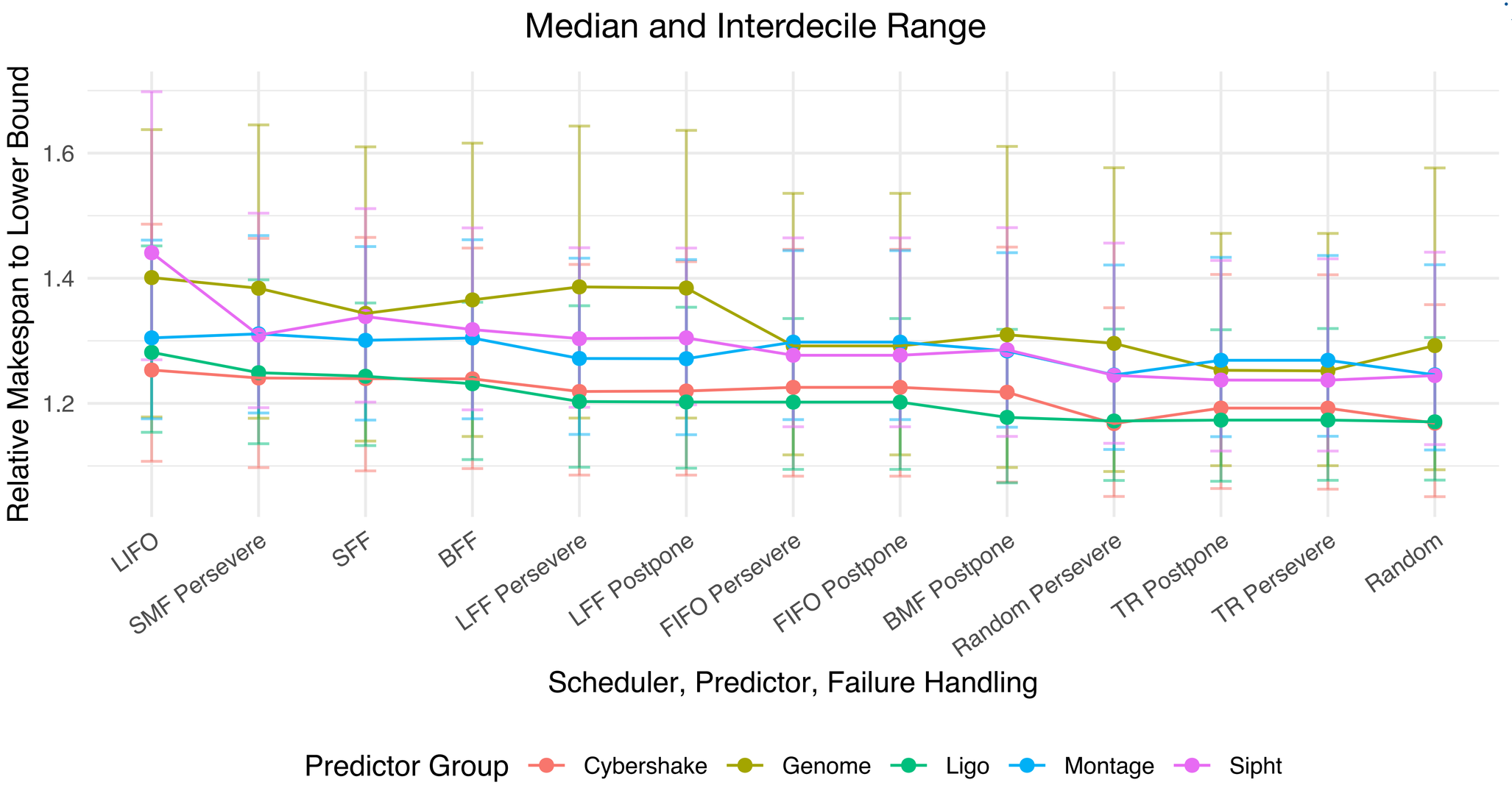
On task submit:
- enqueue ready task

On task succeeded:
- update predictions
- submit new ready tasks

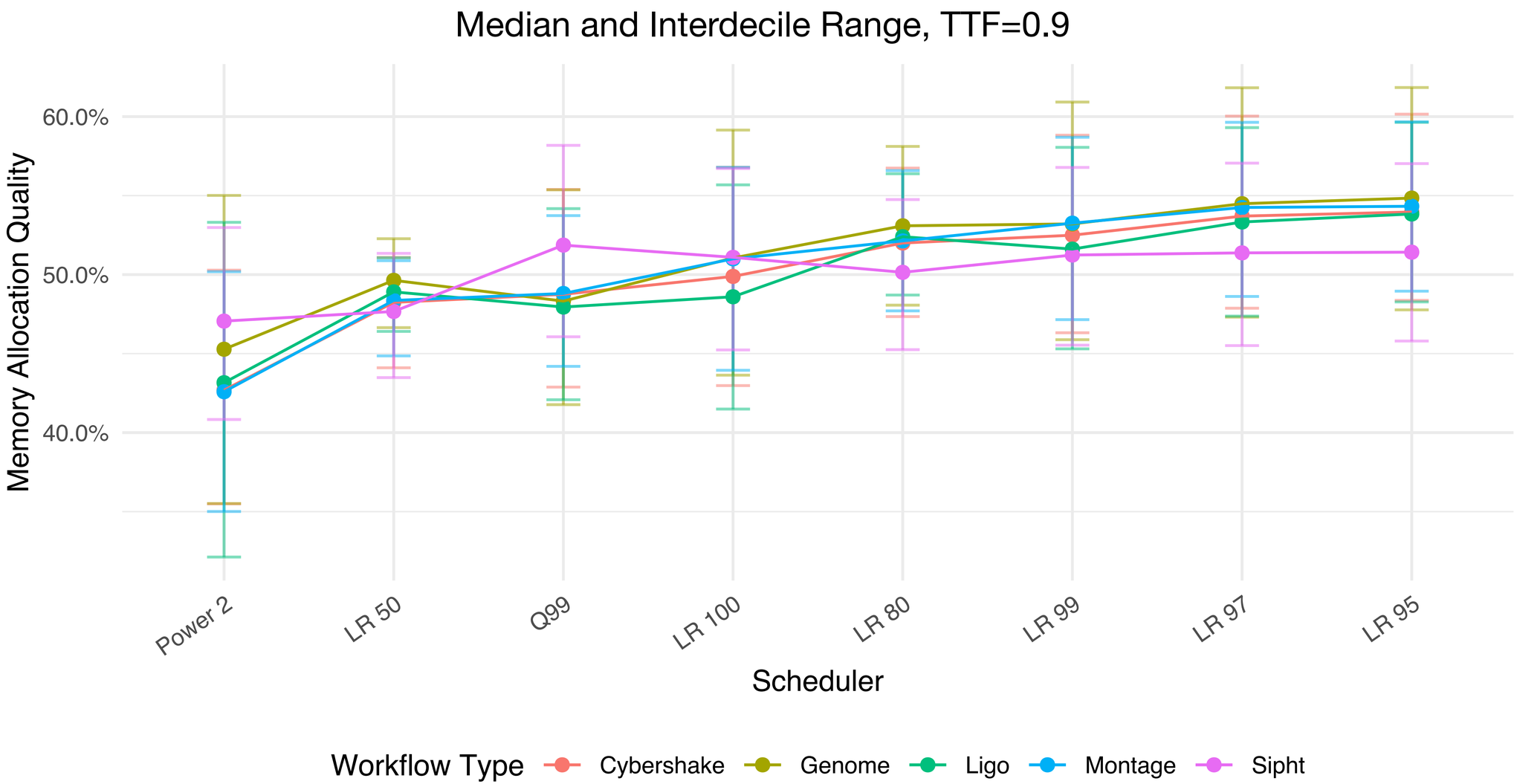
On task failed:
- increment attempts
- double memory request

On free resources on machine *m*:
- consider highest ranking task
- get predicted memory usage
- machine *m* has enough memory:
  - remove task from ready queue
  - start task

MSR with Oracle



Memory Allocation Quality



LWR

Survey on Resource Usage Prediction

Reference	Features	Methods	Task	Evaluation
Chapter 5, [Witt et al., 2019b]	Input size, data set id, program id	LR (asymmetric loss)	R	AOP, AUP, MAQ (≈75%)
[Li et al., 2019]	Program, job, and submission features, user estimates	Random forest	C	ACC (≈90-95%)
			R	MAPE (≈10-35%)
[Tyryshkina et al., 2019]	Program id, version, and parameters, data set id, file size and extension	Random forest	R	R^2 (≈60-85%)
			I	ACC (≈90%)
[Tovar et al., 2018]	Program id or category	LR (intercept only, asymmetric loss)	R	FR, MAQ (≈70%)
[Andresen et al., 2018]	User id, role, and estimates, user statistics	LR, RT (CART)	R	R^2 (≈15-30%)
[Duplyakin et al., 2018]	Number of nodes, method parameters, problem parameters	Bayesian optimization, GP	AL	RMSE convergence speed, total costs
[Rodrigues et al., 2017]	User, group, and queue, submission time, user estimate	Ensemble (SVM, RF, NN, KNN), sliding window	C	ACC (≈75-90%)
[Taghavi et al., 2016]	User statistics and estimates, project, requested processors, job type, recent job resource usage	NN, TS (ARIMA, KF), RT (CART, MARS, CHAID)	R	R^2 (≈93%), RMSE, AIC, AOP, AUP
[Matsunaga and Fortes, 2010]	Input description, hardware characterization	KNN, LR, RT (PQR)	R	MAPE (≈1-5%)

Method

A resource usage measurement set $D = (\tau, \tau^*, r, x)$ specifies the run times $\tau = \{\tau_1, \ldots, \tau_n\}$, times to failure $\tau^* = \{\tau_1^*, \ldots, \tau_n^*\}$, peak memory usages $r = \{r_1, \ldots, r_n\}$, and input sizes $x = \{x_1, \ldots, x_n\}$ for n tasks.

$$f_{\theta,b}(x_i, j) = \begin{cases} \max(\theta_1 x_i + \theta_0, a_l) & j = 1 \\ b \cdot f_{\theta,b}(x_i, j-1) & j > 1 \end{cases}$$

Rectified linear allocation

Exponential allocation

$$k_i(\theta, b) = 1 + \max\left(0, \left\lceil \log_b \frac{r_i}{f_{\theta,b}(x_i, 1)} \right\rceil\right)$$

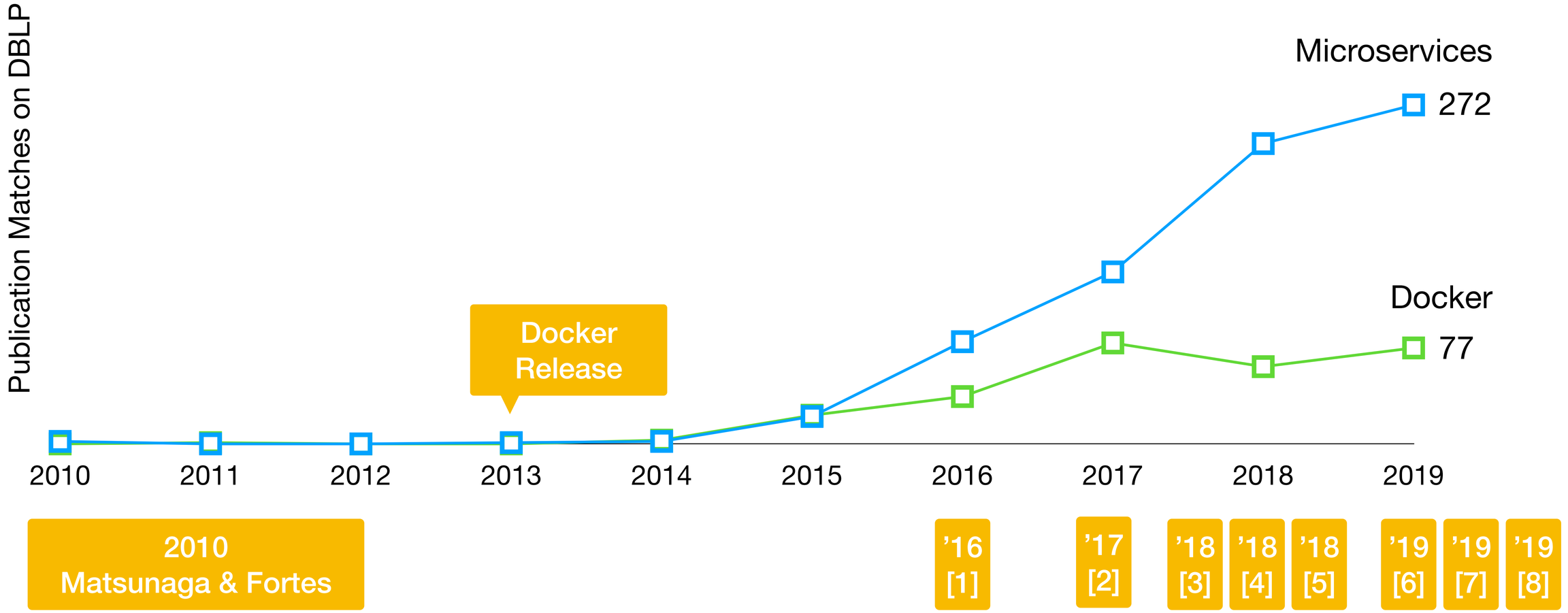
Number of attempts

$$W(D, \theta, b) = \sum_{i=1}^{n} \underbrace{(f_{\theta,b}(x_i, 1) b^{k_i - 1} - r_i)\tau_i}_{over-sizing} + \underbrace{f_{\theta,b}(x_i, 1) \frac{b^{k_i - 1} - 1}{b - 1} \tau_i^*}_{under-sizing}$$

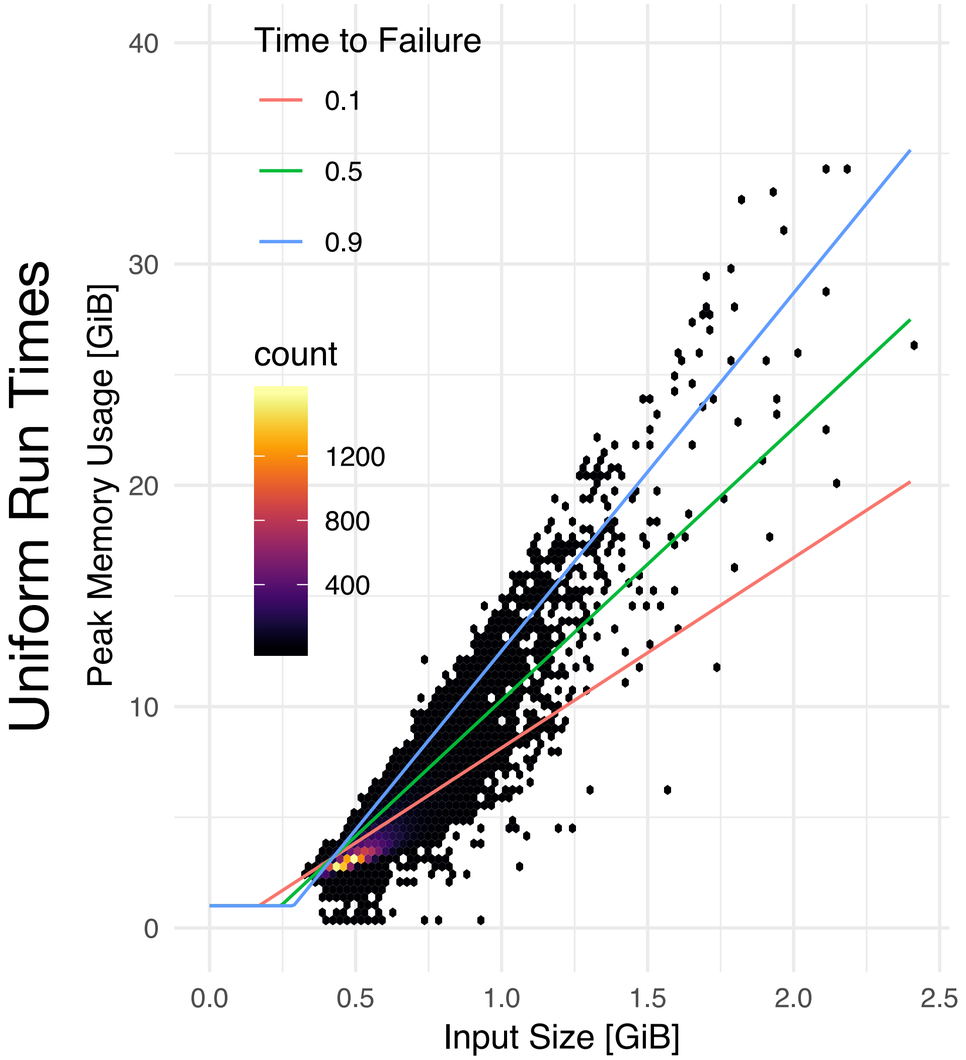
Total wastage over all attempts over all tasks

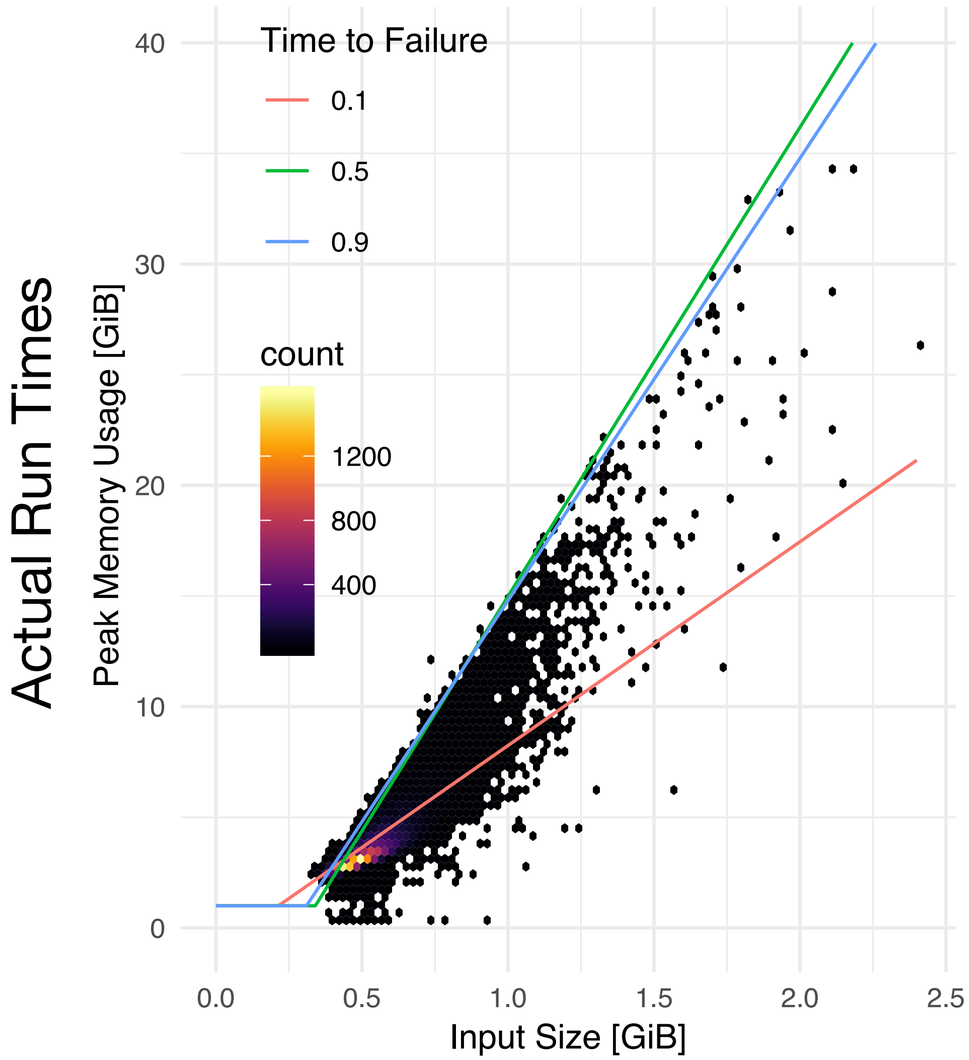
$$\underset{\theta \in \mathbb{R}^2, b \in \mathbb{R}}{\text{minimize}}\ W(D, \theta, b) \text{ s.t. } b > 1$$

Memory Prediction History

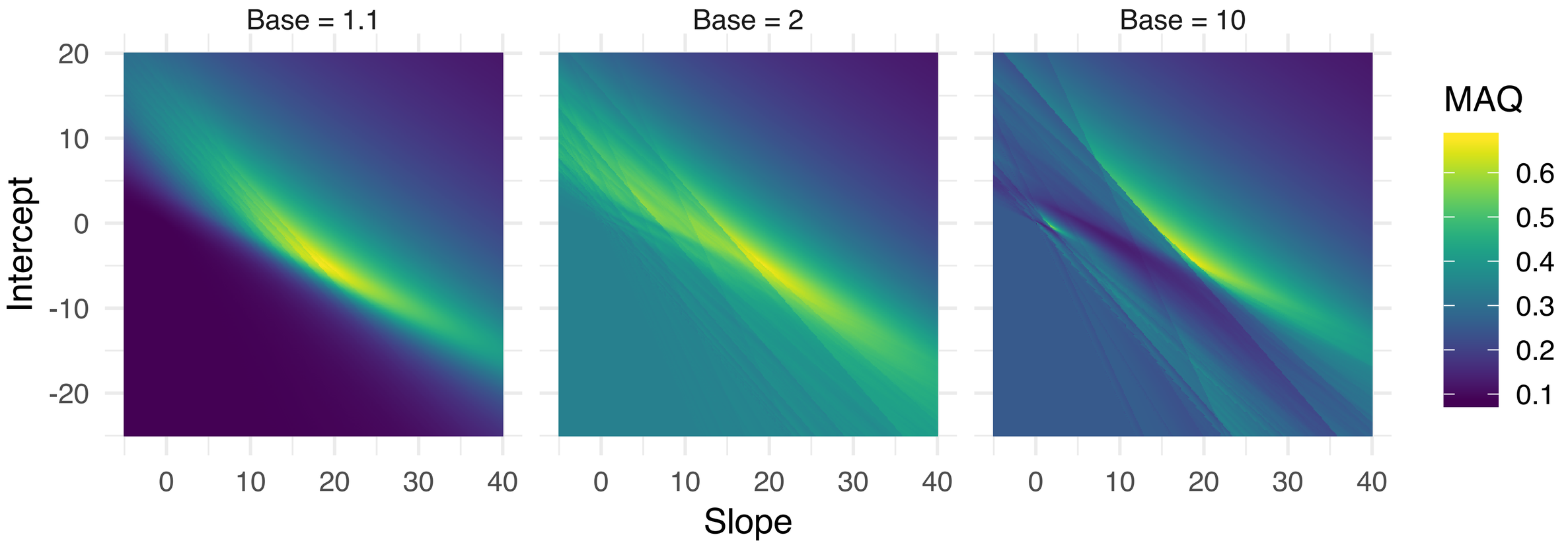


Impact of Run Times

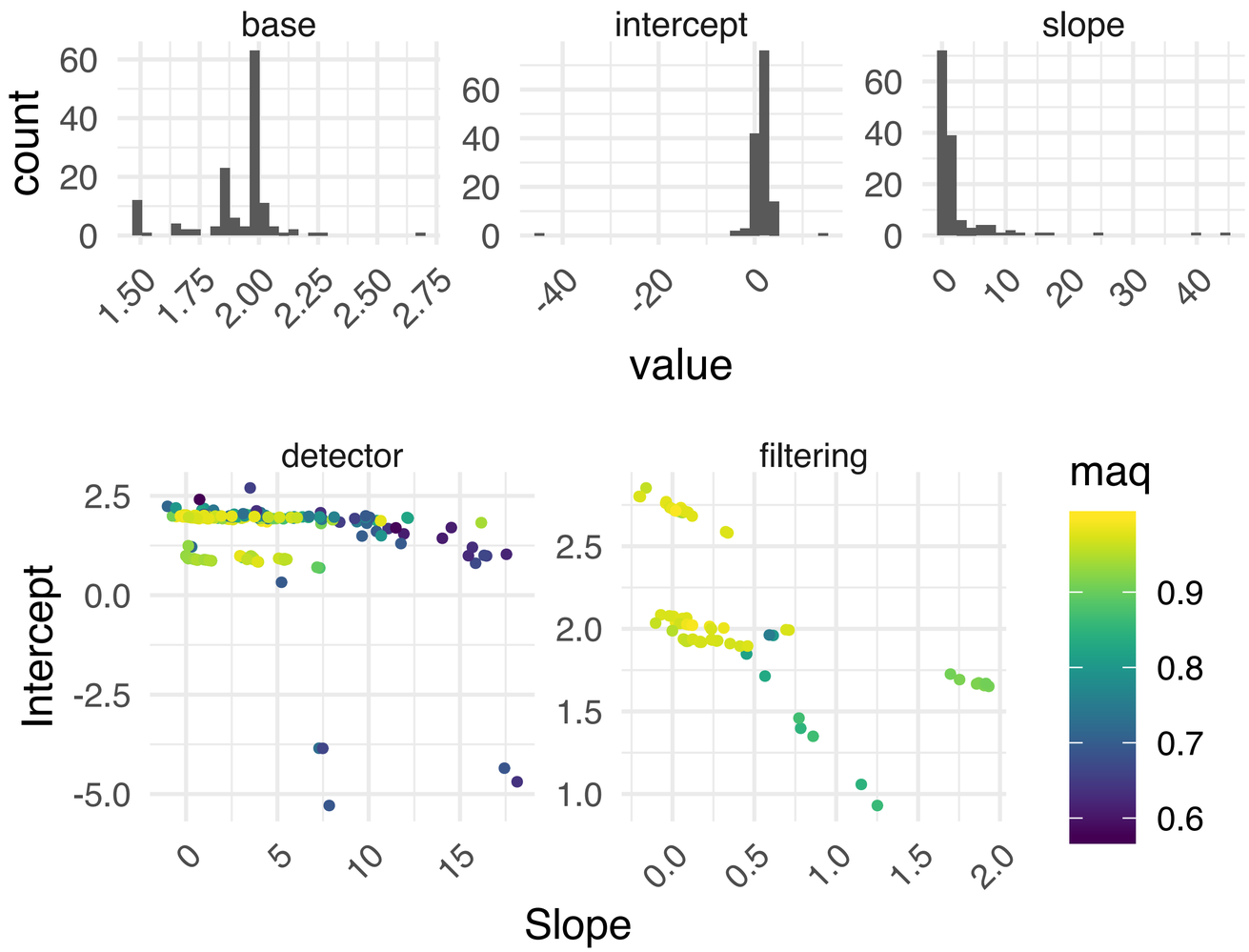




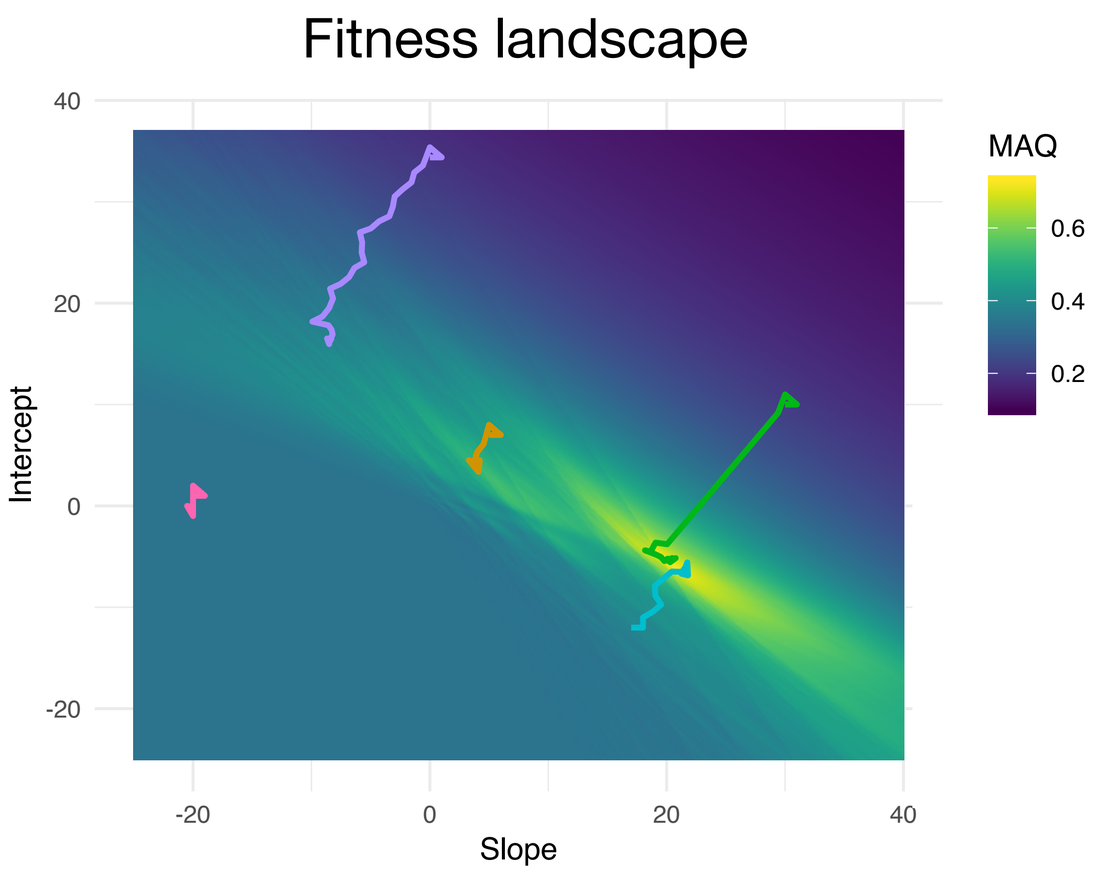
Impact of Re-Allocation Factor

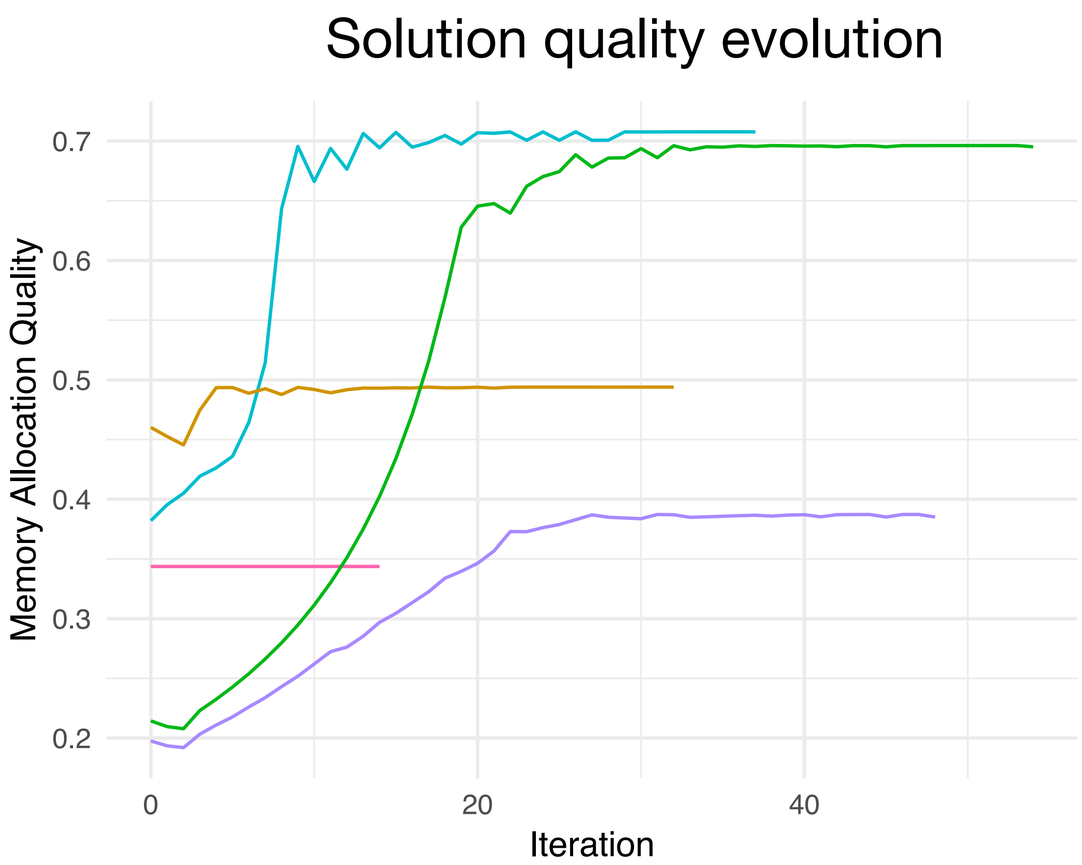


LWR Fitted Model Parameters



Refinement of Initial Solutions





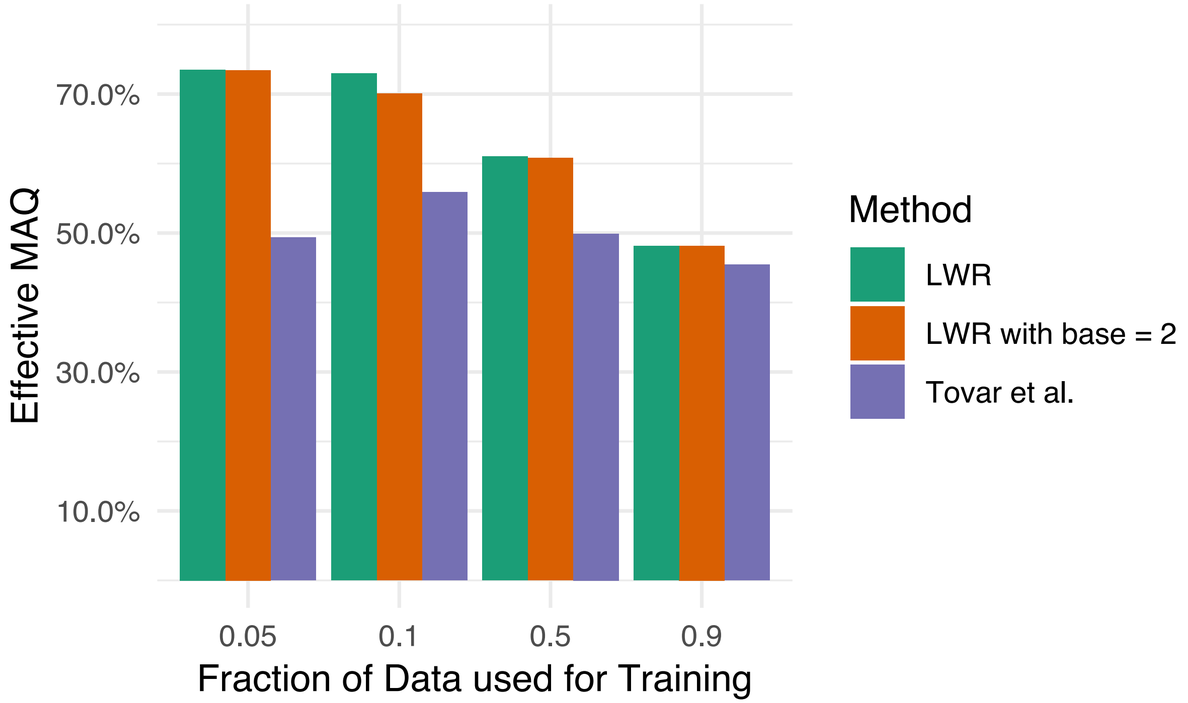
Aggregated Results

Effective Memory Allocation Quality

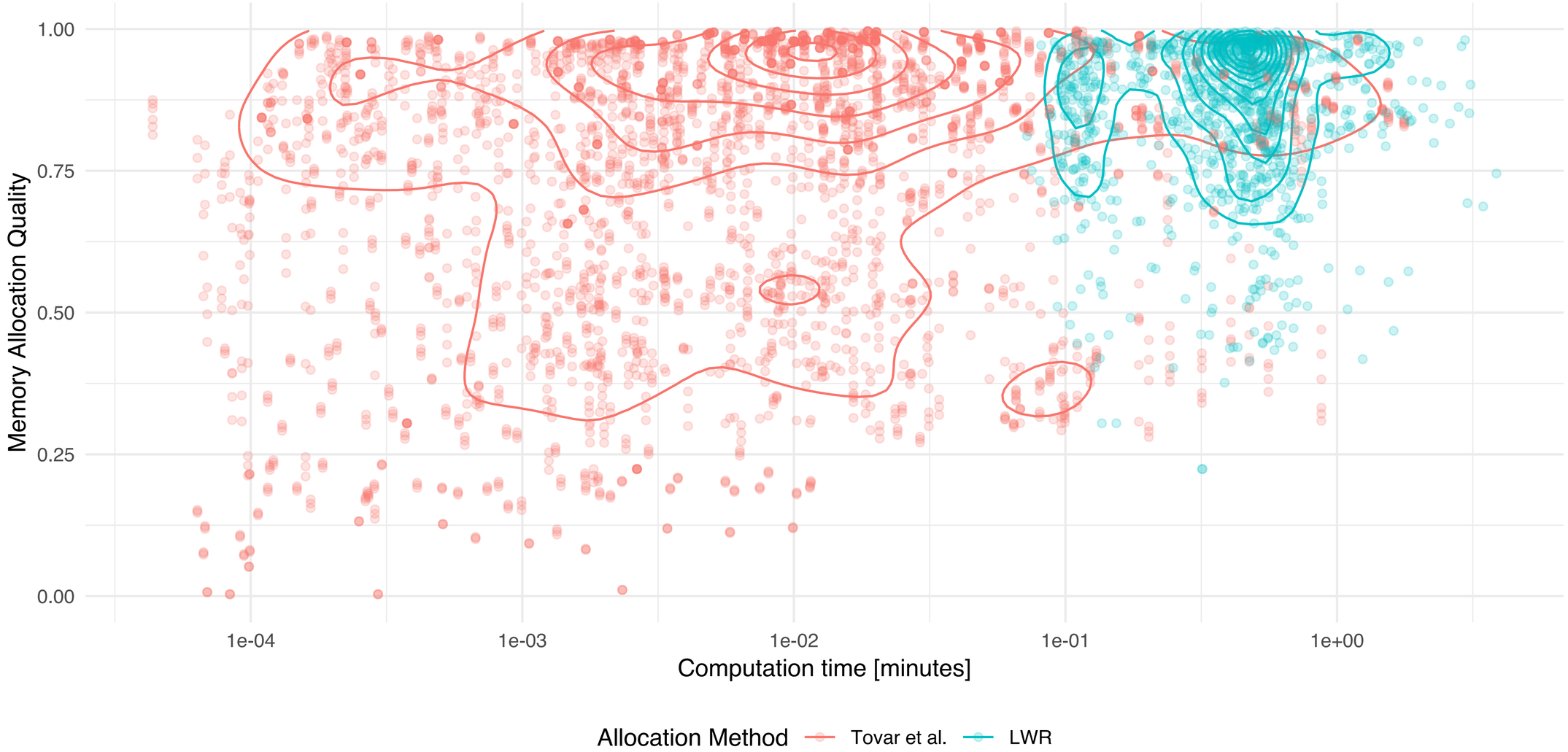
- wastage on test set +
  user wastage during training

Effective MAQ > 70%

- using 5% data for training
- up to 40% throughput gain



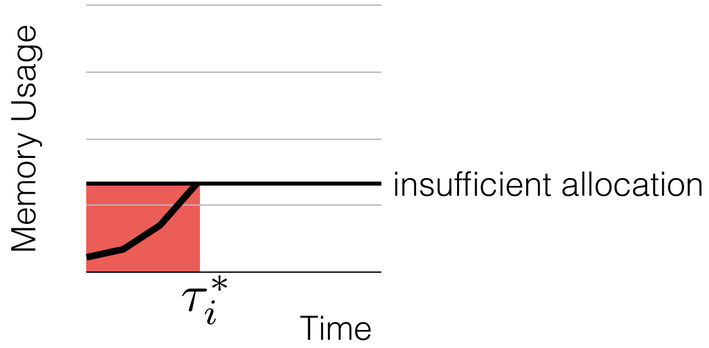
LWR Computation Time vs. Quality



Constrained Memory Execution Model

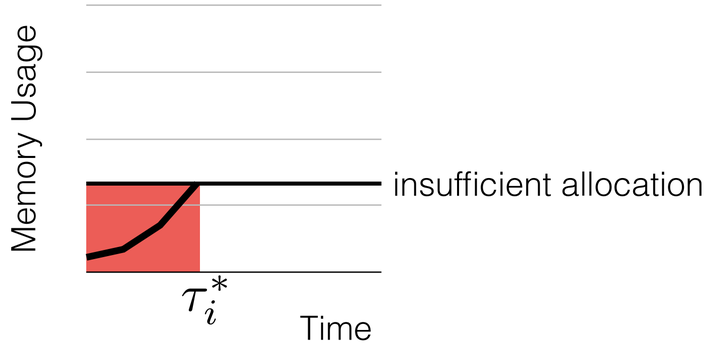
Constrained Memory Execution Model

Failed attempt

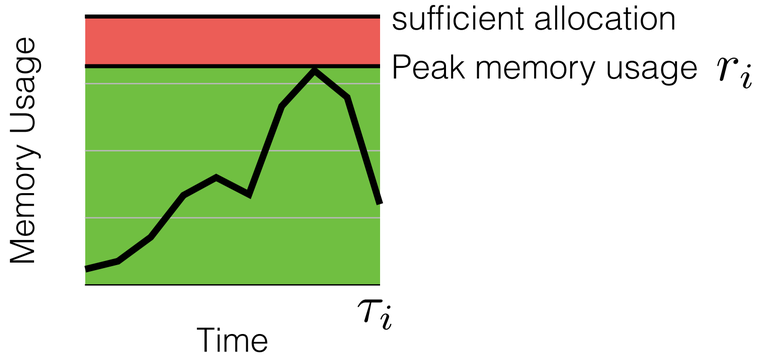


Constrained Memory Execution Model

Failed attempt

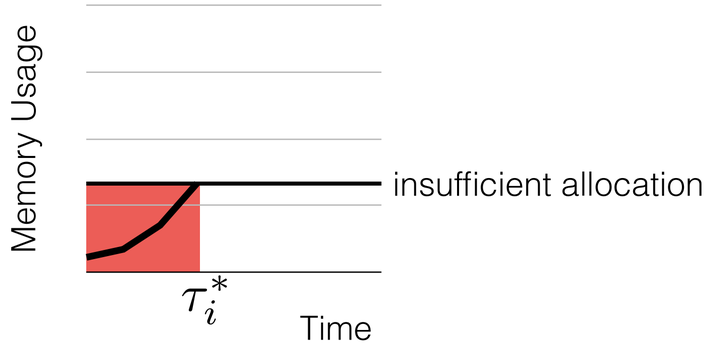


Successful attempt



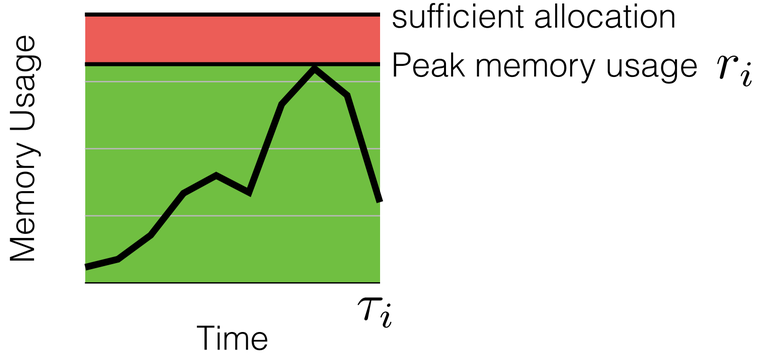
Constrained Memory Execution Model

Failed attempt



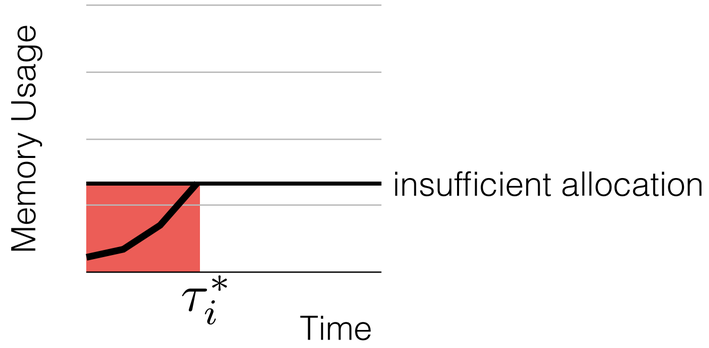
$$\text{wastage}(r_i, \tau_i, \tau_i^*, a_{ij}) = \begin{cases} a_{ij}\tau_i^* & \text{if } r_i > a_{ij} \\ (a_{ij} - r_i)\tau & \text{otherwise} \end{cases}$$

Successful attempt



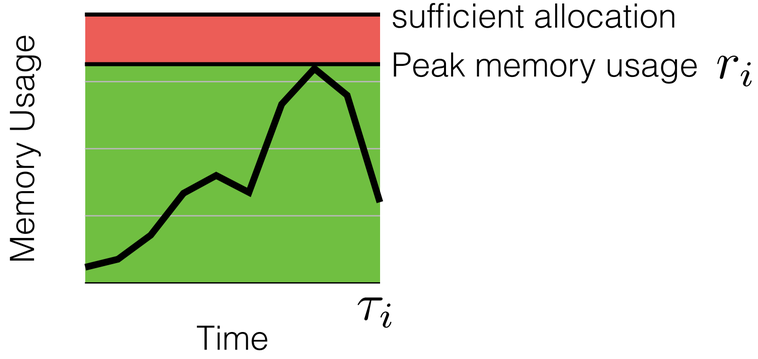
Constrained Memory Execution Model

Failed attempt



$$\text{wastage}(r_i, \tau_i, \tau_i^*, a_{ij}) = \begin{cases} a_{ij}\tau_i^* & \text{if } r_i > a_{ij} \\ (a_{ij} - r_i)\tau & \text{otherwise} \end{cases}$$

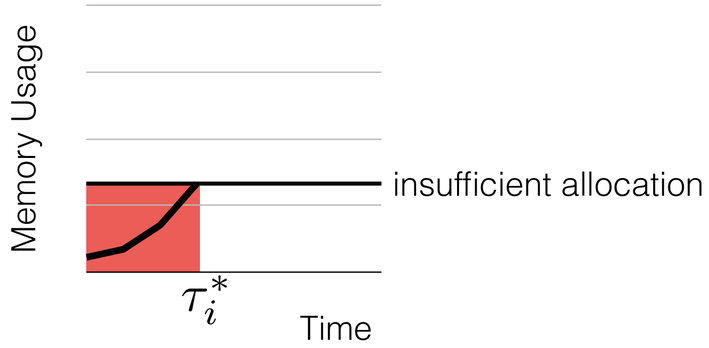
Successful attempt



$$W = \sum_i \sum_{j=1}^{k_i} \text{wastage}(r_i, \tau_i, \tau_i^*, a_{ij})$$

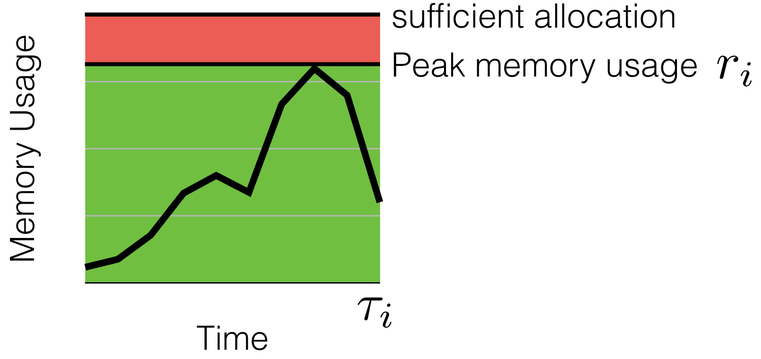
Constrained Memory Execution Model

Failed attempt



$$\text{wastage}(r_i, \tau_i, \tau_i^*, a_{ij}) = \begin{cases} a_{ij}\tau_i^* & \text{if } r_i > a_{ij} \\ (a_{ij} - r_i)\tau & \text{otherwise} \end{cases}$$

Successful attempt



$$W = \sum_i \sum_{j=1}^{k_i} \text{wastage}(r_i, \tau_i, \tau_i^*, a_{ij})$$

$$\text{MAQ} = \frac{\text{usage}}{\text{usage} + \text{wastage}}$$

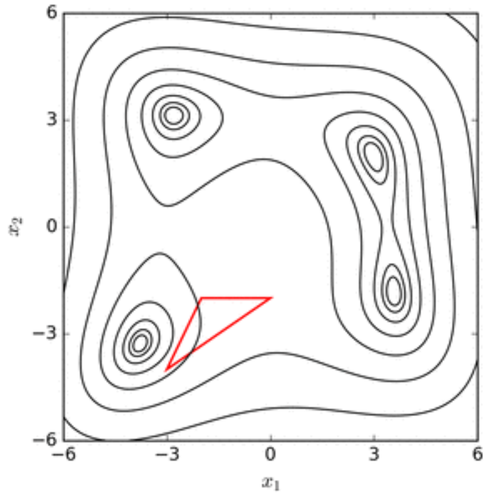
Nelder-Mead / Simplex-Based Optimization

Non-linear constrained optimization

- **Cobyla**
- sequential quadratic programming

Non-linear optimization

- Nelder-Mead
- conjugate gradient descent
- …



en.wikipedia.org/wiki/File:Nelder-Mead_Himmelblau.gif

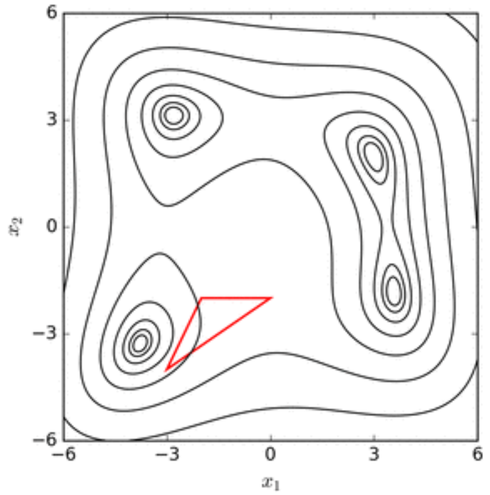
Nelder-Mead / Simplex-Based Optimization

Non-linear constrained optimization

- **Cobyla**
- sequential quadratic programming

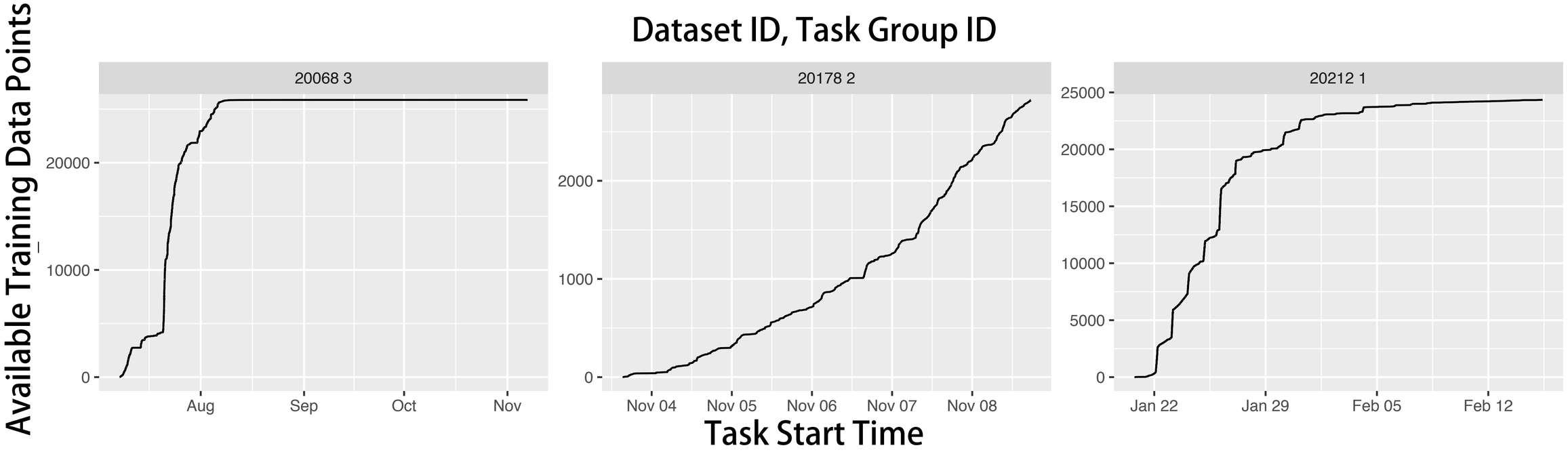
Non-linear optimization

- Nelder-Mead
- conjugate gradient descent
- …

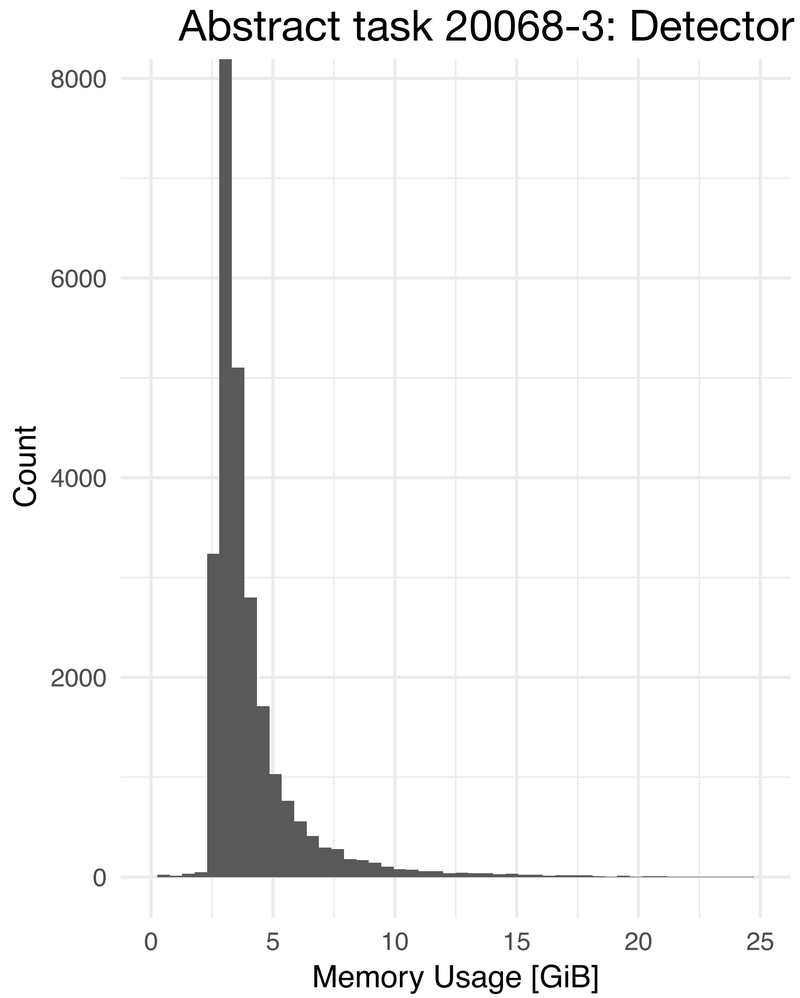


en.wikipedia.org/wiki/File:Nelder-Mead_Himmelblau.gif

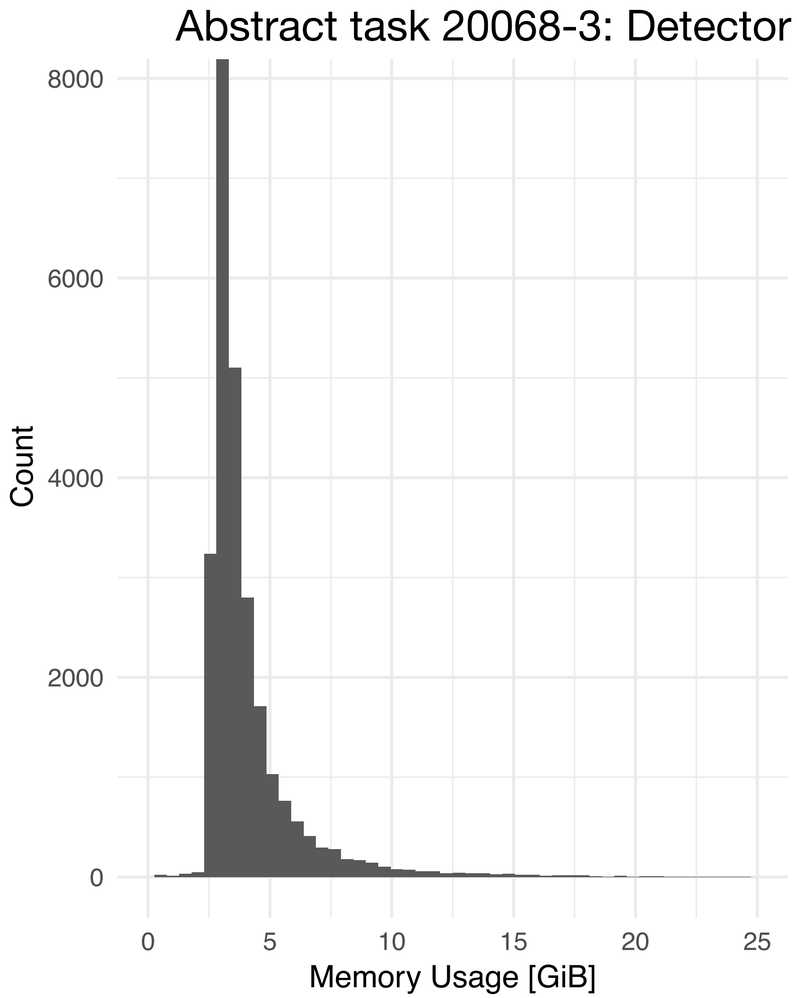
IceCube Training Data Influx

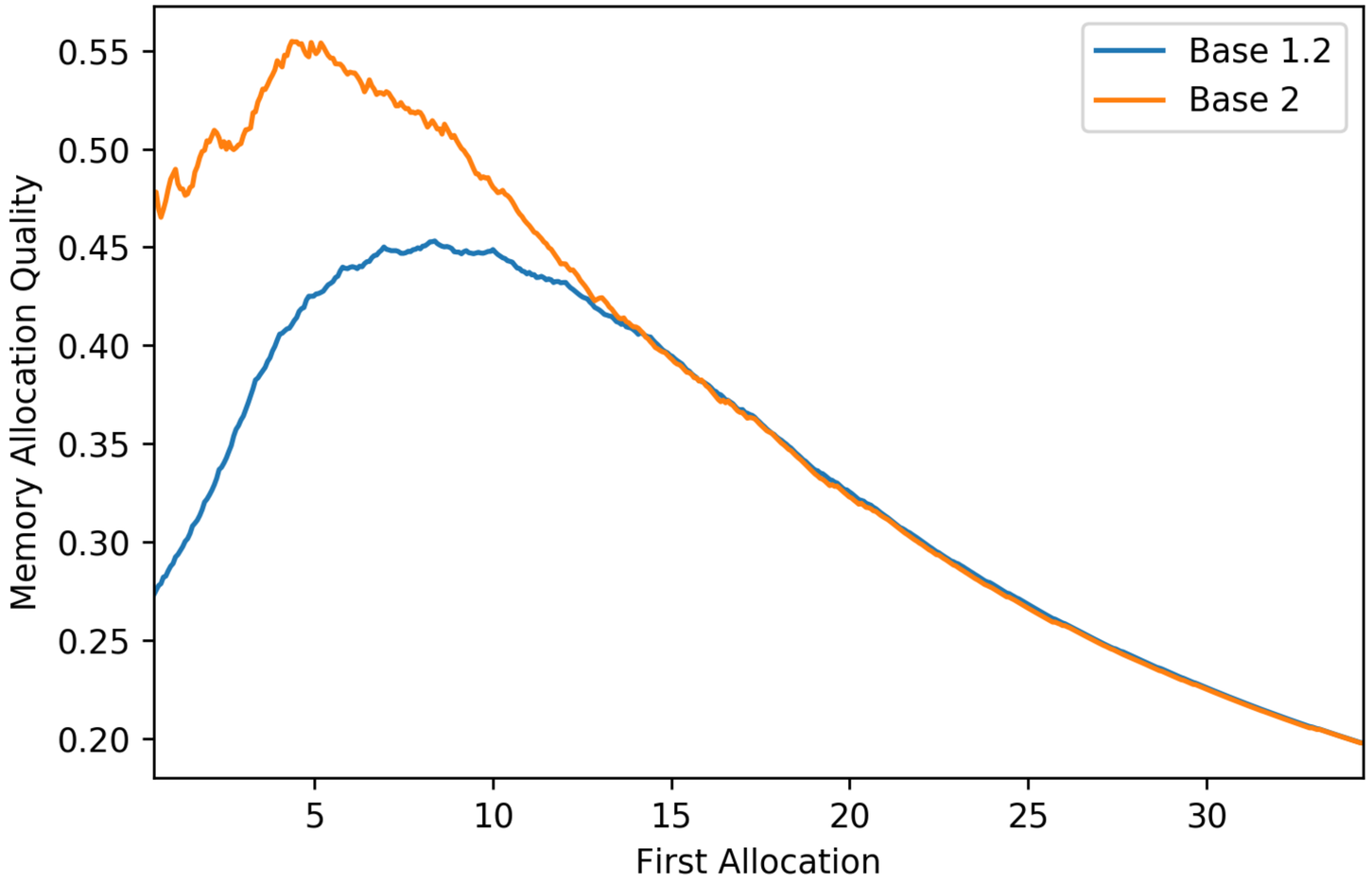


Exponential Re-Allocation: Abstract Task Example

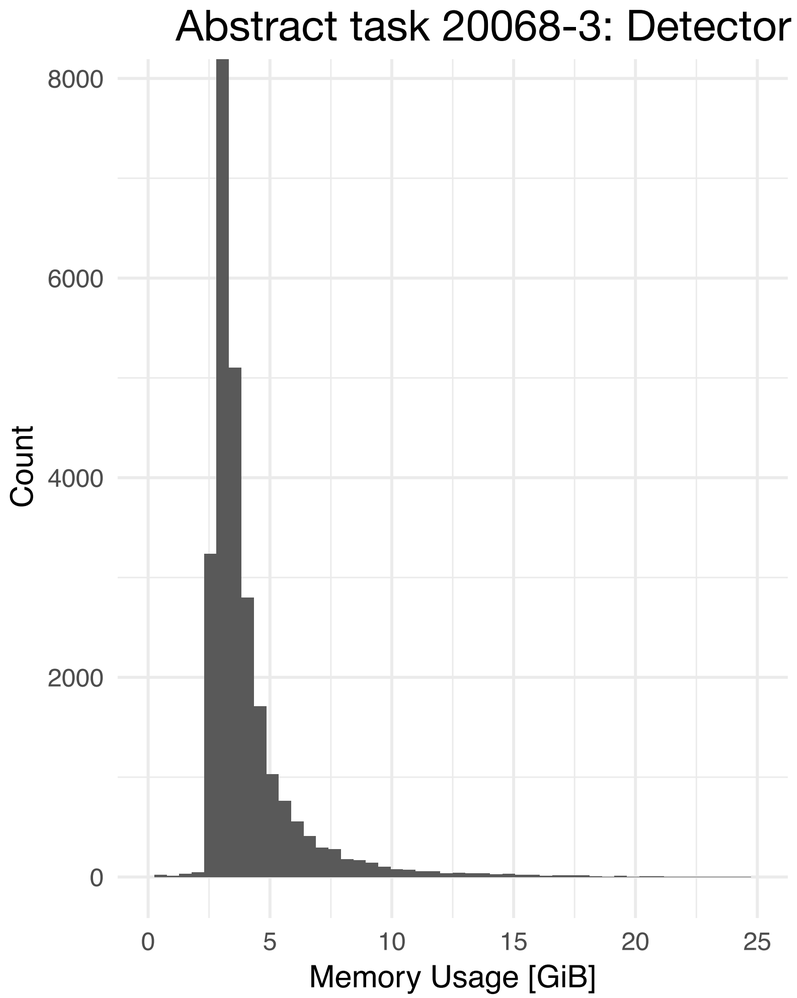


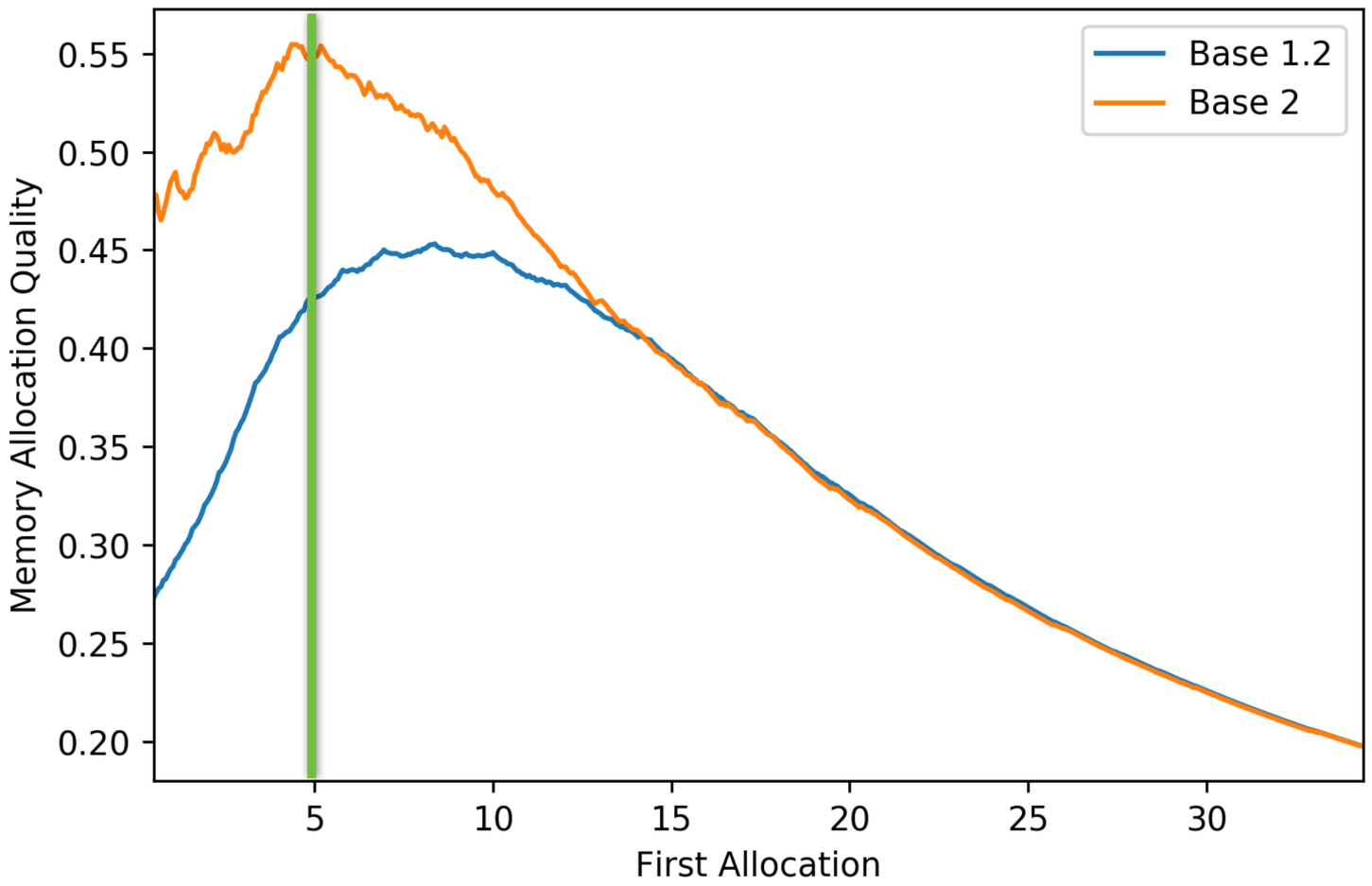
Exponential Re-Allocation: Abstract Task Example



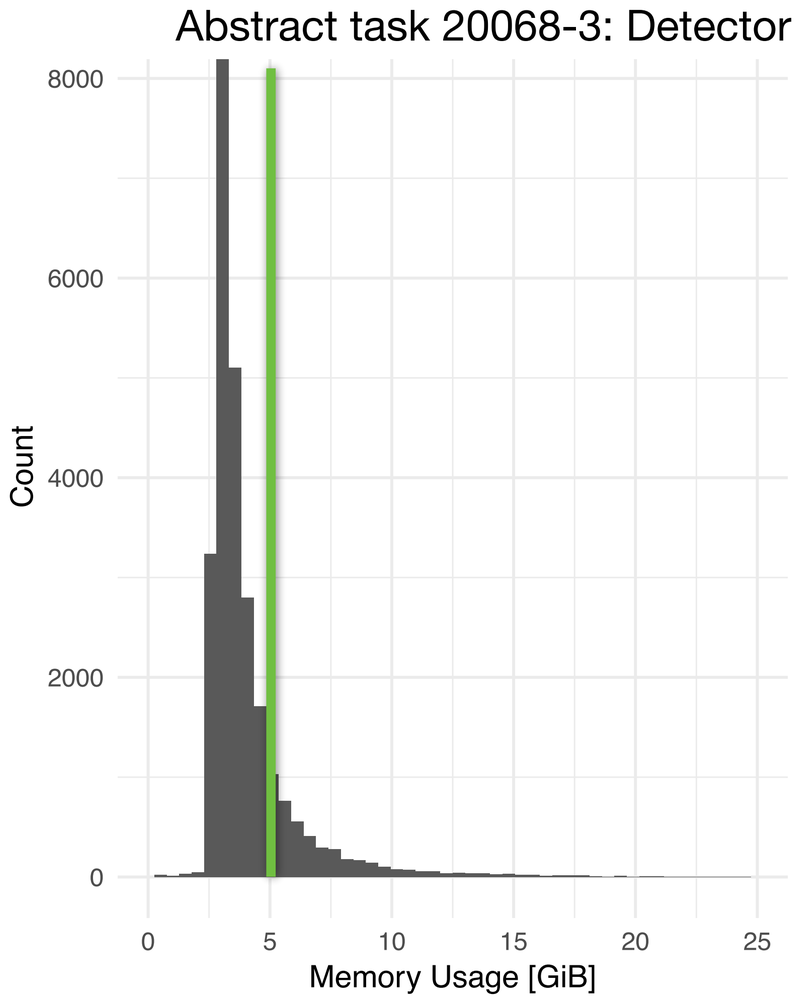


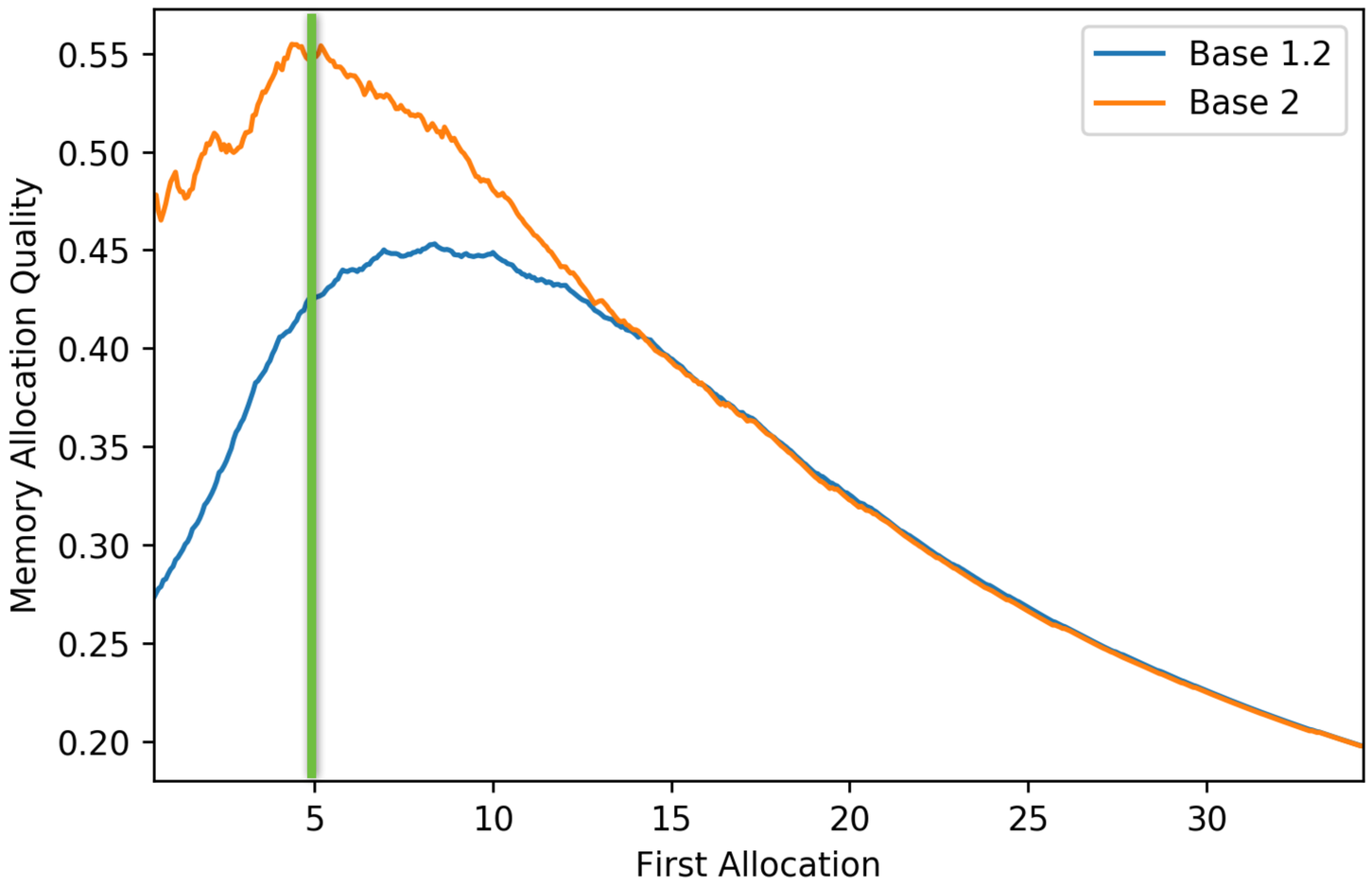
Exponential Re-Allocation: Abstract Task Example



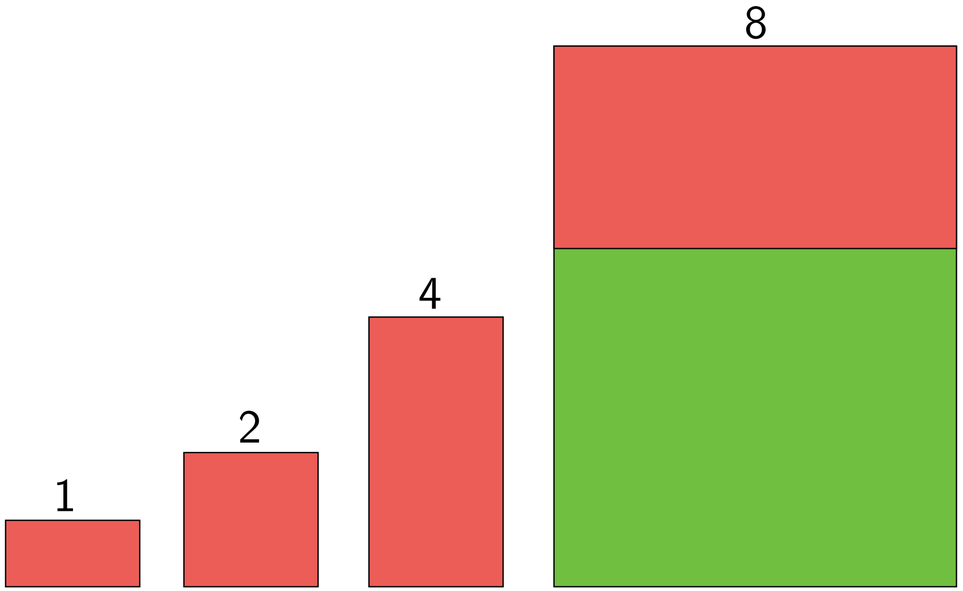


Exponential Re-Allocation: Abstract Task Example

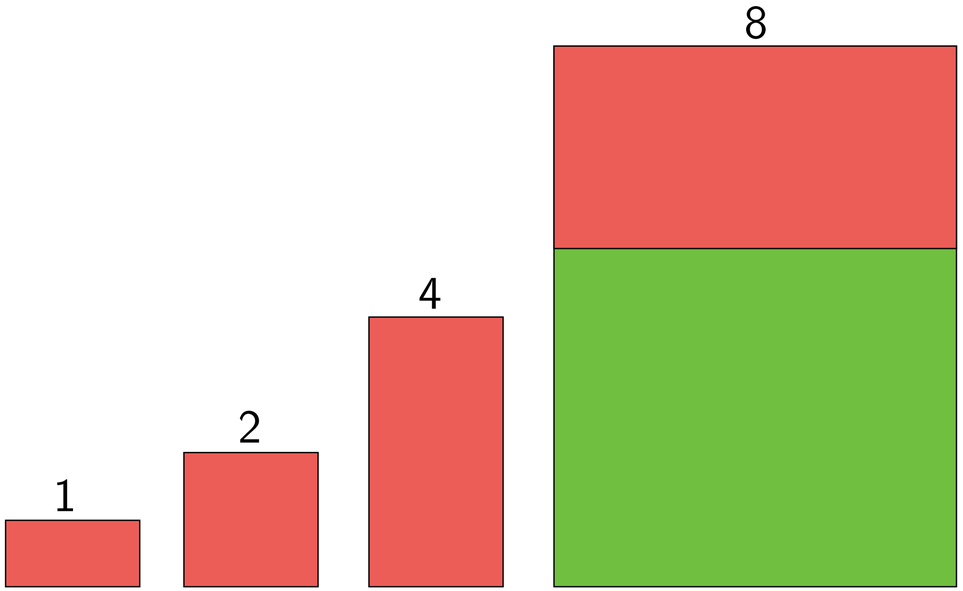


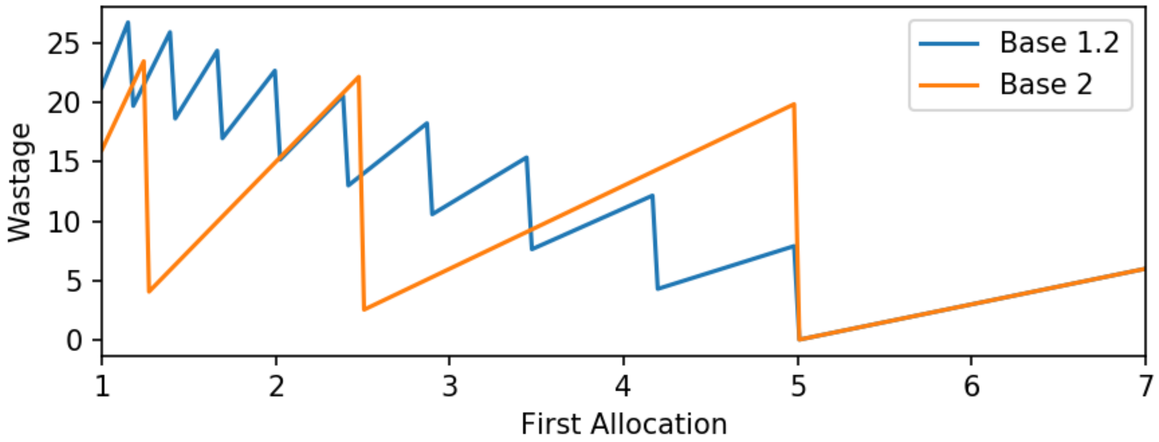


Exponential Allocation: Single Task Example

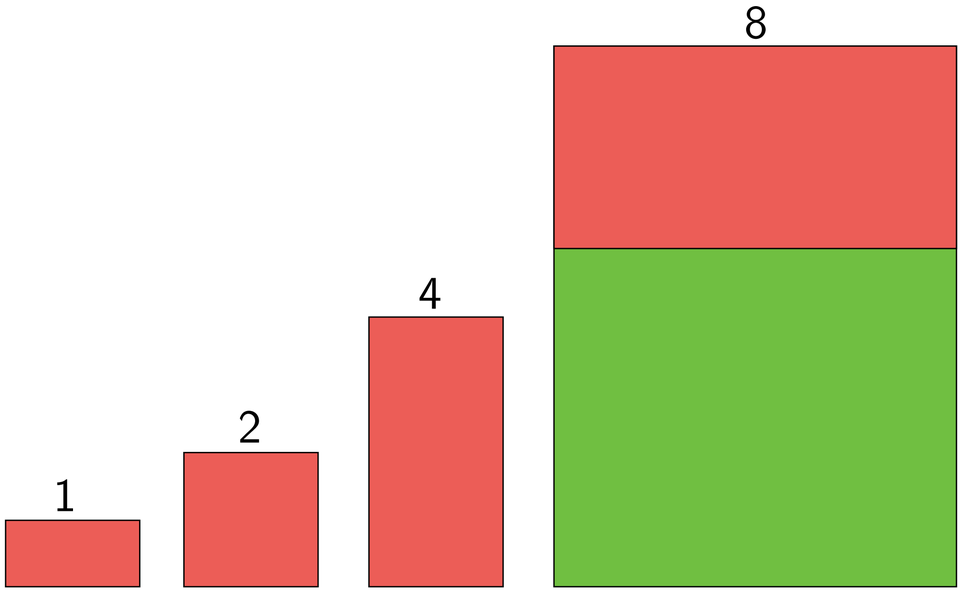


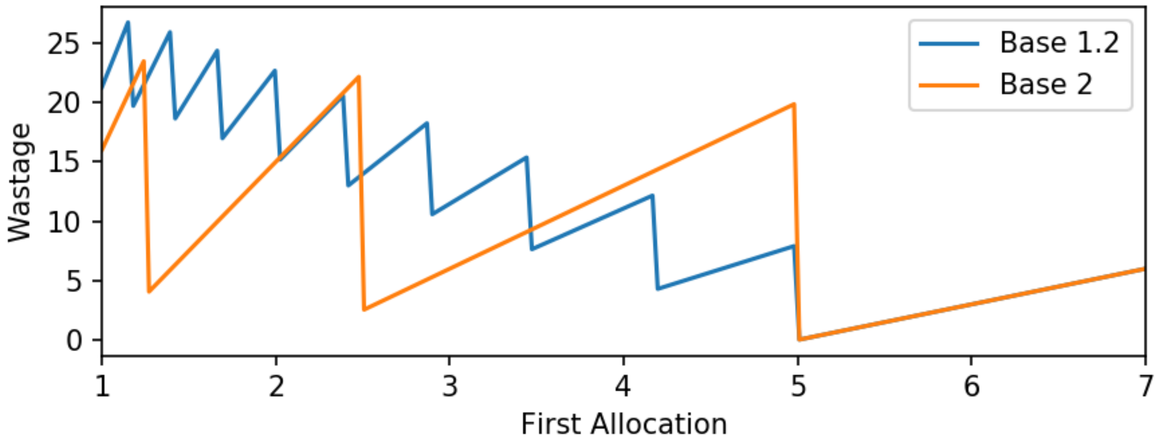
Exponential Allocation: Single Task Example





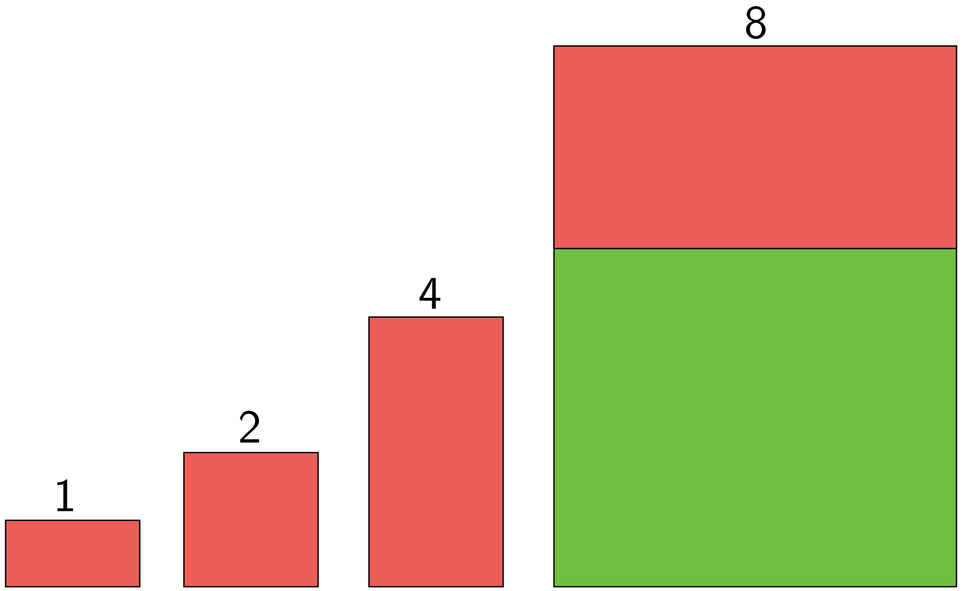
Exponential Allocation: Single Task Example



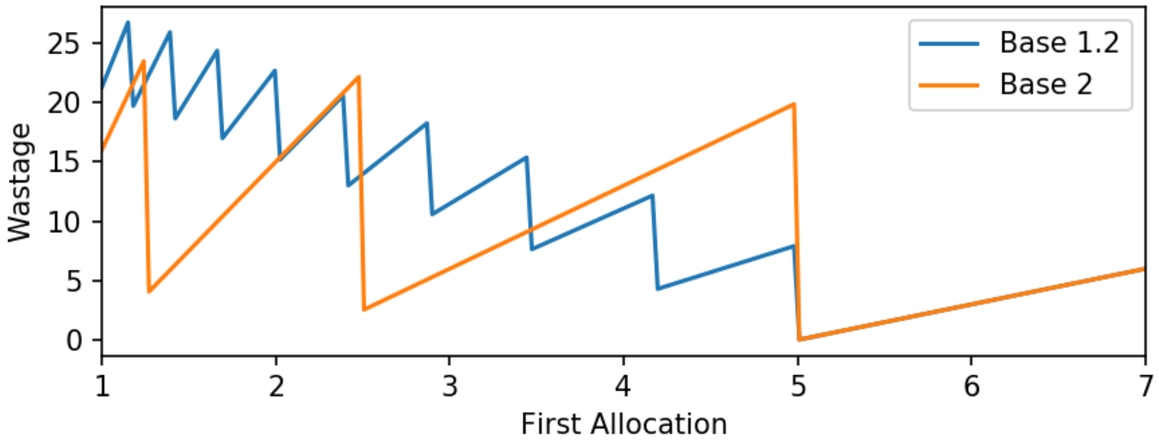


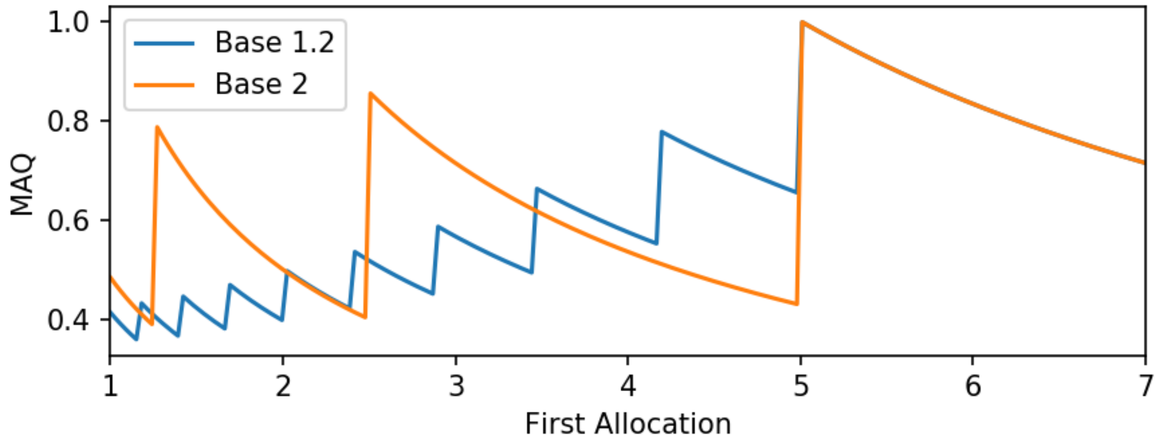
$$\text{MAQ} = \frac{\text{usage}}{\text{usage} + \text{wastage}}$$

Exponential Allocation: Single Task Example

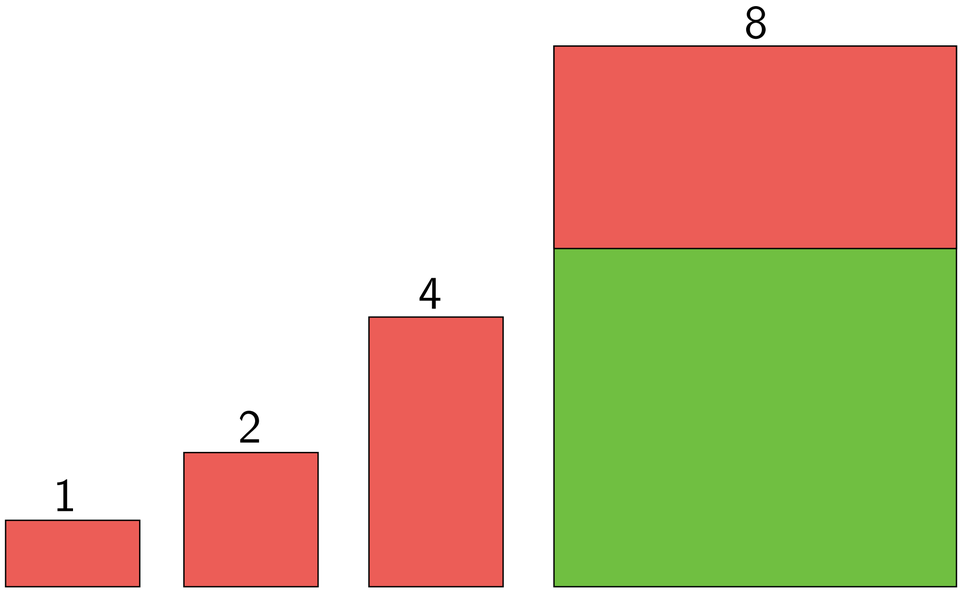


$$\mathrm{MAQ} = \frac{\mathrm{usage}}{\mathrm{usage} + \mathrm{wastage}}$$

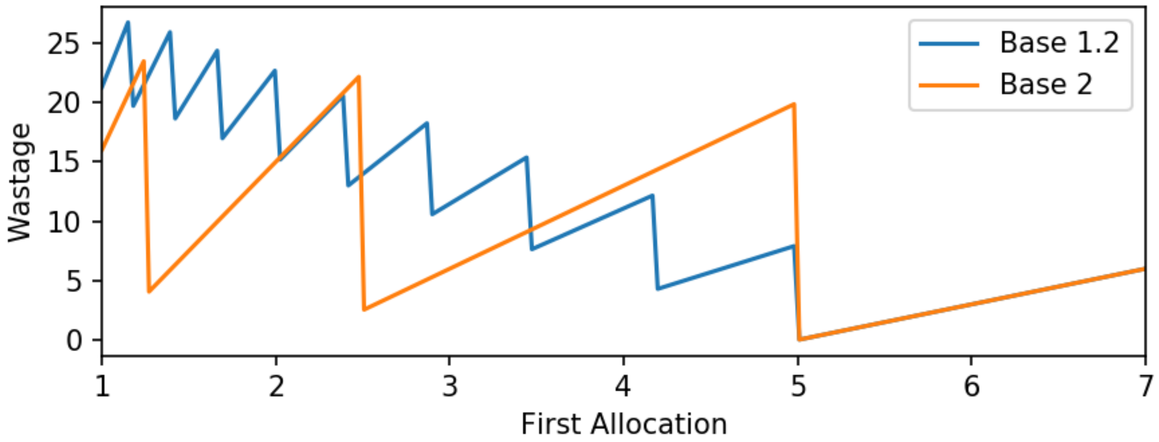


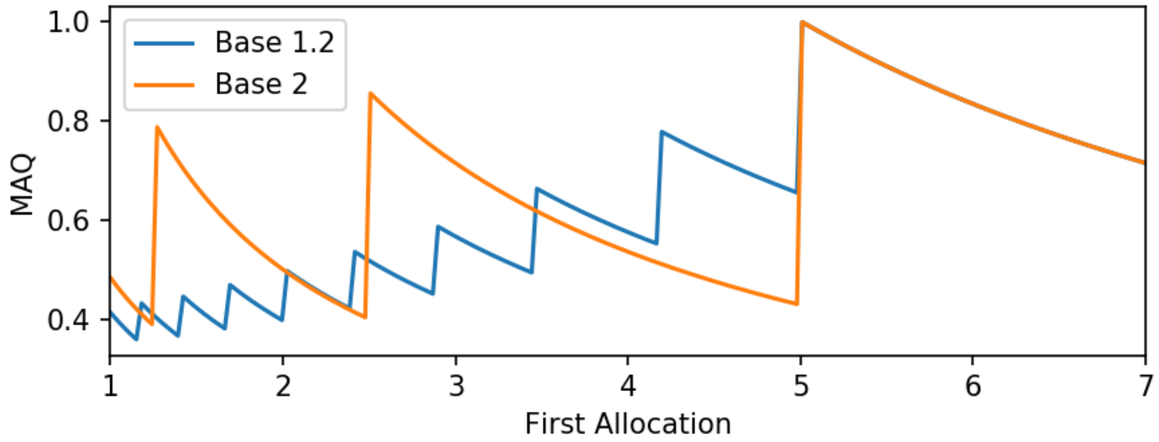


Exponential Allocation: Single Task Example



$$\text{MAQ} = \frac{\text{usage}}{\text{usage} + \text{wastage}}$$

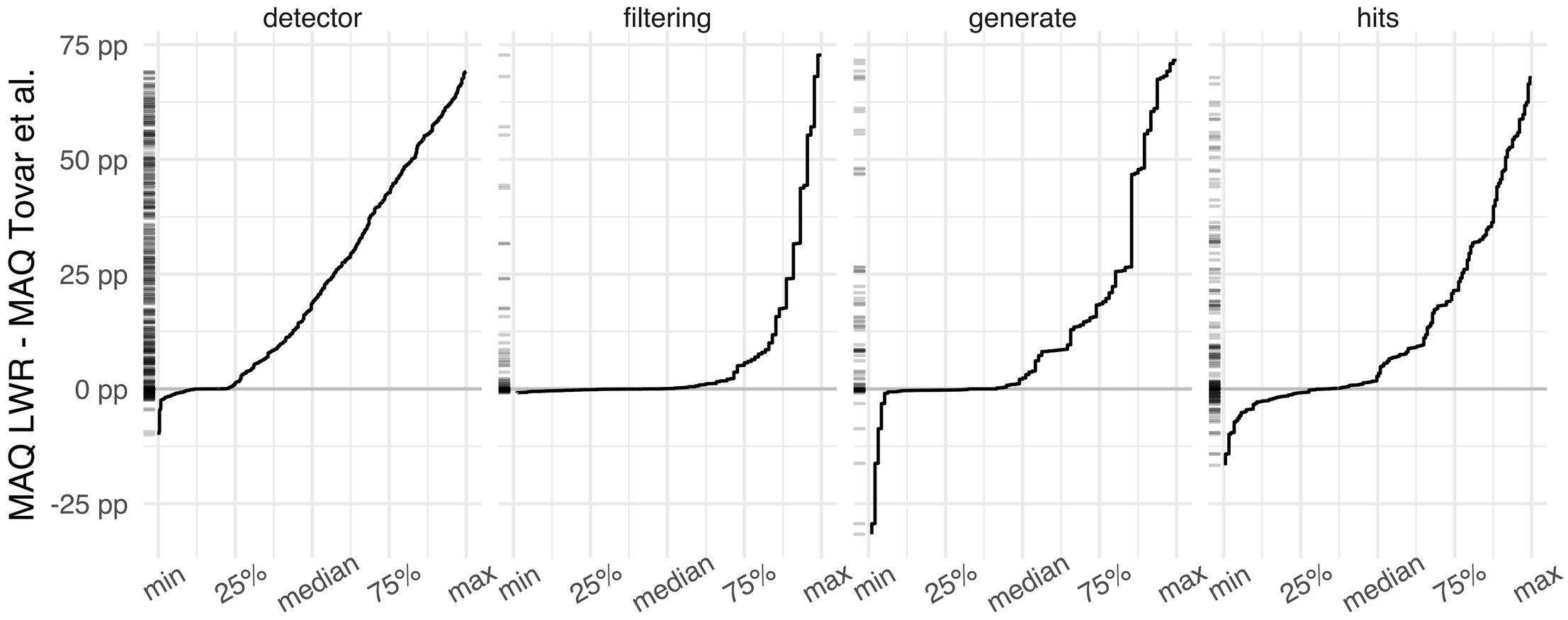




Goal: design prediction model that maximizes MAQ

MAQ Difference ECDFs

142 Abstract Tasks, Faceted by Executable



MAQ ECDF for Modified Tovar et al.

Train after having seen first *k*% of data
Evaluate MAQ on remaining tasks