System Identification and Data Mining with HeuristicLab

An Open Source Optimization Environment for Research and Education

M. Kommenda, A. Scheibenpflug
Heuristic and Evolutionary Algorithms Laboratory (HEAL)
School of Informatics/Communications/Media, Campus Hagenberg
University of Applied Sciences Upper Austria
Instructor Biographies

• Michael Kommenda
  – Diploma in computer sciences (2007)
    Upper Austria University of Applied Sciences, Austria
  – MSc in bioinformatics (2011)
    Upper Austria University of Applied Sciences, Austria
  – Research associate at the Research Center Hagenberg
  – Joined HEAL in 2006
  – One of the main architects of HeuristicLab

• Andreas Scheibenpflug
  – MSc in computer sciences (2011)
    Upper Austria University of Applied Sciences, Austria
  – Research assistant at the Research Center Hagenberg
  – Joined HEAL in 2010
  – One of the main architects of HeuristicLab
HeuristicLab Team

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

Softwarepark 11
A-4232 Hagenberg
AUSTRIA

WWW: http://heal.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
- 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
- Introduce Metaheuristics and Evolutionary Algorithms
- Demonstrate basic features
- Demonstrate editing and analysis of optimization experiments
- Demonstrate custom algorithms and graphical algorithm designer
- Outline some additional features
Introduction to HeuristicLab

• 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 3.3.5 released on July 9th, 2011
Where to get HeuristicLab?

• Download binaries
  – deployed as ZIP archives
  – latest stable version 3.3.5
    • released on July 9th, 2011
  – daily trunk build
  – http://dev.heuristiclab.com/download

• Check out sources
  – SVN repository
  – HeuristicLab 3.3.5 tag
    • http://dev.heuristiclab.com/svn/hl/core/tags/3.3.5
  – current development trunk
    • http://dev.heuristiclab.com/svn/hl/core/trunk

• License
  – GNU General Public License (Version 3)

• System requirements
  – Microsoft .NET Framework 4.0 Full Version
  – enough RAM and CPU power ;(-)
Plugin Infrastructure

- HeuristicLab consists of many assemblies
  - 95 plugins in HeuristicLab 3.3.5
  - 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
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
Graphical User Interface

Algorithm View

Problem View

Parameter Collection 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
Metaheuristics

- There are problems which can’t be exactly solved in feasible time (e.g. NP-hard)
  - High complexity
  - Large search space
  - Curse of dimensionality/Combinatorial explosion
  - But: Solution evaluation is cheap

- → Try to generate a solution
- Check if you can generate a better solution
- Metaheuristics don’t (always) find the best solution, but a good solution
Evolutionary Algorithms

– Idea:
  • Simulation of natural evolution
  • Adaptation of species considered as an optimization process
  • Population-based optimization

Mechanisms of optimization:
• Evaluation on the basis of fitness function
• Selection on the basis of fitness
• Solution manipulation
• Acceptance or rejection

Mechanisms of evolution:
• Selection
• Variation
• Replication
Flowchart of an EA

- Initial population of individuals is generated randomly or heuristically.

- Individuals are evaluated and are assigned a certain fitness value.

- While termination condition is not met
  - Fitter individuals are selected for reproduction and children are produced by applying crossover and mutation on the parents.

  - The new individuals are evaluated.

  - New population is generated from the new and/or old individuals.

End while.
Genetic Algorithms

Population at time t

Population at time t+1

selection

crossover

mutation
Crossover

• Simple but effective:
  – Single-Point Crossover

• Generalizations:
  – Two-Point Crossover
  – n-Point Crossover

• Crossover points are set randomly
Mutation

• Bit-Flip with little probability

• Generally one bit is changed in mutant (in case of very long chromosomes sometimes even more bits)
• Mutation rate defines probability of being mutated
Available Algorithms & Problems

**Algorithms**
- Genetic Algorithm
- Island Genetic Algorithm
- Offspring Selection Genetic Algorithm
- Island Offspring Selection Genetic Algorithm
- SASEGASA
- Evolution Strategy
- NSGA-II
- Particle Swarm Optimization
- Local Search
- Simulated Annealing
- Tabu Search
- Variable Neighborhood Search
- Linear Regression
- Linear Discriminant Analysis
- Support Vector Machine
- k-Means
- User-defined Algorithm

**Problems**
- Single-Objective Test Function
- Traveling Salesman Problem
- Quadratic Assignment Problem
- Vehicle Routing Problem
- Scheduling
- Knapsack
- OneMax
- Data Analysis
- Regression
- Symbolic Regression
- Classification
- Symbolic Classification
- Clustering
- Artificial Ant
- External Evaluation Problem
- User-defined Problem

IMMM 2011
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

double-click to open sample algorithms and problems
Create Algorithm
Create or Load Problem
Import or Parameterize Problem Data
Parameterize Algorithm
Start, Pause, Resume, Stop and Reset
Inspect Results

![Image of HeuristicLab Optimizer 3.3.3.5837 displaying Genetic Algorithm results with a graph showing Qualities over generations.]
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

drag & drop here to add algorithms, problems, 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
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
## RunCollection Tabular View

![Image of RunCollection Tabular View](image)

### Table Content

<table>
<thead>
<tr>
<th>Genetic Algorithm (Mutation Rate 1%) Run 13</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%) Run 14</td>
<td>3110</td>
<td>[Details]</td>
<td>16405</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>16843.13</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 15</td>
<td>3110</td>
<td>[Details]</td>
<td>14783</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>15023.02</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 16</td>
<td>3110</td>
<td>[Details]</td>
<td>14252</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>14282.39</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 17</td>
<td>3110</td>
<td>[Details]</td>
<td>13243</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13245.95</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 18</td>
<td>3110</td>
<td>[Details]</td>
<td>13703</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13749.98</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 19</td>
<td>3110</td>
<td>[Details]</td>
<td>13854</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13951.09</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 20</td>
<td>3110</td>
<td>[Details]</td>
<td>15421</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>15431.74</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 21</td>
<td>3110</td>
<td>[Details]</td>
<td>14409</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>15147</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 22</td>
<td>3110</td>
<td>[Details]</td>
<td>13771</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13954.56</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 23</td>
<td>3110</td>
<td>[Details]</td>
<td>14529</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>14532.3</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 24</td>
<td>3110</td>
<td>[Details]</td>
<td>13055</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13642.7</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 25</td>
<td>3110</td>
<td>[Details]</td>
<td>12403</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>12810.09</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 26</td>
<td>3110</td>
<td>[Details]</td>
<td>14091</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>14653.98</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 27</td>
<td>3110</td>
<td>[Details]</td>
<td>12595</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13257.99</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 28</td>
<td>3110</td>
<td>[Details]</td>
<td>12792</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13264.38</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 29</td>
<td>3110</td>
<td>[Details]</td>
<td>12711</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13151.19</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 30</td>
<td>3110</td>
<td>[Details]</td>
<td>12326</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>12625.78</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 31</td>
<td>3110</td>
<td>[Details]</td>
<td>13346</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13777.85</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 32</td>
<td>3110</td>
<td>[Details]</td>
<td>12807</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13264.81</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 33</td>
<td>3110</td>
<td>[Details]</td>
<td>12741</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>13113.18</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 34</td>
<td>3110</td>
<td>[Details]</td>
<td>15921</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>18894.04</td>
</tr>
<tr>
<td>Genetic Algorithm (Mutation Rate 1%) Run 35</td>
<td>3110</td>
<td>[Details]</td>
<td>16384</td>
<td>334.5903245</td>
<td>OrderCross...</td>
<td>19603.36</td>
</tr>
</tbody>
</table>
RunCollection 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
RunCollection BubbleChart
RunCollection 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
RunCollection BoxPlots

![Image of RunCollection BoxPlots](http://dev.heuristiclab.com)
RunCollection 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)
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
  – ...

IMMM 2011
http://dev.heuristiclab.com
TSPAlleleFrequencyAnalyzer
TSPPopualtionDiversityAnalyzer
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

HeuristicLab Optimizer 3.3.3.5837

Available Operators

- Genetic Algorithm
  - Name: Genetic Algorithm
  - Problem: Test Functions 3.3
  - Parameters: Traveling Salesman 3.3
  - Results: Vehicle Routing 3.3
- HeuristicLab Selection 3.3
  - Best Selector
  - Conditional Selector
  - Crowded Tournament Selector
  - Gender Specific Selection
  - LeftReducer
  - LeftSelector
  - LinearRankSelector
  - Merging Reducer
  - NoSameMatesSelector
  - OffspringSelector
  - ReproductionSelector
  - RandomReducer
  - RandomSelector
  - Replicator
  - RightChildReducer
  - RightReducer
  - RightSelector
  - TournamentSelector
  - WorkReducer
  - WorkSelector

A quality proportional selection operator which considers a single double quality value for selection.

Execution Time: 00:00:03.5182012

Details

IMMM 2011

http://dev.heuristiclab.com
Building User-Defined Algorithms
Programmable Operators

```java
public class ProgrammableSingleSuccessorOperator {
    public static IOperation Execute(ProgrammableSingleSuccessorOperator op, ...
    // implement custom operator

    return op.Successor == null ? null : context.CreateOperation(op.Successor, ...
}
```
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
• Validation techniques
Introduction to Data-based Modeling

\[ y = f(x, w) + \varepsilon \]

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>52.0085</td>
<td>58497.7</td>
<td>26.9956</td>
<td>24.0391</td>
<td>12.9005</td>
<td>66.2633</td>
<td>4.027</td>
</tr>
<tr>
<td>52.9874</td>
<td>59181.1</td>
<td>27.0061</td>
<td>23.7483</td>
<td>12.7239</td>
<td>66.8367</td>
<td>4.154</td>
</tr>
<tr>
<td>54.0505</td>
<td>58879.2</td>
<td>26.8679</td>
<td>23.5869</td>
<td>12.5877</td>
<td>67.6773</td>
<td>4.047</td>
</tr>
<tr>
<td>55.3182</td>
<td>58982.3</td>
<td>27.1852</td>
<td>23.7869</td>
<td>12.6229</td>
<td>67.7128</td>
<td>4.118</td>
</tr>
<tr>
<td>46.5388</td>
<td>59270.3</td>
<td>26.4421</td>
<td>24.2601</td>
<td>12.5892</td>
<td>65.7425</td>
<td>4.743</td>
</tr>
<tr>
<td>56.0167</td>
<td>58234.5</td>
<td>26.1072</td>
<td>23.4412</td>
<td>12.4224</td>
<td>67.9462</td>
<td>4.042</td>
</tr>
<tr>
<td>57.0168</td>
<td>58021.7</td>
<td>26.2513</td>
<td>23.3085</td>
<td>12.3036</td>
<td>69.4392</td>
<td>3.721</td>
</tr>
<tr>
<td>57.0139</td>
<td>59228.5</td>
<td>26.2285</td>
<td>23.4041</td>
<td>12.3685</td>
<td>68.3322</td>
<td>3.593</td>
</tr>
<tr>
<td>62.5214</td>
<td>49855.7</td>
<td>26.3076</td>
<td>22.7812</td>
<td>12.0711</td>
<td>70.3331</td>
<td>3.785</td>
</tr>
<tr>
<td>69.9885</td>
<td>53534.4</td>
<td>25.5916</td>
<td>21.3448</td>
<td>11.0299</td>
<td>73.7138</td>
<td>3.028</td>
</tr>
<tr>
<td>70.0156</td>
<td>55058.2</td>
<td>25.5759</td>
<td>21.4617</td>
<td>11.0915</td>
<td>72.9846</td>
<td>3.137</td>
</tr>
<tr>
<td>70.0446</td>
<td>56099.4</td>
<td>26.059</td>
<td>21.4652</td>
<td>11.0684</td>
<td>73.2351</td>
<td>3.175</td>
</tr>
<tr>
<td>70.0335</td>
<td>55891.6</td>
<td>26.1834</td>
<td>21.3318</td>
<td>10.6288</td>
<td>74.8925</td>
<td>3.165</td>
</tr>
<tr>
<td>70.0164</td>
<td>56841.1</td>
<td>26.1791</td>
<td>21.2119</td>
<td>10.0354</td>
<td>74.8419</td>
<td>3.175</td>
</tr>
<tr>
<td>70.0147</td>
<td>58254.6</td>
<td>26.7381</td>
<td>21.3367</td>
<td>10.9244</td>
<td>73.2341</td>
<td>3.175</td>
</tr>
<tr>
<td>70.0061</td>
<td>54444.9</td>
<td>26.5786</td>
<td>21.2395</td>
<td>10.6505</td>
<td>74.9187</td>
<td>3.175</td>
</tr>
<tr>
<td>70.0325</td>
<td>48858.2</td>
<td>26.3799</td>
<td>21.4135</td>
<td>10.8558</td>
<td>74.0993</td>
<td>3.175</td>
</tr>
</tbody>
</table>

Training

Test
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

- Common tasks
  - Regression (real-valued labels)
  - Classification (discrete labels)
  - Clustering (no labels, group similar observations)
Data-based Modeling Algorithms in HeuristicLab

• Symbolic regression and classification based on genetic programming

• External Libraries:
  – Support Vector Machines for Regression and Classification
  – Linear Regression
  – Linear Discriminate Analysis
  – K-Means clustering
  – Random Forests for Regression and Classification
  – Neural Networks
Case Studies

• Demonstration
  – problem configuration
    • data import
    • target variable
    • input variables
    • data partitions (training and test)
  – analysis of results
    • accuracy metrics
    • visualization of model output
Case Study: Regression

• Poly-10 benchmark problem dataset
  – 10 input variables $x_1 \ldots x_{10}$
  – $y = x_1 \cdot x_2 + x_3 \cdot x_4 + x_5 \cdot x_6 + x_1 \cdot x_7 \cdot x_9 + x_3 \cdot x_6 \cdot x_{10}$
  – non-linear modeling approach necessary

  – frequently used in GP literature

  – download
    http://dev.heuristicslab.com/AdditionalMaterial#IMMM2011
Linear Regression

- Create new algorithm
Import Data from CSV-File

![Import Data from CSV-File](image-url)
Inspect and Configure Dataset

IMMM 2011

http://dev.heuristiclab.com
Inspect Imported Data
Set Target Variable
Select Input Variables
Configure Training and Test Partitions

![Configuration interface](Image)
Run Linear Regression
Inspect Results
Inspect Scatterplot of Predicted and Target Values
Inspect Linechart
Inspect Graphical Representation of Model
Nonlinear Regression Methods

• Random Forests Regression
  – Learns multiple models on different sample subsets

• Support Vector Regression
  – Maps data to higher dimensional space
  – Calculates linear model
  – Transfers the data back to the input space
Symbolic Regression with HeuristicLab

- Linear regression produced an inaccurate model.
- Nonlinear methods produces overfit models
- 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
Genetic Programming Life Cycle

Population of Models

Child selection

New generation

Calculation of correlation

Parents selection

Optional sample selection

Generation of new models

Evaluation

IMMM 2011 http://dev.heuristiclab.com
Generating new Models

\[
\frac{5.0}{\ln X_1(t-1) + X_2(t-2) \cdot X_3(t-2) \cdot X_2(t-2)} - 1.5
\]

\[
\frac{5.0}{X_1(t-1) + X_3(t-2) \cdot X_2(t-2) - \frac{5.0}{X_1(t-1)}} - 1.5
\]
Mutating 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
Inspect Data and Configure Dataset
Set Target and Input Variables
Configure Maximal Model Depth and Length

IMMM 2011  http://dev.heuristiclab.com  86
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
  • MATLAB
  • LaTeX
Detailed Model Analysis and Simplification
Symbolic Simplification and Node Impacts
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{align*}
\text{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{align*}

\begin{align*}
    c_0 &= 0.08133712120642195 \\
    c_1 &= 0.190955016563887 \\
    c_2 &= -0.029881174429839 \\
    c_3 &= 0.07889183541302 \\
    c_4 &= -0.0103065273366223 \\
    c_5 &= 0.0316849536390699 \\
    c_6 &= -0.047070758925129 \\
    c_7 &= -0.0291939124032144 \\
    c_8 &= 0.00157679665070775 \\
    c_9 &= 0.105250443686677 \\
    c_{10} &= 0.0200987846293256 \\
\end{align*}
LaTeX Export

\begin{align*}
\text{Result} &= x_4(t) - x_3(t) - 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 (c_{14} x_4(t) + c_{15} x_5(t) + \frac{1}{c_{17} x_2(t)})} \cdot c_{19} \right) + 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} &= 0.0031437698160385 \\
c_{17} &= -3.00832012161288 \\
c_{18} &= 0.867137925013337 \\
c_{19} &= -5.451900909899249 \\
c_{20} &= -0.204498330755849 \\
c_{21} &= -0.0465539907207764
\end{align*}
Variable Relevance Analysis

• Which variables are important to predict classes correctly?

• Demonstration
  – Variable frequency analyzer
  – Symbol frequency analyzer
  – Variable impacts

![Variable frequency graph](image1)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Relative Variable Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>x4</td>
<td>0.302803869106054</td>
</tr>
<tr>
<td>x3</td>
<td>0.24170172985569</td>
</tr>
<tr>
<td>x1</td>
<td>0.179112367914678</td>
</tr>
<tr>
<td>x10</td>
<td>0.0589664719249172</td>
</tr>
<tr>
<td>x2</td>
<td>0.0544635184742382</td>
</tr>
<tr>
<td>x6</td>
<td>0.044674403657897</td>
</tr>
<tr>
<td>x8</td>
<td>0.0436011597048278</td>
</tr>
<tr>
<td>x7</td>
<td>0.0331173502974243</td>
</tr>
<tr>
<td>x5</td>
<td>0.0226252246461621</td>
</tr>
<tr>
<td>x9</td>
<td>0.01946242278034</td>
</tr>
</tbody>
</table>
Inspect Variable Frequency Chart
Inspect Variable Impacts
Inspect Symbol Frequencies

Symbol frequencies

- ProgramRootSymbol
- StartSymbol
- Variable
- Division
- Constant
- Subtraction

Relative Symbol Frequency

Constant X = 21, Y = 0.3126176270372
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)

  – download
    [http://dev.heuristiclab.com/AdditionalMaterial#IMMM2011](http://dev.heuristiclab.com/AdditionalMaterial#IMMM2011)
Open Sample

HeuristicLab Optimizer 3.3.3.5837

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 ▼ (Reset Algorithm) to prepare a new run

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolution Strategy - Griewank</td>
<td>An evolution strategy which solves the 10-dimensional Griewank test function</td>
</tr>
<tr>
<td>Genetic Algorithm - TSP</td>
<td>A genetic algorithm which solves the &quot;130&quot; traveling salesman problem (imported from TSPLIB)</td>
</tr>
<tr>
<td>Genetic Algorithm - VRP</td>
<td>A genetic algorithm which solves the &quot;C101&quot; vehicle routing problem (imported from Solomon)</td>
</tr>
<tr>
<td>Genetic Programming - Artificial Art</td>
<td>A standard genetic programming algorithm to solve the artificial art problem (Sante-Fe test)</td>
</tr>
<tr>
<td>Genetic Programming - Symbolic Classification</td>
<td>A standard genetic programming algorithm to solve a classification problem (Mammographic-Mass dataset)</td>
</tr>
<tr>
<td>Genetic Programming - Symbolic Regression</td>
<td>A standard genetic programming algorithm to solve a symbolic regression problem (lower dataset)</td>
</tr>
<tr>
<td>Island Genetic Algorithm - TSP</td>
<td>An island genetic algorithm which solves the &quot;130&quot; traveling salesman problem (imported from TSPLIB)</td>
</tr>
<tr>
<td>Local Search - Knapsack</td>
<td>A local search algorithm that solves a randomly generated Knapsack problem</td>
</tr>
</tbody>
</table>
Configure and Run Algorithm
Inspect Quality Linechart
Inspect Best Training Solution
Inspect Model Output and Thresholds
Inspect Confusion Matrix
Inspect ROC Curve

![ROC Curve Image]
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
Inspect Best Model on Validation Partition

[Image of software interface showing a scatter plot and regression analysis details]
Inspect Linechart of Correlation of Training and Validation Fitness
Agenda

- Objectives of the Tutorial
- Introduction to Metaheuristics
- 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, 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
  – scatter search
  – ...

• Cloud Computing
  – port HeuristicLab Hive to Windows Azure

• Linux
  – port HeuristicLab to run on Mono and Linux machines

• Have a look at the HeuristicLab roadmap
  – http://dev.heuristiclab.com/trac/hl/core/roadmap

• Any other ideas, requests or recommendations?
  – please write an e-mail to support@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

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

http://dev.heuristiclab.com

heuristiclab@googlegroups.com

support@heuristiclab.com