Automatic (Offline) Configuration & Design of Optimisation Algorithms

Manuel López-Ibáñez
manuel.lopez-ibanez@manchester.ac.uk

University of Manchester

October 27, 2015
CREST Open Workshop, London
What do we talk when we talk about hyper-heuristics?

- Online tuning / Parameter control, (self-)adaptation / Reactive search / Adaptive Selection
- Algorithm selection / Algorithm portfolios
- Offline tuning / Parameter configuration
Modern high-performance optimisers involve a large number of design choices and parameter settings

- **categorical** parameters
  - recombination $\in \{ \text{uniform, one-point, two-point} \}$
  - local search $\in \{ \text{tabu search, SA, ILS} \}$
- **ordinal** parameters
  - neighborhoods $\in \{ \text{small, medium, large} \}$
- **numerical** parameters
  - weighting factors, population sizes, temperature, hidden constants, ...

Parameters may be *conditional* to specific values of other parameters: --temperature if LS == "SA"

*Configuring algorithms involves setting categorical, ordinal and numerical parameters*
Manual tuning

Human expert + trial-and-error/statistics

Benchmark Problems

Problem Instances

Solver

Automatic Algorithm Configuration & Design
Towards more systematic approaches

**Traditional approaches**

- Trial–and–error design guided by expertise/intuition
  - prone to over-generalizations,
  - limited exploration of design alternatives,
  - human biases

- Guided by theoretical studies
  - often based on over-simplifications,
  - specific assumptions,
  - few parameters

Can we make this approach more principled and automatic?
Automatic Algorithm Configuration

1. Find the best parameter configuration of a solver from a set of *training problem instances*

2. Repeatedly use this configuration to solve *unseen problem instances* of the same problem

3. Performance measured over test (≠ training) instances

**A problem with many names:**

offline parameter tuning, automatic algorithm configuration, automatic algorithm design, hyper-parameter tuning, hyper-heuristics, meta-optimisation, programming by optimisation, ...
Offline tuning / Algorithm configuration

- Learn best parameters before *solving* an instance
- Configuration done on training instances
- Performance measured over test (≠ training) instances

Online tuning / Parameter control / Reactive search

- Learn parameters *while* solving an instance
- No training phase
- Limited to very few crucial parameters

All online methods have parameters that are configured offline
A stochastic black-box optimisation problem

- Mixed decision variables: discrete (categorical, ordinal, integer) and continuous
- Stochasticity from algorithm and problem instances
- Black-box: evaluation requires running the algorithm

Methods for Automatic Algorithm Configuration

- **SPO** [Bartz-Beielstein, Lasarczyk & Preuss, 2005]
- **ParamILS** [Hutter, Hoos & Stützle, 2007]
- **GGA** [Ansótegui, Sellmann & Tierney, 2009]
- **SMAC** [Hutter, Hoos & Leyton-Brown, 2011]
- **IRACE** [López-Ibáñez, Dubois-Lacoste, Stützle & Birattari, 2011]
Automatic Algorithm Configuration: How?

- Complex parameter spaces: numerical, categorical, ordinal, subordinate (conditional)
- Large parameter spaces (hundreds of parameters)
- Heterogeneous instances
- Medium to large tuning budgets (few hundred to thousands of runs)
- Individual runs require from seconds to hours
- Multi-core CPUs, MPI, Grid-Engine clusters

Modern automatic configuration tools are general, flexible, powerful and easy to use
Automatic Algorithm Configuration: Applications

- **Parameter tuning**
  - Exact MIP solvers (CPLEX, SCIP)
  - single-objective optimisation metaheuristics
  - multi-objective optimisation metaheuristics
  - anytime algorithms (improve time-quality trade-offs)

- **Automatic algorithm design**
  - *From a flexible framework of algorithm components*
  - From a grammar description

- **Machine learning**
  - Automatic model selection for high-dimensional survival analysis [Lang et al., 2014]
  - Hyperparameter tuning in **mlr** R package [Bischl et al., 2014]

- **Automatic design of control software for robots**
  [Francesca et al., 2015]
For procedures that require parameter tuning, the available data must be partitioned into a training and a test set. Tuning should be performed in the training set only.


- Essential tool when developing and comparing algorithms:
- First tune, then analyse
- Comparing with untuned algorithms is always unfair
Normally, optimisation algorithms are viewed as this . . .
Automatic Design: Monolithic vs. Component-wise view

...but we prefer this view
Manuel López-Ibáñez and Thomas Stützle.

The automatic design of multi-objective ant colony optimization algorithms.

MOACO Algorithms

- **Multiple objective Ant-Q (MOAQ)**
  [Mariano & Morales, 1999]
  [García-Martínez et al., 2007]

- **MACS-VRPTW**
  [Gambardella et al., 1999]

- **BicriterionAnt**
  [Iredi et al., 2001]

- **SACO**
  [T’Kindt et al., 2002]

- **Multiobjective Network ACO**
  [Cardoso et al., 2003]

- **Multicriteria Population-based ACO**
  [Guntsch & Middendorf, 2003]

- **MACS**
  [Barán & Schaerer, 2003]

- **COMPETants**
  [Doerner et al., 2003]

- **Pareto ACO**
  [Doerner et al., 2004]

- **Multiple Objective ACO Metaheuristic**
  [Gravel et al., 2002]

- **MOACO-bQAP**
  [López-Ibáñez et al., 2004]

- **MOACO-ALBP**
  [Baykasoglu et al., 2005]

- **mACO-\{1, 2, 3, 4\}**
  [Alaya et al., 2007]

- **Population-based ACO**
  [Angus, 2007]
A flexible MOACO framework

- High-level design independent from the problem
  ⇒ Easy to extend to new problems

- Instantiates 9 MOACO algorithms from the literature

- Multi-objective algorithmic design: 10 parameters

- Hundreds of potential papers algorithm designs

- Underlying ACO settings are also configurable

- Implemented for bi-objective TSP and bi-objective Knapsack
Automatic Design: Results

Tuning done with \textit{irace} (1000 runs)

<table>
<thead>
<tr>
<th>Worst</th>
<th>Best MOACO of literature + default ACO settings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\textit{Tuned} MOACO design + default ACO settings</td>
</tr>
<tr>
<td></td>
<td>Best MOACO of literature + \textit{tuned} ACO settings</td>
</tr>
<tr>
<td></td>
<td>\textit{Tuned} MOACO design + \textit{tuned} ACO settings</td>
</tr>
<tr>
<td>Best</td>
<td>\textit{Tuned} (MOACO design + ACO settings)</td>
</tr>
</tbody>
</table>
We propose a new MOACO algorithm that...

We propose an approach to automatically design MOACO algorithms:
1. Synthesize state-of-the-art knowledge into a flexible MOACO framework
2. Explore the space of potential designs automatically using irace

Other examples:
- Single-objective solvers for MIP: CPLEX, SCIP
- Single-objective algorithmic framework for SAT: SATenstein
  \[\text{[KhudaBukhsh, Xu, Hoos & Leyton-Brown, 2009]}\]
- Multi-objective framework for PFSP, TP+PLS
  \[\text{[Dubois-Lacoste, López-Ibáñez & Stützle, 2011]}\]
- Multi-objective Evolutionary Algorithms (MOEAs)
  \[\text{[Bezerra, López-Ibáñez & Stützle, 2015]}\]
The Journal of Heuristics does not endorse the up-the-wall game. [Policies on Heuristic Search Research]

True innovation in metaheuristics research therefore does not come from yet another method that performs better than its competitors, certainly if it is not well understood why exactly this method performs well. [Sörensen, 2013]

Finding a state-of-the-art algorithm is “easy”:

- problem modeling + algorithmic components + computing power

Conclusions

- Hyper-heuristics is a cool name, but not very explanatory
- Interesting works that may fall within Hyper-heuristics, but not called as such
- Automatic algorithm configuration is working today: Use it!
- Automatic design will be the end of the up-the-wall game
- Paradigm shift in optimisation research:

  From monolithic algorithms
  to flexible frameworks of algorithmic components
Automatic (Offline) Configuration & Design of Optimisation Algorithms

Manuel López-Ibáñez
manuel.lopez-ibanez@manchester.ac.uk

University of Manchester

October 27, 2015
CREST Open Workshop, London

http://iridia.ulb.ac.be/irace
References


