Algorithm and Experiment Design with HeuristicLab
An Open Source Optimization Environment for Research and Education

Stefan Wagner & Gabriel Kronberger
Heuristic and Evolutionary Algorithms Laboratory (HEAL)
School of Informatics/Communications/Media, Campus Hagenberg
University of Applied Sciences Upper Austria
Hagenberg, Austria
{swagner, gkronber}@heuristiclab.com
http://dev.heuristiclab.com

Copyright is held by the author/owner(s).
GECCO’12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

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 (2006)
  - 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
  - Research assistant (2005 – 2011)
  - 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 3.3.7 released on July 8th, 2012

Where to get HeuristicLab?

• Download binaries
  – deployed as ZIP archives
  – latest stable version 3.3.7
  – released on July 8th, 2012
  – daily trunk builds
  – http://dev.heuristiclab.com/download

• Check out sources
  – SVN repository
  – HeuristicLab 3.3.7 tag
    http://dev.heuristiclab.com/svn/hl/core/tags/3.3.7
  – 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
  – 119 plugins in HeuristicLab 3.3.7
  – 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
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

• 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 & Problems

Algorithms

- Evolution Strategy
- Genetic Algorithm
- Genetic Programming
- Hungarian Algorithm
- Island Genetic Algorithm
- Island Offspring Selection Genetic Algorithm
- Local Search
- Neighborhood Optimization
- Island Offspring Selection Genetic Algorithm
- Problem-Specific Optimization
- Robust Future Search
- NEHSMOA
- Local Search
- Tabu Search
- User-Defined Algorithm
- Variable Neighborhood Search
- Performance Benchmark
- Cross Validation
- Invalidation
- Automated Parameter Analysis
- Linear Regression
- Multivariate Linear Regression
- Nearest Neighbor Regression and Classification
- Neural Network Regression and Classification
- Random Forest Regression and Classification
- Support Vector Regression and Classification

Problems

- Artificial Ant
- Classification
- Clustering
- External Evaluation Problem
- Knapsack
- Lawn Mower
- Linear Assignment
- Onemax
- Quadratic Assignment
- Regression
- Single-Objective Test Function
- Symbolic Classification
- Symbolic Regression
- Traveling Salesman
- User-defined Problem
- Vehicle Routing
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

Create Algorithm

double-click to open sample algorithms and problems
Inspection Results

• 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

Compare Runs

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
Create Batch Runs and Experiments

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
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
    - RunCollection DataTable View

- Filtering is automatically applied to all open run collection views

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

• 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

• 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)
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
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
  - ...
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
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

Symbolic regression with HeuristicLab
Model simplification
Variable relevance analysis
Classification with HeuristicLab

Reminder: Data-based Modeling

Learn association of input variables to labels
Common tasks
- Regression (real-valued labels)
- Classification (discrete labels)
- Clustering (no labels, group similar observations)

\[ y = f(x, w) + \varepsilon \]
Data-based Modeling in HeuristicLab

- Symbolic regression and classification based on genetic programming
- External libraries (alglib, libSVM):
  - Linear Regression
  - Linear Discriminate Analysis
  - Support Vector Machines
  - Random Forest
  - Neural Networks
  - K-Means Clustering

Case Studies

- Demonstration
  - problem configuration
  - analysis of results
    - accuracy metrics
    - visualization of model output

Linear Regression

- Create new algorithm

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
Symbolic Regression with HeuristicLab

- Goal: produce a nonlinear symbolic regression model using genetic programming
- Reminder: Symbolic Regression
  - 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
- Analysis of results
  - model accuracy
  - model structure
Case Study

- Symbolic regression GP benchmark instances available
- In the following: Poly-10 benchmark problem
  - 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
  - download: [http://dev.heuristiclab.com/AdditionalMaterial#GECCO2012](http://dev.heuristiclab.com/AdditionalMaterial#GECCO2012)

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

Configure Function Set (Grammar)
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

Automatic Symbolic Simplification

Manual Simplification

Textual Representations Are Also Available
- Use ViewHost to switch to textual representation view.
Variable Relevance Analysis

- Which variables are important to predict classes correctly?
- Demonstration
  - variable frequency analyzer
  - symbol frequency analyzer
  - variable impacts
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#GECCO2012
Inspect Best Training Solution

Inspect Model Output and Thresholds

Inspect Confusion Matrix

Inspect ROC Curve
Validation of Results

• Overfitting

• Strategies to reduce overfitting
  – validation partition
  – cross-validation

GECCO 2012 http://dev.heuristiclab.com 109

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

GECCO 2012 http://dev.heuristiclab.com 110

Configuration of Validation Partition

Inspect Best Model on Validation Partition

GECCO 2012 http://dev.heuristiclab.com 111

GECCO 2012 http://dev.heuristiclab.com 112
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, AnyLogic, ...)
  - supports different protocols (command line parameters, TCP, ...)
- Meta-optimization
  - 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
  - job shop scheduling
- 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?
  - join our HeuristicLab Google group heuristiclab@googlegroups.com or write an e-mail to support@heuristiclab.com
  - Friend our Facebook group
  - Follow our youtube channel www.youtube.com/user/HeuristicLab
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

• S. Wagner, S. Winkler, R. Braune, M. Kofler, A. Beham, M. Affenzeller
  Benefits of plugin based heuristic optimization software systems

• S. Wagner, G. Kronberger, A. Beham, S. Winkler, M. Affenzeller
  Benefits of plugin based heuristic optimization software systems

• S. Wagner, G. Kronberger, A. Beham, S. Winkler, M. Affenzeller
  Benefits of plugin based heuristic optimization software systems

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

• S. Wagner, A. Beham, H. Kronberger, M. Kofler, S. Winkler, O. Pitz, D. Affenzeller
  Heuristic optimization software systems - Modeling of heuristic optimization algorithms in the HeuristicLab software environment

• S. Wagner
  A unified approach to metaheuristic optimization
  Actas del séptimo congreso español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'2010), 2010.

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

Questions & Answers

http://dev.heuristiclab.com
heuristiclab@googlegroups.com