Evaluating the Performance of Reinforcement Learning Algorithms

Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, Philip Thomas

Why do we care?

Performance evaluations:

1. Justify novel algorithms or enhancements
2. Tell us what algorithms to use

If done correctly:

- Can identify solved problems
- Place emphasis on areas that need more research

RL Algorithms for the Real-world

Want:

1. High levels of performance
2. No expert knowledge required

As a result:

1. less time tuning algorithms
2. More time solving harder problems

Algorithm Performance Evaluations

Typical evaluation procedure:

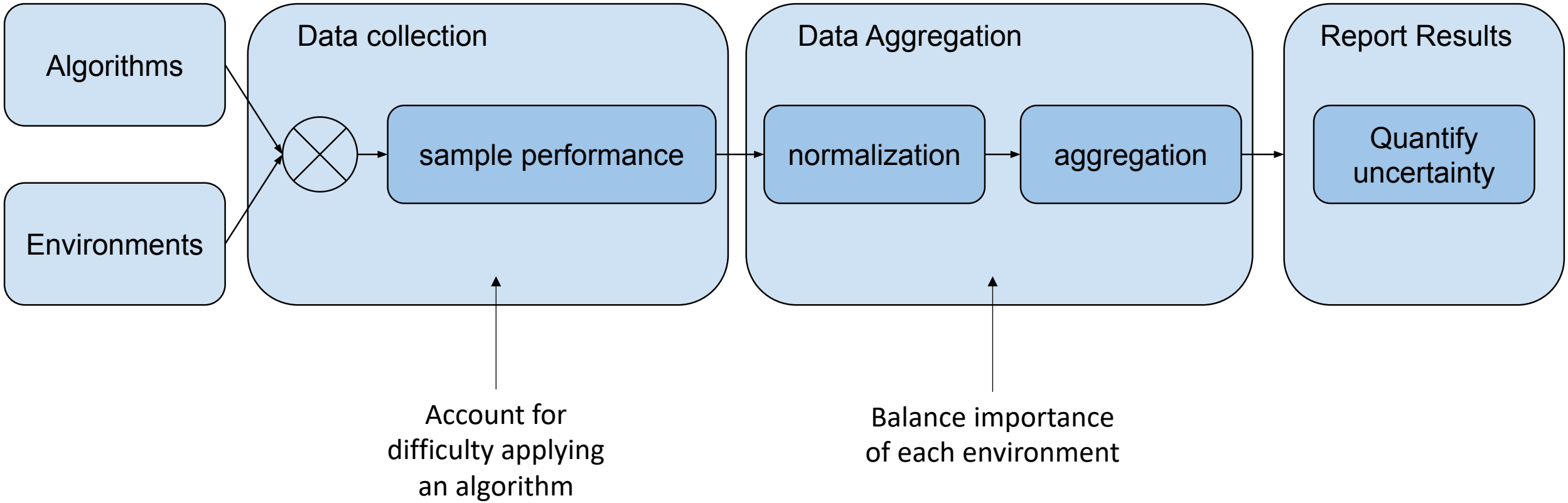
1. Tune each algorithm's hyperparameters (e.g., policy structure, learning rate)
2. Run several trials of using tune parameters
3. Report performance (metrics, learning curve, etc.)

Need a new evaluation procedure!

This does not fit our needs:

- Ignores the difficulty of applying algorithms

Evaluation Pipeline



A General Evaluation Question

Which algorithm(s) perform well across a wide variety of environments with little or no environment–specific tuning?

Existing evaluation procedures cannot answer this question

We develop techniques for:

1. Sampling performance metrics that reflect knowledge of how to use the algorithm
2. Normalizing scores to account for the intrinsic difficulties of each environment
3. Balancing the importance of each environment in the aggregate measure
4. Computing uncertainty over the whole process

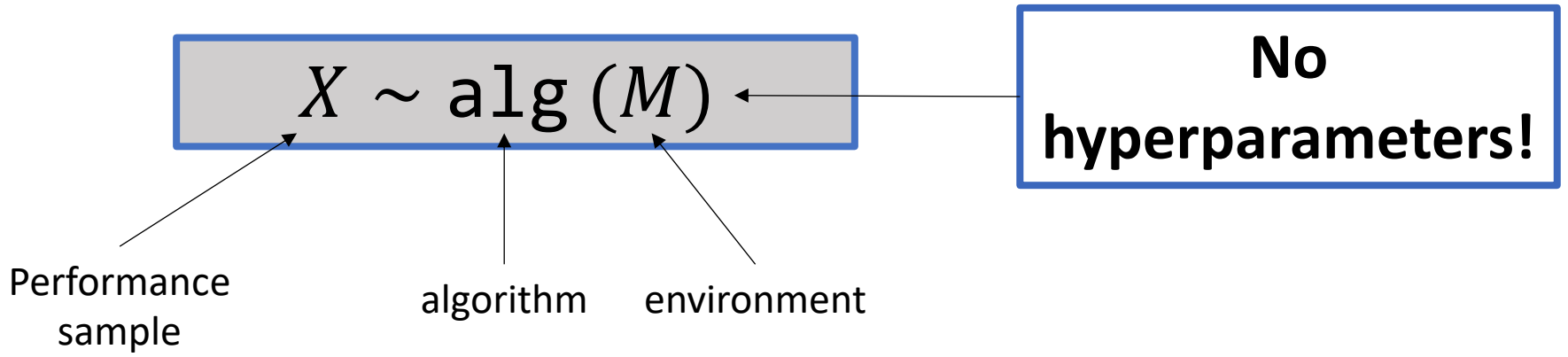
Sampling Performance Without Tuning

- Formalize knowledge to use an algorithm

Complete algorithm definition

Sampling Performance Without Tuning

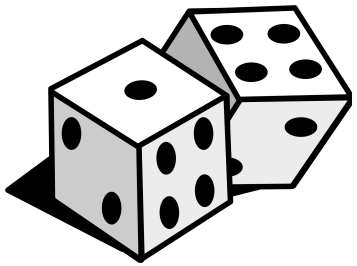
An algorithm is complete on an environment, when defined such that the only required input to the algorithm is the environment.



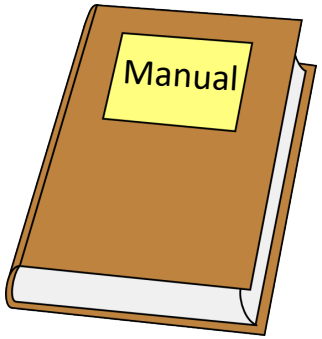
Making Complete Algorithm Definitions

- Open research question!

methods



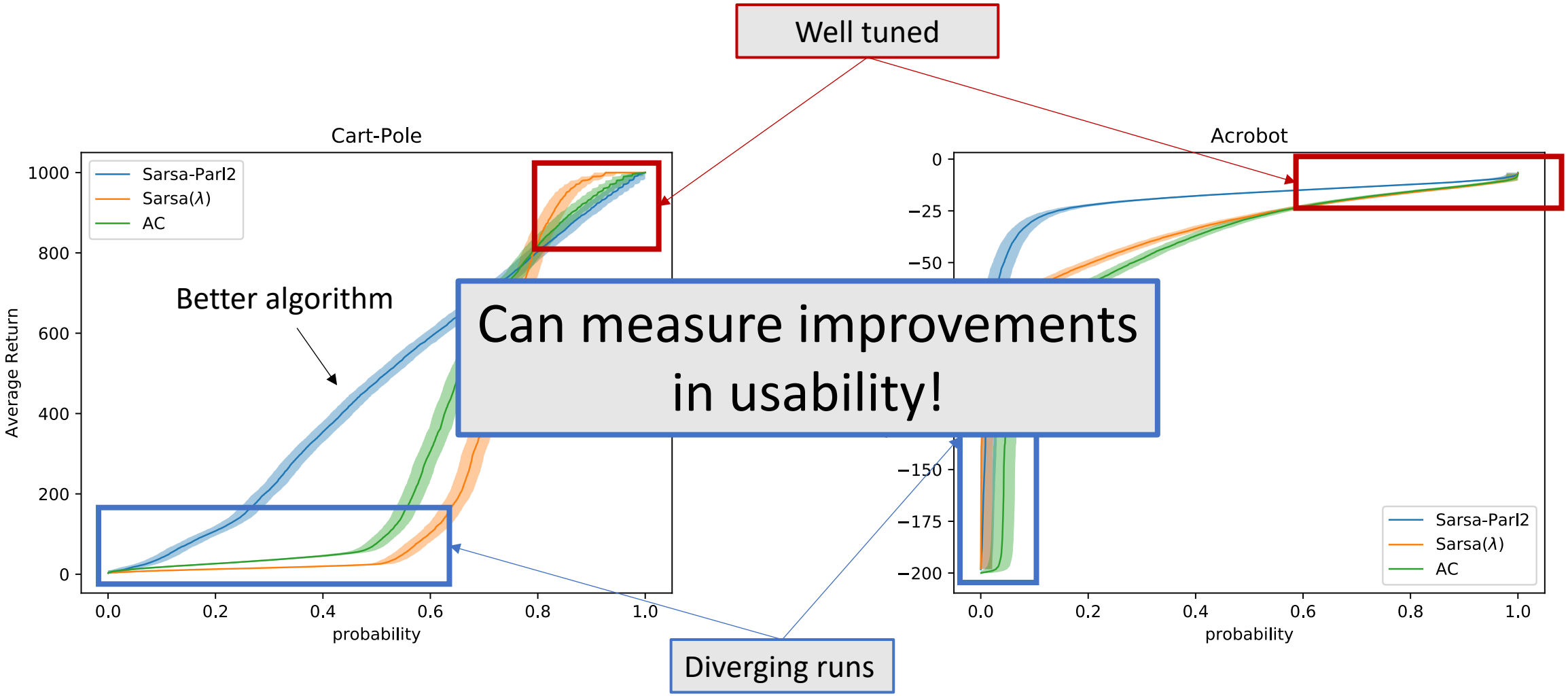
random sampling



smart heuristics

adaptive methods

Performance of Complete Algorithms



Comparisons Over Multiple Environments

Problem:

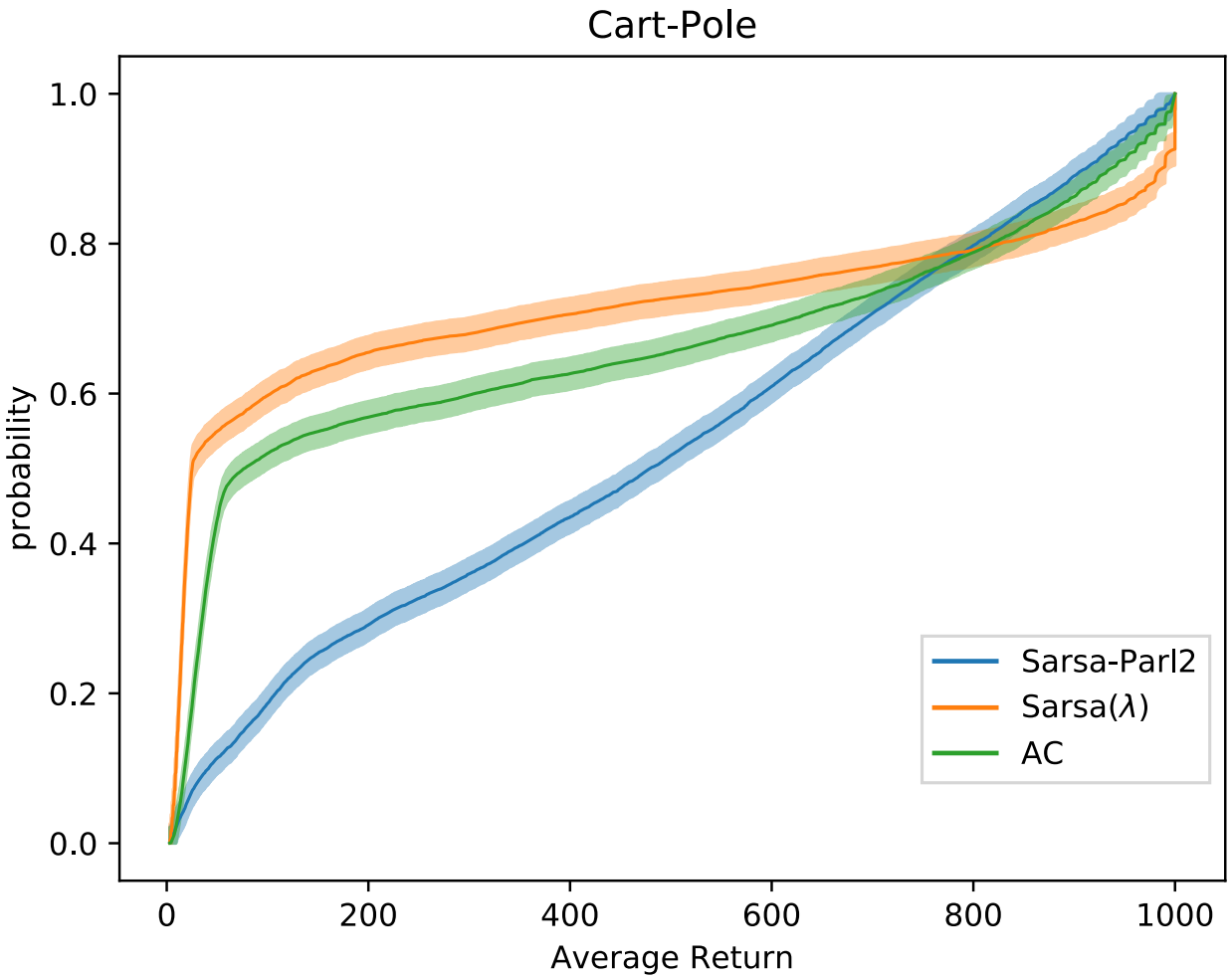
- No common measure of performance

Desired normalization properties:

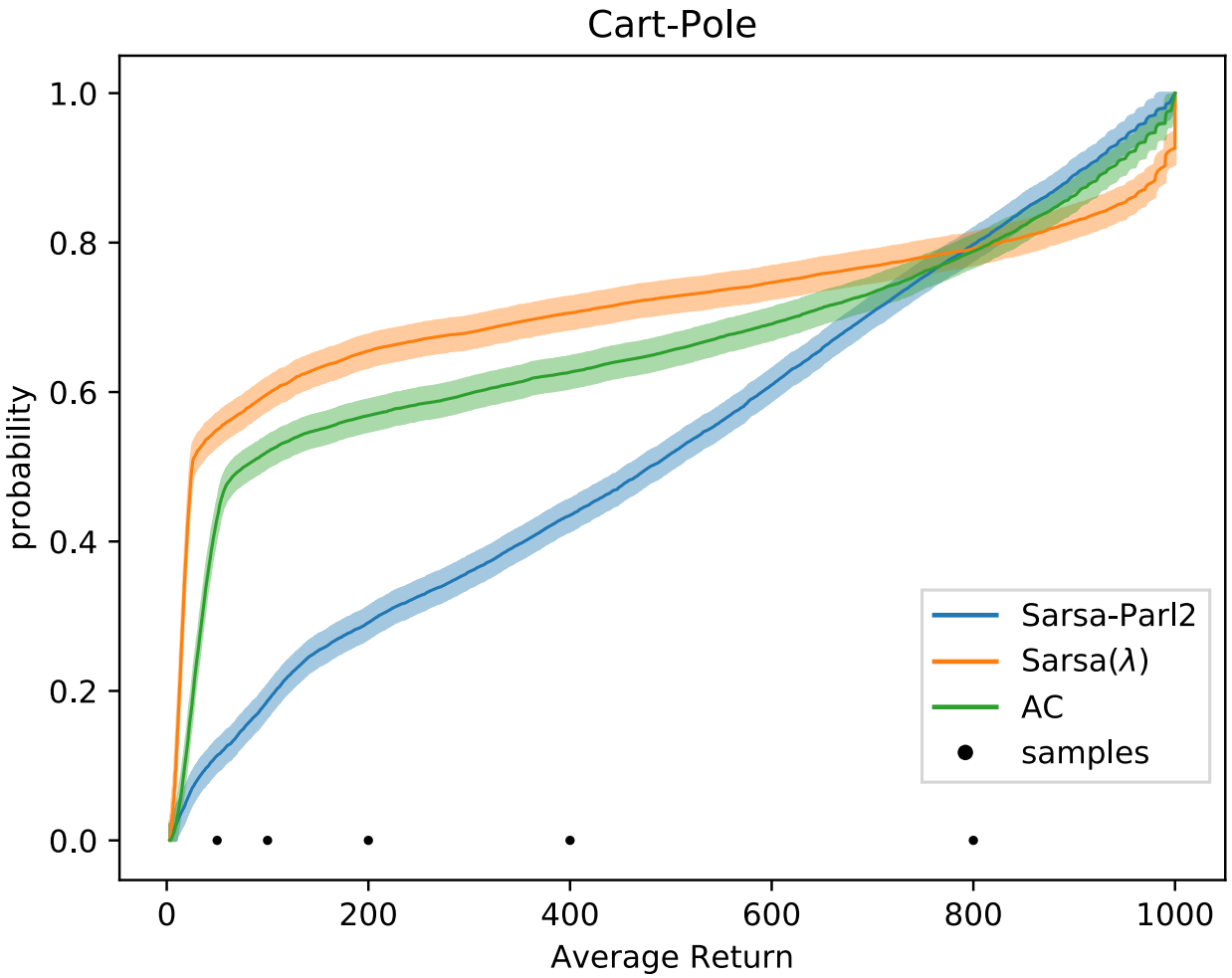
- Same scale and center
- Capture intr

Use cumulative distribution function

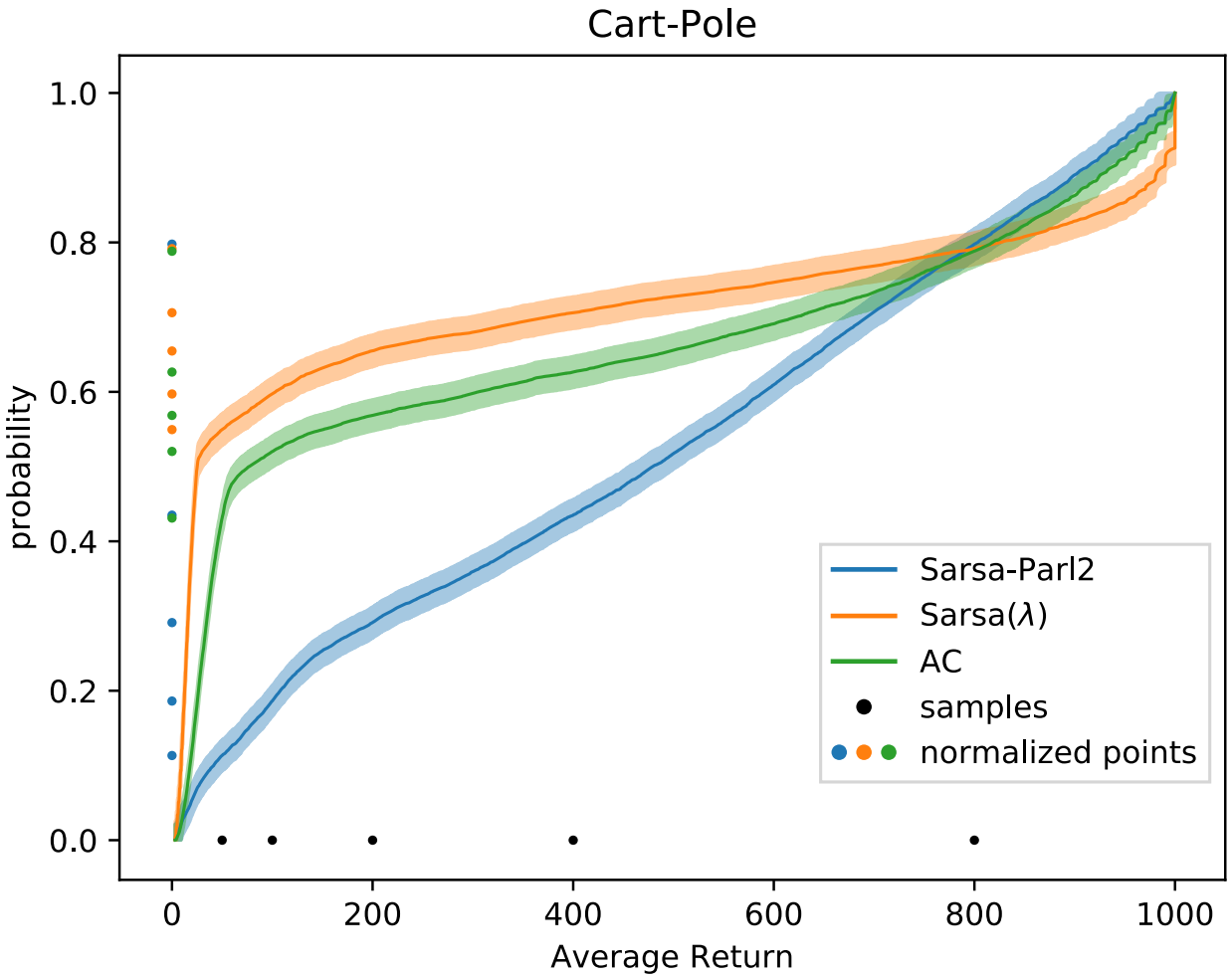
Normalizing Scores



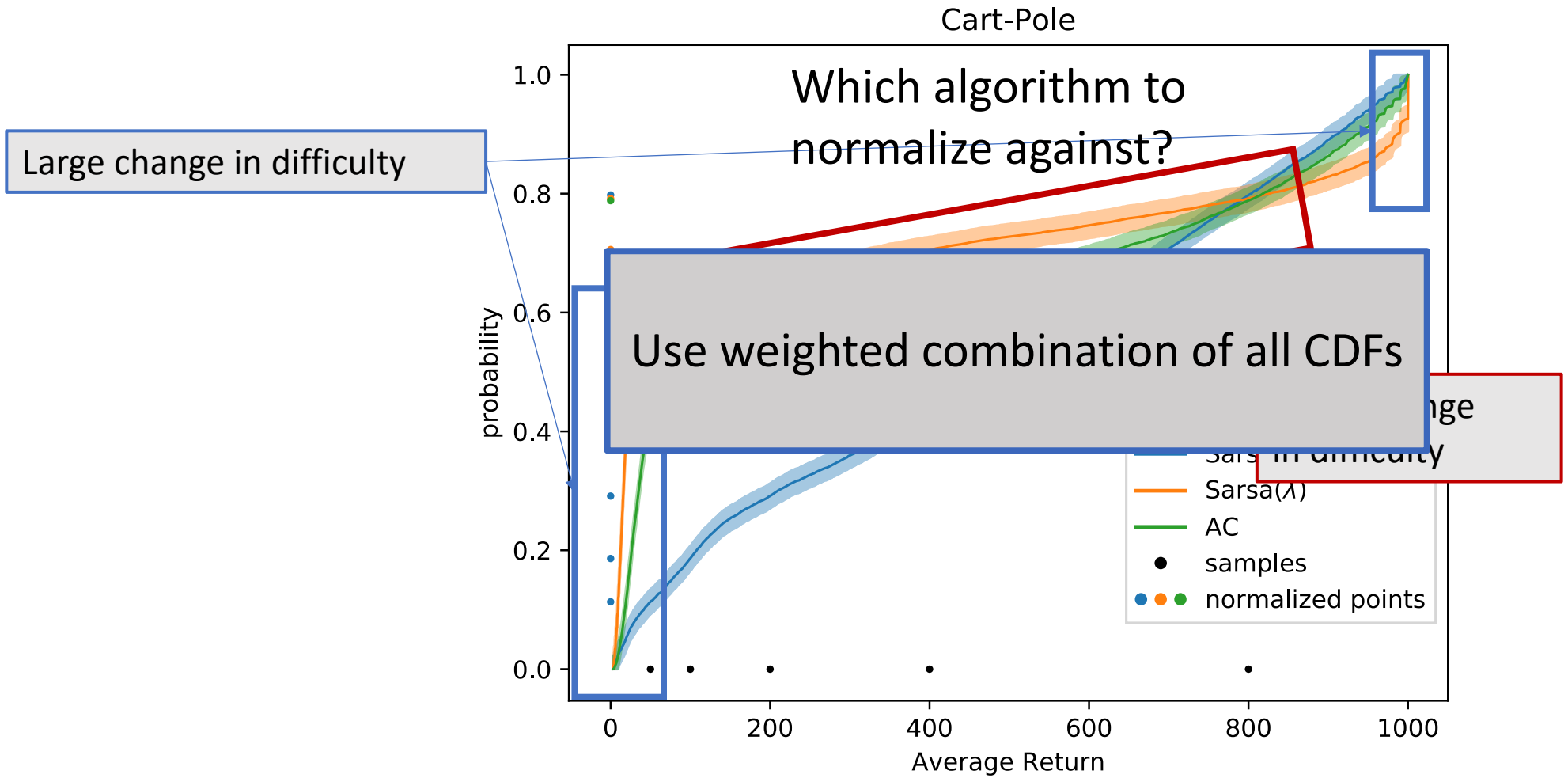
Normalizing Scores



Normalizing Scores



Normalizing Scores



Aggregating Performance Measures

- Need to weight normalization functions
- Need to weight environments
- Avoid unintentional bias in weightings

Use game theory!

$$Z_x = \sum_{m}^{\mathcal{M}} q_m \sum_{y}^{\mathcal{A}} q_y \mathbf{E}\left[F_y(x(m))\right]$$

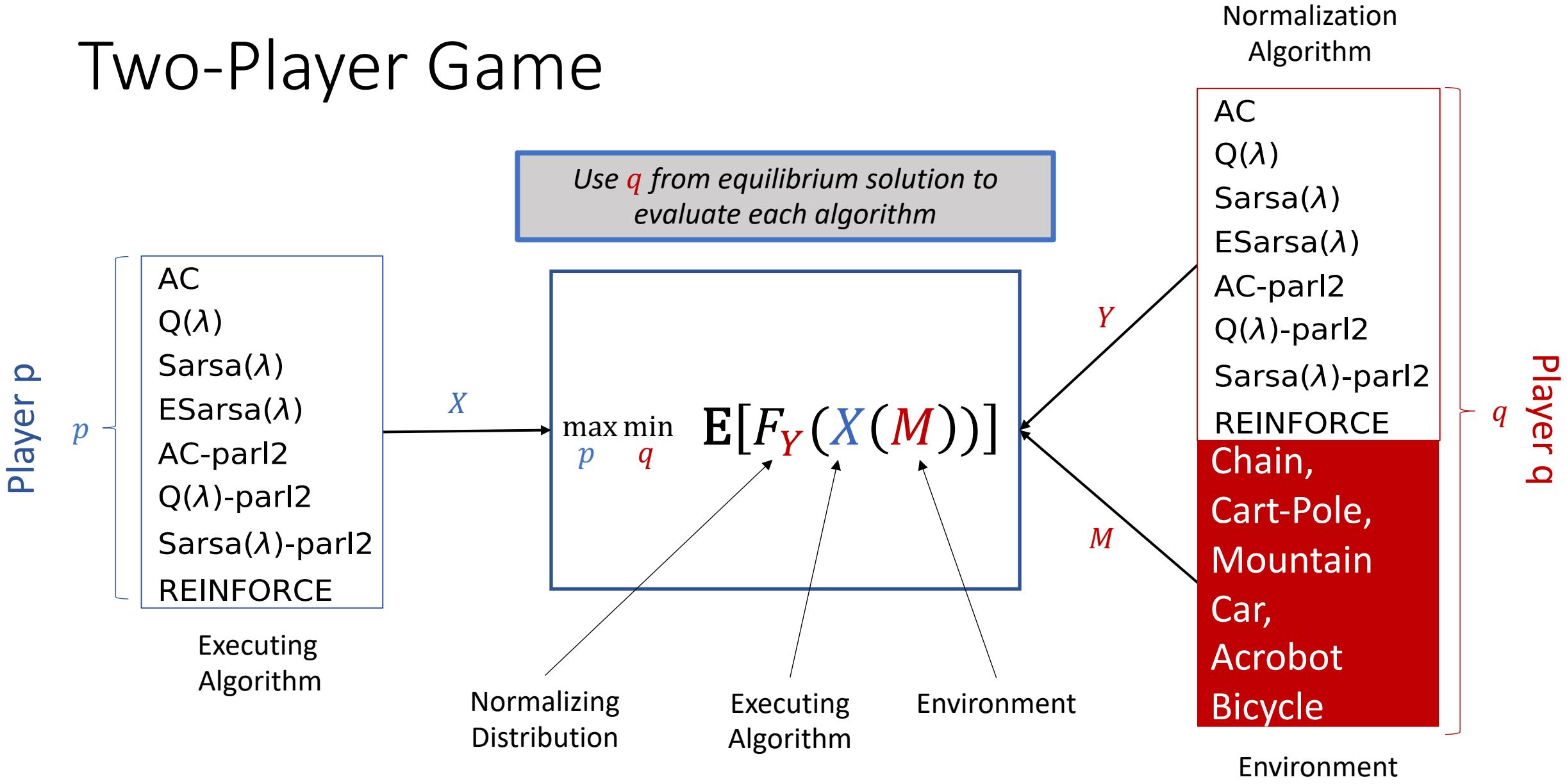
Aggregate performance of algorithm x

Environment weights

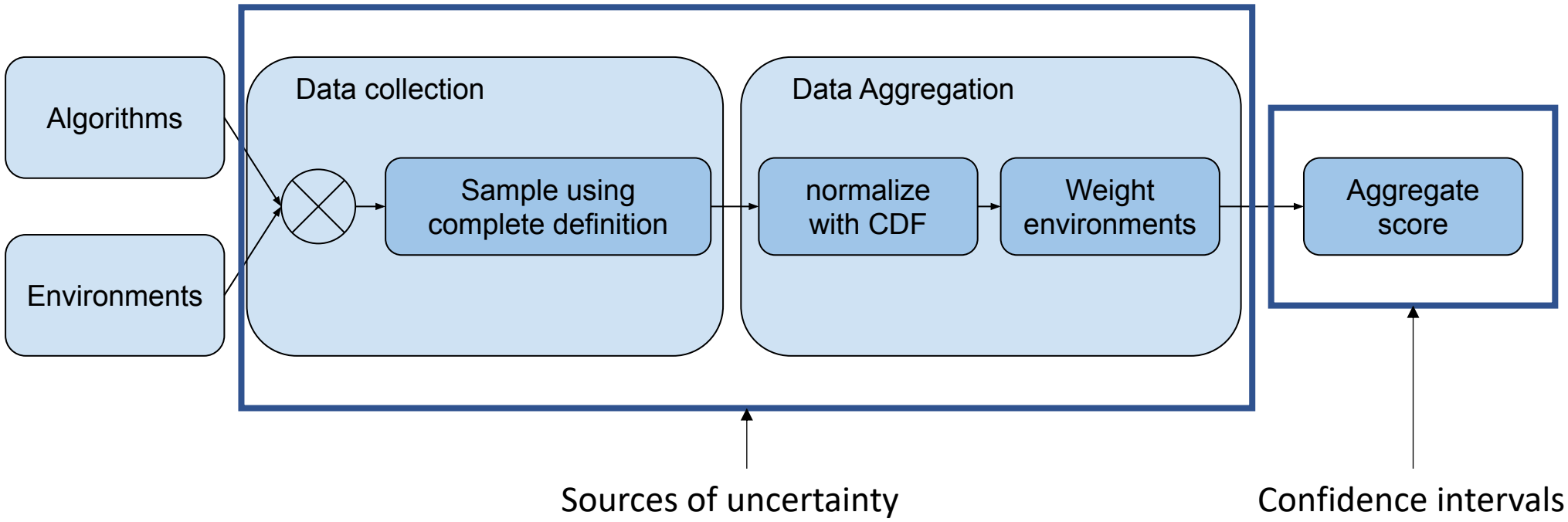
Normalization weights

Normalization function

Two-Player Game



Quantifying Uncertainty

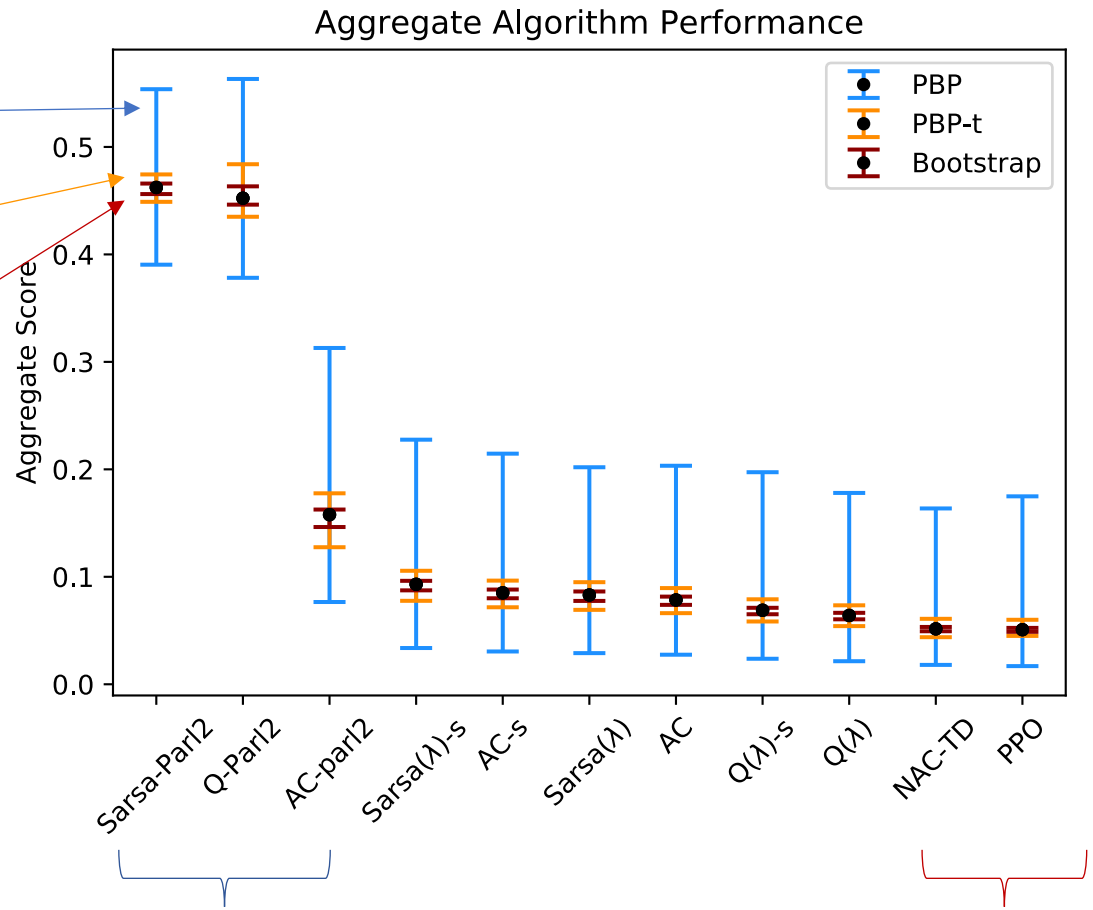


Quantifying Uncertainty

Valid for any distribution

Assumptions of normality

No guarantee



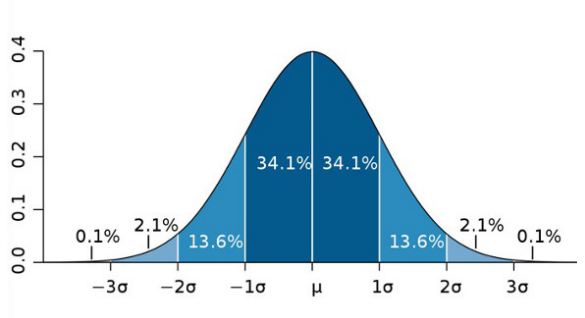
Adapt step sizes

Lots of hyperparameters

Takeaways

No need to tune hyperparameters

Can measure improvement in usability



Reliable estimates of uncertainty

Acknowledgements

Yash Chandak

Daniel Cohen

Mengxue Zhang

Prof. Philip S. Thomas

Questions?

Scott Jordan

sjordan@cs.umass.edu

http://cics.umass.edu/sjordan | @UMassScott