Empirical Hardness Models: Methods and Uses

Lars Kotthoff

University of British Columbia
larsko@cs.ubc.ca

1QBit, 18 October 2016
Big Picture

▷ advance the state of the art through meta-algorithmic techniques
▷ rather than inventing new things, use existing things better
▷ “supplement” our understanding of how algorithms work
Big Picture

- advance the state of the art through meta-algorithmic techniques
- rather than inventing new things, use existing things better
- “supplement” our understanding of how algorithms work
- Algorithm Selection: choose the best algorithm to solve a given problem
- Algorithm Configuration: find the best parameterisation
- at the core of both: empirical performance models
Leveraging the Differences

Performance Improvements

Algorithm Selection

Given a problem, choose the best algorithm to solve it.
Contemporary Model

- $x \in \mathcal{P}$: Problem space
- $A \in \mathcal{A}$: Algorithm space
- $S(x) = A$: Selection model
- $p \in \mathcal{R}^n$: Performance measure

Feature extraction

∀$x \in \mathcal{P}' \subset \mathcal{P}$, $A \in \mathcal{A}$: $p(A, x)$

Training data

Feedback
Portfolios

- instead of a single algorithm, use several complementary algorithms
- idea from Economics – minimise risk by spreading it out across several securities
- same for computational problems – minimise risk of algorithm performing poorly
- in practice often constructed from competition winners

---

Key Components of an Algorithm Selection System

▷ feature extraction
▷ performance model
▷ prediction-based selector/scheduler

optional:
▷ presolver
▷ secondary/hierarchical models and predictors (e.g. for feature extraction time)
Features

- relate properties of problem instances to performance
- relatively cheap to compute
- specified by domain expert
- syntactic – analyse instance description
- probing – run algorithm for short time
- dynamic – instance changes while algorithm is running
Syntactic Features

▷ number of variables, number of clauses/constraints/…
▷ ratios
▷ order of variables/values
▷ clause/constraints–variable graph or variable graph:
  ▷ node degrees
  ▷ connectivity
  ▷ clustering coefficient
  ▷ ...
▷ ...
▷ ...
Probing Features

- number of nodes/propagations within time limit
- estimate of search space size
- tightness of problem/constraints
- ...

...
Dynamic Features

- change of variable domains
- number of constraint propagations
- number of failures a clause participated in
- ...
What Features do we need in Practice?

- trade-off between complex features and complex models
- in practice, very simple features (e.g. problem size) can perform well, depending on the model used
Types of Performance Models

▷ models for entire portfolios
▷ models for individual algorithms
▷ models that are somewhere in between (e.g. pairs of algorithms)
Models for Entire Portfolios

▷ predict the best algorithm in the portfolio
▷ alternatively: cluster and assign best algorithms to clusters

optional (but important):
▷ attach a “weight” during learning (e.g. the difference between best and worst solver) to bias model towards the “important” instances
▷ special loss metric


Models for Individual Algorithms

- predict the performance for each algorithm separately
- combine the predictions to choose the best one
- for example: predict the runtime for each algorithm, choose the one with the lowest runtime

---

Hybrid Models

▷ get the best of both worlds
▷ for example: consider pairs of algorithms to take relations into account
▷ for each pair of algorithms, learn model that predicts which one is faster


Types of Predictions/Algorithm Selectors

▷ best algorithm
▷ $n$ best algorithms ranked
▷ allocation of resources to $n$ algorithms
▷ change the currently running algorithm?


Time of Prediction

before problem is being solved
▷ select algorithm(s) once
▷ no recourse if predictions are bad

while problem is being solved
▷ continuously monitor problem features and/or performance
▷ can remedy bad initial choice or react to changing problem
Example System – SATzilla

- 7 SAT solvers, 4811 problem instances
- syntactic (33) and probing features (15)
- ridge regression to predict log runtime for each solver, choose the solver with the best predicted performance
- later version uses random forests to predict better algorithm for each pair, aggregation through simple voting scheme
- pre-solving, feature computation time prediction, hierarchical model
- won several competitions

---

Algorithm Configuration

Given a (set of) problem(s), find the best parameter configuration.
Algorithm Configuration

Local Search

- start with random configuration
- change a single parameter (local search step)
- if better, keep the change, else revert
- repeat
- optional (but important): restart with new random configurations

Local Search Example

graphics by Holger Hoos
Local Search Example

Initialisation

graphics by Holger Hoos
Local Search Example

Local Search

graphics by Holger Hoos
Local Search Example

Local Search

graphics by Holger Hoos
Local Search Example

Perturbation

graphics by Holger Hoos
Local Search Example

Local Search

graphics by Holger Hoos
Local Search Example

Going Beyond Local Optima: Iterated Local Search

Animation credit: Holger Hoos

Hutter & Lindauer AC-Tutorial AAAI 2016, Phoenix, USA 33

Local Search

graphics by Holger Hoos
Local Search Example

Local Search

Animation credit: Holger Hoos
Hutter & Lindauer AC-Tutorial AAAI 2016, Phoenix, USA

graphics by Holger Hoos
Local Search Example

Selection (using Acceptance Criterion)

graphics by Holger Hoos
Model-Based Search

▷ build model of parameter-response surface
▷ allows targeted exploration of new configurations
▷ (can take instance features into account like algorithm selection)

Model-Based Search Example

Iter = 1, Gap = 0.0000e+00

graphics by Bernd Bischl with mlrMBO R package
Model-Based Search Example

Iter = 6, Gap = 0.0000e+00

Graphics by Bernd Bischl with mlrMBO R package
Model-Based Search Example

Iter = 13, Gap = 0.0000e+00

graphics by Bernd Bischl with mlrMBO R package
Model-Based Search Example

Iter = 20, Gap = 0.0000e+00

graphics by Bernd Bischl with mlrMBO R package
Overtuning

▷ similar to overfitting in machine learning
▷ performance improves on training instances, but not on test instances
▷ configuration is too “tailored”, e.g. specific to satisfiable instances
▷ but can be combined with Algorithm Selection
Algorithm Selection  choose the best *algorithm* for solving a problem

Algorithm Configuration  choose the best *parameter configuration* for solving a problem with an algorithm

▷ mature research areas
▷ can combine configuration and selection
▷ effective tools are available
Tools and Resources

Algorithm Selection

- **autofolio** [https://bitbucket.org/mlindauer/autofolio/](https://bitbucket.org/mlindauer/autofolio/)
- **LLAMA** [https://bitbucket.org/lkotthoff/llama](https://bitbucket.org/lkotthoff/llama)
- **SATzilla** [http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/](http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/)

Algorithm Configuration

- **iRace** [http://iridia.ulb.ac.be/irace/](http://iridia.ulb.ac.be/irace/)
- **mlrMBO** [https://github.com/mlr-org/mlrMBO](https://github.com/mlr-org/mlrMBO)
- **Spearmint** [https://github.com/HIPS/Spearmint](https://github.com/HIPS/Spearmint)
- **TPE** [https://jaberg.github.io/hyperopt/](https://jaberg.github.io/hyperopt/)

---

Interested? Join the COSEAL group!

http://coseal.net