Generalization Bounds for Model-based Algorithm Configuration

Zhiyang Chen, Hailong Yao, Xia Yin

Tsinghua University, University of Science and Technology Beijing

NEURAL INFORMATION PROCESSING SYSTEMS

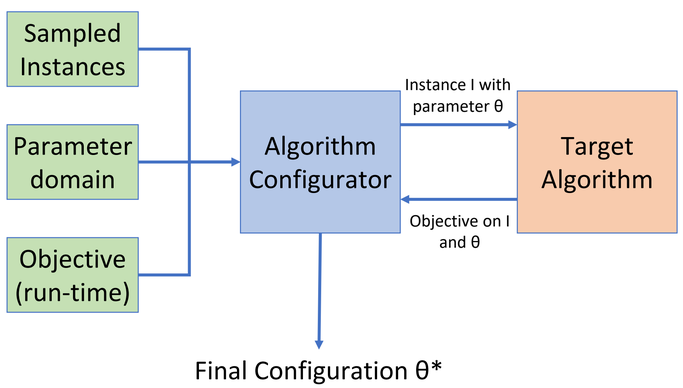
Algorithm Configuration

Algorithms in practice have **tunable parameters** which greatly affect the performance.
A carefully selected parameter configuration makes the algorithm perform strong on particular instances.

Domain	Default vs. Optimized
Answer Set Solving [Gebser et al. 2011]	up to $14\times$ speedup
AI Planning [Vallati et al. 2013]	up to $40\times$ speedup
Mixed Integer Programming [Hutter et al. 2010]	up to $52\times$ speedup
Satisfiability Solving [Hutter et al. 2017]	up to $3000\times$ speedup
Minimum Vertex Cover [Wagner et al. 2017]	up to 9% absolute impr.
Machine Learning [Feuer et al. 2015]	up to 35% absolute impr.
Deep Learning [Zimmer et al. 2020]	up to 49% absolute impr.

(Lindauer and Biedenkapp, 2020)

Manually tuning parameters are time-consuming.
Automated **algorithm configurators** (AC) are proposed to find promising parameters.
- Sample a set of problem instances as a training set.
- Find a high-performing configuration on this set.



Model-based Algorithm Configuration

Model-based AC is the most popular approach in practice, based on a Bayesian optimization framework:

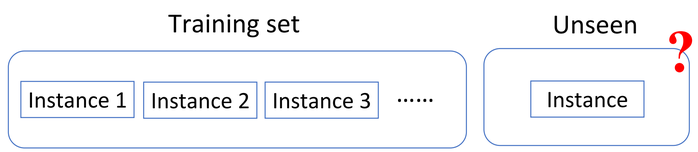
1. Randomly sample a number of initial parameters.
2. Evaluate these params on the training set.
3. Build a dataset of tuples <param, instance feature, algorithm performance>, and train a **surrogate model** predicting the performance.
4. Use a local search method to find a promising param configuration based on the surrogate model.
5. Evaluate this configuration, update the dataset and Goto Step 3 until the iteration limit is exceeded.

Generalization Bounds

Though model-based AC performs well in practice, a theoretical question remains unsolved:

The parameters found by algorithm configurators are evaluated on sampled problem instances from a distribution. Why do they perform well on other unseen instances?

Our contributions: We answer this question by presenting a generalization guarantee for model-based AC under mild assumptions.



Expected performance on distribution — Average performance on training set

$$|\mathbb{E}_{I\sim\mathcal{D}}[u(\tilde{\theta},I)] - \frac{1}{m}\sum_{i=1}^{m} u(\tilde{\theta},I_i)|$$

Our Settings

- Continuous parameter space in $[0,1]^n$ and bounded performance metric in [0,1]
- Random forest surrogate model
- Expected improvement acquisition function
- Sampling promising parameters using Metropolis-Hastings

Our Main Theorem

- m i.i.d. instances
- Q decision trees in the random forest model
- n parameters and d features of problem instances
- T iterations
- The expected generalization error is

$$\tilde{O}\left(\sqrt{T\left(\frac{1}{m}+\sqrt{\frac{n+d}{Q}}\right)}\right)$$

Our Technique

- Generalization bound via algorithmic stability
- Prove the stability of model-based AC via bounding the KL divergence between two perturbed path
- A uniform stability result on the random forest model via a dual Rademacher complexity bound
- A stability result of the Metropolis-Hasting