# Towards White-Box Benchmarks for Algorithm Control

André Biedenkapp, H. Furkan Bozkurt, Frank Hutter, Marius Lindauer,
University of Freiburg

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

## In a Nutshell

- **Algorithm *configuration***, often necessary to achieve peak performance over a set of instances (e.g. AI Planning and SAT)
- It has been shown that **different parameter settings are optimal at different stages** of an algorithm
- **Algorithm *control*** adjusts parameters depending on an observed state
- **Goal:** Learn this control policy from data

## Related Work

- RL for Algorithm Control:
  - *Battiti and Campigotto (2012)* applied Least Squares Policy Iteration to learn a policy of **one SAT parameter**
  - *Daniel et al. (2016)* used Relative Entropy Policy Search to **learn a controller for the step size of NN optimizers**.
- *Jaderberg et al. (2017)* use Population Based Training to adjust the hyperparameters of RL agents over time

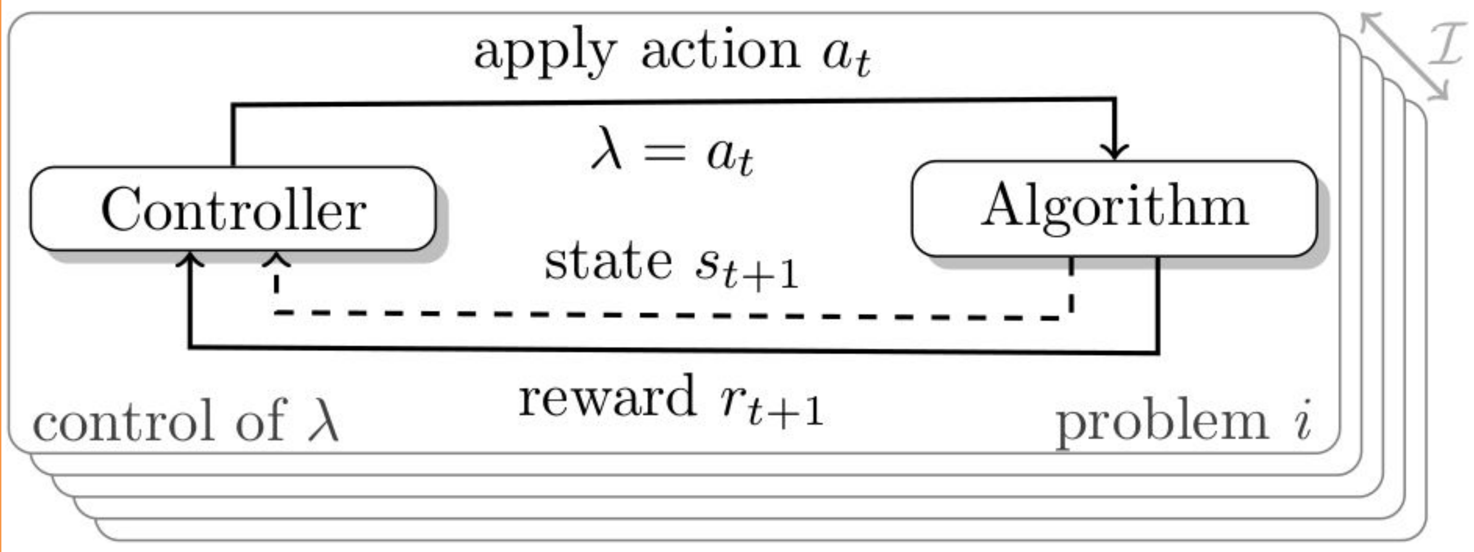
## Definition: Algorithm Control

Given:

- a parameterized algorithm $\mathcal{A}$ with a parameter **configuration space** $\Lambda$,
- a **state description** $s_t \in \mathcal{S}$ of algorithm $\mathcal{A}$ at each time point $t$,
- a space of **control policies** $\mathbf{\Pi}$: $\mathcal{S} \rightarrow \Lambda$ mapping from states to configurations, and
- a **cost metric** $c$ assessing the cost of a control policy $\boldsymbol{\pi}$ by running $\mathcal{A}$ with $\boldsymbol{\pi}$ (e.g., runtime or accuracy), and
- a **set of problems** $\mathcal{I}$

**Goal**: obtain a control policy $\boldsymbol{\pi}^*_{\mathcal{I}} \in \mathbf{\Pi}$ with minimal cost $c$.

## RL for Algorithm Control



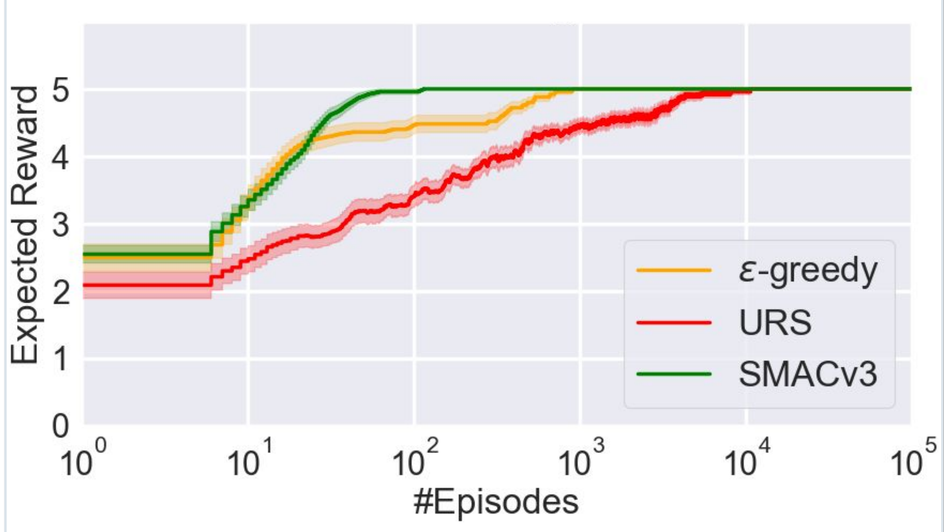

Control of hyperparameter λ at time step $t$ on problem $i$.
State $s_t$ is given by some internal statistics of the Algorithm.

## Insights Gained on White-Box Benchmarks

### Agents

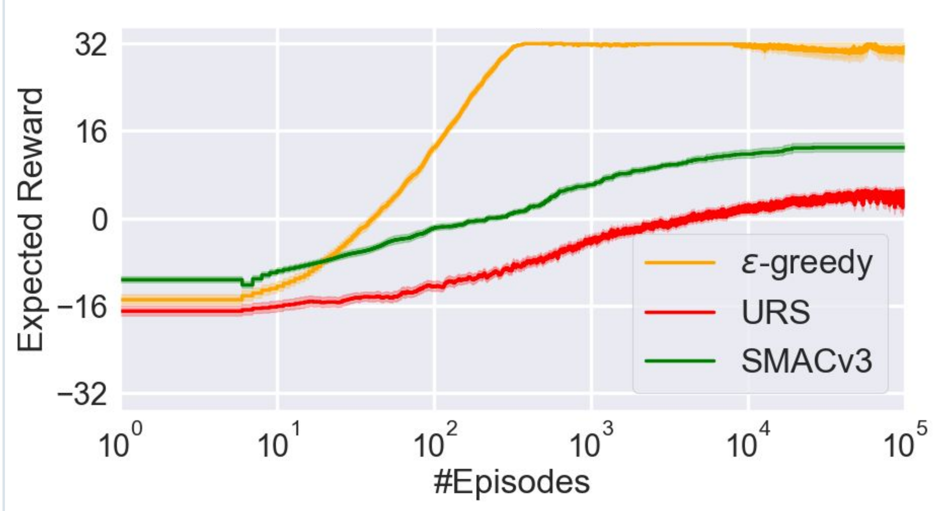
- *ε*-greedy : Tabular Q-learning with exploration factor 0.1
- URS : Tabular Q-learning with exploration factor 1.0
- SMACv3 : Black-box Bayesian optimization (BO)
- DQN : Q-learning with NN function approximator

### Counting



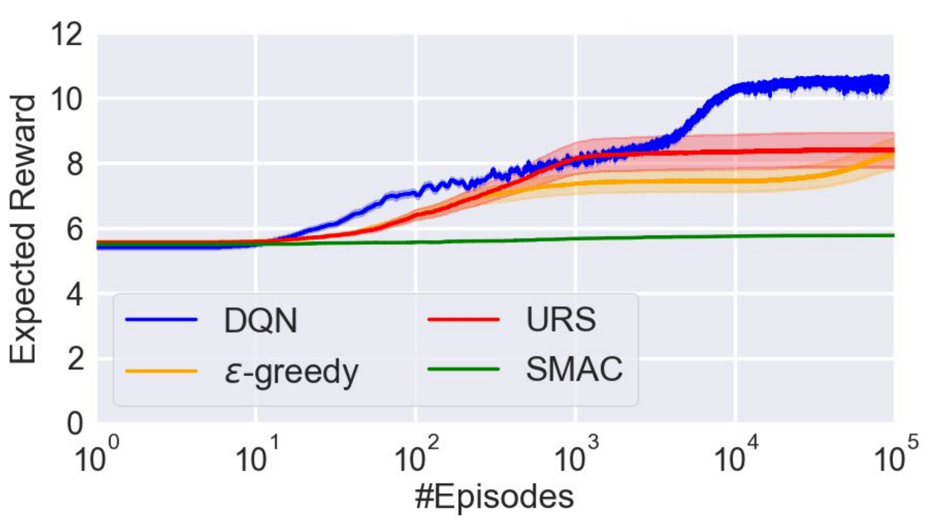

BO already performs well for short policies and small action spaces

### Luby



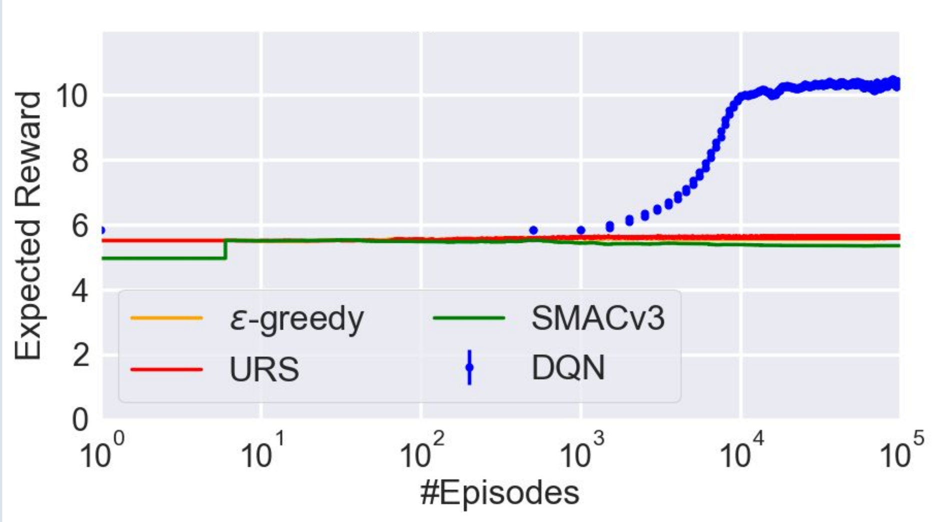

RL quickly learns to handle larger action spaces and long policies

### Sigmoid TRAIN




Learning across heterogeneous problems impossible in a black-box view

### Sigmoid TEST




Function approximation is needed for generalization

### State Features

- RL not only depends on a good reward function but also good state features
- Plethora of instance-feature we can use as part of the state-space
- What is a good feature describing algorithms?
- What temporal information can we encode?

Paper:

AutoML.org

@automlfreiburg