The Genetic and Evolutionary Computation Conference

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 and Media
Upper Austria University of Applied Sciences
Hagenberg, Austria

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

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

1

Instructor Biographies

- Stefan Wagner
  - MSc in computer science (2004)
    Johannes Kepler University Linz, Austria
  - PhD in technical sciences (2009)
    Johannes Kepler University Linz, Austria
  - Associate professor (2005 – 2009)
    Upper Austria University of Applied Sciences
  - Full professor for complex software systems (since 2009)
    Upper Austria University of Applied Sciences
  - Co-founder of the HEAL research group
  - Project manager and chief architect of HeuristicLab
  - http://heal.heuristiclab.com/team/wagner

- Gabriel Kronberger
  - MSc in computer science (2005)
    Johannes Kepler University Linz, Austria
  - PhD in technical sciences (2010)
    Johannes Kepler University Linz, Austria
  - Research assistant (since 2005)
    Upper Austria University of Applied Sciences
  - Member of the HEAL research group
  - Architect of HeuristicLab
  - http://heal.heuristiclab.com/team/kronberger

2

Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- Demonstration
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

HeuristicLab

3

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

4

Introduction

HeuristicLab
A Paradigm-Independent and Extensible Environment for Heuristic Optimization

- 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 in June 2011

5

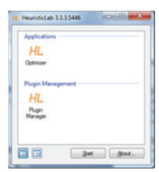
Where to get HeuristicLab?

- Download binaries
  - deployed as ZIP archives
  - latest stable version 3.3.5
    - released in June 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 ;-)

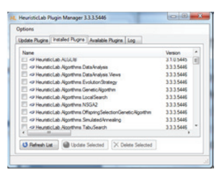
6

Plugin Infrastructure

- HeuristicLab consists of many assemblies
  - for example 74 plugins in HeuristicLab 3.3.3
  - 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



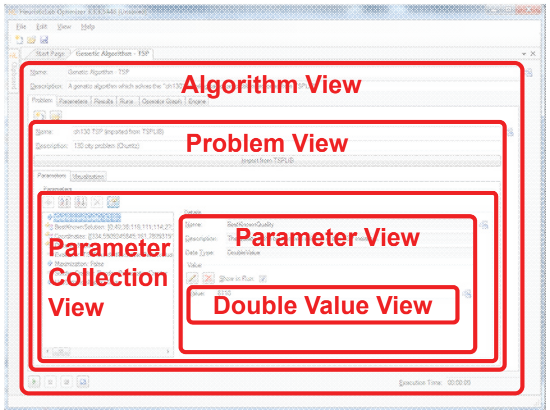
7

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 (default) or moved (shift key)
  - enabled for collection items and content objects

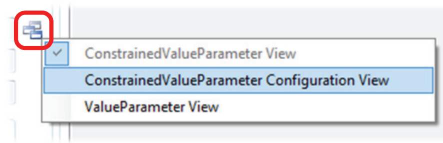
8

Graphical User Interface



Graphical User Interface

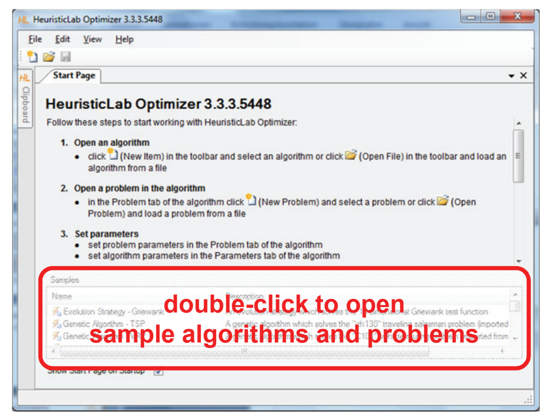
- ViewHost
  - control which hosts views
  - right-click on windows icon to switch views
  - double-click on windows icon to open another view
  - drag & drop windows icon to copy contents



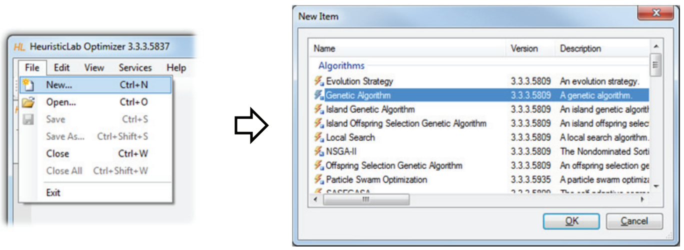
Available Algorithms & Problems

- Algorithms
  - 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

HeuristicLab Optimizer

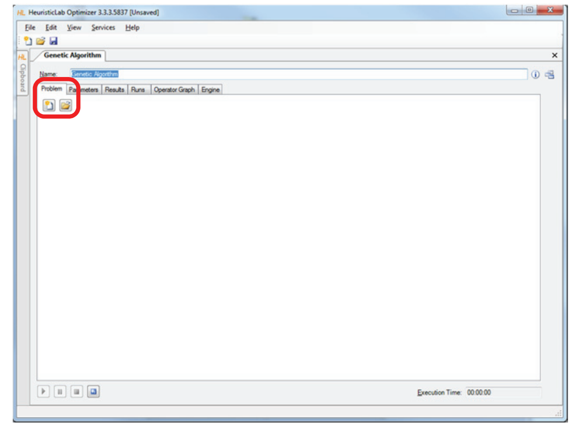


Create Algorithm



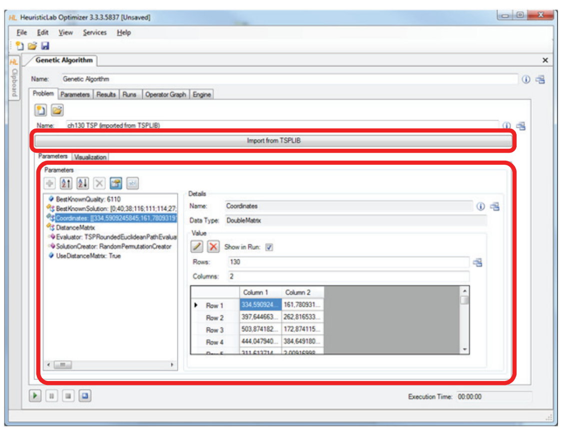
13

Create or Load Problem



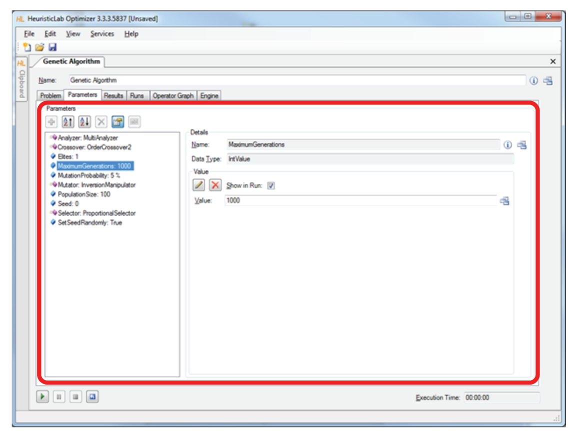
14

Import or Parameterize Problem Data



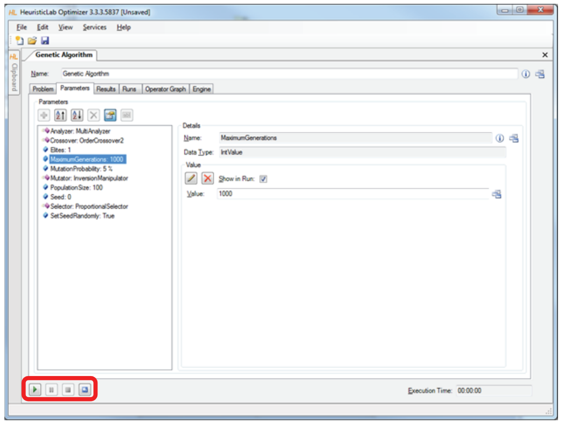
15

Parameterize Algorithm



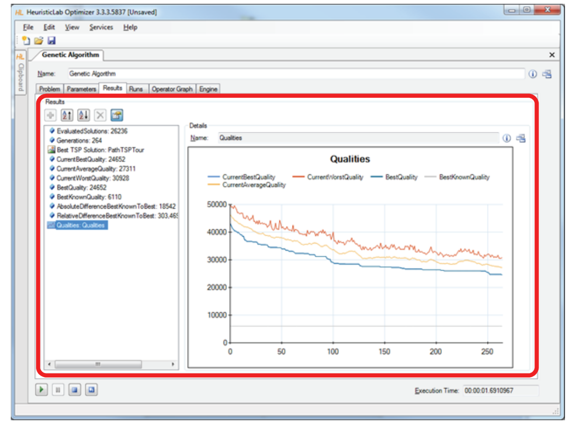
16

Start, Pause, Resume, Stop and Reset



17

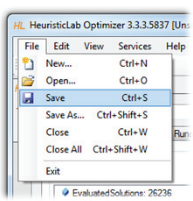
Inspect Results



18

Save and Load

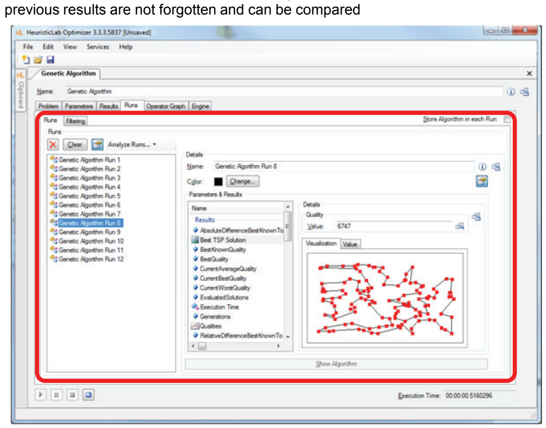
- Save to and load from disk
  - HeuristicLab items (i.e., algorithms, problems, experiments, …) can be saved to and loaded from a file
  - algorithms can be paused, saved, loaded and resumed
  - data format is custom compressed XML
  - saving and loading files might take several minutes
  - saving and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



19

Compare Runs

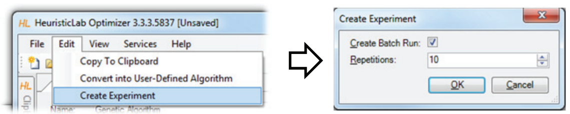
- A run is created each time when the algorithm is stopped
  - runs contain all results and parameter settings
  - previous results are not forgotten and can be compared



20

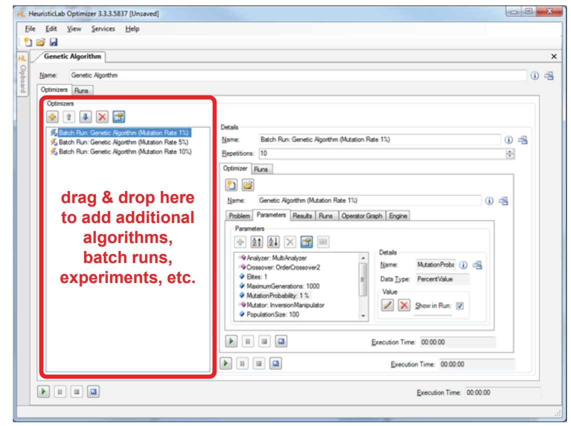
Create Batch Runs and Experiments

- 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



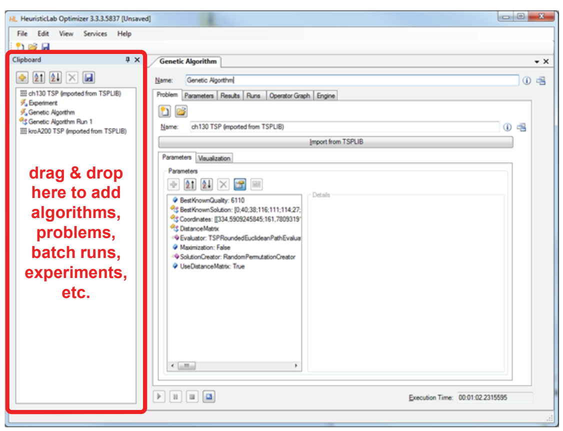
21

Create Batch Runs and Experiments



22

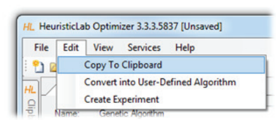
Clipboard



23

Clipboard

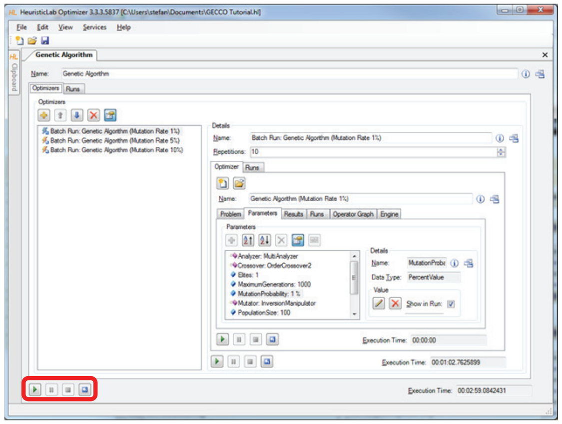
- Store items
  - click on the buttons to add or remove items
  - drag & drop items on the clipboard
  - use the menu to add a copy of a shown item to the clipboard



- 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

24

Start, Pause, Resume, Stop, Reset



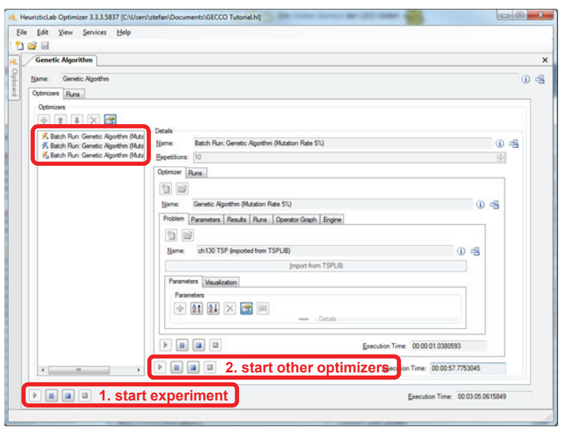
25

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

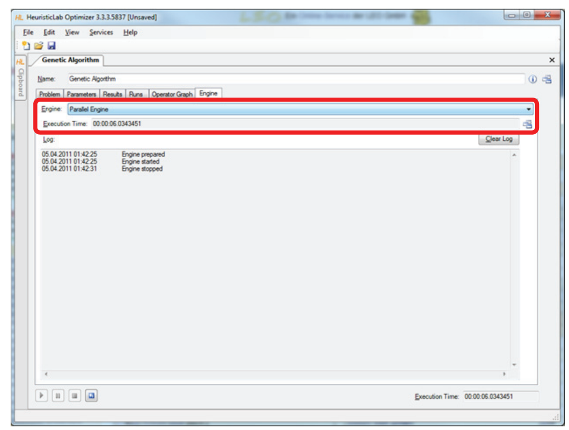
26

Parallel Execution of Experiments



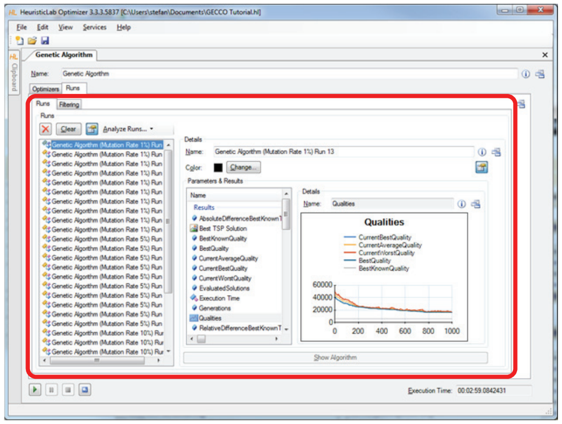
27

Parallel Execution of Algorithms



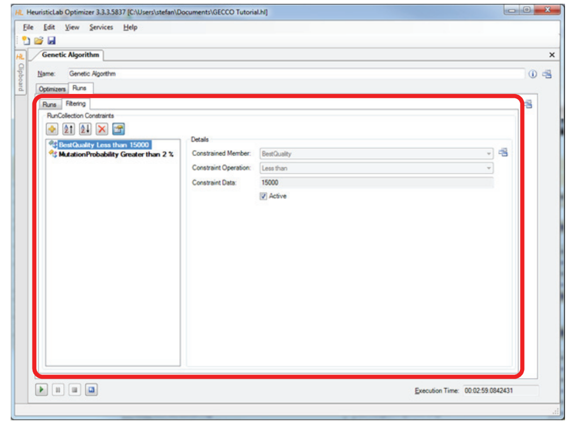
28

Compare Runs



29

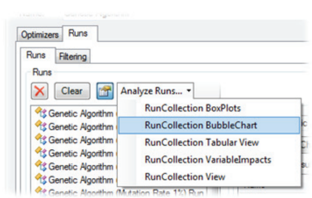
Filter Runs



30

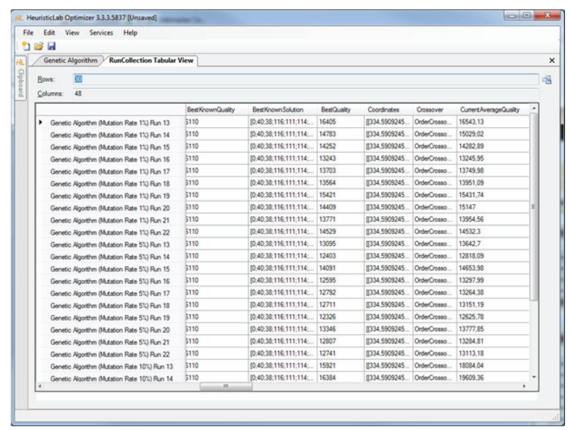
Analyze Runs

- 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



31

RunCollection Tabular View

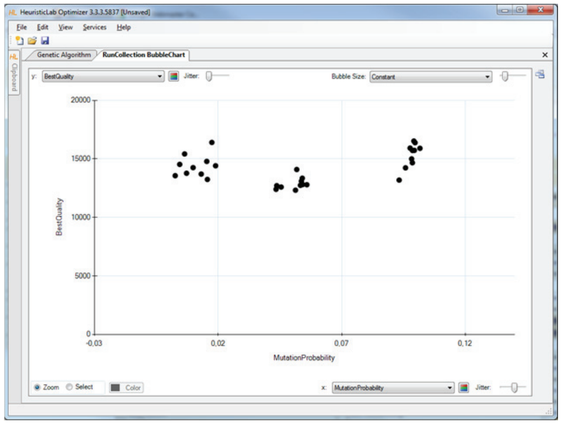


32

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

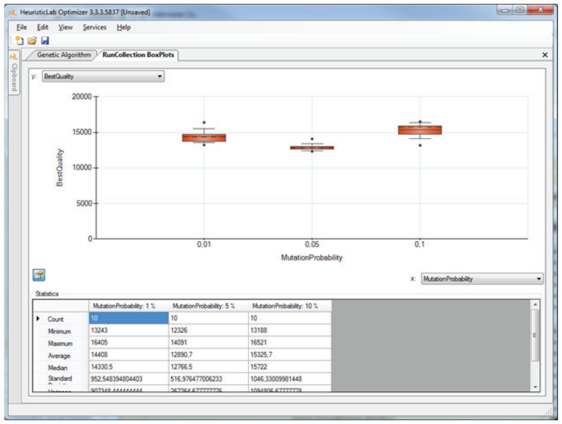
RunCollection BubbleChart



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



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)

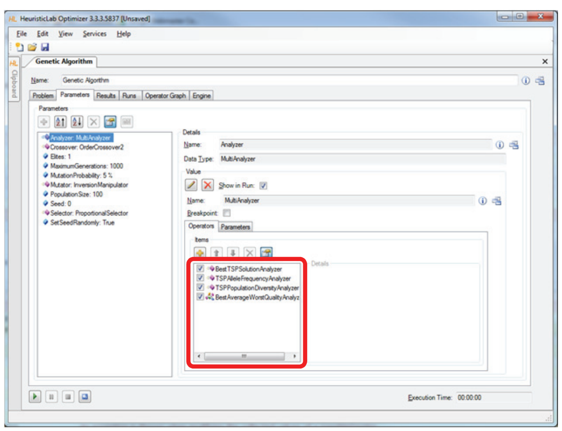
37

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
  - …

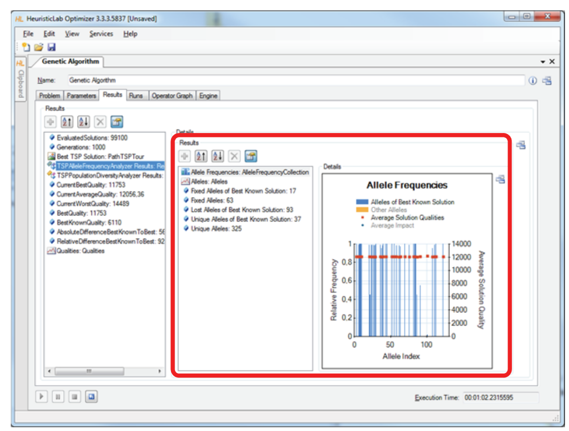
38

Analyzers



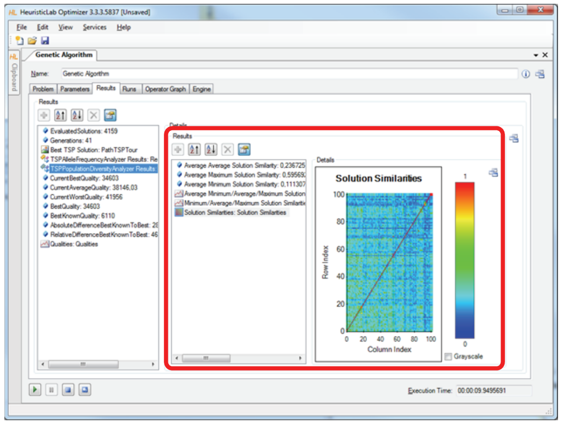
39

TSPAlleleFrequencyAnalyzer



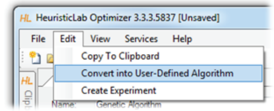
40

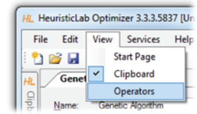
TSPPopulationDiversityAnalyzer



Building User-Defined Algorithms

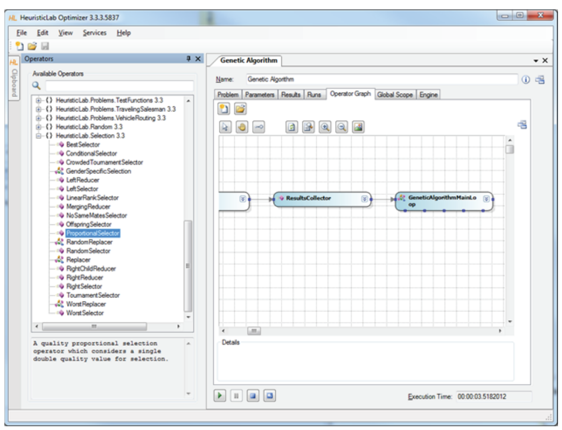
- Operator graphs
  - algorithms are represented as operator graphs
  - operator graphs of user-defined algorithms can be changed
  - algorithms can be defined in the graphical algorithm designer
  - use the menu to convert a standard algorithm into a user-defined algorithm



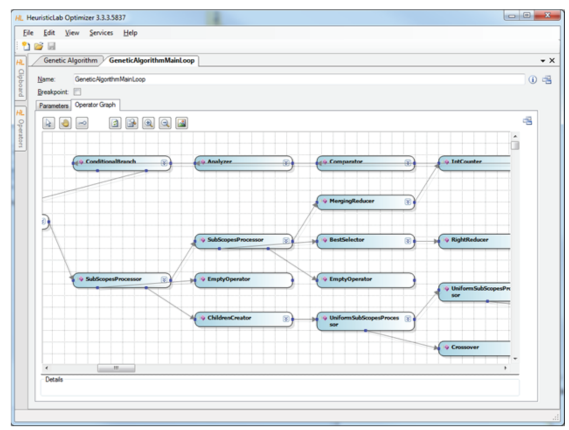


- Operators sidebar
  - drag & drop operators into an operator graph
- Programmable operators
  - add programmable operators in order to implement custom logic in an algorithm
  - no additional development environment needed
- Debug algorithms
  - use the debug engine to obtain detailed information during algorithm execution

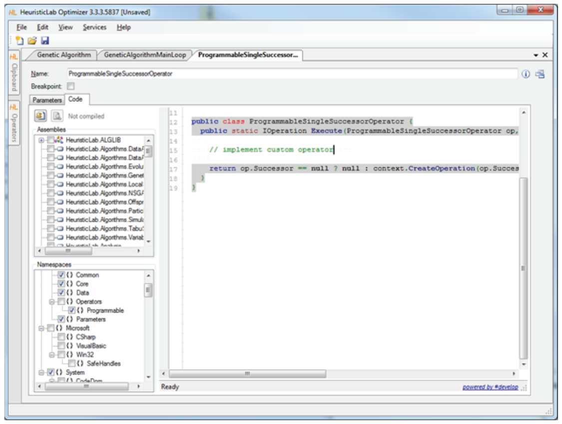
Building User-Defined Algorithms



Building User-Defined Algorithms

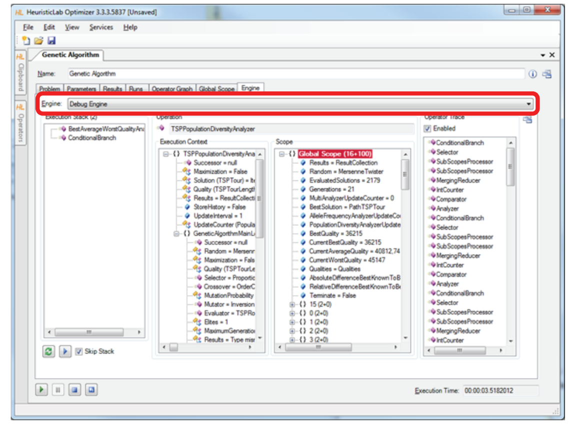


Programmable Operators



45

Debugging Algorithms



46

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 $(y_1 \ldots y_N)^T$
- Goal: learn association of input variable values to labels
- Common tasks
  - Regression (real-valued labels)
  - Classification (discrete labels)
  - Clustering (no labels, group similar observations)

47

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

48

In the next 60 minutes: case studies

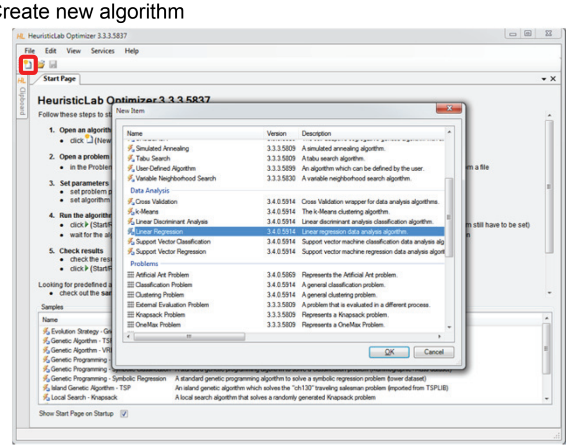
- Regression
  - Artificial benchmark problem dataset *Poly-10*
  - Algorithms:
    - Linear regression
    - Symbolic regression using Genetic Programming
- Classification
  - Real world medical *Mammographic Mass* dataset from the UCI Machine Learning Repository
  - Algorithms:
    - Symbolic classification

Case study: Regression

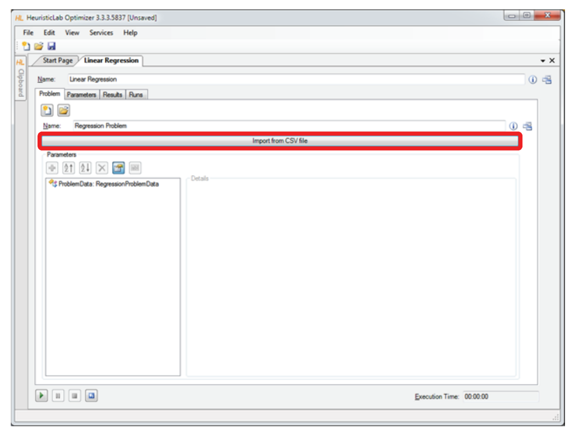
- Poly-10 benchmark problem dataset
  - 10 input variables $x_1 \ldots x_{10}$
  - $y = x_1 x_2 + x_3 x_4 + x_5 x_6 + x_1 x_7 x_9 + x_3 x_6 x_{10}$
  - Non-linear modeling approach necessary
  - Frequently used in GP literature
  - Download: http://dev.heuristiclab.com/AdditionalMaterial#GECCO2011

Linear Regression

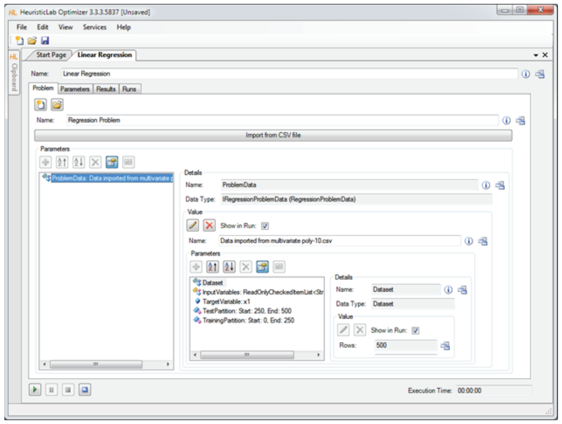
- Create new algorithm



Import Data from CSV-File

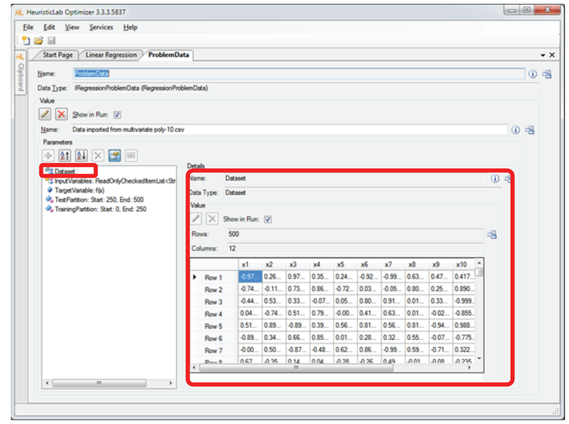


Inspect And Configure Dataset



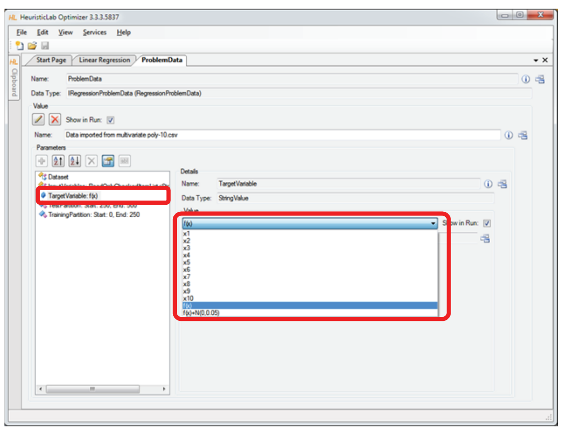
53

Inspect Imported Data



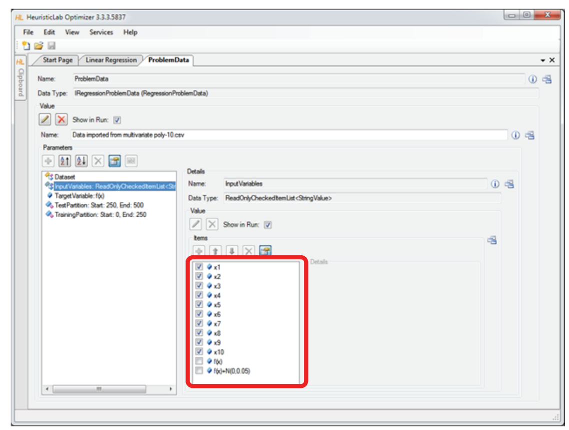
54

Set Target Variable



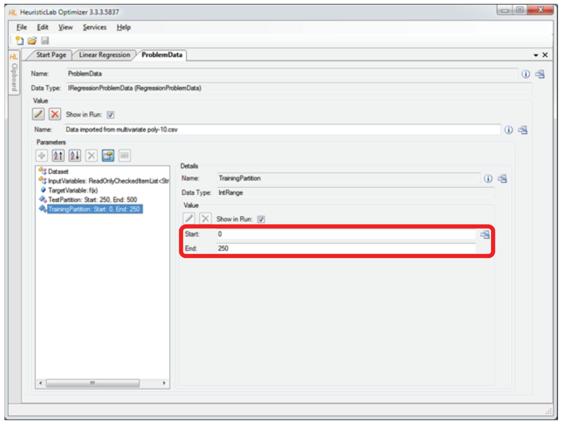
55

Select Input Variables



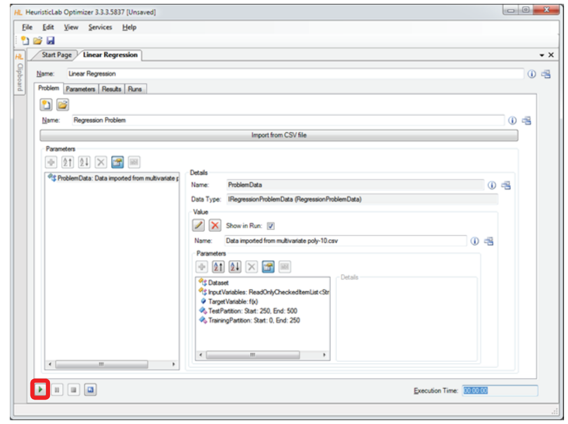
56

Configure Training and Test Partitions



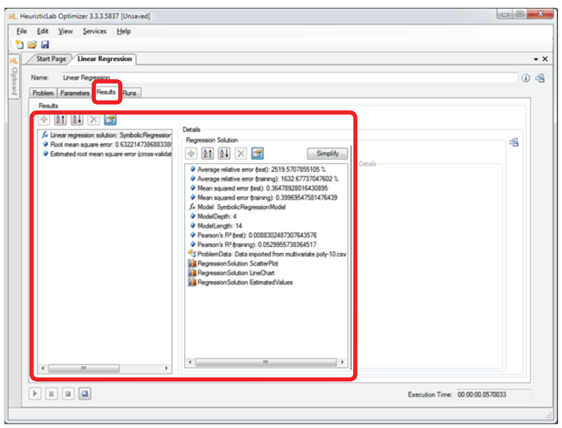
57

Run Linear Regression



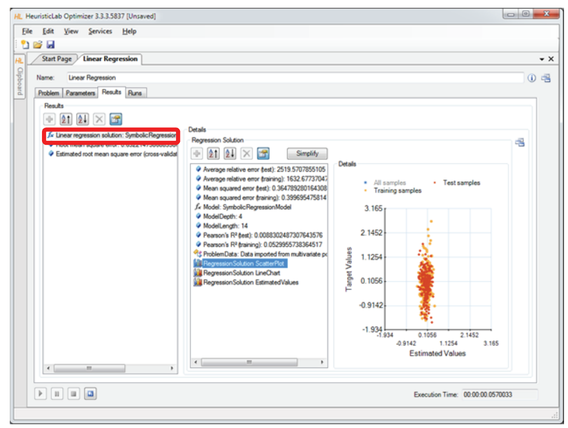
58

Inspect Results



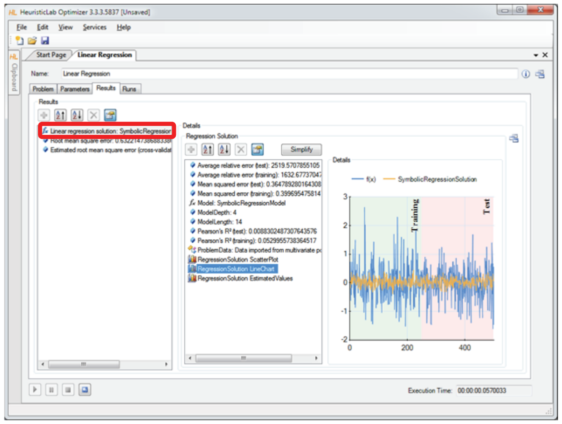
59

Inspect Scatterplot of Predicted and Target Values



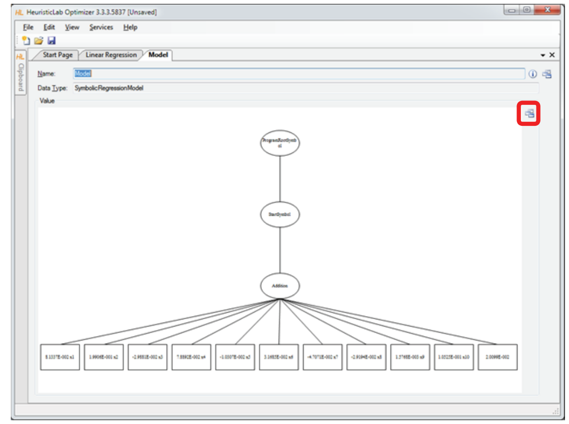
60

Inspect Linechart



61

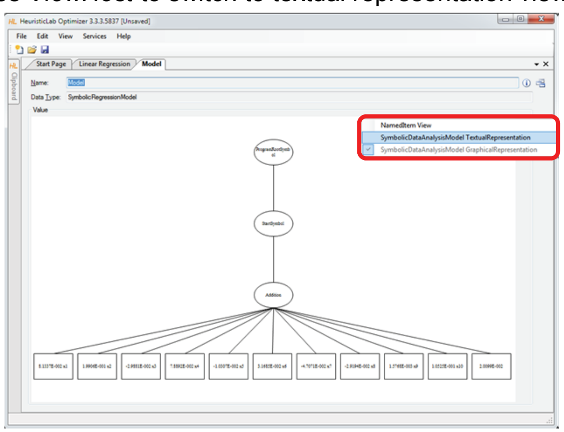
Inspect Graphical Representation of Model



62

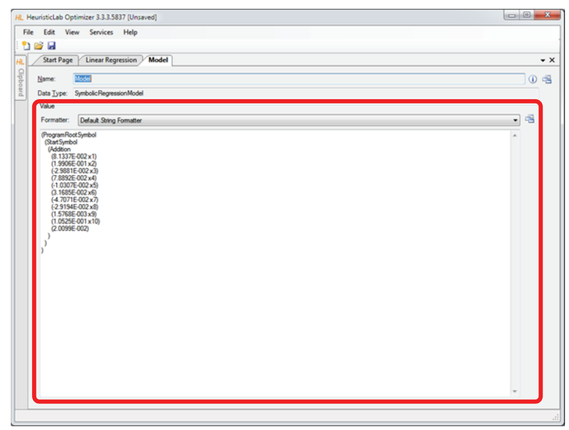
Textual Representations Are Also Available

- Use *ViewHost* to switch to textual representation view.



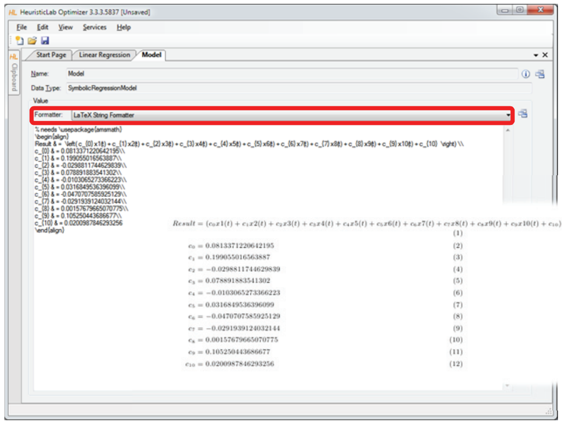
63

Default Textual Representation for Model Export



64

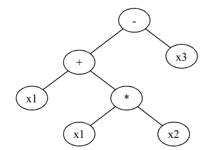
Textual Representation for Export to LaTeX



65

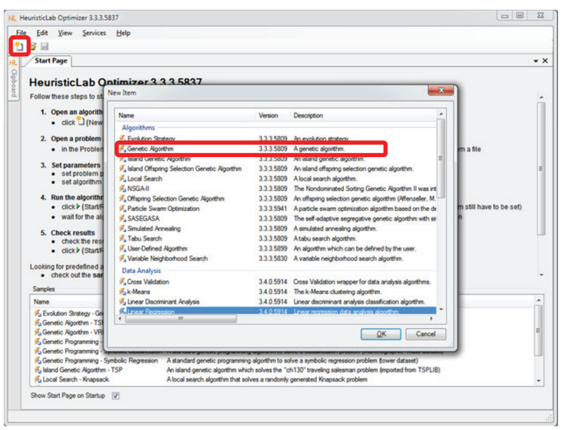
Nonlinear Modeling: Symbolic Regression

- 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



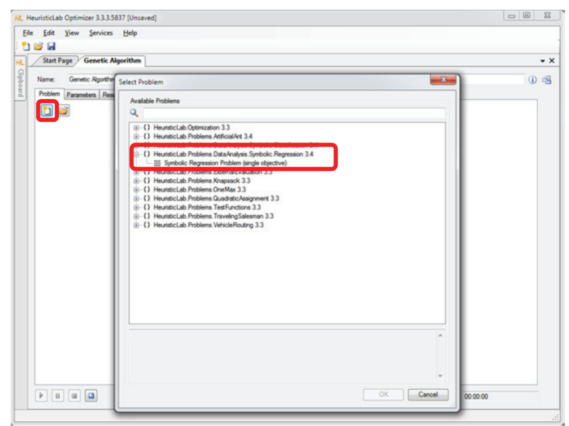
66

Create New Genetic Algorithm



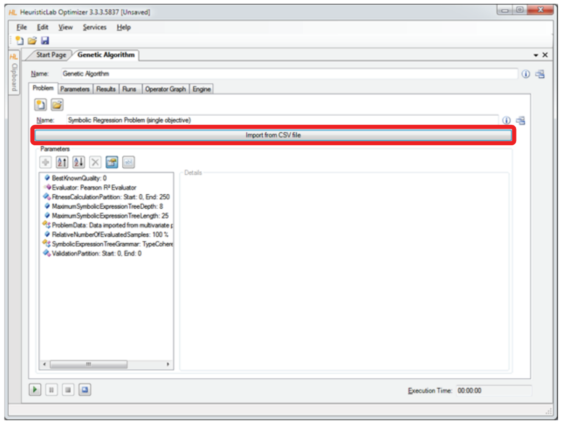
67

Create New Symbolic Regression Problem



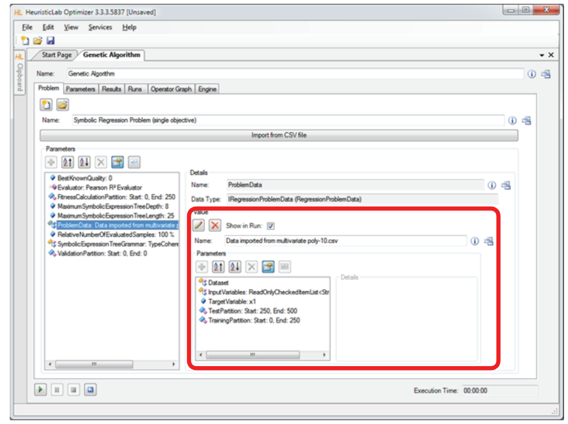
68

Import Data



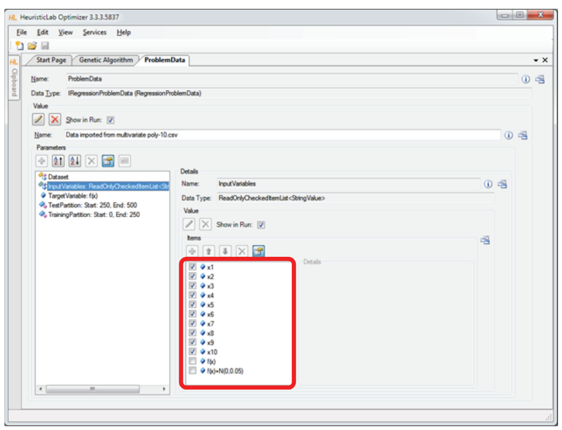
69

Inspect Data and Configure Dataset



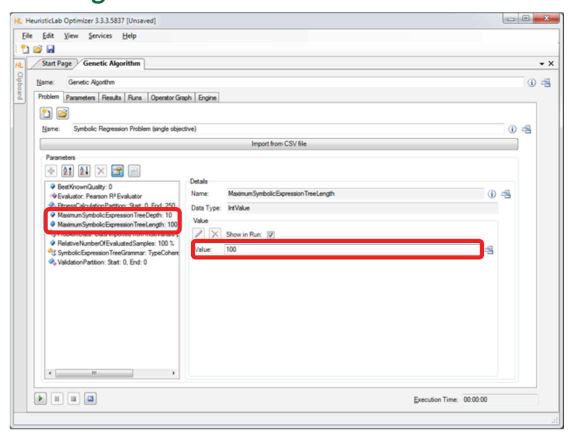
70

Set Target and Input Variables



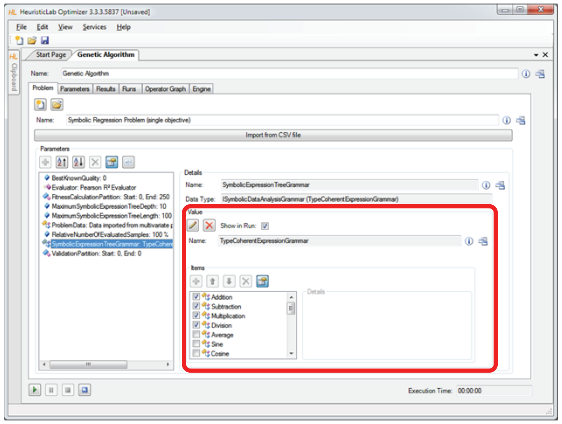
71

Configure Maximal Model Depth and Length



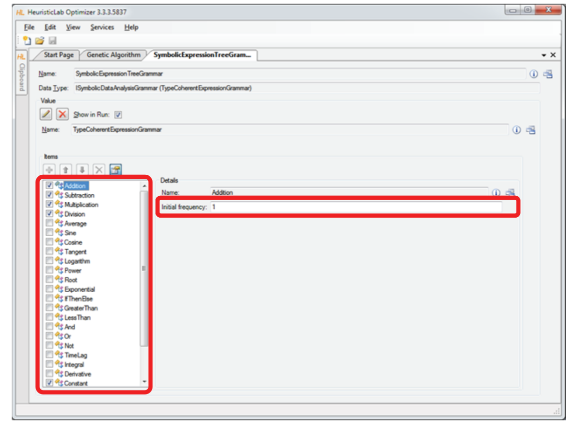
72

Configure Function Set (Grammar)



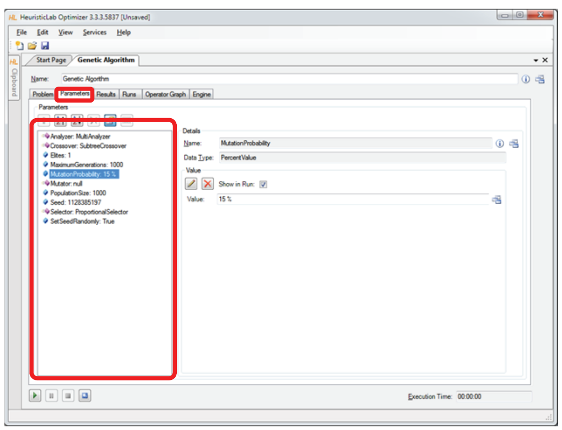
73

Configure Function Set (Grammar)



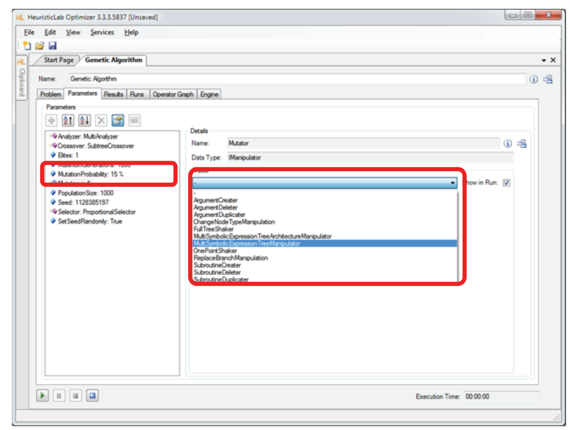
74

Configure Algorithm Parameters



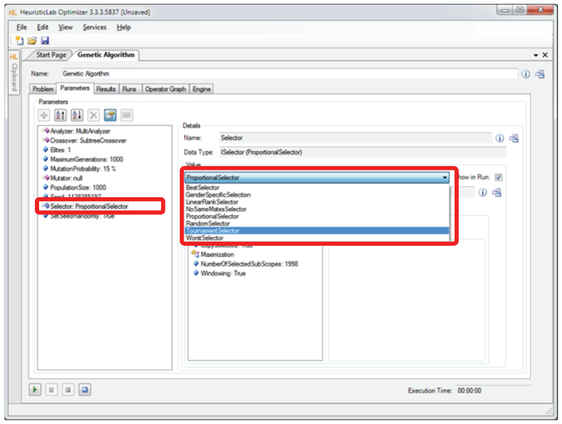
75

Configure Mutation Operator



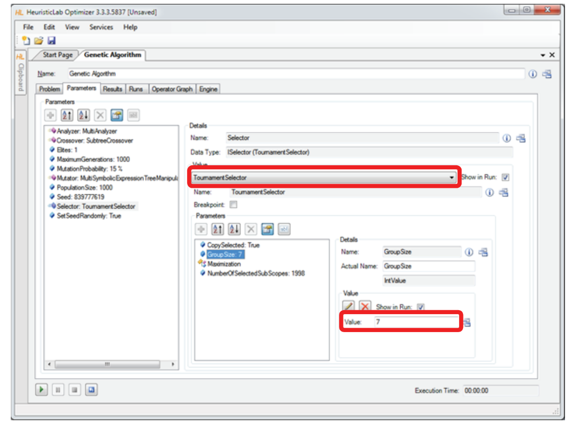
76

Configure Selection Operator



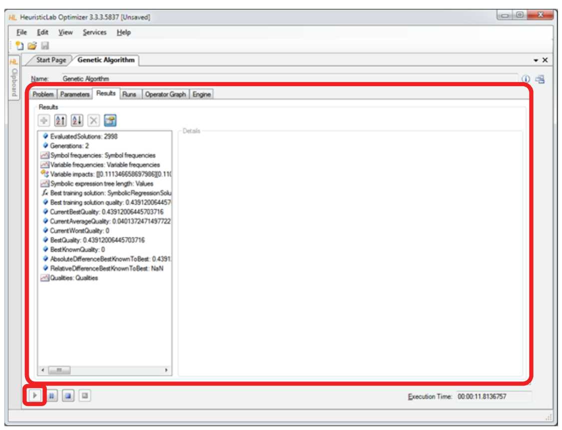
77

Configure Tournament Group Size



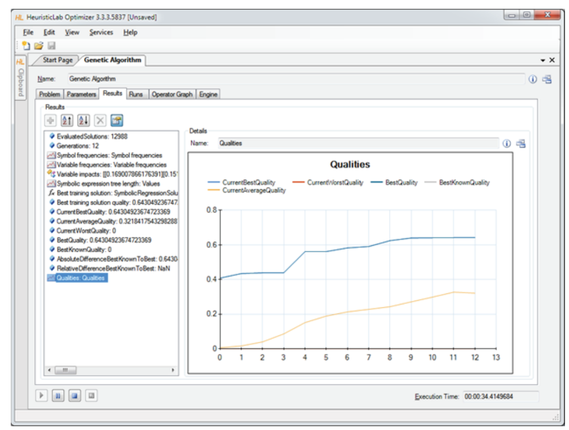
78

Start Algorithm and Inspect Results



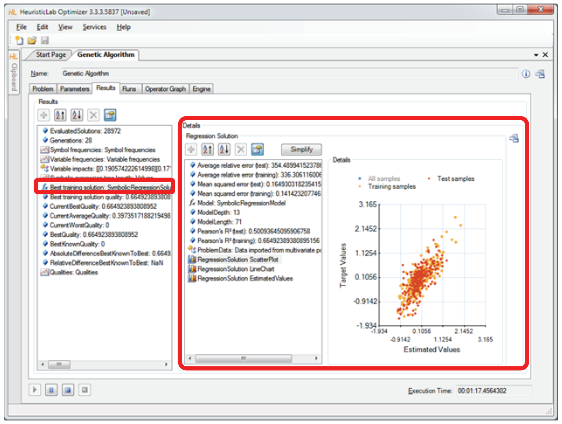
79

Inspect Quality Chart



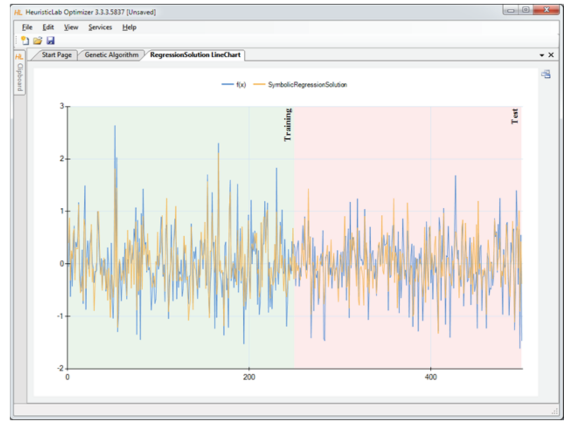
80

Inspect Best Model on Training Partition



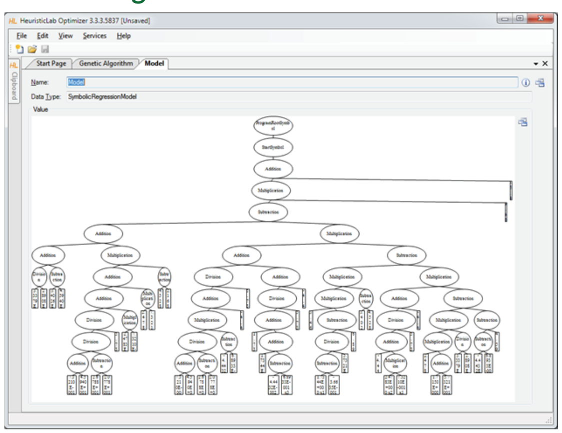
81

Inspect Linechart of Best Model on Trainingset



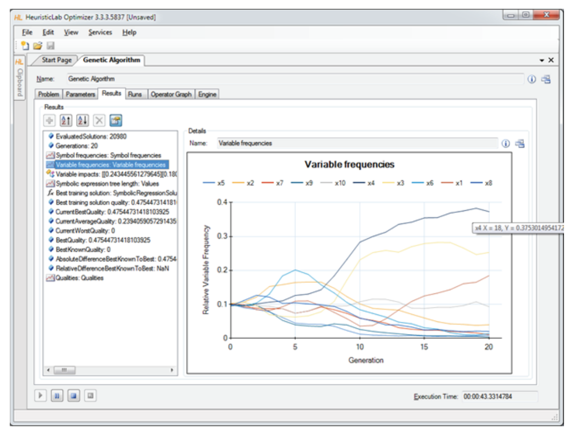
82

Inspect Structure of Best Model on Trainingset



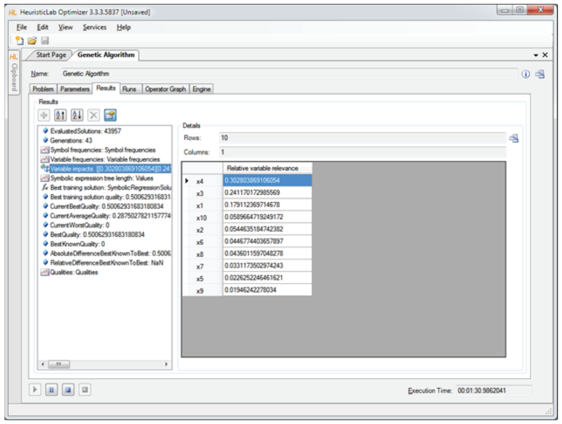
83

Inspect Variable Frequency Chart



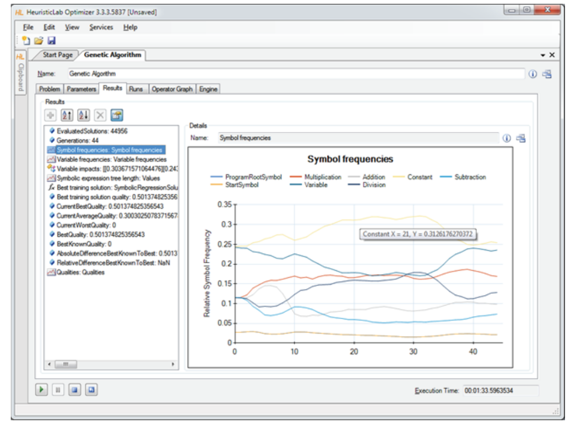
84

Inspect Variable Impacts



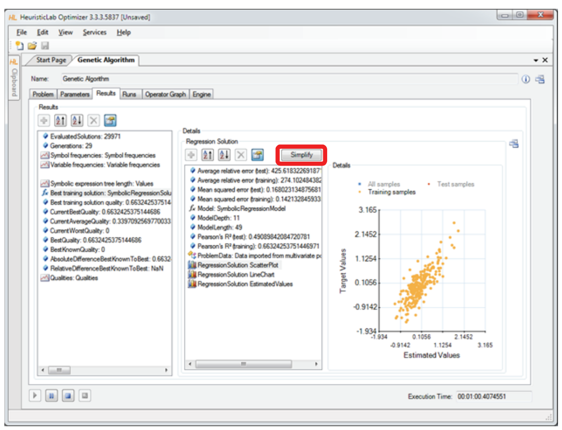
85

Inspect Symbol Frequencies



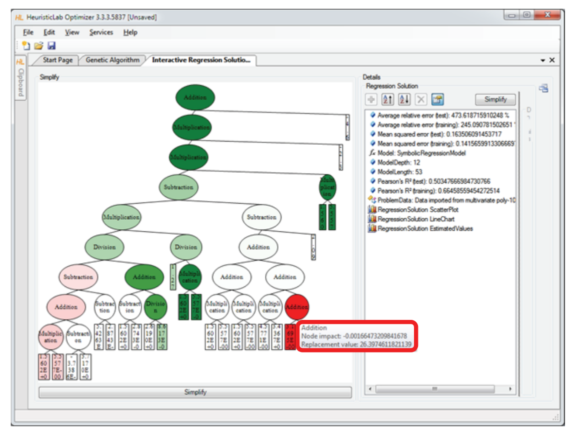
86

Detailed Model Analysis and Simplification



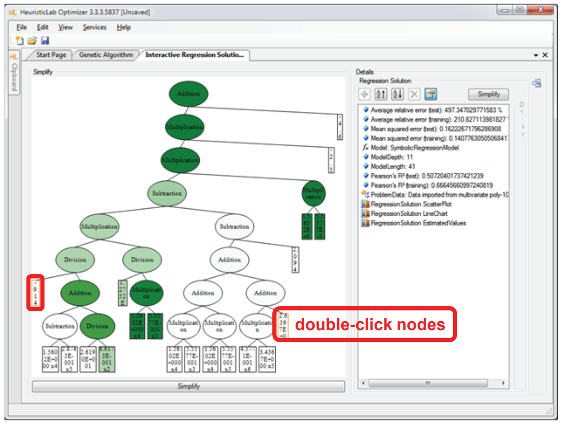
87

Symbolic Simplification and Node Impacts



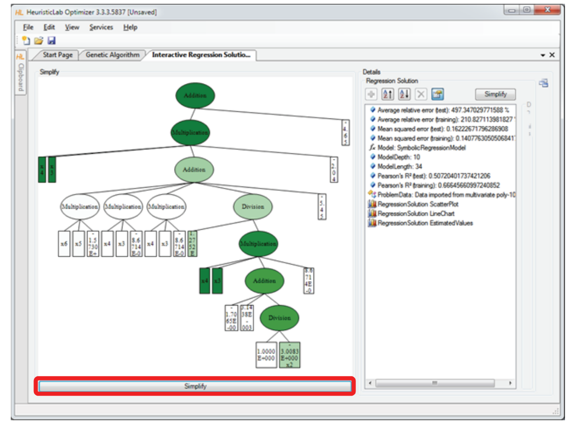
88

Manual Simplification



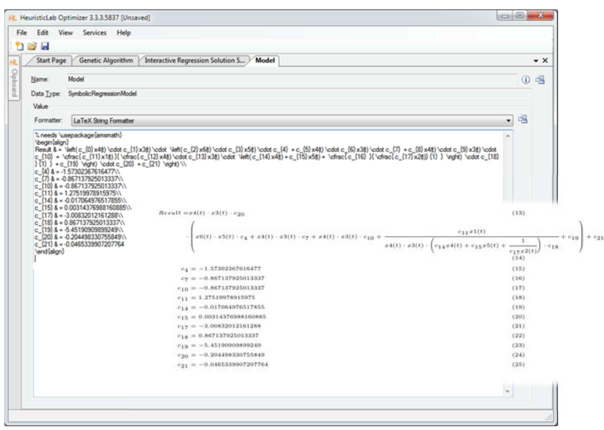
89

Automatic Symbolic Simplification



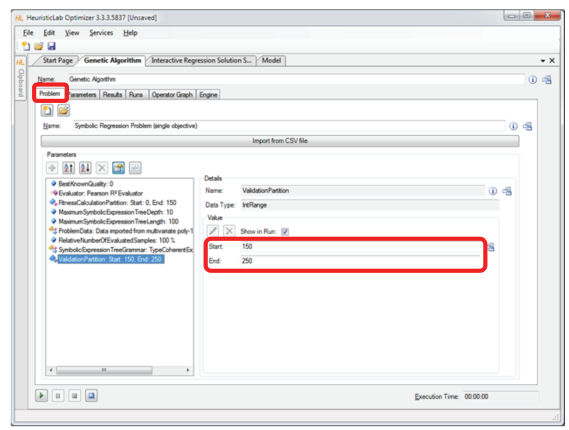
90

LaTeX Export



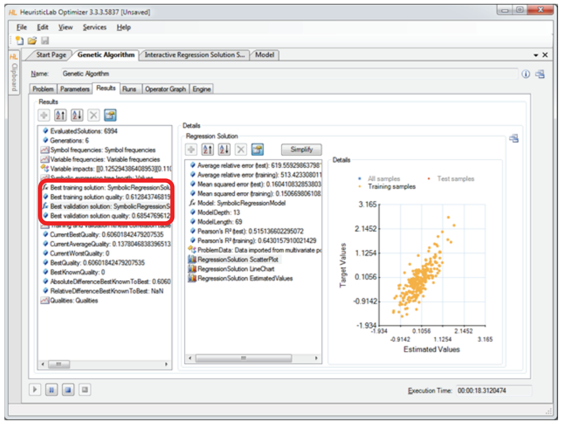
91

Configuration of Validation Partition



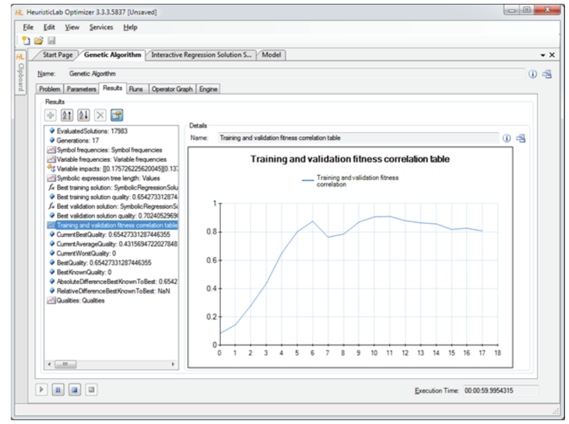
92

Inspect Best Model on Validation Partition



93

Inspect Linechart of Correltion of Training and Validation Fitness



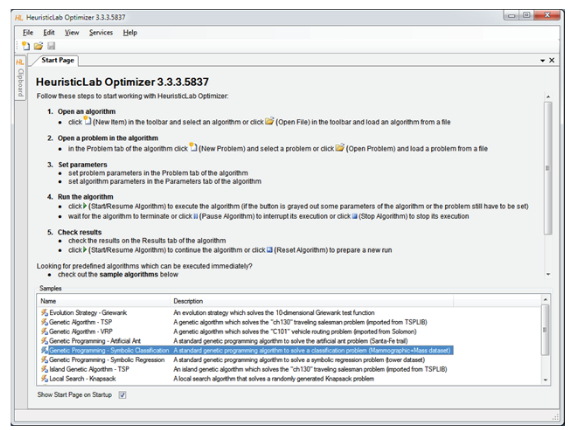
94

Case study: Classification

- Real world medical dataset (*Mammographic Mass*) from UCI Machine Learning Repository (Frank & Asuncion)
  - 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#GECCO2011

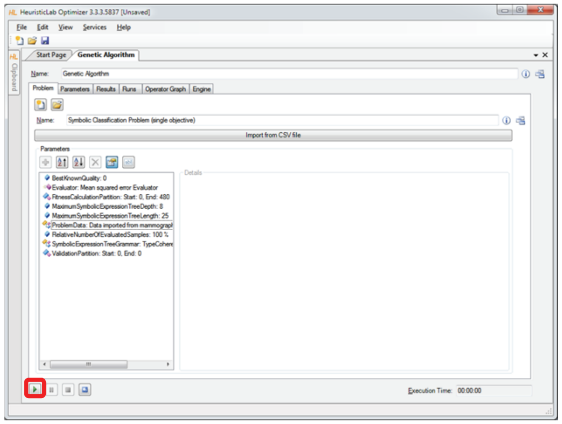
95

Open Sample



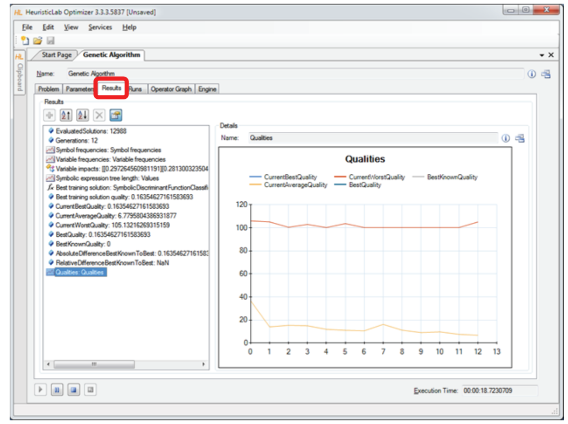
96

Configure and Run Algorithm



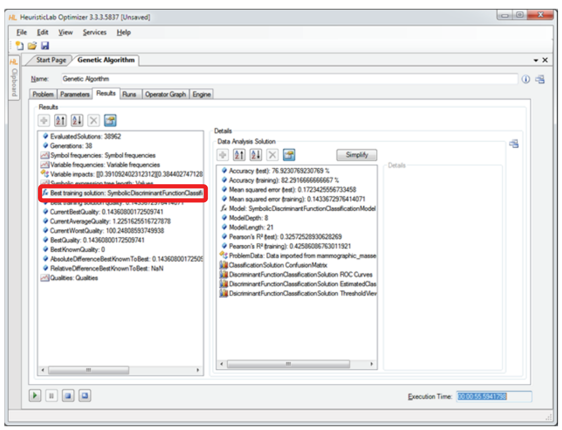
97

Inspect Quality Linechart



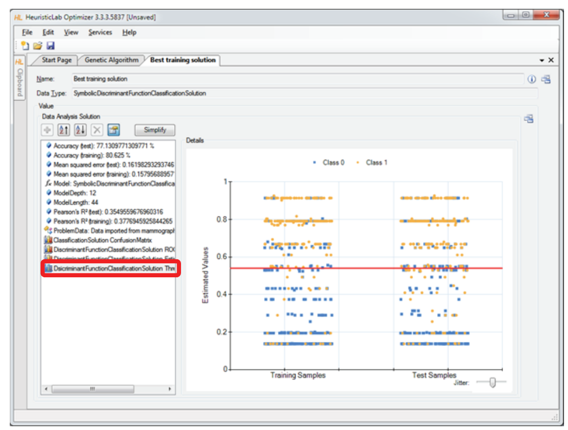
98

Inspect Best Training Solution



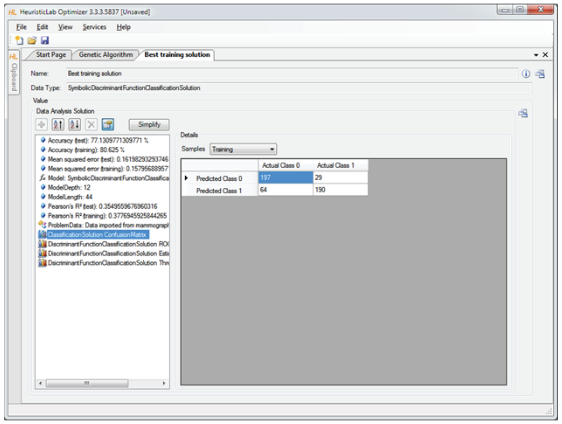
99

Inspect Model Output and Thresholds



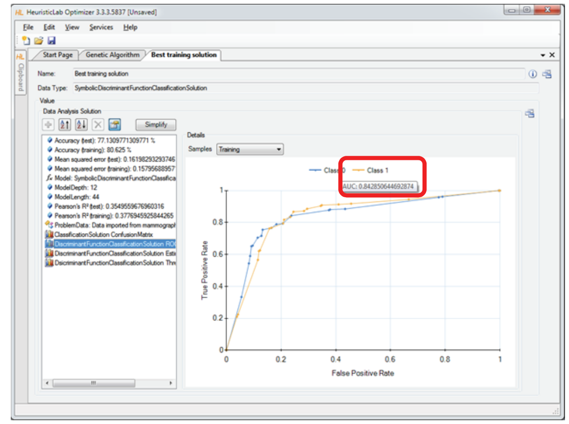
100

Inspect Confusion Matrix



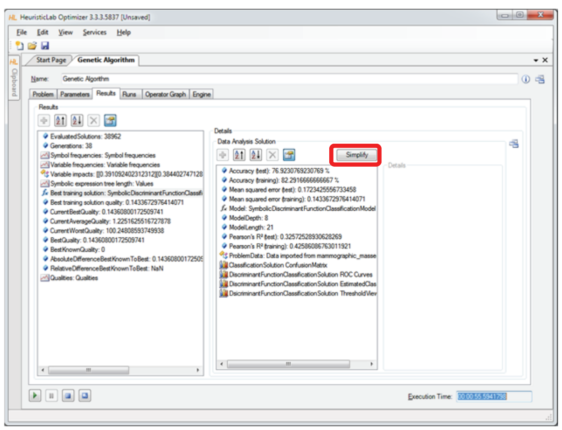
101

Inspect ROC curve



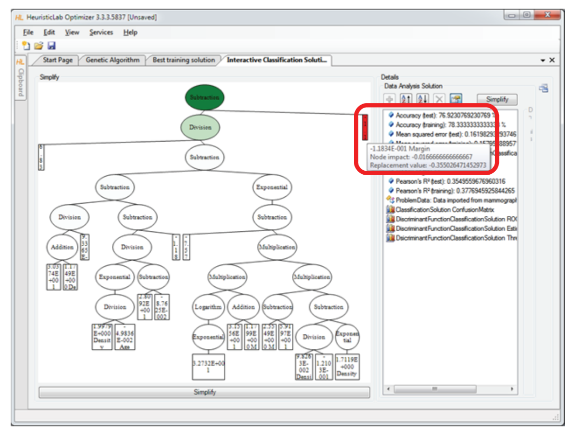
102

Analyse and Simplify Best Training Solution



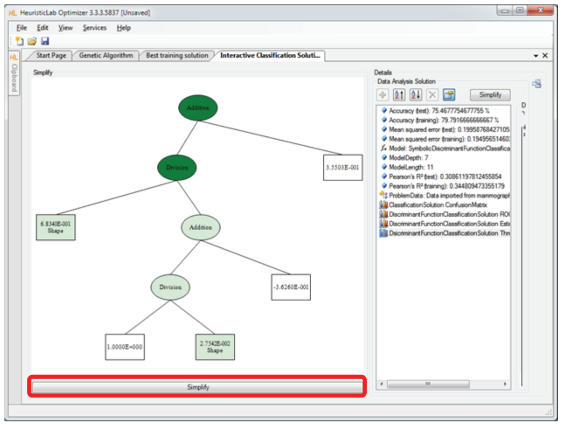
103

Analyse and Simplify Model



104

Symbolically Simplified Model



105

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

106

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

107

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

http://heal.heuristiclab.com

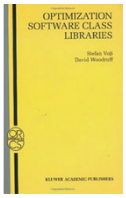
HEAL Heuristic and Evolutionary Algorithms Laboratory

FH OBERÖSTERREICH

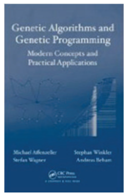
108

Suggested Readings

- S. Voß, D. Woodruff (Edts.)
  Optimization Software Class Libraries
  Kluwer Academic Publishers, 2002



- 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
  Computer Aided Systems Theory - EUROCAST 2007, Lecture Notes in Computer Science, vol. 4739, pp. 747-754
  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
  Computer Aided Systems Theory - EUROCAST 2009, Lecture Notes in Computer Science, vol. 5717, pp. 729-736
  Springer, 2009

- S. Wagner
  Heuristic optimization software systems - Modeling of heuristic optimization algorithms in the HeuristicLab software environment
  Ph.D. thesis, Johannes Kepler University Linz, Austria, 2009.

- 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

Questions & Answers

http://dev.heuristiclab.com

support@heuristiclab.com