Generalization in Portfolio-based Algorithm Selection

Nina Balcan, Tuomas Sandholm, **Ellen Vitercik**

Carnegie Mellon University

Algorithm portfolios

Algorithms often have **many tunable parameters**

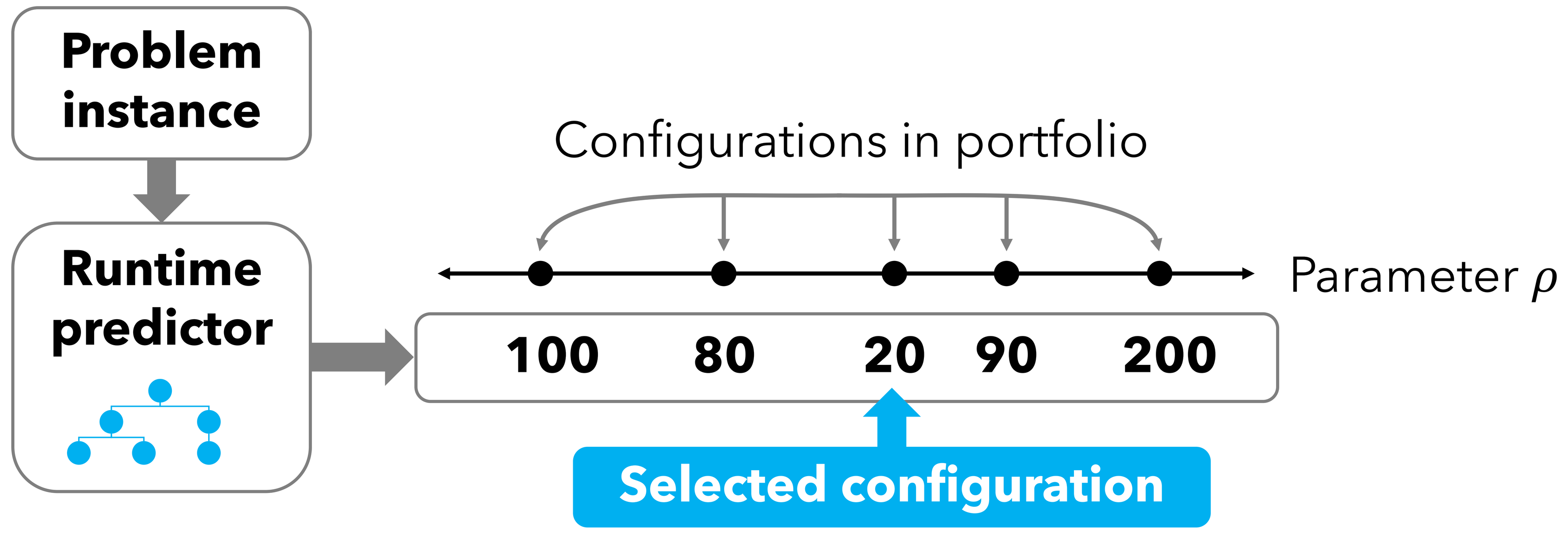
Significant impact on runtime, solution quality, …

Best configuration for one problem is rarely optimal for another

Portfolio-based algorithm selection

1. Compile diverse portfolio of parameter settings
2. At runtime, select one with strong predicted performance

Example:



Fueled breakthroughs in:

- SAT [Xu, Hutter, Hoos, Leyton-Brown, '08]
- Integer programming [Xu, Hoos, Leyton-Brown, '10]
- Combinatorial auction winner determination [Sandholm '13]
- Planning [Cenamor, De La Rosa, Fernández, '16]

Learning a portfolio and selector

1. Fix parameterized algorithm
2. Receive training set S of "typical" inputs

Training set: Problem instance 1 | Problem instance 2 | Problem instance 3 | • • •

Unseen ?: Problem instance

3. Use S to learn a **portfolio** $\hat{\mathcal{P}}$ of configurations and a **selector** $\hat{f}$ that maps problem instances to $\hat{\mathcal{P}}$

Key question: On **future** inputs,

Will the configuration $\hat{f}$ selects have good performance?

Sources of generalization error

1. Portfolio **size**
2. Learning-theoretic complexity of the **algorithm selector**
3. Learning-theoretic complexity of: the algorithm's **performance** as a function of its parameters

Model

$\mathcal{Z}$: Set of all inputs (e.g., IPs)

$\mathbb{R}$: Set of all parameter settings (e.g., CPLEX parameter)

Unknown distribution $\mathcal{D}$ over inputs

E.g., airline scheduling problems

$u_\rho(z)$ = utility of algorithm parameterized by $\rho \in \mathbb{R}$ on input z

Runtime, solution quality, …

$u_z^*(\rho)$ = utility as a function of parameter

Assumption: $u_z^*(\rho)$ is piecewise constant with $\leq t$ pieces

Holds in:

- Integer programming — Balcan, Dick, Sandholm, **Vitercik**, '18
- Clustering — Balcan, Nagarajan, **Vitercik**, White, '17; Balcan, Dick, White, 18; Balcan, Dick, Lang, '20
- Computational biology — Balcan, DeBlasio, Dick, Kingsford, Sandholm, **Vitercik**, '20
- Greedy algorithm configuration — Gupta, Roughgarden, '16

Main result

WHP over $z_1, \ldots, z_N \sim \mathcal{D}$:

|**Avg** utility of configurations selected by $\hat{f}$ - **expected** utility|

$$\leq \tilde{O}\left(\sqrt{(d + \kappa \log t)/N}\right)$$

d, κ: *Intrinsic complexity* of the set of algorithm selectors; t: Portfolio size

Strong **average** performance ➡ Strong **future** performance

Takeaway:

As portfolio grows, can have good configuration for any input,

…but it becomes **impossible** to avoid **overfitting**

Linear performance model

Xu, Hutter, Hoos, Leyton-Brown '08

Configurations in portfolio $\hat{\mathcal{P}}$ along parameter ρ

Input z with **features** $\phi(z) \in \mathbb{R}^m$

Linear models	$\hat{w}^1 \in \mathbb{R}^m$	$\hat{w}^2$	$\hat{w}^3$
Predicted performance	$\hat{w}^1 \cdot \phi(z)$	$\hat{w}^2 \cdot \phi(z)$	$\hat{w}^3 \cdot \phi(z)$

Error bound: $\tilde{O}\left(\sqrt{m\kappa/N}\right)$

In the paper:

- Regression tree performance models [Hutter, Xu, Hoos, Leyton-Brown, '14]
- Clustering-based algorithm selectors [Kadioglu, Malitsky, Sellmann, Tierney, '10]

Experiments: Branch-and-bound