AI-Augmented Algorithms – How I Learned to Stop Worrying and Love Choice

Lars Kotthoff

University of Wyoming
larsko@uwyo.edu

St Andrews, 25 July 2018
Outline

▷ Big Picture
▷ Motivation
▷ Algorithm Selection and Portfolios
▷ Algorithm Configuration
▷ Outlook
Big Picture

▷ advance the state of the art through meta-algorithmic techniques
▷ rather than inventing new things, use existing things more intelligently – automatically
▷ invent new things through combinations of existing things
Big Picture

▷ advance the state of the art through meta-algorithmic techniques
▷ rather than inventing new things, use existing things more intelligently – automatically
▷ invent new things through combinations of existing things
Motivation – What Difference Does It Make?
Performance Differences

Leveraging the Differences

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean PAR10</th>
<th>Number Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB Proteus</td>
<td>97</td>
<td>1493</td>
</tr>
<tr>
<td>Proteus</td>
<td>1774</td>
<td>1424</td>
</tr>
<tr>
<td>VB CSP</td>
<td>3577</td>
<td>1349</td>
</tr>
<tr>
<td>VB SAT</td>
<td>17373</td>
<td>775</td>
</tr>
<tr>
<td>VB DirectOrder Encoding</td>
<td>17637</td>
<td>764</td>
</tr>
<tr>
<td>VB Direct Encoding</td>
<td>21736</td>
<td>593</td>
</tr>
<tr>
<td>VB Support Encoding</td>
<td>21986</td>
<td>583</td>
</tr>
</tbody>
</table>

Performance Improvements

Common Theme

Performance models of black-box processes

▷ also called surrogate models
▷ replace expensive underlying process with cheap approximate model
▷ build approximate model based on real evaluations using machine learning techniques
▷ no knowledge of what the underlying process does required (but can be helpful)
▷ allow better understanding of the underlying process through interrogation of the model
Algorithm Selection
Algorithm Selection

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

Algorithm Selection

Portfolio

Algorithm 1  Algorithm 2  Algorithm 3

Training Instances

Instance 1  Instance 2  Instance 3

Feature Extraction

Algorithm Selection

Performance Model

Instance 4: Algorithm 2
Instance 5: Algorithm 3
Instance 6: Algorithm 3

Feature Extraction

Instance 4
Instance 5
Instance 6
...
Algorithm 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

“algorithm” used in a very loose sense
▷ algorithms
▷ heuristics
▷ machine learning models
▷ consistency levels
▷ ...

Algorithms
Parallel Portfolios

Why not simply run all algorithms in parallel?
- not enough resources may be available/waste of resources
- algorithms may be parallelized themselves
- memory contention
Building an Algorithm Selection System

▷ most approaches rely on machine learning
▷ train with representative data, i.e. performance of all algorithms in portfolio on a number of instances
▷ evaluate performance on separate set of instances
▷ potentially large amount of prep work
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)
Types of Performance Models

Regression Models

- A1: 1.2
- A2: 4.5
- A3: 3.9

Classification Model

Pairwise Classification Models

- A1 vs. A2: A1: 1 vote, A2: 0 votes
- A1 vs. A3: A3: 2 votes

Pairwise Regression Models

- A1 - A2: 0
- A1 - A3: 0
- A3: 1.7

Instance 1: Algorithm 2
Instance 2: Algorithm 1
Instance 3: Algorithm 3
Benchmark Library – ASlib

- currently 29 data sets/scenarios with more in preparation
- SAT, CSP, QBF, ASP, MAXSAT, OR, machine learning...
- includes data used frequently in the literature that you may want to evaluate your approach on
- performance of common approaches that you can compare to
- http://aslib.net

### Algorithm Selection literature summary

<table>
<thead>
<tr>
<th>citation</th>
<th>domain</th>
<th>features</th>
<th>predict what</th>
<th>predict how</th>
<th>predict when</th>
<th>portfolio</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Langley 1989.</td>
<td>search</td>
<td>past efficiency</td>
<td>algorithm</td>
<td>hand-crafted and learned rules</td>
<td>offline and online</td>
<td>dynamic</td>
<td>1989</td>
</tr>
<tr>
<td>Carbonell et al. 1991</td>
<td>planning</td>
<td>problem domain features, search statistics</td>
<td>control rules</td>
<td>explanation-based rule construction</td>
<td>online</td>
<td>dynamic</td>
<td>1991</td>
</tr>
<tr>
<td>Gratch and DeJong 1992</td>
<td>planning</td>
<td>problem domain features, search statistics</td>
<td>control rules</td>
<td>probabilistic rule construction</td>
<td>online</td>
<td>dynamic</td>
<td>1992</td>
</tr>
<tr>
<td>Smith and Selby 1992</td>
<td>software design</td>
<td>features of abstract representation</td>
<td>algorithms and data structures</td>
<td>simulated annealing</td>
<td>offline</td>
<td>static</td>
<td>1992</td>
</tr>
<tr>
<td>Aine 1992</td>
<td>machine learning</td>
<td>instance features</td>
<td>algorithm</td>
<td>learned rules</td>
<td>offline</td>
<td>static</td>
<td>1992</td>
</tr>
<tr>
<td>Brodie 1993</td>
<td>machine learning</td>
<td>instance features</td>
<td>algorithm</td>
<td>hand-crafted rules</td>
<td>offline</td>
<td>static</td>
<td>1993</td>
</tr>
<tr>
<td>Kamel et al. 1993</td>
<td>differential equations</td>
<td>past performance, instance features</td>
<td>algorithm</td>
<td>hand-crafted rules</td>
<td>offline</td>
<td>static</td>
<td>1993</td>
</tr>
<tr>
<td>Cuhil 1994</td>
<td>software design</td>
<td>instance features</td>
<td>algorithms and data structures</td>
<td>frame-based knowledge base</td>
<td>offline</td>
<td>static</td>
<td>1994</td>
</tr>
<tr>
<td>Tang et al. 1995</td>
<td>constraints</td>
<td>instance features</td>
<td>algorithms and data structures</td>
<td>-</td>
<td>static</td>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>Breier 1995</td>
<td>software design</td>
<td>runtime performance</td>
<td>algorithms, data structures and their parameters</td>
<td>-</td>
<td>static</td>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>Huberman et al. 1997</td>
<td>graph colouring</td>
<td>resource allocation</td>
<td>statistical model</td>
<td>offline</td>
<td>static</td>
<td>1997</td>
<td></td>
</tr>
<tr>
<td>Gomes and Selman 1997b, Gomes and Selman 1997a</td>
<td>constraints</td>
<td>problem size and past performance</td>
<td>algorithm</td>
<td>statistical model</td>
<td>offline</td>
<td>static</td>
<td>1997</td>
</tr>
<tr>
<td>Cook and Varel 1997</td>
<td>parallel search</td>
<td>probing</td>
<td>set of search strategies</td>
<td>decision trees, Bayesian classifier, nearest neighbour, neural net</td>
<td>online</td>
<td>static</td>
<td>1997</td>
</tr>
<tr>
<td>Fox 1997, Fox et al. 1998</td>
<td>planning</td>
<td>branch and bound</td>
<td>resource allocation</td>
<td>statistical model, regression</td>
<td>offline</td>
<td>static</td>
<td>1997</td>
</tr>
<tr>
<td>Cesaia et al. 1999</td>
<td>vehicle routing problem</td>
<td>runtime performance</td>
<td>algorithm</td>
<td>genetic algorithms</td>
<td>offline</td>
<td>static</td>
<td>1999</td>
</tr>
<tr>
<td>Howe et al. 1999</td>
<td>planning</td>
<td>instance features</td>
<td>resource allocation</td>
<td>linear regression</td>
<td>offline</td>
<td>static</td>
<td>1999</td>
</tr>
<tr>
<td>Taatila-Martín et al. 1999</td>
<td>scheduling</td>
<td>instance and search features</td>
<td>algorithm</td>
<td>genetic algorithms</td>
<td>offline</td>
<td>dynamic</td>
<td>1999</td>
</tr>
<tr>
<td>Wilson et al. 2000</td>
<td>software design</td>
<td>instance features</td>
<td>data structures</td>
<td>nearest neighbour</td>
<td>offline</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Beck and Fox 2000</td>
<td>job shop scheduling</td>
<td>instance feature changes</td>
<td>algorithm scheduling policy</td>
<td>hand-crafted rules</td>
<td>online</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Brazil and Goaes 2000</td>
<td>classification</td>
<td>past performance</td>
<td>ranking</td>
<td>distribution model</td>
<td>offline</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Lagoudakis and Littman 2000</td>
<td>order selection, sorting</td>
<td>instance features</td>
<td>remaining cost for each sub-problem</td>
<td>MDP</td>
<td>online</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Silio 2000</td>
<td>constraints</td>
<td>probing</td>
<td>cost of solving problem</td>
<td>statistical model</td>
<td>offline</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Pfehringer et al. 2000</td>
<td>classification</td>
<td>instance features, probing</td>
<td>algorithm</td>
<td>9 different classifiers</td>
<td>online</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Furukawa 2000</td>
<td>TSP</td>
<td>past performance</td>
<td>resource allocation</td>
<td>performance simulation for different allocations</td>
<td>offline</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Soares and Brazil 2000</td>
<td>machine learning</td>
<td>instance features</td>
<td>ranking</td>
<td>nearest neighbour</td>
<td>offline</td>
<td>static</td>
<td>2000</td>
</tr>
<tr>
<td>Gomes and Selman 2001</td>
<td>constraints, mixed integer programming</td>
<td>past performance</td>
<td>algorithm</td>
<td>statistical model</td>
<td>offline</td>
<td>static</td>
<td>2001</td>
</tr>
<tr>
<td>Lagoudakis and Littman 2001</td>
<td>optimal branching rules</td>
<td>instance features</td>
<td>remaining cost for each sub-problem</td>
<td>MDP</td>
<td>online</td>
<td>static</td>
<td>2001</td>
</tr>
<tr>
<td>Nau et al. 2001</td>
<td>optimisation</td>
<td>search statistics</td>
<td>expected utility of algorithm</td>
<td>reinforcement learning</td>
<td>offline and online</td>
<td>static</td>
<td>2001</td>
</tr>
<tr>
<td>Horvitz et al. 2001</td>
<td>constraints</td>
<td>instance and instance generator features, search statistics</td>
<td>runtime performance, reward parameters</td>
<td>Bayesian model</td>
<td>offline and online</td>
<td>static</td>
<td>2001</td>
</tr>
<tr>
<td>Barrett and Tsang 2001</td>
<td>constraints</td>
<td>instance features, search</td>
<td>redundant constraints to add</td>
<td>hand-crafted rules</td>
<td>offline</td>
<td>static</td>
<td>2001</td>
</tr>
</tbody>
</table>

http://larskotthoff.github.io/assurvey/

Algorithm Configuration
Algorithm Configuration

Given a (set of) problem(s), find the best parameter configuration.
Parameters?

▷ anything you can change that makes sense to change
▷ e.g. search heuristic, variable ordering, type of global constraint decomposition
▷ not random seed, whether to enable debugging, etc.
▷ some will affect performance, others will have no effect at all
Automated Algorithm Configuration

▷ no background knowledge on parameters or algorithm
▷ as little manual intervention as possible
 ▷ failures are handled appropriately
 ▷ resources are not wasted
 ▷ can run unattended on large-scale compute infrastructure
Algorithm Configuration

Parameter domains & starting values

Configurator

Calls with different parameter settings

Configuration scenario

Target algorithm

Solves

Problem instances

Returns solution cost

General Approach

▷ evaluate algorithm as black box function
▷ observe effect of parameters without knowing the inner workings
▷ decide where to evaluate next
▷ balance diversification/exploration and intensification/exploitation
When are we done?

▷ most approaches incomplete
▷ cannot prove optimality, not guaranteed to find optimal solution (with finite time)
▷ performance highly dependent on configuration space

→ How do we know when to stop?
Time Budget

How much time/how many function evaluations?
▷ too much → wasted resources
▷ too little → suboptimal result
▷ use statistical tests
▷ evaluate on parts of the instance set
▷ for runtime: adaptive capping
Grid and Random Search

▷ evaluate certain points in parameter space

**Grid Layout**

**Random Layout**

Model-Based Search

▷ evaluate small number of configurations
▷ build model of parameter-performance surface based on the results
▷ use model to predict where to evaluate next
▷ repeat
▷ allows targeted exploration of new configurations
▷ can take instance features into account like algorithm selection

Model-Based Search Example

Iter = 1, Gap = 1.9909e−01

$\begin{align*}
\text{type} & \quad \text{init} & \quad \text{prop} \\
y & \quad y & \quad yhat \\
ei & \quad e & \quad ei
\end{align*}$
Model-Based Search Example

Iter = 2, Gap = 1.9909e−01

- $\scriptstyle y$
- $\scriptstyle x$
- $\scriptstyle yhat$
- $\scriptstyle ei$
- $\scriptstyle type$
- $\scriptstyle init$
- $\scriptstyle prop$
- $\scriptstyle seq$

- $\scriptstyle y$
- $\scriptstyle yhat$
- $\scriptstyle ei$

- $\scriptstyle x$
- $\scriptstyle e_i$

- $\scriptstyle Iter = 2$, $\scriptstyle Gap = 1.9909e−01$
Model-Based Search Example

Iter = 3, Gap = 1.9909e−01

![Graph showing iterated and gap values](image)
Model-Based Search Example

Iter = 4, Gap = 1.9992e−01
Model-Based Search Example

Iter = 5, Gap = 1.9992e−01
Model-Based Search Example

Iter = 6, Gap = 1.9996e−01

Graph showing the search process with iterations and gap values.
Model-Based Search Example

Iter = 7, Gap = 2.0000e−01
Model-Based Search Example

Iter = 8, Gap = 2.0000e−01

- $x$ - $y$ - $y\text{hat}$
- $\varepsilon_i$

- init
- prop
- seq

Iter = 8, Gap = 2.0000e−01
Model-Based Search Example

Iter = 9, Gap = 2.0000e−01
Model-Based Search Example

Iter = 10, Gap = 2.0000e−01

Graph showing a curve with marked points for different types:
- **init** (red circle)
- **prop** (blue triangle)
- **seq** (green square)

X-axis: -1.0 to 1.0
Y-axis: -1.0 to 0.0

Graph also shows a yhat line (dashed) and an ei line (dotted) with specific values indicated.

Values:
- y: 0e+00, 1e−07, 2e−07, 3e−07, 4e−07
- ei: −1.0, −0.5, 0.0, 0.5, 1.0

Graph also includes annotations for different types:
Benchmark Library – AClib

▷ ASP, MIP, planning, machine learning, …
▷ 4 algorithm configuration tools from the literature already integrated
▷ https://bitbucket.org/mlindauer/aclib2

Outlook
Quo Vadis, Software Engineering?

Run

Meta-Algorithmics in the Physical Realm – AI and Lasers
Tools and Resources

LLAMA  https://bitbucket.org/lkotthoff/llama
SATzilla http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/
   iRace http://iridia.ulb.ac.be/irace/
mlrMBO https://github.com/mlr-org/mlrMBO
SMAC http://www.cs.ubc.ca/labs/beta/Projects/SMAC/
Spearmint https://github.com/HIPS/Spearmint
   TPE https://jaberg.github.io/hyperopt/
   autofolio https://bitbucket.org/mlindauer/autofolio/
Auto-WEKA http://www.cs.ubc.ca/labs/beta/Projects/autoweka/
Auto-sklearn https://github.com/automl/auto-sklearn
Summary

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
▷ COnfiguration and SElection of ALgorithms group COSEAL
  http://www.coseal.net

Don’t set parameters prematurely, embrace choice!
I’m hiring!

Several funded graduate positions available.