AutoML: Beyond AutoML

Overview: Algorithm Configuration

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance
  2. you care about the empirical performance of an algorithm

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
    1. the algorithm at hand has parameters that influence its performance
    2. you care about the empirical performance of an algorithm
- a limitation of HPO is that we assume that we care only about a single task (i.e., dataset or input to the algorithm)

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance
  2. you care about the empirical performance of an algorithm
- a limitation of HPO is that we assume that we care only about a single task (i.e., dataset or input to the algorithm)

$\rightsquigarrow$ Can we find an algorithm's configuration that performs well and robustly across a set of tasks?

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance
  2. you care about the empirical performance of an algorithm
- a limitation of HPO is that we assume that we care only about a single task (i.e., dataset or input to the algorithm)

$\rightsquigarrow$ Can we find an algorithm's configuration that performs well and robustly across a set of tasks?

- A hyperparameter configuration for a set of datasets

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance
  2. you care about the empirical performance of an algorithm
- a limitation of HPO is that we assume that we care only about a single task (i.e., dataset or input to the algorithm)

$\leadsto$ Can we find an algorithm's configuration that performs well and robustly across a set of tasks?

- A hyperparameter configuration for a set of datasets
- A parameter configuration of a SAT solver for a set of SAT instances

Generalization of HPO

- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance
  2. you care about the empirical performance of an algorithm
- a limitation of HPO is that we assume that we care only about a single task (i.e., dataset or input to the algorithm)

$\rightsquigarrow$ Can we find an algorithm's configuration that performs well and robustly across a set of tasks?

- A hyperparameter configuration for a set of datasets
- A parameter configuration of a SAT solver for a set of SAT instances
- A parameter configuration of an AI planning solver for a set of planning problems
- ...

Generalization of HPO

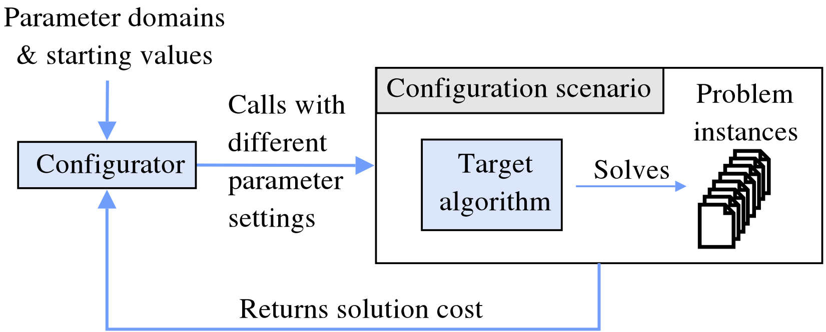
- hyperparameter optimization (HPO) is not limited to ML
- in fact, you can optimize the performance of any algorithm by means of HPO if
  1. the algorithm at hand has parameters that influence its performance
  2. you care about the empirical performance of an algorithm
- a limitation of HPO is that we assume that we care only about a single task (i.e., dataset or input to the algorithm)

$\rightsquigarrow$ Can we find an algorithm's configuration that performs well and robustly across a set of tasks?

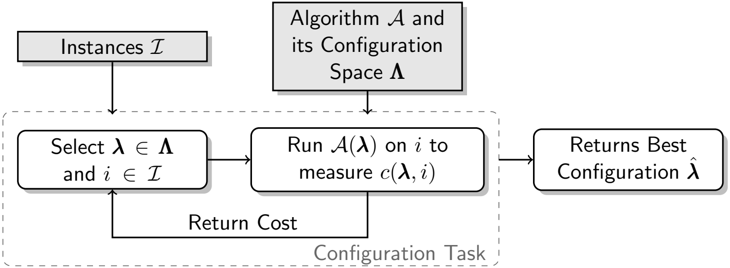
- A hyperparameter configuration for a set of datasets
- A parameter configuration of a SAT solver for a set of SAT instances
- A parameter configuration of an AI planning solver for a set of planning problems
- ...

$\rightsquigarrow$ Algorithm configuration

Algorithm Configuration Visualized



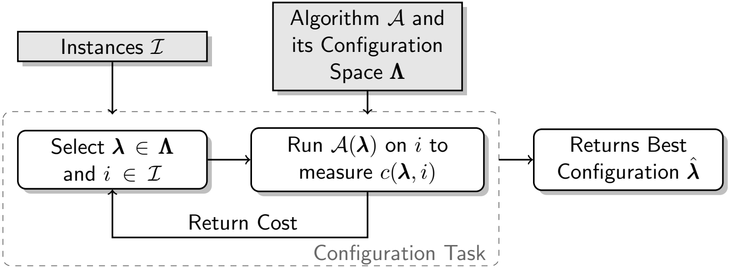
Algorithm Configuration – in More Detail



Definition

Given a parameterized algorithm $\mathcal{A}$ with possible (hyper-)parameter settings $\mathbf{\Lambda}$,

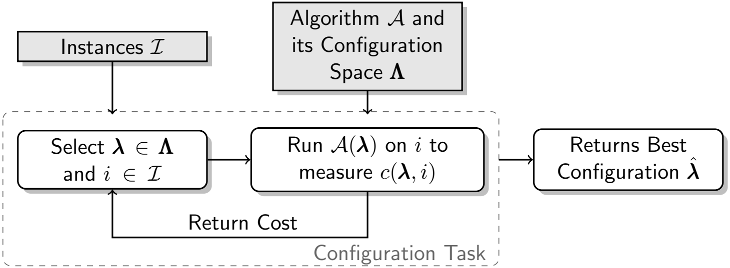
Algorithm Configuration – in More Detail



Definition

Given a parameterized algorithm $\mathcal{A}$ with possible (hyper-)parameter settings $\boldsymbol{\Lambda}$, a set of training problem instances $\mathcal{I}$,

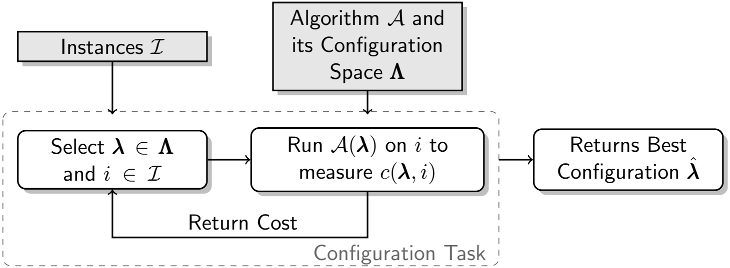
Algorithm Configuration – in More Detail



Definition

Given a parameterized algorithm $\mathcal{A}$ with possible (hyper-)parameter settings $\boldsymbol{\Lambda}$, a set of training problem instances $\mathcal{I}$, and a cost metric $c : \boldsymbol{\Lambda} \times \mathcal{I} \to \mathbb{R}$,

Algorithm Configuration – in More Detail



Definition

Given a parameterized algorithm $\mathcal{A}$ with possible (hyper-)parameter settings $\boldsymbol{\Lambda}$, a set of training problem instances $\mathcal{I}$, and a cost metric $c : \boldsymbol{\Lambda} \times \mathcal{I} \to \mathbb{R}$, the algorithm configuration problem is to find a parameter configuration $\boldsymbol{\lambda}^* \in \boldsymbol{\Lambda}$ that minimizes c across the instances in $\mathcal{I}$.

Algorithm Configuration – Full Formal Definition

Definition

An instance of the algorithm configuration problem is a 5-tuple $(\mathcal{A}, \mathbf{\Lambda}, \mathcal{D}, \kappa, c)$ where:

- $\mathcal{A}$ is a parameterized algorithm;
- $\mathbf{\Lambda}$ is the (hyper-)parameter configuration space of $\mathcal{A}$;
- $\mathcal{D}$ is a distribution over problem instances with domain $\mathcal{I}$;

Algorithm Configuration – Full Formal Definition

Definition

An instance of the algorithm configuration problem is a 5-tuple $(\mathcal{A}, \mathbf{\Lambda}, \mathcal{D}, \kappa, c)$ where:

- $\mathcal{A}$ is a parameterized algorithm;
- $\mathbf{\Lambda}$ is the (hyper-)parameter configuration space of $\mathcal{A}$;
- $\mathcal{D}$ is a distribution over problem instances with domain $\mathcal{I}$;
- $\kappa < \infty$ is a cutoff time, after which each run of $\mathcal{A}$ will be terminated if still running

Algorithm Configuration – Full Formal Definition

Definition

An instance of the algorithm configuration problem is a 5-tuple $(\mathcal{A}, \mathbf{\Lambda}, \mathcal{D}, \kappa, c)$ where:

- $\mathcal{A}$ is a parameterized algorithm;
- $\mathbf{\Lambda}$ is the (hyper-)parameter configuration space of $\mathcal{A}$;
- $\mathcal{D}$ is a distribution over problem instances with domain $\mathcal{I}$;
- $\kappa < \infty$ is a cutoff time, after which each run of $\mathcal{A}$ will be terminated if still running
- $c : \mathbf{\Lambda} \times \mathcal{I} \rightarrow \mathbb{R}$ is a function that measures the observed cost of running $\mathcal{A}(\boldsymbol{\lambda})$ on an instance $i \in \mathcal{I}$ with cutoff time κ

Algorithm Configuration – Full Formal Definition

Definition

An instance of the algorithm configuration problem is a 5-tuple $(\mathcal{A}, \mathbf{\Lambda}, \mathcal{D}, \kappa, c)$ where:

- $\mathcal{A}$ is a parameterized algorithm;
- $\mathbf{\Lambda}$ is the (hyper-)parameter configuration space of $\mathcal{A}$;
- $\mathcal{D}$ is a distribution over problem instances with domain $\mathcal{I}$;
- $\kappa < \infty$ is a cutoff time, after which each run of $\mathcal{A}$ will be terminated if still running
- $c : \mathbf{\Lambda} \times \mathcal{I} \to \mathbb{R}$ is a function that measures the observed cost of running $\mathcal{A}(\boldsymbol{\lambda})$ on an instance $i \in \mathcal{I}$ with cutoff time κ

The cost of a candidate solution $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ is $f(\boldsymbol{\lambda}) = \mathbb{E}_{i \sim \mathcal{D}}(c(\boldsymbol{\lambda}, i))$.
The goal is to find $\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \mathbf{\Lambda}} f(\boldsymbol{\lambda})$.

Distribution of Instances

We usually have a finite number of instances from a given application

- We want to do well on that type of instances
- Future instances of this type should be solved well

Distribution of Instances

We usually have a finite number of instances from a given application

- We want to do well on that type of instances
- Future instances of this type should be solved well

Like in machine learning

- We split the instances into a training set and a test set
- We configure algorithms on the training instances
- We only use the test instances afterwards
  $\rightarrow$ unbiased estimate of generalization performance for unseen instances

Challenges of Algorithm Configuration

- Structured high-dimensional parameter space
  - categorical vs. continuous parameters
  - conditionals between parameters

Challenges of Algorithm Configuration

- Structured high-dimensional parameter space
  - categorical vs. continuous parameters
  - conditionals between parameters

- Stochastic optimization
  - Randomized algorithms: optimization across various seeds
  - Distribution of benchmark instances (often wide range of hardness)
  - Subsumes so-called *multi-armed bandit problem*

Challenges of Algorithm Configuration

- Structured high-dimensional parameter space
  - categorical vs. continuous parameters
  - conditionals between parameters
- Stochastic optimization
  - Randomized algorithms: optimization across various seeds
  - Distribution of benchmark instances (often wide range of hardness)
  - Subsumes so-called *multi-armed bandit problem*
- Generalization across instances
  - apply algorithm configuration to homogeneous instance sets
  - Instance sets can also be heterogeneous,
    i.e., no single configuration performs well on all instances

  $\leadsto$ combination of algorithm configuration and selection

Challenges of Algorithm Configuration

- Structured high-dimensional parameter space
  - categorical vs. continuous parameters
  - conditionals between parameters
- Stochastic optimization
  - Randomized algorithms: optimization across various seeds
  - Distribution of benchmark instances (often wide range of hardness)
  - Subsumes so-called *multi-armed bandit problem*
- Generalization across instances
  - apply algorithm configuration to homogeneous instance sets
  - Instance sets can also be heterogeneous,
    i.e., no single configuration performs well on all instances

  $\leadsto$ combination of algorithm configuration and selection

$\leadsto$ Hyperparameter optimization is a subproblem of algorithm configuration

[Eggensperger et al. 2019]