Algorithm and Experiment Design with HeuristicLab

An Open Source Optimization Environment for Research and Education

S. Wagner, G. Kronberger
Heuristic and Evolutionary Algorithms Laboratory (HEAL)
School of Informatics/Communications/Media, Campus Hagenberg
University of Applied Sciences Upper Austria
Instructor Biographies

• Stefan Wagner
  – Full professor for complex software systems (since 2009)
    University of Applied Sciences Upper Austria
  – Co-founder of the HEAL research group
  – Project manager and chief architect of HeuristicLab
  – PhD in technical sciences (2009)
    Johannes Kepler University Linz, Austria
  – Associate professor (2005 – 2009)
    University of Applied Sciences Upper Austria
  – http://heal.heuristiclab.com/team/wagner

• Gabriel Kronberger
  – Full professor for business intelligence (since 2011)
    University of Applied Sciences Upper Austria
  – Member of the HEAL research group
  – Architect of HeuristicLab
  – PhD in technical sciences (2010)
    Johannes Kepler University Linz, Austria
    University of Applied Sciences Upper Austria
  – http://heal.heuristiclab.com/team/kronberger
Agenda

• Objectives of the Tutorial
• Introduction
• Where to get HeuristicLab?
• Plugin Infrastructure
• Graphical User Interface
• Available Algorithms & Problems

• **Demonstration Part I: Working with HeuristicLab**
• **Demonstration Part II: Data-based Modeling**

• Some Additional Features
• Planned Features
• Team
• Suggested Readings
• Bibliography
• Questions & Answers
Objectives of the Tutorial

• Introduce general motivation and design principles of HeuristicLab
• Show where to get HeuristicLab
• Explain basic GUI usability concepts
• Demonstrate basic features
• Demonstrate editing and analysis of optimization experiments
• Demonstrate custom algorithms and graphical algorithm designer
• Demonstrate data-based modeling features
• Outline some additional features
Introduction

• Motivation and Goals
  – graphical user interface
  – paradigm independence
  – multiple algorithms and problems
  – large scale experiments and analyses
  – parallelization
  – extensibility, flexibility and reusability
  – visual and interactive algorithm development
  – multiple layers of abstraction

• Facts
  – development of HeuristicLab started in 2002
  – based on Microsoft .NET and C#
  – used in research and education
  – second place at the Microsoft Innovation Award 2009
  – open source (GNU General Public License)
  – version 3.3.0 released on May 18th, 2010
  – latest version 3.3.16 "Prague"
Where to get HeuristicLab?

- **Download binaries**
  - deployed as ZIP archives
  - latest stable version 3.3.16 "Prague"
    - released on July 31st, 2019
  - daily trunk builds
  - [http://dev.heuristiclab.com/download](http://dev.heuristiclab.com/download)

- **Check out sources**
  - SVN repository
  - HeuristicLab 3.3.16 tag
    - [https://src.heuristiclab.com/svn/core/tags/3.3.16](https://src.heuristiclab.com/svn/core/tags/3.3.16)
  - Stable development version
    - [https://src.heuristiclab.com/svn/core/stable](https://src.heuristiclab.com/svn/core/stable)

- **License**
  - GNU General Public License (Version 3)

- **System requirements**
  - Microsoft .NET Framework 4.5
  - enough RAM and CPU power ;-)

HeuristicLab Tutorial [http://dev.heuristiclab.com](http://dev.heuristiclab.com)
Plugin Infrastructure

• HeuristicLab consists of many assemblies
  – 160+ plugins in HeuristicLab 3.3.16
  – plugins can be loaded or unloaded at runtime
  – plugins can be updated via internet
  – application plugins provide GUI frontends

• Extensibility
  – developing and deploying new plugins is easy
  – dependencies are explicitly defined, automatically checked and resolved
  – automatic discovery of interface implementations (service locator pattern)

• Plugin Manager
  – GUI to check, install, update or delete plugins
Plugin Architecture

HeuristicLab Tutorial

http://dev.heuristiclab.com
Graphical User Interface

• HeuristicLab GUI is made up of views
  – views are visual representations of content objects
  – views are composed in the same way as their content
  – views and content objects are loosely coupled
  – multiple different views may exist for the same content

• Drag & Drop
  – views support drag & drop operations
  – content objects can be copied or moved (shift key)
  – enabled for collection items and content objects

HeuristicLab Tutorial
http://dev.heuristiclab.com
Graphical User Interface

Algorithm View

Problem View

Parameter View

Double Value View
Graphical User Interface

• ViewHost
  – control which hosts views
  – right-click on windows icon to switch views
  – double-click on windows icon to open another view
  – drag & drop windows icon to copy contents
Available Algorithms

Population-based
• CMA-ES
• Evolution Strategy
• Genetic Algorithm
• Offspring Selection Genetic Algorithm (OSGA)
• Island Genetic Algorithm
• Island Offspring Selection Genetic Algorithm
• Parameter-less Population Pyramid (P3)
• SASEGASA
• Relevant Alleles Preserving GA (RAPGA)
• Age-Layered Population Structure (ALPS)
• Genetic Programming
• NSGA-II
• Scatter Search
• Particle Swarm Optimization

Trajectory-based
• Local Search
• Tabu Search
• Robust Tabu Search
• Variable Neighborhood Search
• Simulated Annealing

Data Analysis
• Linear Discriminant Analysis
• Linear Regression
• Multinomial Logit Classification
• k-Nearest Neighbor
• k-Means
• Neighbourhood Component Analysis
• Artificial Neural Networks
• Random Forests
• Support Vector Machines
• Gaussian Processes
• Gradient Boosted Trees
• Gradient Boosted Regression
• Barnes-Hut t-SNE
• Kernel Ridge Regression
• Elastic-net Regression

Additional Algorithms
• Performance Benchmarks
• Hungarian Algorithm
• Cross Validation
• LM-BFGS
Available Problems

Combinatorial Problems
- Test Problems (1-max, NK, HIFF, Deceptive Trap)
- Traveling Salesman
- Probabilistic Traveling Salesman
- Vehicle Routing (MDCVRPTW, PDPTW, ...)
- Knapsack
- Bin Packing
- Graph Coloring
- Job Shop Scheduling
- Linear Assignment
- Quadratic Assignment
- Orienteering

Genetic Programming Problems
- Test Problems (Even Parity, MUX)
- Symbolic Classification
- Symbolic Regression
- Symbolic Time-Series Prognosis
- Artificial Ant
- Lawn Mower
- Robocode
- Grammatical Evolution

Additional Problems
- Single-/Multi-Objective Test Function
  - Ackley, Griewank, Rastrigin, Sphere, etc.
- Programmable Problem
- External Evaluation Problem
  (generic TCP/IP, Scilab, MATLAB)
- Regression, Classification, Clustering
- Trading

http://dev.heuristiclab.com
Agenda

• Objectives of the Tutorial
• Introduction
• Where to get HeuristicLab?
• Plugin Infrastructure
• Graphical User Interface
• Available Algorithms & Problems

• Demonstration Part I: Working with HeuristicLab
• Demonstration Part II: Data-based Modeling

• Some Additional Features
• Planned Features
• Team
• Suggested Readings
• Bibliography
• Questions & Answers
Demonstration Part I: Working with HeuristicLab

• Create, Parameterize and Execute Algorithms
• Save and Load Items
• Create Batch Runs and Experiments
• Multi-core CPUs and Parallelization
• Analyze Runs
• Analyzers
• Building User-Defined Algorithms
HeuristicLab Optimizer

Follow these steps to start working with HeuristicLab Optimizer:

1. Open an algorithm
   - click (New Item) in the toolbar and select an algorithm or click (Open File) in the toolbar and load an algorithm from a file

2. Open a problem in the algorithm
   - in the Problem tab of the algorithm click (New Problem) and select a problem or click (Open Problem) and load a problem from a file

3. Set parameters
   - set problem parameters in the Problem tab of the algorithm
   - set algorithm parameters in the Parameters tab of the algorithm

4. Run the algorithm
   - click (Start/Resume Algorithm) to execute the algorithm (if the button is grayed out some parameters of the algorithm or the problem still have to be set)
   - wait for the algorithm to terminate or click (Pause Algorithm) to interrupt its execution or click (Stop Algorithm) to stop its execution

5. Check results
   - check the results on the Results tab of the algorithm
   - click (Start/Resume Algorithm) to continue the algorithm or click (Resume Algorithm) to prepare a new run

Looking for predefined algorithms which can be executed immediately?
- check out the sample algorithms below

Any feedback, questions, problems or requests for new features?
- visit the HeuristicLab tracker at http://dev.heuristiclab.com
- watch the HeuristicLab video tutorials at <http://www.youtube.com/heuristiclab>
- join the HeuristicLab mailing list mailto:heuristiclab@googlegroups.com
- visit the HeuristicLab facebook site <http://www.facebook.com/heuristiclab>
- write an e-mail to mailto:support@heuristiclab.com to contact the HeuristicLab team

Double-click to open sample algorithms and problems.
Create Algorithm

[Image of the HeuristicLab Optimizer interface]

[Image of the New Item dialog box with various algorithms listed]

- Benchmark Algorithm
- CMA Evolution Strategy
- Evolution Strategy
- Genetic Algorithm
- Hungarian Algorithm
- Island Genetic Algorithm
- Island Offspring Selection Genetic Algorithm
- LM-BFGS

- Version: 3.3.10.11173
- Description: An algorithm to execute
- Description: An evolution strategy
- Description: An evolution strategy
- Description: A genetic algorithm
- Description: The Hungarian algorithm
- Description: An island genetic algorithm
- Description: An island offspring selection genetic algorithm
- Description: The limited-memory BFGS

http://dev.heuristiclab.com
Create or Load Problem
Import or Parameterize Problem Data
Parameterize Algorithm
Start, Pause, Resume, Stop and Reset
Inspect Results

HeuristicLab Tutorial  
http://dev.heuristiclab.com
Compare Runs

- A run is created each time when the algorithm is stopped
  - runs contain all results and parameter settings
  - previous results are not forgotten and can be compared
Save and Load

• Save to and load from disk
  – HeuristicLab items (i.e., algorithms, problems, experiments, ...) can be saved to and loaded from a file
  – algorithms can be paused, saved, loaded and resumed
  – data format is custom compressed XML
  – saving and loading files might take several minutes
  – saving and loading large experiments requires some memory
Create Batch Runs and Experiments

• Batch runs
  – execute the same optimizer (e.g. algorithm, batch run, experiment) several times

• Experiments
  – execute different optimizers
  – suitable for large scale algorithm comparison and analysis

• Experiments and batch runs can be nested

• Generated runs can be compared afterwards
Create Batch Runs and Experiments

drag & drop here to add additional algorithms, batch runs, experiments, etc.
Clipboard

• Store items
  – click on the buttons to add or remove items
  – drag & drop items on the clipboard
  – use the menu to add a copy of a shown item to the clipboard

• Show items
  – double-click on an item in the clipboard to show its view

• Save and restore clipboard content
  – click on the save button to write the clipboard content to disk
  – clipboard is automatically restored when HeuristicLab is started the next time
Clipboard

drag & drop here to add algorithms, problems, batch runs, experiments, etc.
Start, Pause, Resume, Stop, Reset
Compare Runs
Analyze Runs

- HeuristicLab provides interactive views to analyze and compare all runs of a run collection
  - textual analysis
    - RunCollection Tabular View
  - graphical analysis
    - RunCollection BubbleChart
    - RunCollection BoxPlots

- Filtering is automatically applied to all open run collection views
# Runs – Tabular View

![Image of HeuristicLab Tabular View](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Run Number</th>
<th>BestKnownQuality</th>
<th>BestKnownSolution</th>
<th>BestQuality</th>
<th>Coordinates</th>
<th>Crossover</th>
<th>Current/AverageQuality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 13</td>
<td>3110</td>
<td>16405</td>
<td>16843.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 14</td>
<td>3110</td>
<td>14783</td>
<td>15029.02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 15</td>
<td>3110</td>
<td>14252</td>
<td>14282.89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 16</td>
<td>3110</td>
<td>13243</td>
<td>13245.96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 17</td>
<td>3110</td>
<td>13703</td>
<td>13749.88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 18</td>
<td>3110</td>
<td>13564</td>
<td>13951.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 19</td>
<td>3110</td>
<td>15421</td>
<td>15431.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 20</td>
<td>3110</td>
<td>14409</td>
<td>15147</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 21</td>
<td>3110</td>
<td>13771</td>
<td>13954.56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%)</td>
<td>Run 22</td>
<td>3110</td>
<td>14529</td>
<td>14532.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 13</td>
<td>3110</td>
<td>13095</td>
<td>13642.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 14</td>
<td>3110</td>
<td>12403</td>
<td>12810.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 15</td>
<td>3110</td>
<td>14091</td>
<td>14653.98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 16</td>
<td>3110</td>
<td>12595</td>
<td>13257.99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 17</td>
<td>3110</td>
<td>12792</td>
<td>13264.38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 18</td>
<td>3110</td>
<td>12711</td>
<td>13151.19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 19</td>
<td>3110</td>
<td>12326</td>
<td>13264.78</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 20</td>
<td>3110</td>
<td>13346</td>
<td>13777.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 21</td>
<td>3110</td>
<td>12807</td>
<td>13264.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 5%)</td>
<td>Run 22</td>
<td>3110</td>
<td>12741</td>
<td>13113.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 10%)</td>
<td>Run 13</td>
<td>3110</td>
<td>15921</td>
<td>18884.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 10%)</td>
<td>Run 14</td>
<td>3110</td>
<td>16354</td>
<td>19693.36</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Runs – Tabular View

• Sort columns
  – click on column header to sort column
  – Ctrl-click on column header to sort multiple columns

• Show or hide columns
  – right-click on table to open dialog to show or hide columns

• Compute statistical values
  – select multiple numerical values to see count, sum, minimum, maximum, average and standard deviation

• Select, copy and paste into other applications
Runs – BubbleChart
Runs – BubbleChart

- Choose values to plot
  - choose which values to show on the x-axis, the y-axis and as bubble size
  - possible values are all parameter settings and results

- Add jitter
  - add jitter to separate overlapping bubbles

- Zoom in and out
  - click on Zoom and click and drag in the chart area to zoom in
  - double click on the chart area background or on the circle buttons beside the scroll bars to zoom out

- Color bubbles
  - click on Select, choose a color and click and drag in the chart area to select and color bubbles
  - apply coloring automatically by clicking on the axis coloring buttons

- Show runs
  - double click on a bubble to open its run

- Export image
  - right-click to open context menu to copy or save image
  - save image as pixel (BMP, JPG, PNG, GIF, TIF) or vector graphics (EMF)

- Show box plots
  - right-click to open context menu to show box plots view
Runs – BoxPlots
Runs – BoxPlots

• Choose values to plot
  – choose which values to show on the x-axis and y-axis
  – possible values are all parameter settings and results

• Zoom in and out
  – click on Zoom and click and drag in the chart area to zoom in
  – double click on the chart area background or on the circle buttons beside the scroll bars to zoom out

• Show or hide statistical values
  – click on the lower left button to show or hide statistical values

• Export image
  – right-click to open context menu to copy or save image
  – save image as pixel (BMP, JPG, PNG, GIF, TIF) or vector graphics (EMF)
Runs – Multi-Line Chart
Filter Runs
Multi-core CPUs and Parallelization

• Parallel execution of optimizers in experiments
  – optimizers in an experiment are executed sequentially from top to bottom per default
  – experiments support parallel execution of their optimizers
  – select a not yet executed optimizer and start it manually to utilize another core
  – execution of one of the next optimizers is started automatically after an optimizer is finished

• Parallel execution of algorithms
  – HeuristicLab provides special operators for parallelization
  – engines decide how to execute parallel operations
  – sequential engine executes everything sequentially
  – parallel engine executes parallel operations on multiple cores
  – Hive engine (under development) executes parallel operations on multiple computers
  – all implemented algorithms support parallel solution evaluation
Parallel Execution of Experiments

1. start experiment
2. start other optimizers
Parallel Execution of Algorithms
Analyzers

• Special operators for analysis purposes
  – are executed after each iteration
  – serve as general purpose extension points of algorithms
  – can be selected and parameterized in the algorithm
  – perform algorithm-specific and/or problem-specific tasks
  – some analyzers are quite costly regarding runtime and memory
  – implementing and adding custom analyzers is easy

• Examples
  – TSPAlleleFrequencyAnalyzer
  – TSPPopulationDiversityAnalyzer
  – SuccessfulOffspringAnalyzer
  – SymbolicDataAnalysisVariableFrequencyAnalyzer
  – SymbolicRegressionSingleObjectiveTrainingBestSolutionAnalyzer
  – ...

HeuristicLab Tutorial  http://dev.heuristiclab.com  43
Analyzers
TSPAlleleFrequencyAnalyzer
TSPPopulationDiversityAnalyzer
Building User-Defined Algorithms

• Operator graphs
  – algorithms are represented as operator graphs
  – operator graphs of user-defined algorithms can be changed
  – algorithms can be defined in the graphical algorithm designer
  – use the menu to convert a standard algorithm into a user-defined algorithm

• Operators sidebar
  – drag & drop operators into an operator graph

• Programmable operators
  – add programmable operators in order to implement custom logic in an algorithm
  – no additional development environment needed

• Debug algorithms
  – use the debug engine to obtain detailed information during algorithm execution
Building User-Defined Algorithms
Building User-Defined Algorithms
Programmable Operators
Scripts

```csharp
for (int g = 0; g < generations; g++) {
    var parents = population.SampleProportional(random, 2 * popSize, quality);
    for (int i = 0; i < popSize; i++) {
        nextGen[i] = PartiallyMatchedCrossover.Apply(random, parents[i * 2],
                         parents[i * 2 + 1],
                         (random.NextDouble() < mutationRate) ? SwapManipulator.Apply(random,
                                                                                   parents[i * 2],
                                                                                   parents[i * 2 + 1]);
        if (nextQual[i] < bestQuality) {
            bestQuality = nextQual[i];
            bestQualityGeneration = g;
        }
    }
}
qualityRow.Values.Add(bestQuality);
Array.Copy(nextGen, population, popSize);
Array.Copy(nextQual, qualities, popSize);
}
```

HeuristicLab Tutorial  http://dev.heuristiclab.com
Debugging Algorithms
Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems

- Demonstration Part I: Working with HeuristicLab
- Demonstration Part II: Data-based Modeling

- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers
Demonstration Part II: Data-based Modeling

- Introduction
- Regression with HeuristicLab
- Model simplification and export
- Variable relevance analysis
- Classification with HeuristicLab
Introduction to Data-based Modeling

• Dataset: Matrix \((x_{i,j})_{i=1..N, j=1..K}\)
  - \(N\) observations of \(K\) input variables
  - \(x_{i,j}\) = \(i\)-th observation of \(j\)-th variable
  - Additionally: Vector of labels \((y_1...y_N)^T\)

• Goal: learn association of input variable values to labels
Data Analysis in HeuristicLab

• Symbolic regression and classification using genetic programming

• External Libraries:
  – Linear Regression, Logistic Regression,
  – k-Nearest Neighbours, k-Means,
  – Random Forest, Support Vector Machines, Neural Networks, Gaussian Processes
Case Study: Regression

• Poly-10 benchmark problem dataset
  – 10 input variables $x_1 \ldots x_{10}$
  – $y = x_1 x_2 + x_3 x_4 + x_5 x_6 + x_1 x_7 x_9 + x_3 x_6 x_{10}$
  – non-linear modeling approach necessary
  – frequently used in GP literature
  – available as benchmark problem instance in HeuristicLab
Demonstration

- problem configuration
  - data import
  - target and input variables
  - data partitions (training and test)

- algorithm configuration

- analysis of results
  - accuracy metrics
  - visualization of model output
Linear Regression

- Create new algorithm

HeuristicLab Tutorial http://dev.heuristiclab.com
Import Data from CSV-File
Inspect and Configure Dataset
Inspect Imported Data
Set Target Variable
Select Input Variables
Configure Training and Test Partitions
Run Linear Regression
Inspect Results
Inspect Scatterplot of Predicted and Target Values
Inspect Linechart
Inspect the Model

\[ Result = (c_0 \cdot x_1 + c_1 \cdot x_2 + c_2 \cdot x_3 + c_3 \cdot x_4 + c_4 \cdot x_5 + c_5 \cdot x_6 + c_6 \cdot x_7 + c_7 \cdot x_8 + c_8 \cdot x_9 + c_9 \cdot x_{10} + c_{10}) \]

\[
\begin{align*}
c_0 &= 0.081337 \\
c_1 &= 0.19906 \\
c_2 &= -0.029881 \\
c_3 &= 0.078892 \\
c_4 &= -0.010307 \\
c_5 &= 0.031685 \\
c_6 &= -0.047071 \\
c_7 &= -0.029194 \\
c_8 &= 0.0015768 \\
c_9 &= 0.10525 \\
c_{10} &= 0.020099
\end{align*}
\]
Symbolic Regression with HeuristicLab

• Linear regression produced an inaccurate model.
• Next: produce a nonlinear symbolic regression model using genetic programming

• Genetic programming
  – evolve variable-length models
  – model representation: symbolic expression tree
  – structure and model parameters are evolved side-by-side
  – white-box models
Symbolic Regression with HeuristicLab

• Demonstration
  – problem configuration
  – function set and terminal set
  – model size constraints
  – evaluation

• Algorithm configuration
  – selection
  – mutation

• Analysis of results
  – model accuracy
  – model structure and parameters
Create New Genetic Algorithm
Create New Symbolic Regression Problem
Import Data

HeuristicLab Tutorial

http://dev.heuristiclab.com
Inspect Data and Configure Dataset
Set Target and Input Variables
Configure Maximal Model Depth and Length
Configure Function Set (Grammar)
Configure Function Set (Grammar)
Configure Algorithm Parameters
Configure Mutation Operator
Configure Selection Operator
Configure Tournament Group Size
Start Algorithm and Inspect Results
Inspect Quality Chart
Inspect Best Model on Training Partition
Inspect Linechart of Best Model on Training Partition
Inspect Structure of Best Model on Training Partition
Model Simplification and Export

- Demonstration
  - automatic simplification
  - visualization of node impacts
  - manual simplification
    - online update of results
- model export
  - Excel
  - MATLAB
  - LaTeX
Detailed Model Analysis and Simplification
Symbolic Simplification and Node Impacts

HeuristicLab Tutorial
Manual Simplification

double-click nodes
Automatic Symbolic Simplification
Textual Representations Are Also Available

- Use ViewHost to switch to textual representation view.
Default Textual Representation for Model Export
Textual Representation for Export to \LaTeX

\begin{quote}
\begin{verbatim}
\texttt{Result} = (c_0 x_1 (t) + c_1 x_2 (t) + c_2 x_3 (t) + c_3 x_4 (t) + c_4 x_5 (t) + c_5 x_6 (t) + c_6 x_7 (t) + c_7 x_8 (t) + c_8 x_9 (t) + c_9 x_10 (t) + c_{10})
\end{verbatim}
\end{quote}

\begin{verbatim}
(1)
\end{verbatim}

\begin{verbatim}
c_0 = 0.0813371220642195
(2)
c_1 = 0.199055016563887
(3)
c_2 = -0.0298811744629839
(4)
c_3 = 0.0788918883541302
(5)
c_4 = -0.0103065273366223
(6)
c_5 = 0.0316849536396099
(7)
c_6 = -0.047070585925129
(8)
c_7 = -0.0291939214032144
(9)
c_8 = 0.0015767966507775
(10)
c_9 = 0.105250443686677
(11)
c_{10} = 0.0200987846293256
(12)
\end{verbatim}
LaTeX Export

\begin{align*}
\text{Result} &= x_4(t) \cdot x_3(t) \cdot c_{20} \\
&= \left( x_6(t) \cdot x_5(t) \cdot c_4 + x_4(t) \cdot x_3(t) \cdot c_7 + x_4(t) \cdot x_3(t) \cdot c_{10} + \frac{c_{11} x_1(t)}{x_4(t) \cdot x_3(t) \cdot \left( c_{14} x_4(t) + c_{15} x_5(t) + \frac{1}{c_{17} x_2(t)} \right) \cdot c_{18}} \right) \cdot c_{19} + c_{21}
\end{align*}

\begin{align*}
c_4 &= -1.57302367616477 \\
c_7 &= -0.867137925013337 \\
c_{10} &= -0.867137925013337 \\
c_{11} &= 1.27519978915975 \\
c_{14} &= -0.017064976517855 \\
c_{15} &= 0.00314376988160885 \\
c_{17} &= -3.00832012161288 \\
c_{18} &= 0.867137925013337 \\
c_{19} &= -5.451900998999249 \\
c_{20} &= -0.204498330755849 \\
c_{21} &= -0.0465339907207764
\end{align*}

HeuristicLab Tutorial  http://dev.heuristiclab.com  98
Variable Relevance Analysis

• Which variables are important for correct predictions?

• Demonstration
  – Variable frequency analyzer
  – symbol frequency analyzer
  – variable impacts
Inspect Variable Frequency Chart
Inspect Variable Impacts

![Image of HeuristicLab interface showing variable impacts analysis]

- Evaluated Solutions: 43,567
- Generations: 43
- Symbol frequencies:
- Variable frequencies:

**Variable Impacts:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Relative Variable Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>x4</td>
<td>0.328303889106054</td>
</tr>
<tr>
<td>x3</td>
<td>0.241170172585569</td>
</tr>
<tr>
<td>x1</td>
<td>0.17912399714678</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>x9</td>
<td>0.01946242278034</td>
</tr>
</tbody>
</table>

Execution Time: 00:01:30.9692041
Inspect Symbol Frequencies

![HeuristicLab Optimization Window](image-url)
Classification with HeuristicLab

- Symbolic classification
  - evolve discriminating function using GP
  - find thresholds to assign classes

- Demonstration
  - real world medical application
  - model accuracy
  - visualization of model output
    - discriminating function output
    - ROC-curve
    - confusion matrix
Case Study: Classification

• Real world medical dataset (*Mammographic Mass*) from UCI Machine Learning Repository
  – data from non-invasive mammography screening
  – variables:
    • patient age
    • visual features of inspected mass lesions: shape, margin, density
  – target variable: severity (malignant, benign)

  – available as a benchmark problem instance in HeuristicLab
Open Sample

HeuristicLab Tutorial

http://dev.heuristiclab.com
Configure and Run Algorithm
Inspect Quality Linechart
Inspect Best Training Solution

![Image of HeuristicLab Optimizer with highlighted best training solution: Symbolic Discriminant Function Classification.](image_url)
Inspect Model Output and Thresholds
Inspect Confusion Matrix
Inspect ROC Curve
Validation of Results

• Overfitting = memorizing data

• Strategies to reduce overfitting
  – validation partition
  – cross-validation
Validation of Results

- Demonstration
  - Configuration of a validation set
  - Inspection of best solution on validation set
  - Analysis of training- and validation fitness correlation
- Cross-validation
  - Configuration
  - Analysis of results
Configuration of Validation Partition

![Configuration of Validation Partition](image)
Inspect Best Model on Validation Partition
Inspect Linechart of Correlation of Training and Validation Fitness
Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems

- Demonstration Part I: Working with HeuristicLab
- Demonstration Part II: Data-based Modeling

- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers
Some Additional Features

• **HeuristicLab Hive**
  - parallel and distributed execution of algorithms and experiments on many computers in a network

• **Optimization Knowledge Base (OKB)**
  - database to store algorithms, problems, parameters and results
  - open to the public
  - open for other frameworks
  - analyze and store characteristics of problem instances and problem classes

• **External solution evaluation and simulation-based optimization**
  - interface to couple HeuristicLab with other applications (MATLAB, Simulink, SciLab, AnyLogic, …)
  - supports different protocols (command line parameters, TCP, …)

• **Parameter grid tests and meta-optimization**
  - automatically create experiments to test large ranges of parameters
  - apply heuristic optimization algorithms to find optimal parameter settings for heuristic optimization algorithms
Planned Features

- **Algorithms & Problems**
  - steady-state genetic algorithm
  - unified tabu search for vehicle routing
  - estimation of distribution algorithms
  - evolution of arbitrary code (controller, etc.)
  - ...

- **Cloud Computing**
  - port HeuristicLab Hive to Windows Azure

- **Have a look at the HeuristicLab roadmap**
  - [http://dev.heuristiclab.com/trac.fcgi/roadmap](http://dev.heuristiclab.com/trac.fcgi/roadmap)

- **Any other ideas, requests or recommendations?**
  - join our HeuristicLab Google group [heuristiclab@googlegroups.com](mailto:heuristiclab@googlegroups.com)
  - write an e-mail to [support@heuristiclab.com](mailto:support@heuristiclab.com)
HeuristicLab Team

Heuristic and Evolutionary Algorithms Laboratory (HEAL)
School of Informatics, Communications and Media
University of Applied Sciences Upper Austria

Softwarepark 11
A-4232 Hagenberg
AUSTRIA

WWW: http://heal.heuristiclab.com
Suggested Readings

• S. Voß, D. Woodruff (Edts.)
  Optimization Software Class Libraries

• M. Affenzeller, S. Winkler, S. Wagner, A. Beham
  Genetic Algorithms and Genetic Programming
  Modern Concepts and Practical Applications
  CRC Press, 2009
Bibliography

- S. Wagner, M. Affenzeller
  HeuristicLab: A generic and extensible optimization environment
  Adaptive and Natural Computing Algorithms, pp. 538-541
  Springer, 2005

- S. Wagner, S. Winkler, R. Braune, G. Kronberger, A. Beham, M. Affenzeller
  Benefits of plugin-based heuristic optimization software systems
  Springer, 2007

- S. Wagner, G. Kronberger, A. Beham, S. Winkler, M. Affenzeller
  Modeling of heuristic optimization algorithms
  Proceedings of the 20th European Modeling and Simulation Symposium, pp. 106-111
  DIPTEM University of Genova, 2008

- S. Wagner, G. Kronberger, A. Beham, S. Winkler, M. Affenzeller
  Model driven rapid prototyping of heuristic optimization algorithms
  Springer, 2009

- S. Wagner
  Heuristic optimization software systems - Modeling of heuristic optimization algorithms in the HeuristicLab software environment

- S. Wagner, A. Beham, G. Kronberger, M. Kommenda, E. Pitzer, M. Kofler, S. Vonolfen, S. Winkler, V. Dorfer, M. Affenzeller
  HeuristicLab 3.3: A unified approach to metaheuristic optimization
  Actas del séptimo congreso español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB’2010), 2010

- S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, M. Affenzeller
  Architecture and Design of the HeuristicLab Optimization Environment
  Advanced Methods and Applications in Computational Intelligence, vol. 6, pp. 197-261, Springer, 2014

- Detailed list of all publications of the HEAL research group: http://research.fh-ooe.at/de/orgunit/356#showpublications
Questions & Answers

http://dev.heuristiclab.com
heuristiclab@googlegroups.com
http://www.youtube.com/heuristiclab
http://www.facebook.com/heuristiclab