Sequential Model-based Optimization for General Algorithm Configuration

Frank Hutter, Holger Hoos, Kevin Leyton-Brown

University of British Columbia

LION 5, Rome
January 18, 2011
Motivation

Most optimization algorithms have parameters
- E.g. IBM ILOG CPLEX:
  • Preprocessing, balance of branching vs. cutting, type of cuts, etc.
  • 76 parameters, mostly categorical

Use machine learning to predict algorithm runtime, given
- parameter configuration used
- characteristics of the instance being solved

Use these predictions for general algorithm configuration
- E.g. optimize CPLEX parameters for given benchmark set
- Two new methods for general algorithm configuration
Related work

**General algorithm configuration**
- Racing algorithms, F-Race [Birattari et al., GECCO‘02-present]
- Iterated Local Search, ParamILS [Hutter et al., AAAI’07 & JAIR ‘09]
- Genetic algorithms, GGA [Ansotegui et al, CP’09]

**Model-based optimization of algorithm parameters**
- Sequential Parameter Optimization [Bartz-Beielstein et al., '05-present]
  - SPO toolbox: interactive tools for parameter optimization
- Our own previous work
  - SPO+: fully automated & more robust [Hutter et al., GECCO’09]
  - TB-SPO: reduced computational overheads [Hutter et al., LION 2010]
- Here: extend to general algorithm configuration
  - Sets of problem instances
  - Many, categorical parameters
Outline

1. ROAR

2. SMAC

3. Experimental Evaluation
A key component of ROAR and SMAC

Compare a configuration $\theta$ vs. the current incumbent, $\theta^*$:

- **Racing approach:**
  - Few runs for poor $\theta$
  - Many runs for good $\theta$
    - once confident enough: update $\theta^* \leftarrow \theta$

- **Agressively rejects poor configurations $\theta$**
  - Very often after a single run
ROAR: a simple method for algorithm configuration

Main ROAR loop:

• Select a configuration $\theta$ uniformly at random
• Compare $\theta$ to current $\theta^*$ (online, one $\theta$ at a time)
  - Using aggressive racing from previous slide

Random
Online
Aggressive
Racing
Outline

1. ROAR

2. SMAC
   Sequential Model-based Algorithm Configuration

3. Experimental Evaluation
SMAC in a Nutshell

Construct a model to predict algorithm performance
- Supervised machine learning
- Gaussian processes (aka kriging)
- Random forest model $f : \Theta \rightarrow \mathbb{R}$

Use that model to select promising configurations

Compare each selected configuration to incumbent
- Using same aggressive racing as ROAR
Fitting a Regression Tree to Data: Example

\[
\begin{array}{|c|c|c|c|}
\hline
\text{param 1} & \text{param 2} & \text{param 3} & \text{runtime} \\
\hline
\text{false} & 2 & \text{red} & 3.7 \\
\text{false} & 2.5 & \text{blue} & 20 \\
\text{true} & 5.5 & \text{red} & 2.1 \\
\text{false} & 5.5 & \text{blue} & 25 \\
\text{false} & 5 & \text{red} & 1.2 \\
\text{true} & 4.5 & \text{green} & 19 \\
\text{true} & 4 & \text{blue} & 12 \\
\text{true} & 3.5 & \text{green} & 17 \\
\hline
\end{array}
\]

\text{param}_{3} \in \{\text{red}\}

\text{param}_{3} \in \{\text{blue, green}\}

\[
\begin{array}{|c|c|c|c|}
\hline
\text{param 1} & \text{param 2} & \text{param 3} & \text{runtime} \\
\hline
\text{false} & 2 & \text{red} & 3.7 \\
\text{true} & 5.5 & \text{red} & 2.1 \\
\text{false} & 5 & \text{red} & 1.2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{param 1} & \text{param 2} & \text{param 3} & \text{runtime} \\
\hline
\text{false} & 2.5 & \text{blue} & 20 \\
\text{true} & 5.5 & \text{blue} & 25 \\
\text{true} & 4.5 & \text{green} & 19 \\
\text{true} & 4 & \text{blue} & 12 \\
\text{true} & 3.5 & \text{green} & 17 \\
\hline
\end{array}
\]
Fitting a Regression Tree to Data: Example

- In each internal node: only store split criterion used

\[ \text{param}_3 \in \{\text{red}\} \]

\[ \text{param}_3 \in \{\text{blue, green}\} \]

\[ \text{param}_2 \leq 3.5 \]

\[ \text{param}_2 > 3.5 \]
Fitting a Regression Tree to Data: Example

- In each internal node: only store split criterion used

param_3 ∈ \{red\}

param_3 ∈ \{blue, green\}

param_2 ≤ 3.5

param_2 > 3.5

<table>
<thead>
<tr>
<th>param 1</th>
<th>param 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>2</td>
<td>red</td>
<td>3.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>param 1</th>
<th>param 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>5.5</td>
<td>red</td>
<td>2.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>param 1</th>
<th>param 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>5</td>
<td>red</td>
<td>1.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>param 1</th>
<th>param 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>blue</td>
<td>blue</td>
<td>20</td>
</tr>
<tr>
<td>true</td>
<td>blue</td>
<td>green</td>
<td>25</td>
</tr>
<tr>
<td>true</td>
<td>green</td>
<td>blue</td>
<td>19</td>
</tr>
<tr>
<td>true</td>
<td>blue</td>
<td>green</td>
<td>12</td>
</tr>
<tr>
<td>true</td>
<td>green</td>
<td>green</td>
<td>17</td>
</tr>
</tbody>
</table>
Fitting a Regression Tree to Data: Example

- In each internal node: only store split criterion used
- In each leaf: store mean of runtimes

\[
\begin{align*}
\text{param}_3 & \in \{\text{red}\} \\
\text{param}_2 & \leq 3.5 \\
3.7 & \\
\text{param}_3 & \in \{\text{blue, green}\} \\
\text{param}_2 & > 3.5
\end{align*}
\]
Fitting a Regression Tree to Data: Example

- In each internal node: only store split criterion used
- In each leaf: store mean of runtimes

param_3 ∈ \{\text{red}\}

param_3 ∈ \{\text{blue, green}\}

param_2 ≤ 3.5

3.7

param_2 > 3.5

1.65

<table>
<thead>
<tr>
<th>param_1</th>
<th>param_2</th>
<th>param_3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>2.5</td>
<td>blue</td>
<td>20</td>
</tr>
<tr>
<td>false</td>
<td>5.5</td>
<td>blue</td>
<td>25</td>
</tr>
<tr>
<td>true</td>
<td>4.5</td>
<td>green</td>
<td>19</td>
</tr>
<tr>
<td>true</td>
<td>4</td>
<td>blue</td>
<td>12</td>
</tr>
<tr>
<td>true</td>
<td>3.5</td>
<td>green</td>
<td>17</td>
</tr>
</tbody>
</table>
Fitting a Regression Tree to Data: Example

- In each internal node: only store split criterion used
- In each leaf: store mean of runtimes

param_3 \in \{\text{red}\}

param_3 \in \{\text{blue, green}\}

param_2 \leq 3.5

param_2 > 3.5

3.7

1.65
Predictions for a new parameter configuration

E.g. $\theta_{n+1} = (\text{true, 4.7, red})$
- Walk down tree, return mean runtime stored in leaf $\Rightarrow 1.65$
Random Forests: sets of regression trees

Training
- Subsample the data $T$ times (with repetitions)
- For each subsample, fit a regression tree

Prediction
- Predict with each of the $T$ trees
- Return empirical mean and variance across these $T$ predictions
Predictions For Different Instances

Runtime data now also includes instance features:
- Configuration $\theta_i$, runtime $r_i$, and *instance features* $x_i = (x_{i,1}, \ldots, x_{i,m})$

Fit a model $g$: $\Theta \times \mathbb{R}^m \rightarrow \mathbb{R}$
- Predict runtime for previously unseen combinations $(\theta_{n+1}, x_{n+1})$

```
feat_2 \leq 3.5
feat_2 > 3.5
```

```
param_3 \in \{\text{blue, green}\}
param_3 \in \{\text{red}\}
```

```
3.7
```

```
feat_7 \leq 17
feat_7 > 17
```

```
2
1
```
Visualization of Runtime Across Instances and Parameter Configurations

Performance of configuration $\theta$ across instances:
- Average of $\theta$’s predicted row

True $\log_{10}$ runtime

Predicted $\log_{10}$ runtime

Darker is faster
Summary of SMAC Approach

Construct model to predict algorithm performance
- Random forest model \( g : \Theta \times \mathbb{R}^m \rightarrow \mathbb{R} \)
- Marginal predictions \( f : \Theta \rightarrow \mathbb{R} \)

Use that model to select promising configurations
- Standard “expected improvement (EI)” criterion
  - combines predicted mean and uncertainty
- Find configuration with highest EI: optimization by local search

Compare each selected configuration to incumbent \( \theta^* \)
- Using same aggressive racing as ROAR
- Save all run data \( \rightarrow \) use to construct models in next iteration
Outline

1. ROAR
2. SMAC
3. Experimental Evaluation
Experimental Evaluation: Setup

Compared SMAC, ROAR, FocusedILS, and GGA

- On 17 small configuration scenarios:
  - Local search and tree search SAT solvers SAPS and SPEAR
  - Leading commercial MIP solver CPLEX

- For each configurator and each scenario
  - 25 configuration runs with 5-hour time budget each
  - Evaluate final configuration of each run on independent test set

Over a year of CPU time

- Will be available as a reproducible experiment package in HAL
- HAL: see Chris Nell’s talk tomorrow @ 17:20
Experimental Evaluation: Results

y-axis: test performance (runtime, smaller is better)
S=SMAC, R=ROAR, F=FocusedILS, G=GGA

• Improvement (means over 25 runs)
  - $0.93 \times - 2.25 \times$ (vs FocusedILS), $1.01 \times - 2.76 \times$ (vs GGA)
• Significant (never significantly worse)
  - $11/17$ (vs FocusedILS), $13/17$ (vs GGA)
• But: SMAC’s performance depends on instance features
Conclusion

Generalized model-based parameter optimization:
- Sets of benchmark instances
- Many, categorical parameters

Two new procedures for general algorithm configuration
- Random Online Aggressive Racing (ROAR)
  • Simple yet surprisingly effective
- Sequential Model-based Algorithm Configuration (SMAC)
  • State-of-the-art configuration procedure
  • Improvements over FocusedILS and GGA
Future Work

Improve algorithm configuration further
- Cut off poor runs early (like adaptive capping in ParamILS)
  • Handle “censored” data in the models
- Combine model-free and model-based methods

Use SMAC’s models to gain scientific insights
- Importance of each parameter
- Interaction of parameters and instance features

Use SMAC’s models for per-instance algorithm configuration
- Compute instance features
- Pick configuration predicted to be best