



HeuristicLab

A Paradigm-Independent and Extensible
Environment for Heuristic Optimization

Algorithm and Experiment Design with HeuristicLab

An Open Source Optimization Environment for
Research and Education

S. Wagner, G. Kronberger

Heuristic and Evolutionary Algorithms Laboratory (HEAL)

School of Informatics/Communications/Media, Campus Hagenberg

University of Applied Sciences Upper Austria



HEAL

Heuristic and Evolutionary
Algorithms Laboratory



Josef Ressel-Zentrum
HEUREKA!

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 (2005 – 2011)
Upper Austria University of Applied Sciences
- Research assistant (2005 – 2011)
Upper Austria University of Applied Sciences
- Full professor for business intelligence (since 2011)
Upper Austria University of Applied Sciences
- Member of the HEAL research group
- Architect of HeuristicLab
- <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

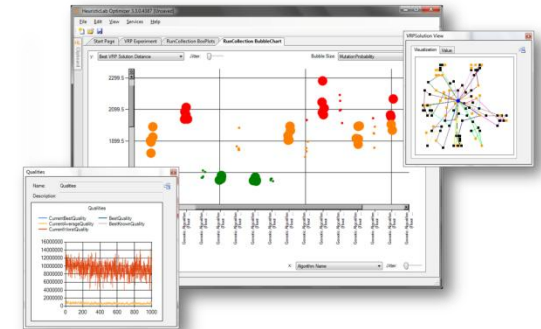
Objectives of the Tutorial



- 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 3.3.5 released on July 9th, 2011



Where to get HeuristicLab?

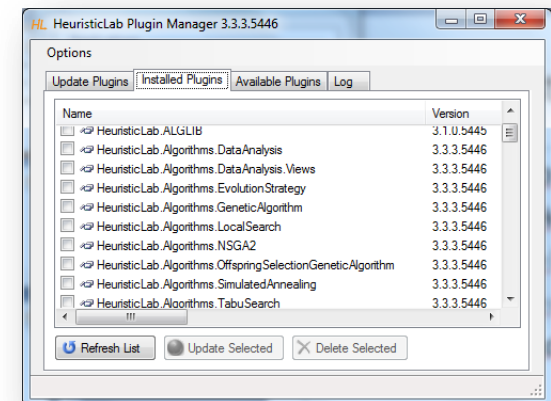
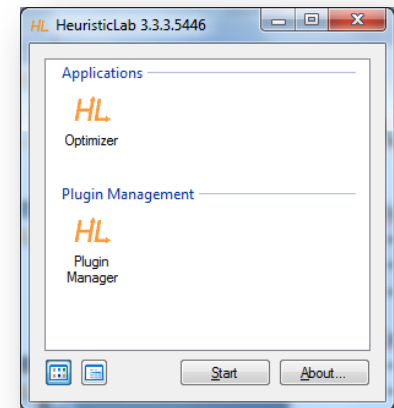


- Download binaries
 - deployed as ZIP archives
 - latest stable version 3.3.5
 - released on July 9th, 2011
 - daily trunk build
 - <http://dev.heuristiclab.com/download>
- Check out sources
 - SVN repository
 - HeuristicLab 3.3.5 tag
 - <http://dev.heuristiclab.com/svn/hl/core/tags/3.3.5>
 - current 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

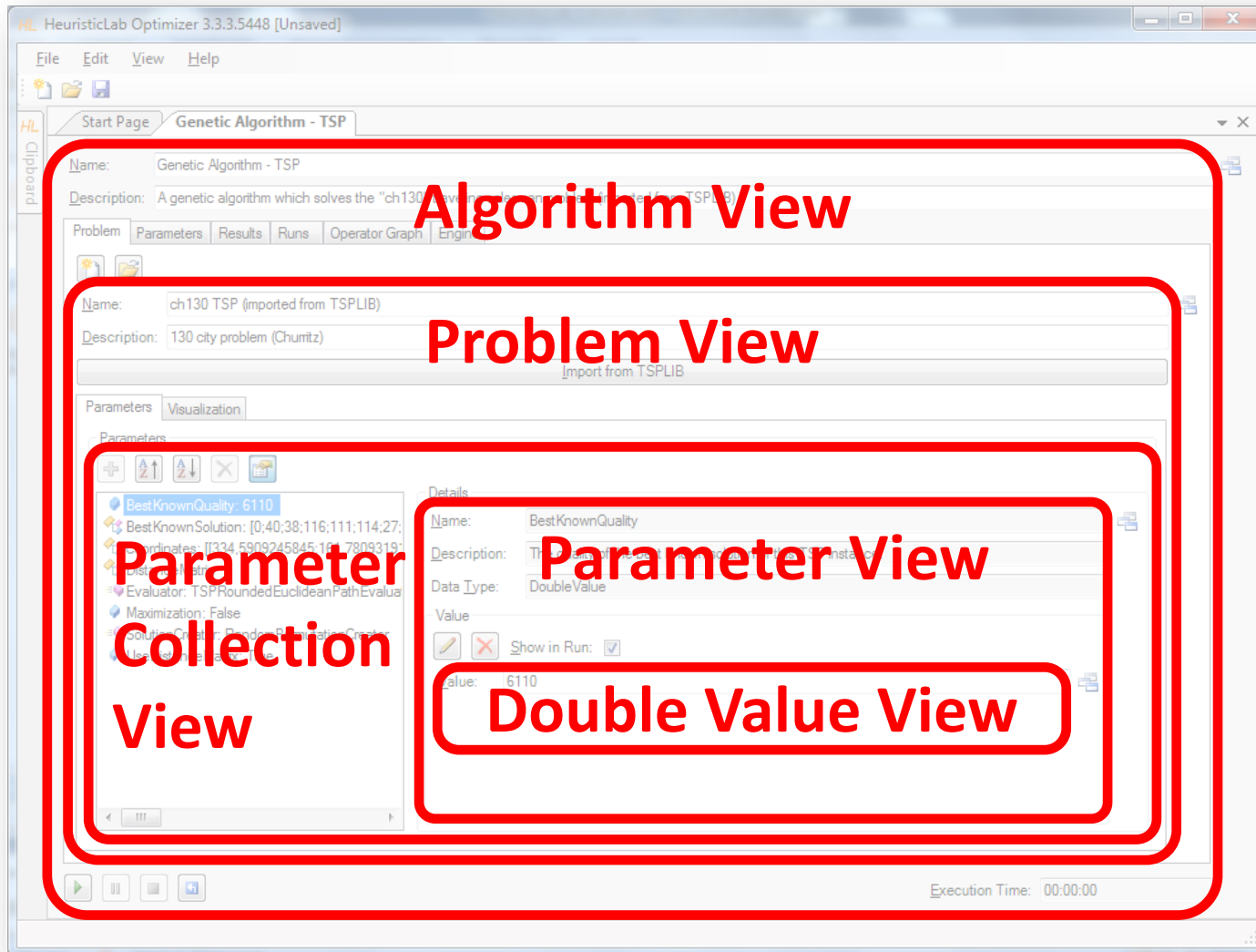


Graphical User Interface



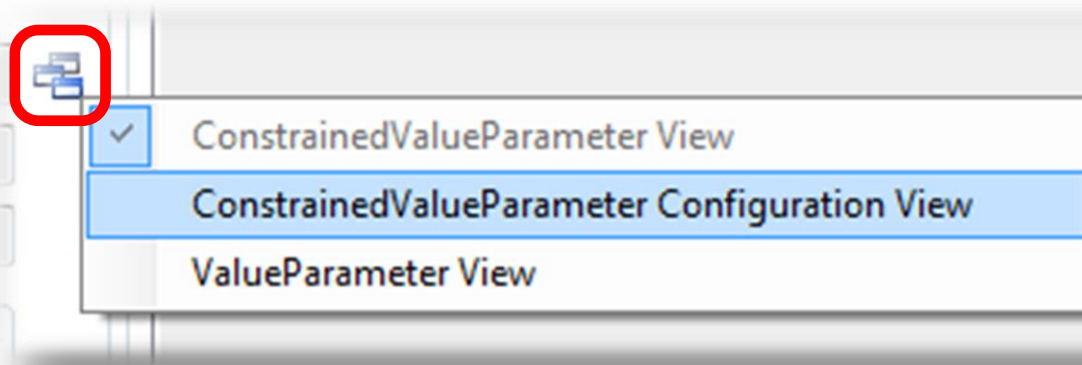
- HeuristicLab GUI is made up of views
 - views are visual representations of content objects
 - views are composed in the same way as their content
 - views and content objects are loosely coupled
 - multiple different views may exist for the same content
- Drag & Drop
 - views support drag & drop operations
 - content objects can be copied or moved (shift key)
 - enabled for collection items and content objects

Graphical User Interface



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

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**

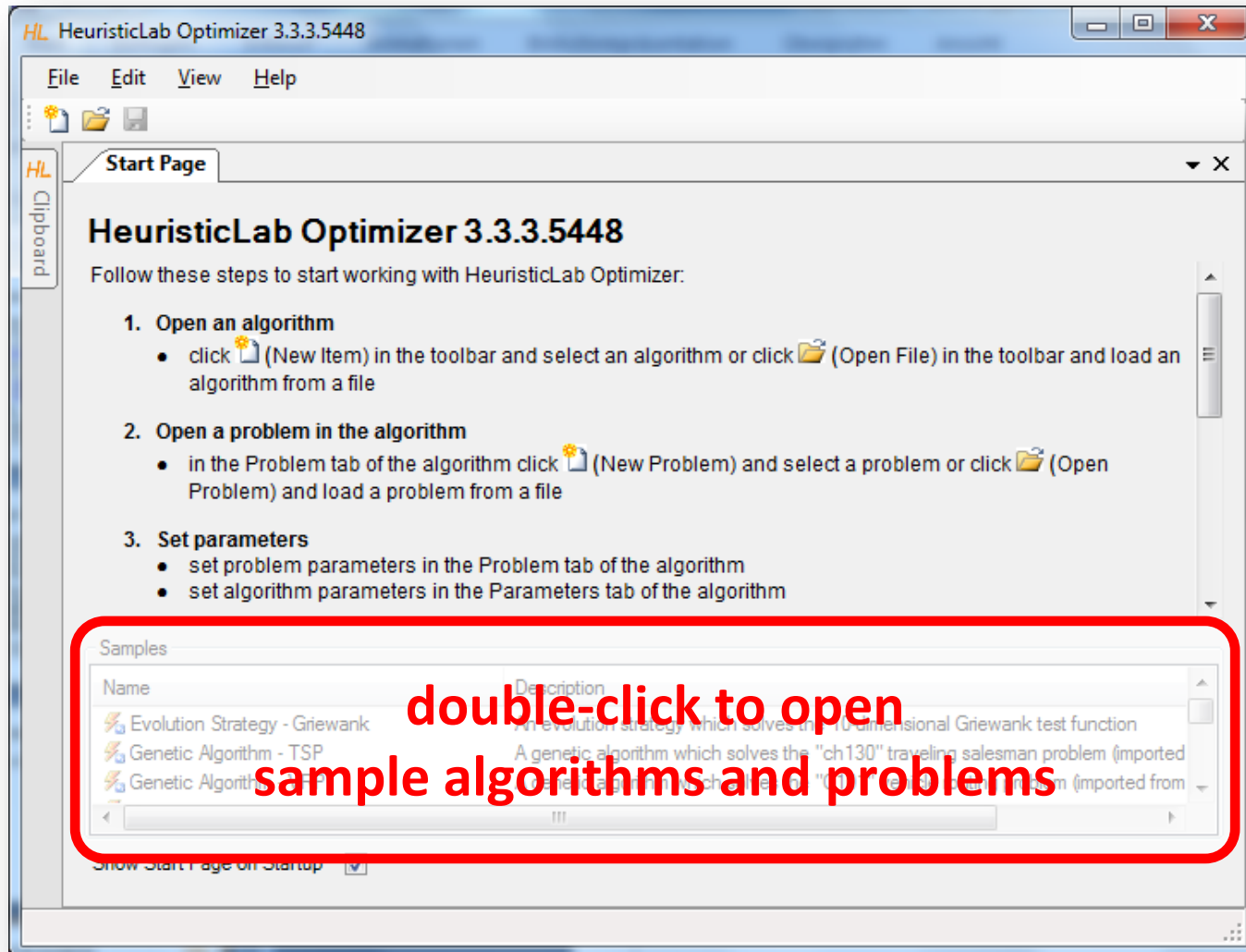
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

Demonstration Part I: Working with HeuristicLab

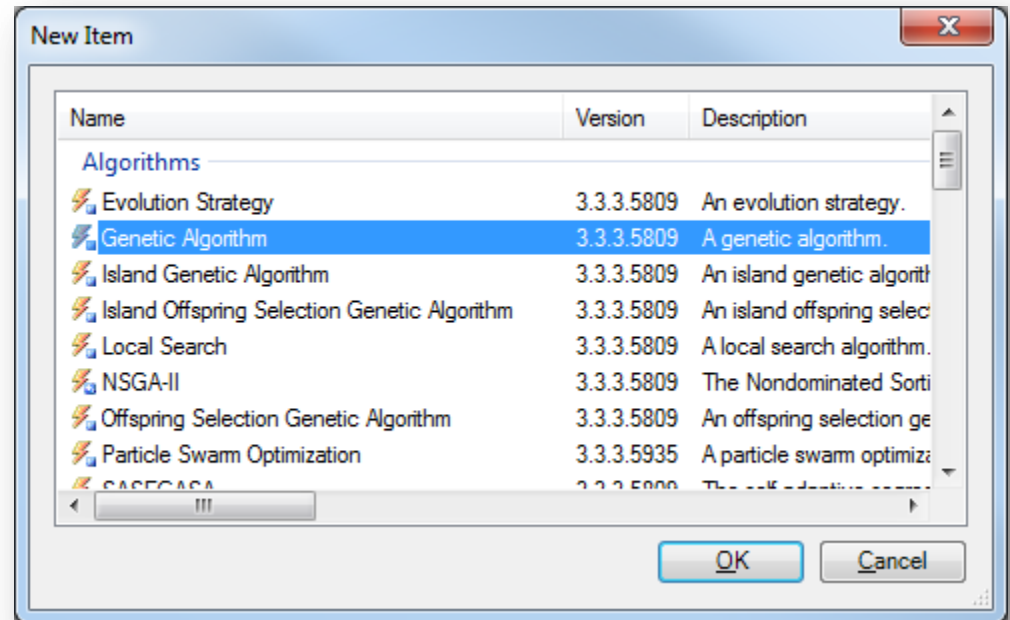
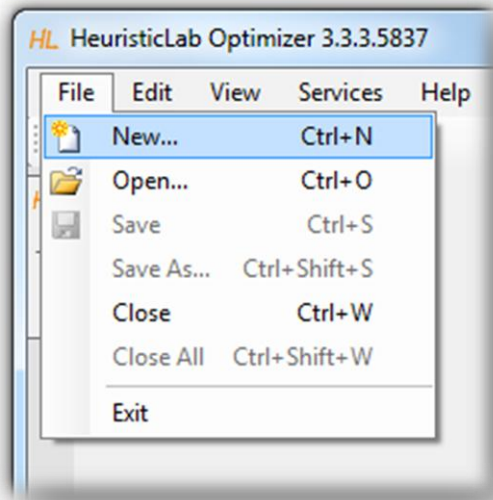


- Create, Parameterize and Execute Algorithms
- Save and Load Items
- Create Batch Runs and Experiments
- Multi-core CPUs and Parallelization
- Analyze Runs
- Analyzers
- Building User-Defined Algorithms

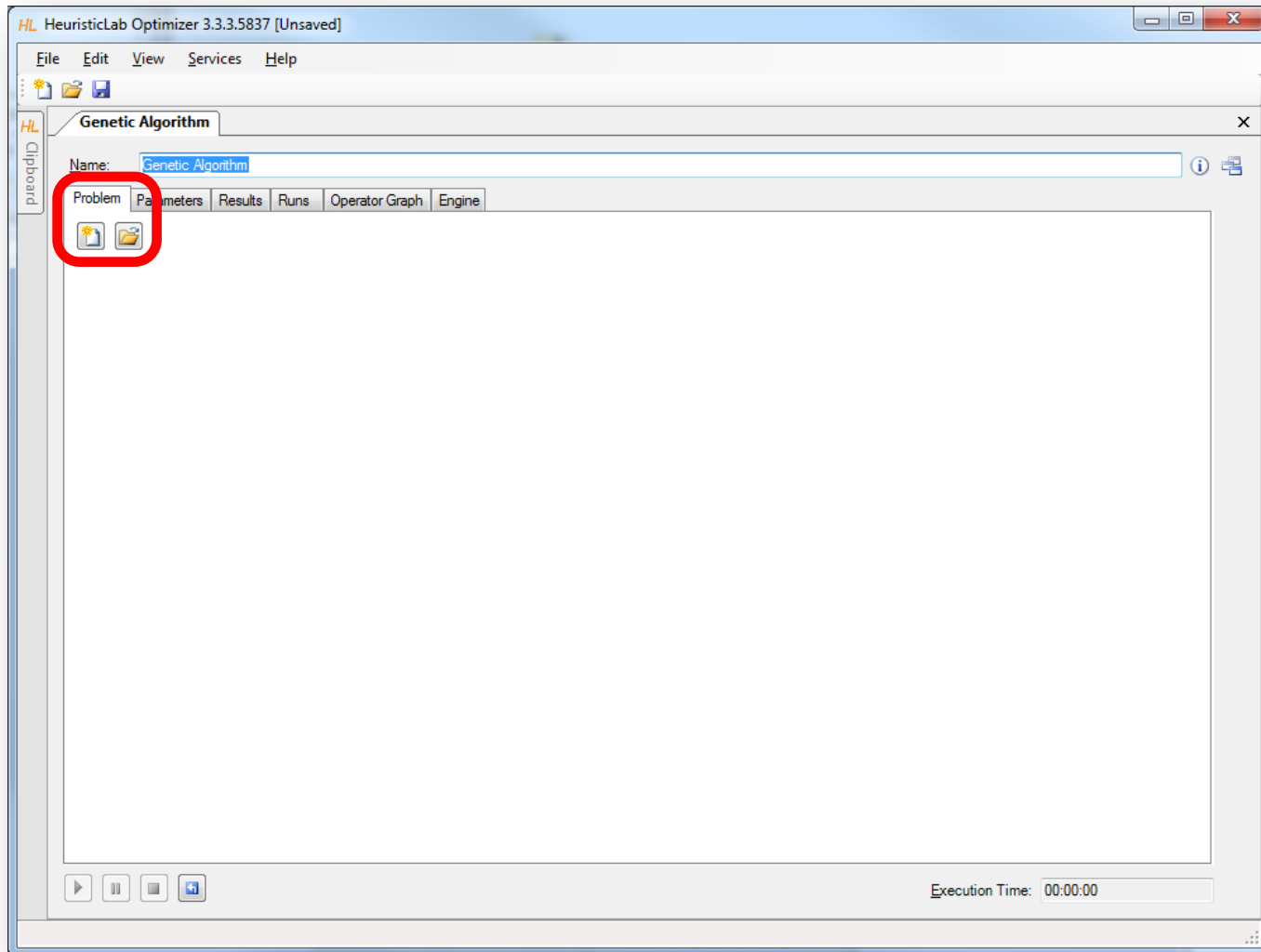
HeuristicLab Optimizer



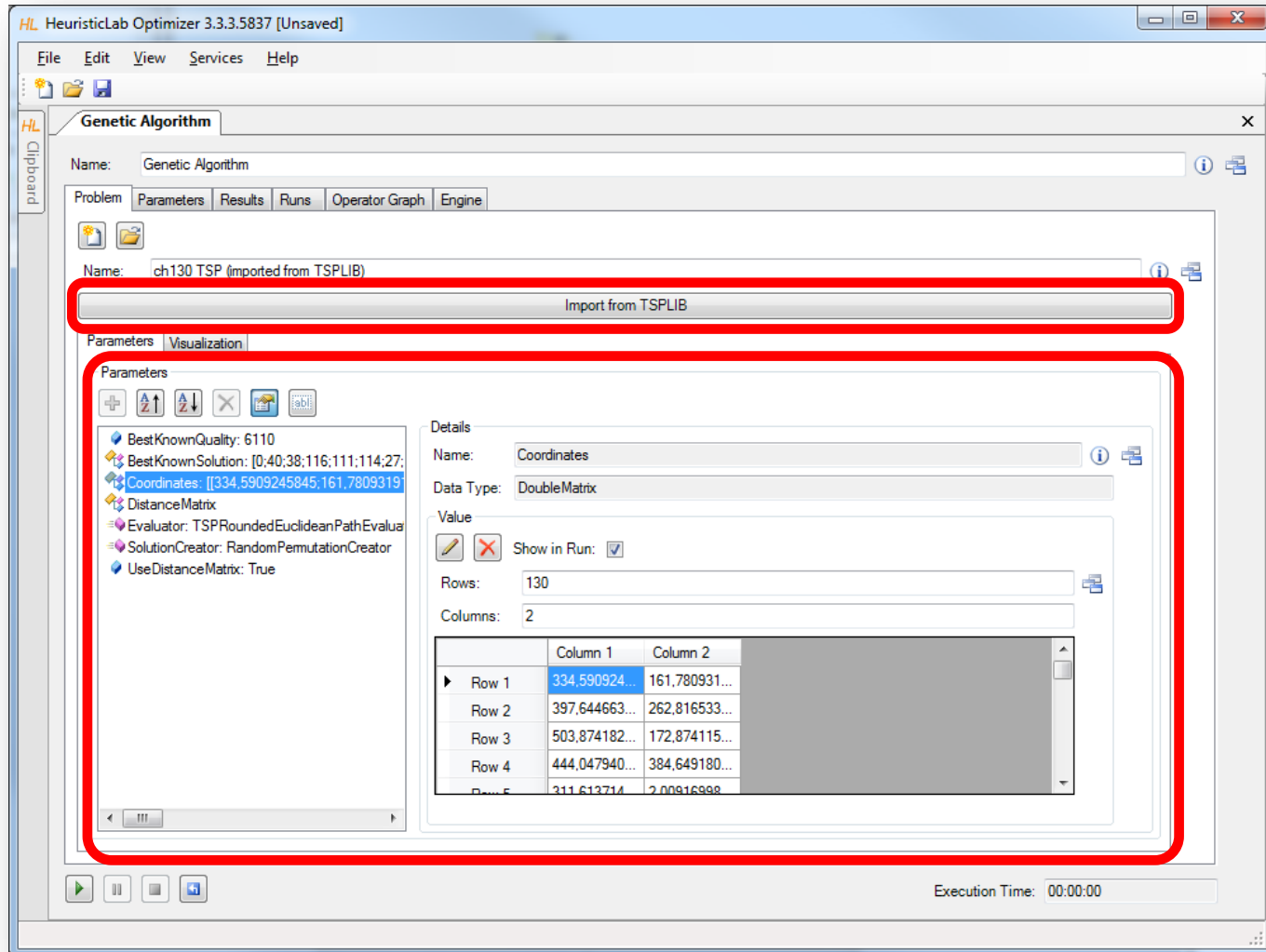
Create Algorithm



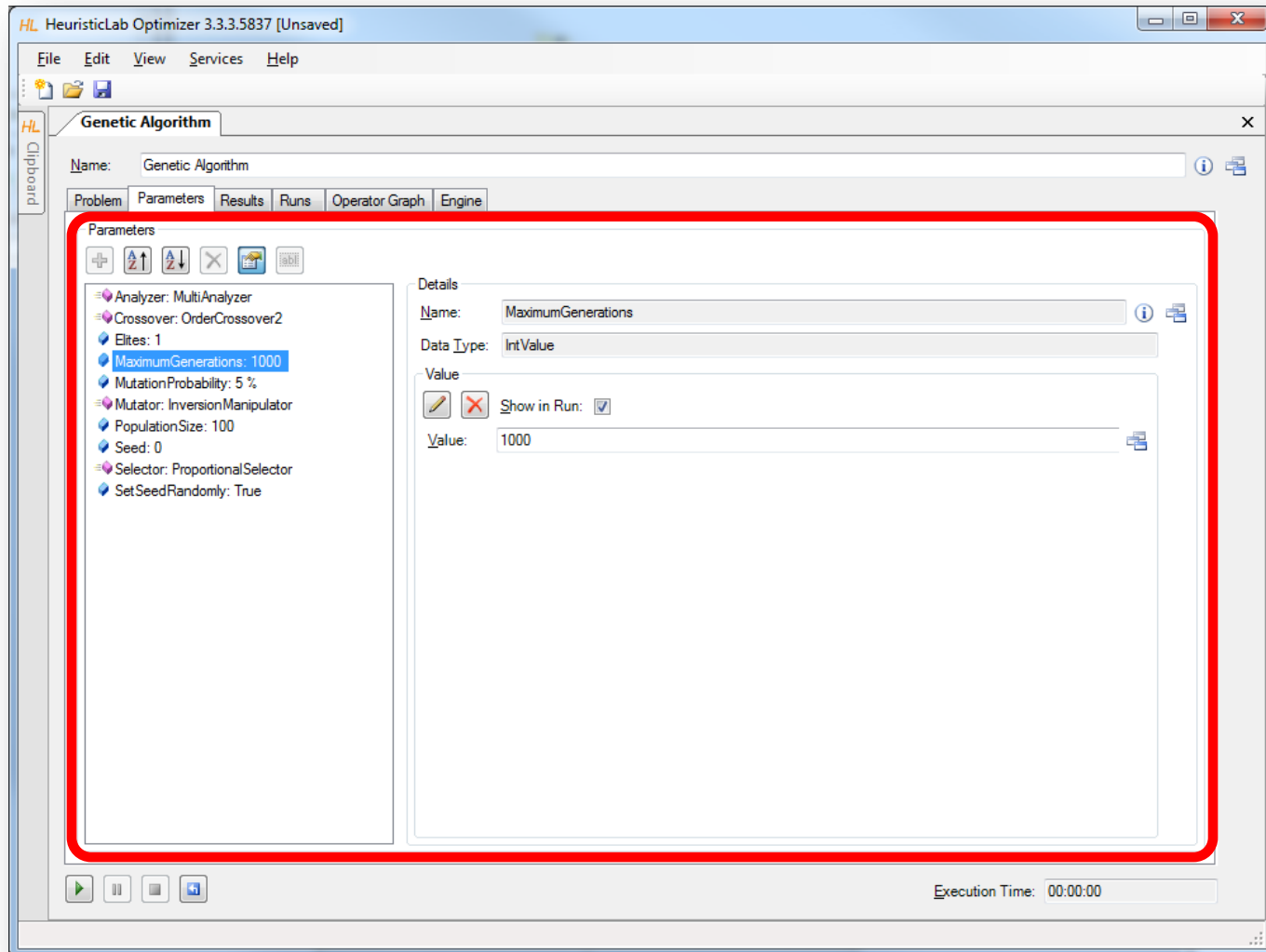
Create or Load Problem



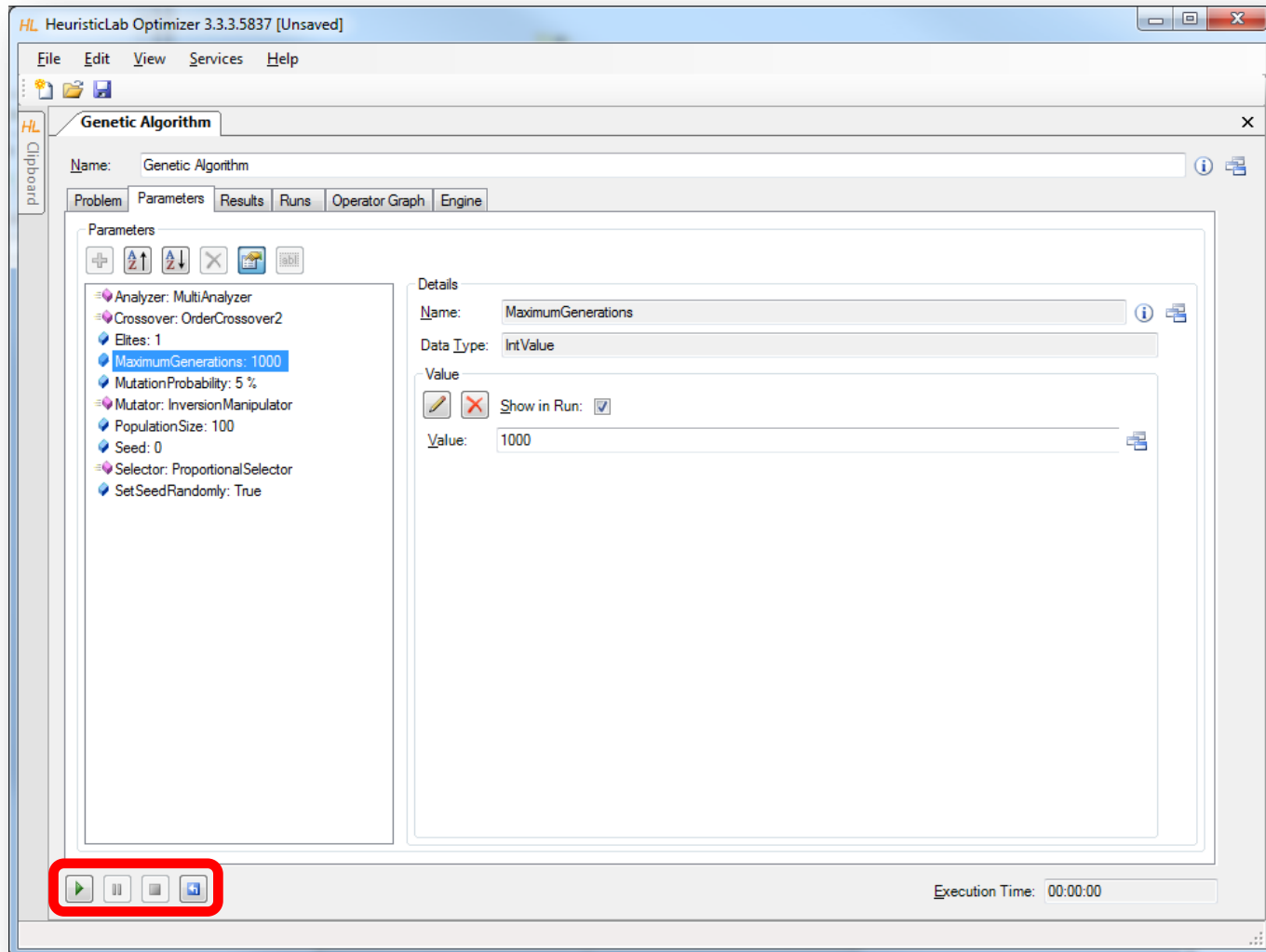
Import or Parameterize Problem Data



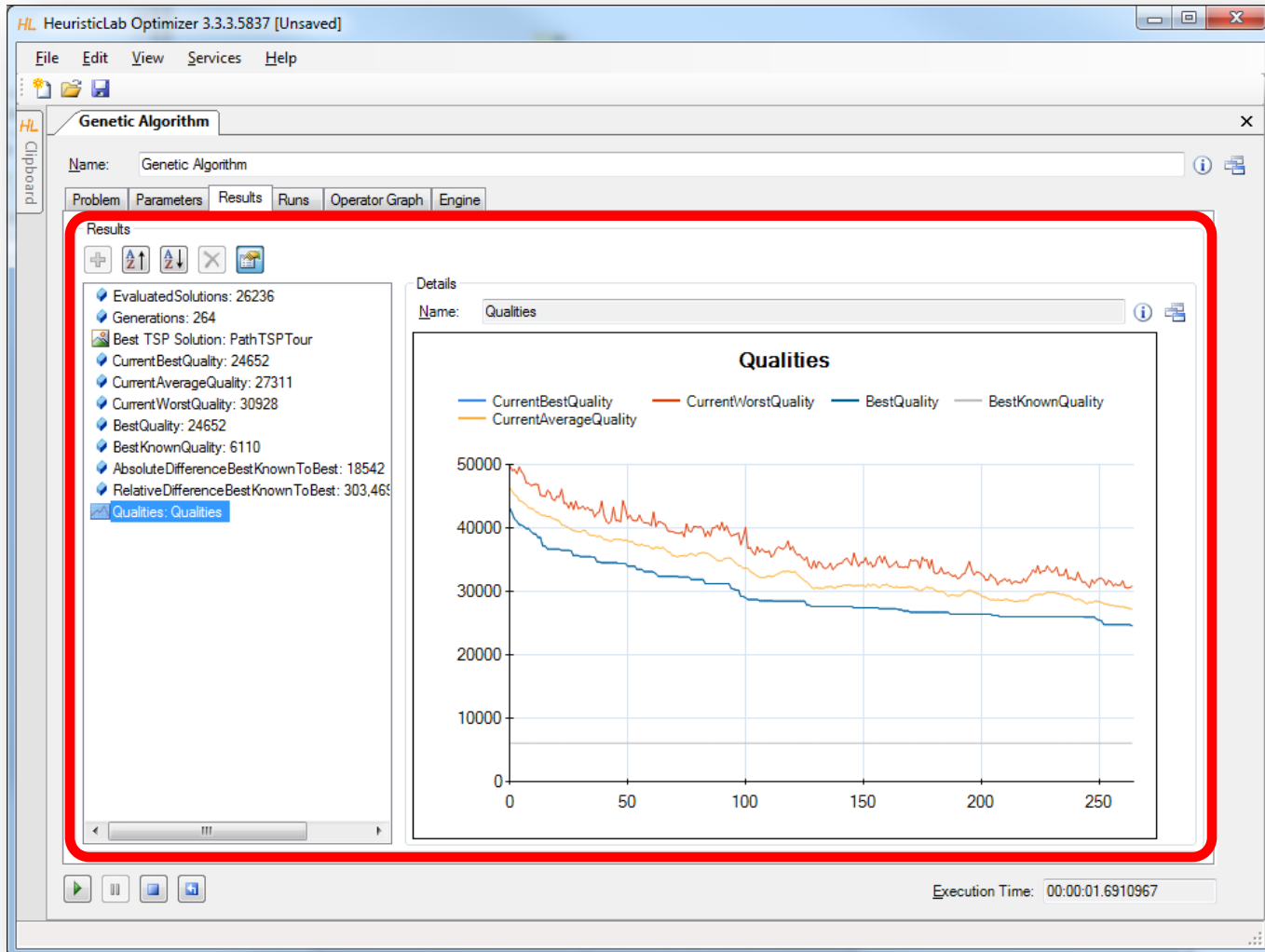
Parameterize Algorithm



Start, Pause, Resume, Stop and Reset

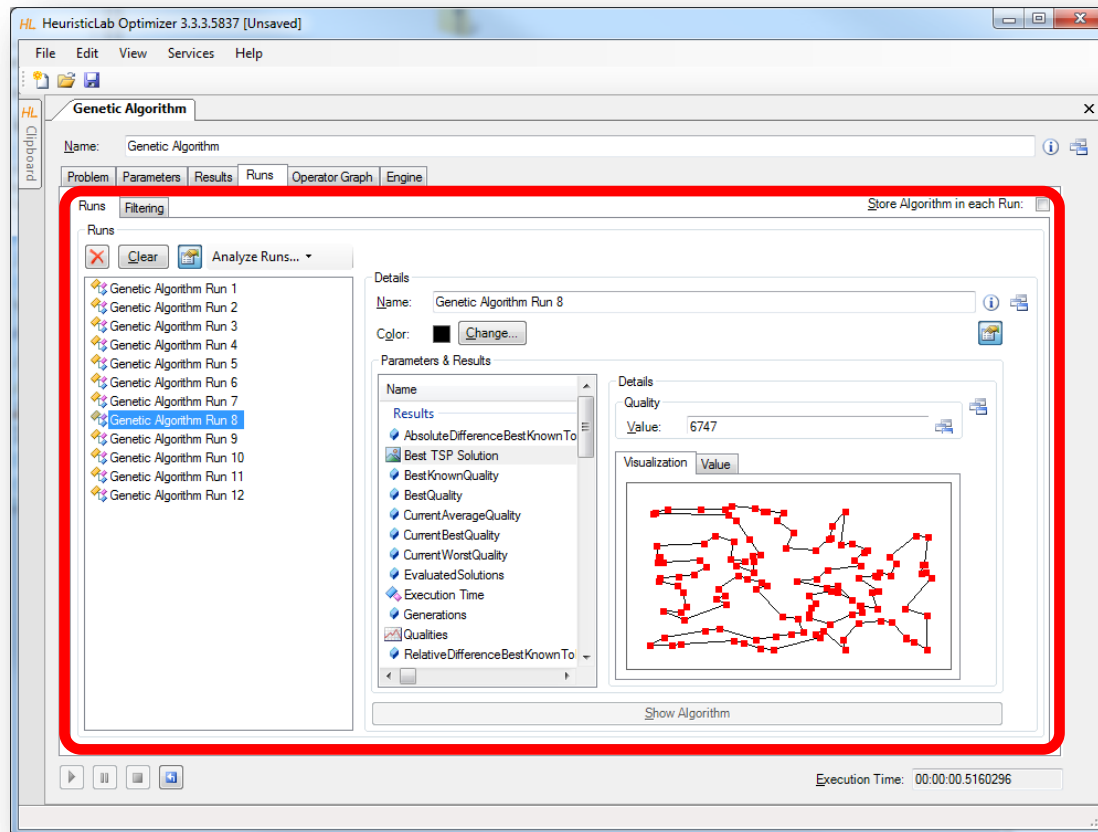


Inspect Results



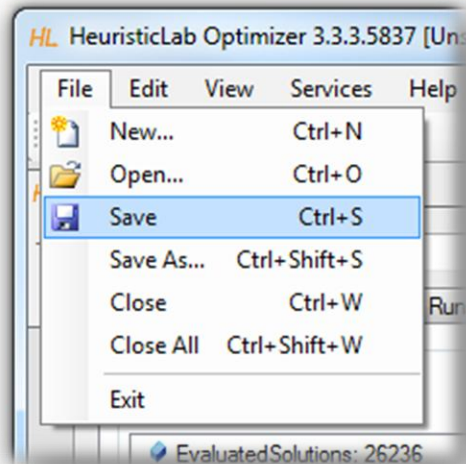
Compare Runs

- 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



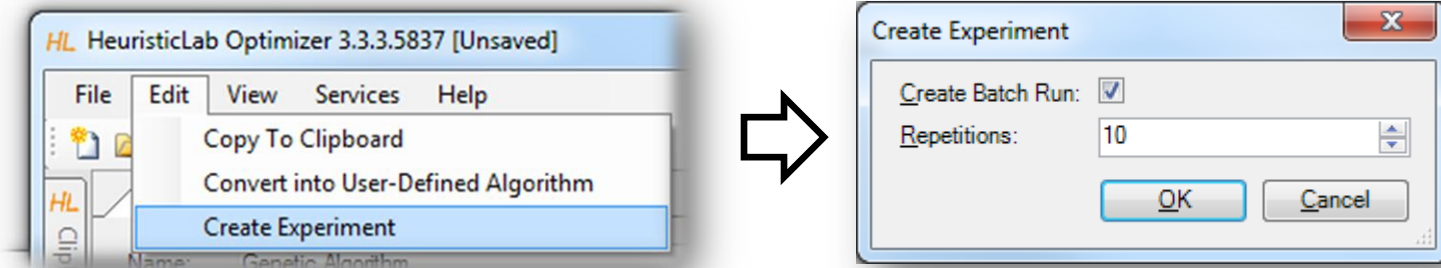
Save and Load

- 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

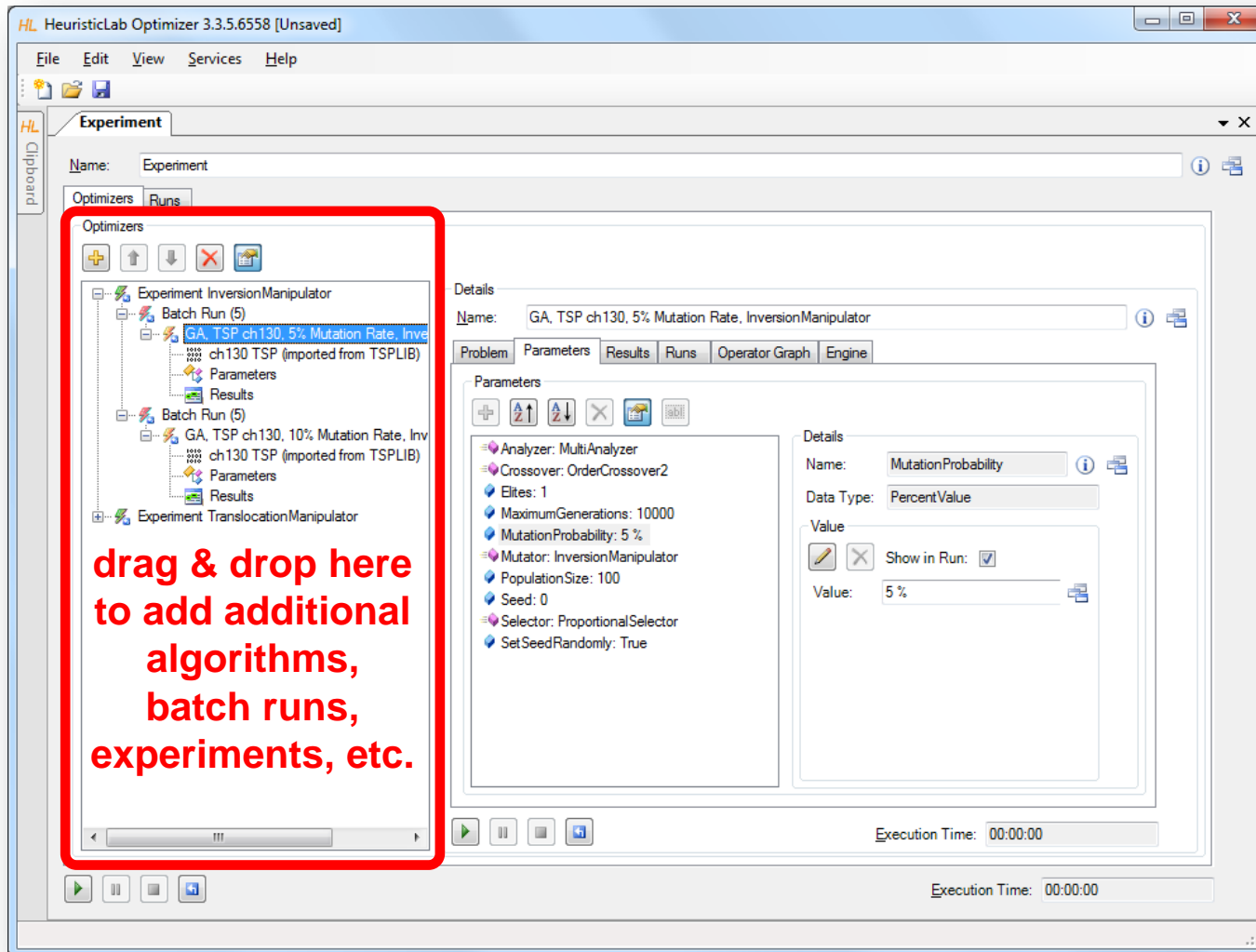


Create Batch Runs and Experiments

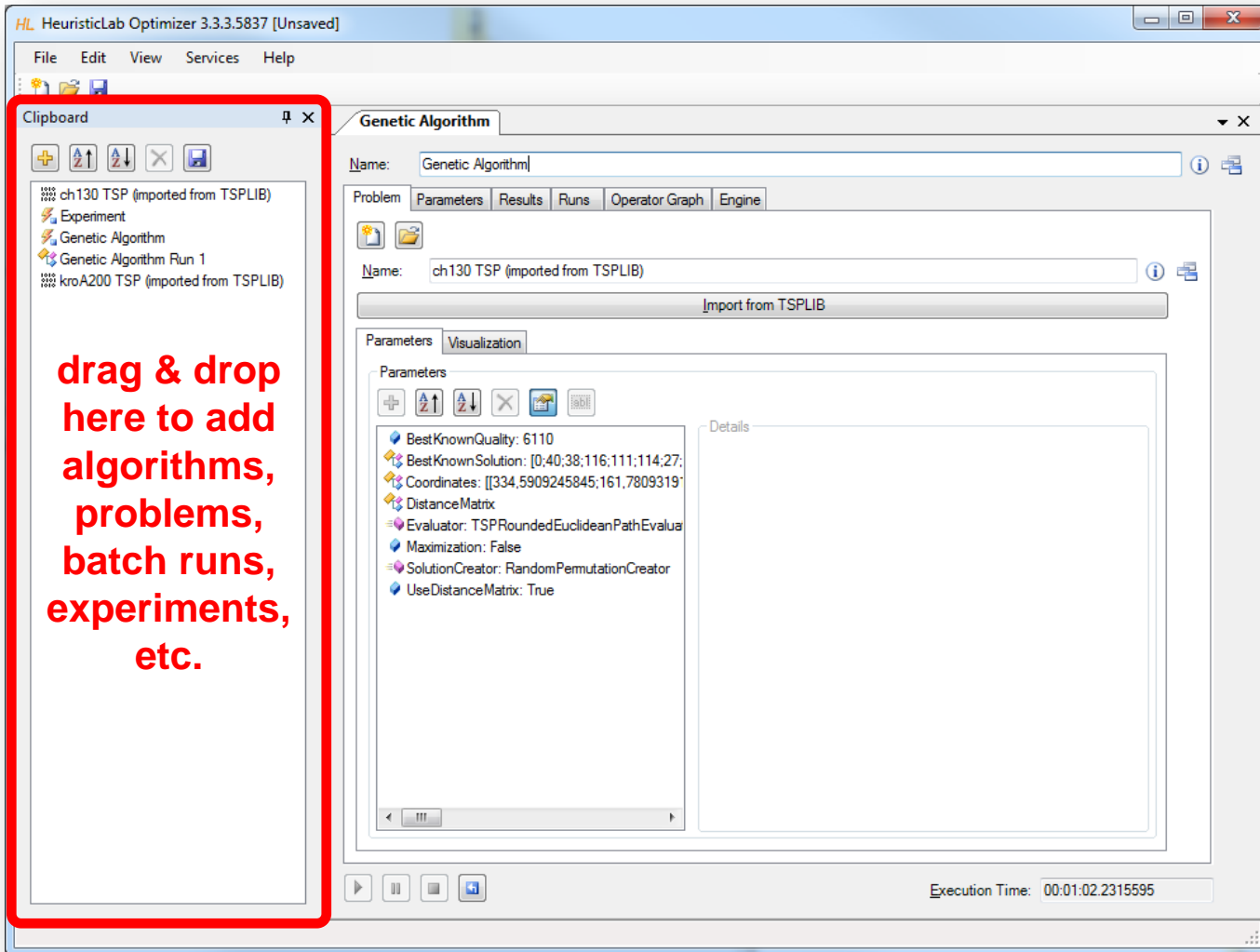
- Batch runs
 - execute the same optimizer (e.g. algorithm, batch run, experiment) several times
- Experiments
 - execute different optimizers
 - suitable for large scale algorithm comparison and analysis
- Experiments and batch runs can be nested
- Generated runs can be compared afterwards



Create Batch Runs and Experiments



Clipboard

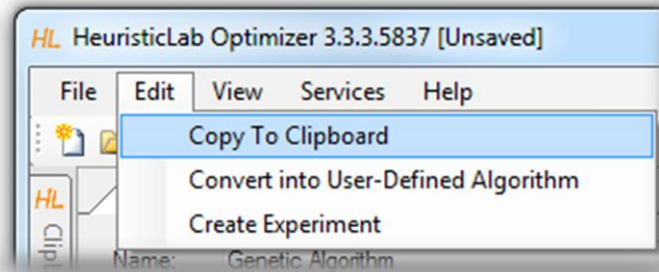


The screenshot shows the HeuristicLab Optimizer interface. A red box highlights the Clipboard window on the left, which contains a list of items: ch130 TSP (imported from TSPLIB), Experiment, Genetic Algorithm, Genetic Algorithm Run 1, and kroA200 TSP (imported from TSPLIB). The main window displays the configuration for a Genetic Algorithm, including the problem name 'ch130 TSP (imported from TSPLIB)' and various parameters such as BestKnownQuality, BestKnownSolution, Coordinates, DistanceMatrix, Evaluator, Maximization, SolutionCreator, and UseDistanceMatrix.

**drag & drop
here to add
algorithms,
problems,
batch runs,
experiments,
etc.**

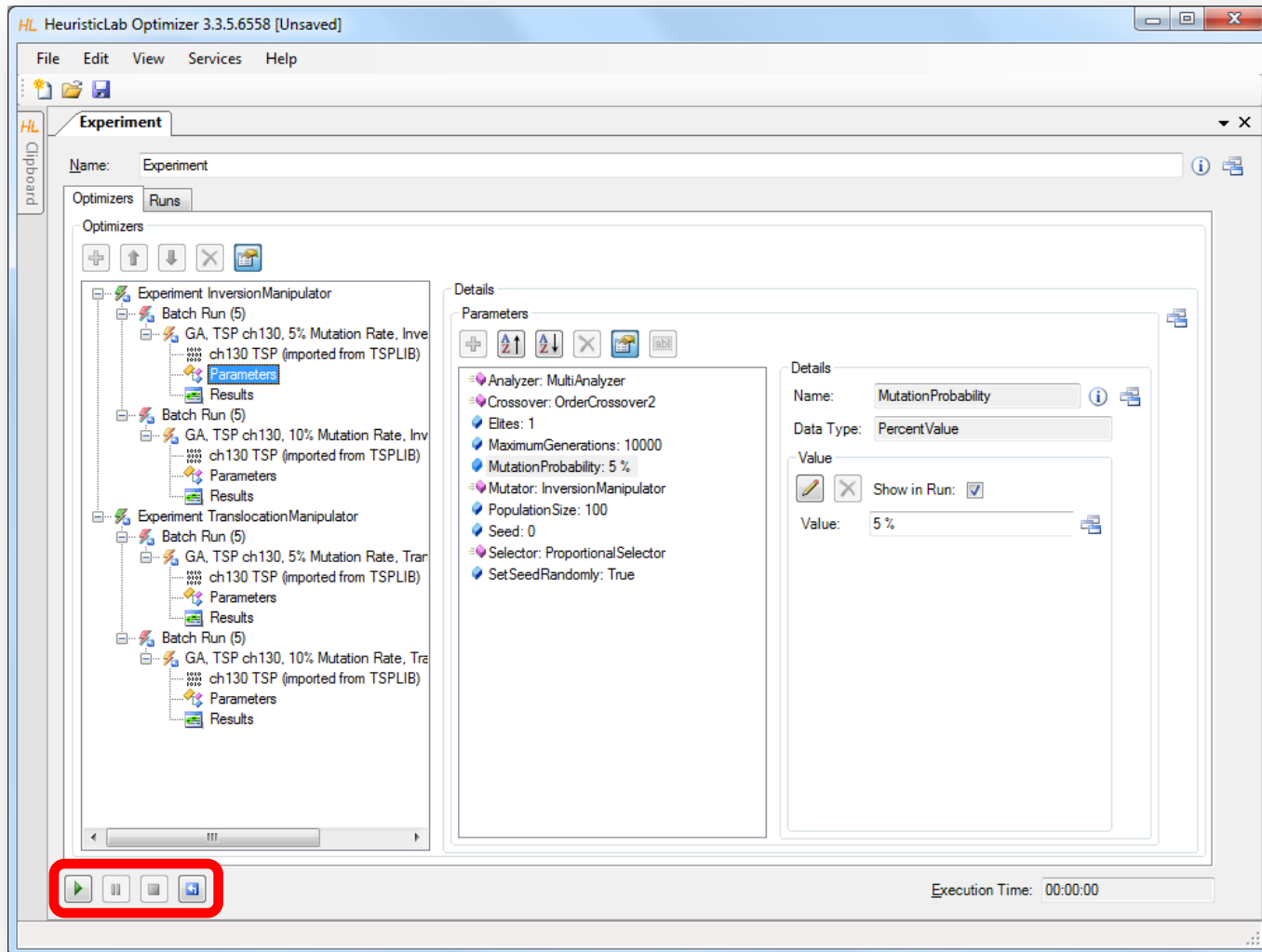
Clipboard

- 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

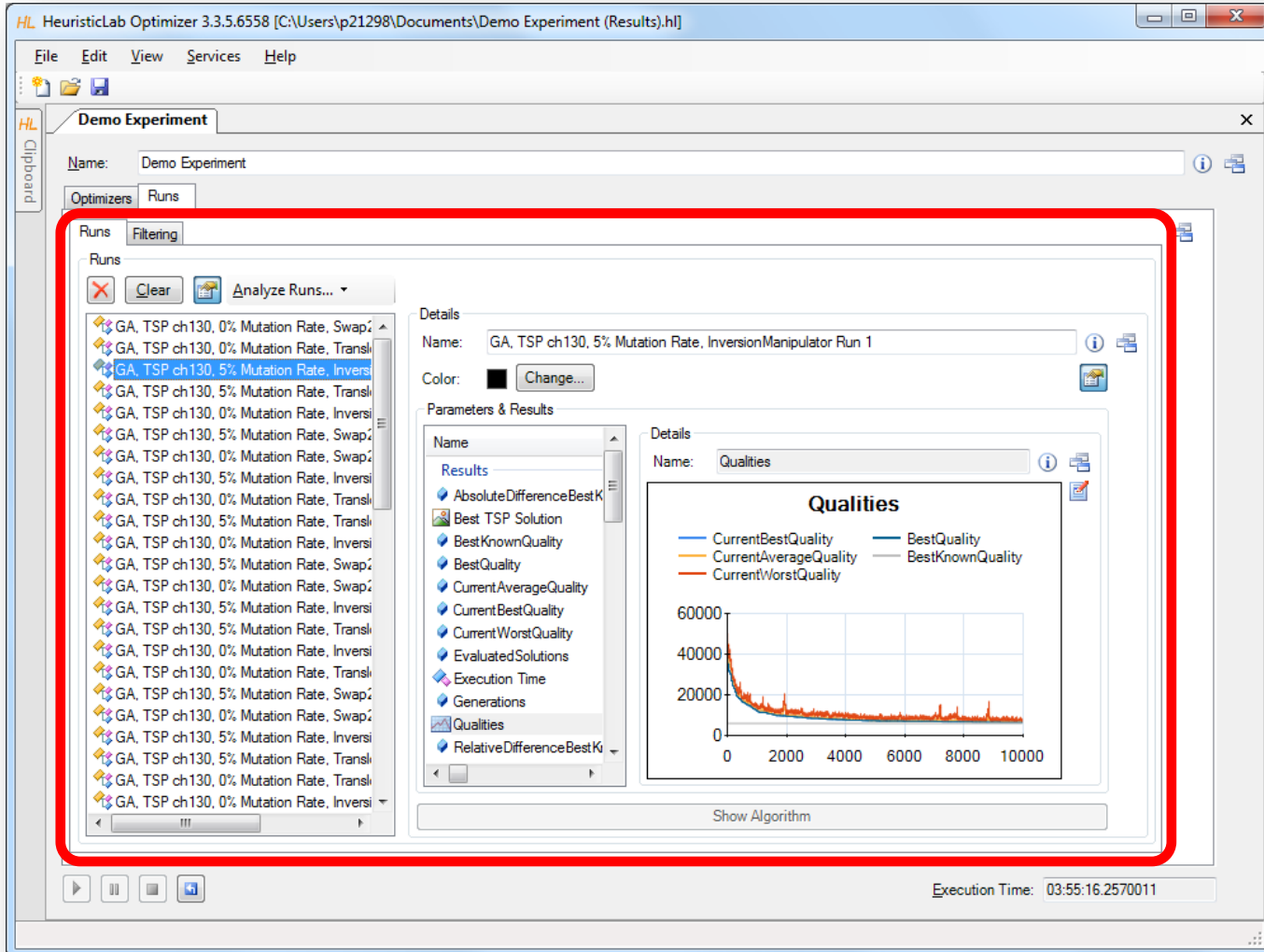


- Show items
 - double-click on an item in the clipboard to show its view
- Save and restore clipboard content
 - click on the save button to write the clipboard content to disk
 - clipboard is automatically restored when HeuristicLab is started the next time

Start, Pause, Resume, Stop, Reset



Compare Runs



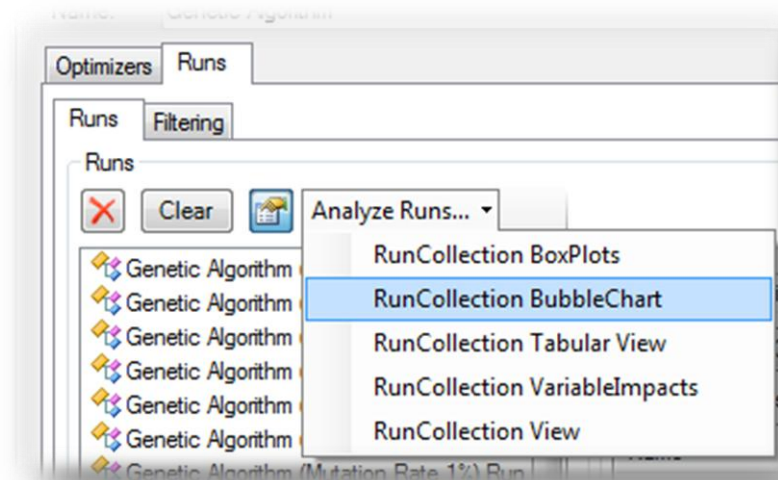
The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.5.6558 [C:\Users\p21298\Documents\Demo Experiment (Results).hl]". The interface is divided into several sections:

- File Edit View Services Help**: The top menu bar.
- Demo Experiment**: The main workspace, containing:
 - Name:** Demo Experiment
 - Optimizers** and **Runs** tabs.
 - Runs** tab: A list of runs with columns for Name, Color, and Parameters & Results. A red box highlights this list. The runs are labeled "GA, TSP ch130, 0% Mutation Rate, Swap;" and "GA, TSP ch130, 5% Mutation Rate, Invers...".
 - Details** panel: Shows details for the selected run: "GA, TSP ch130, 5% Mutation Rate, InversionManipulator Run 1". It includes a "Color" selector and a "Parameters & Results" list.
 - Parameters & Results** panel: Shows a list of results including "AbsoluteDifferenceBestK", "Best TSP Solution", "BestKnownQuality", "BestQuality", "CurrentAverageQuality", "CurrentBestQuality", "CurrentWorstQuality", "EvaluatedSolutions", "Execution Time", "Generations", "Qualities", and "RelativeDifferenceBestK".
 - Qualities** graph: A line graph showing the quality of the solution over time. The Y-axis is labeled "Qualities" and ranges from 0 to 60000. The X-axis is labeled "Generations" and ranges from 0 to 10000. The graph shows four data series: "CurrentBestQuality" (blue), "CurrentAverageQuality" (orange), "CurrentWorstQuality" (red), and "BestQuality" (green). The graph shows a sharp initial drop in quality, followed by a period of relative stability with some fluctuations.

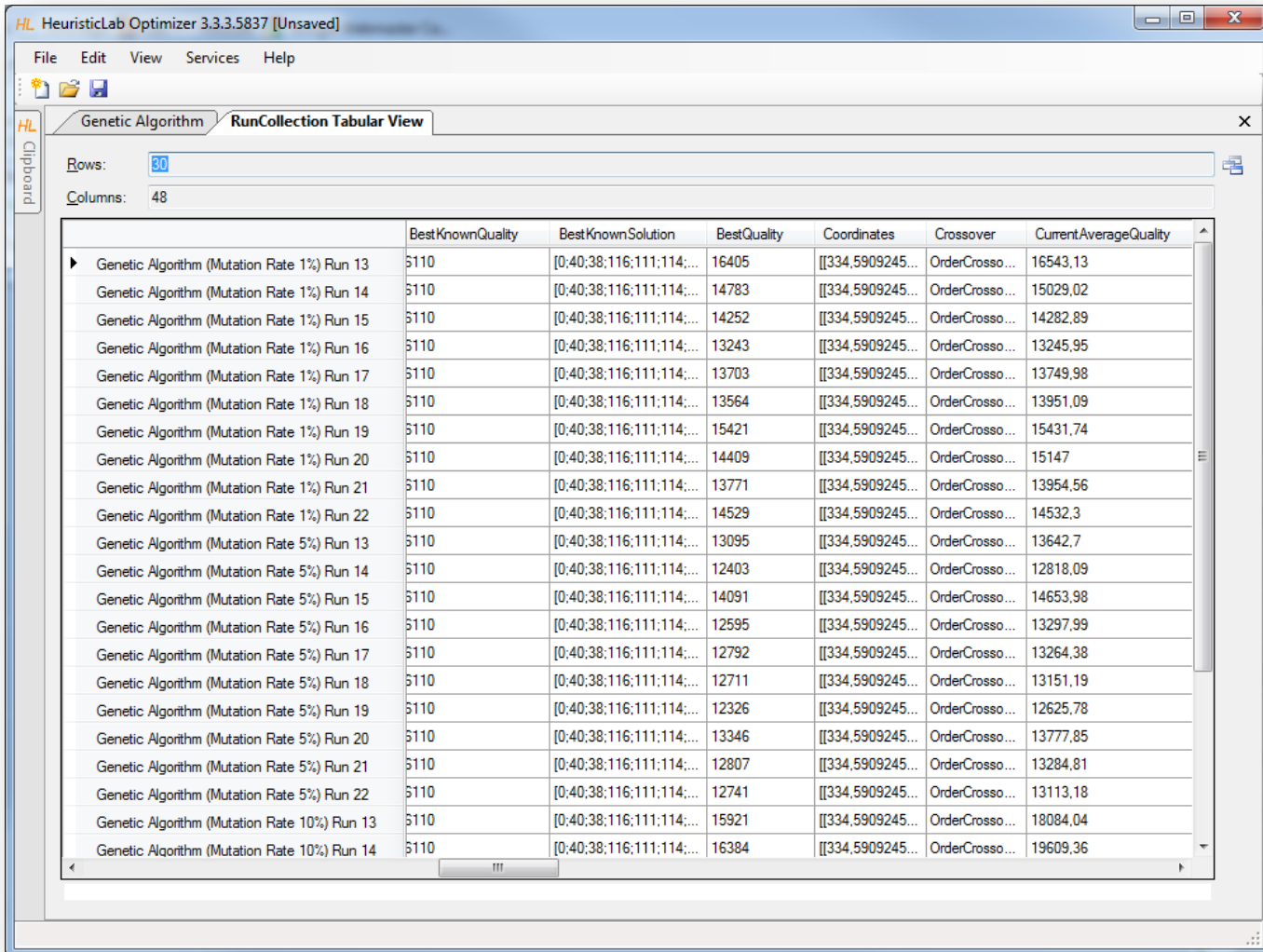
At the bottom of the interface, the **Execution Time** is displayed as 03:55:16.2570011.

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



HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

HL Clipboard

Genetic Algorithm RunCollection Tabular View

Rows: 30

Columns: 48

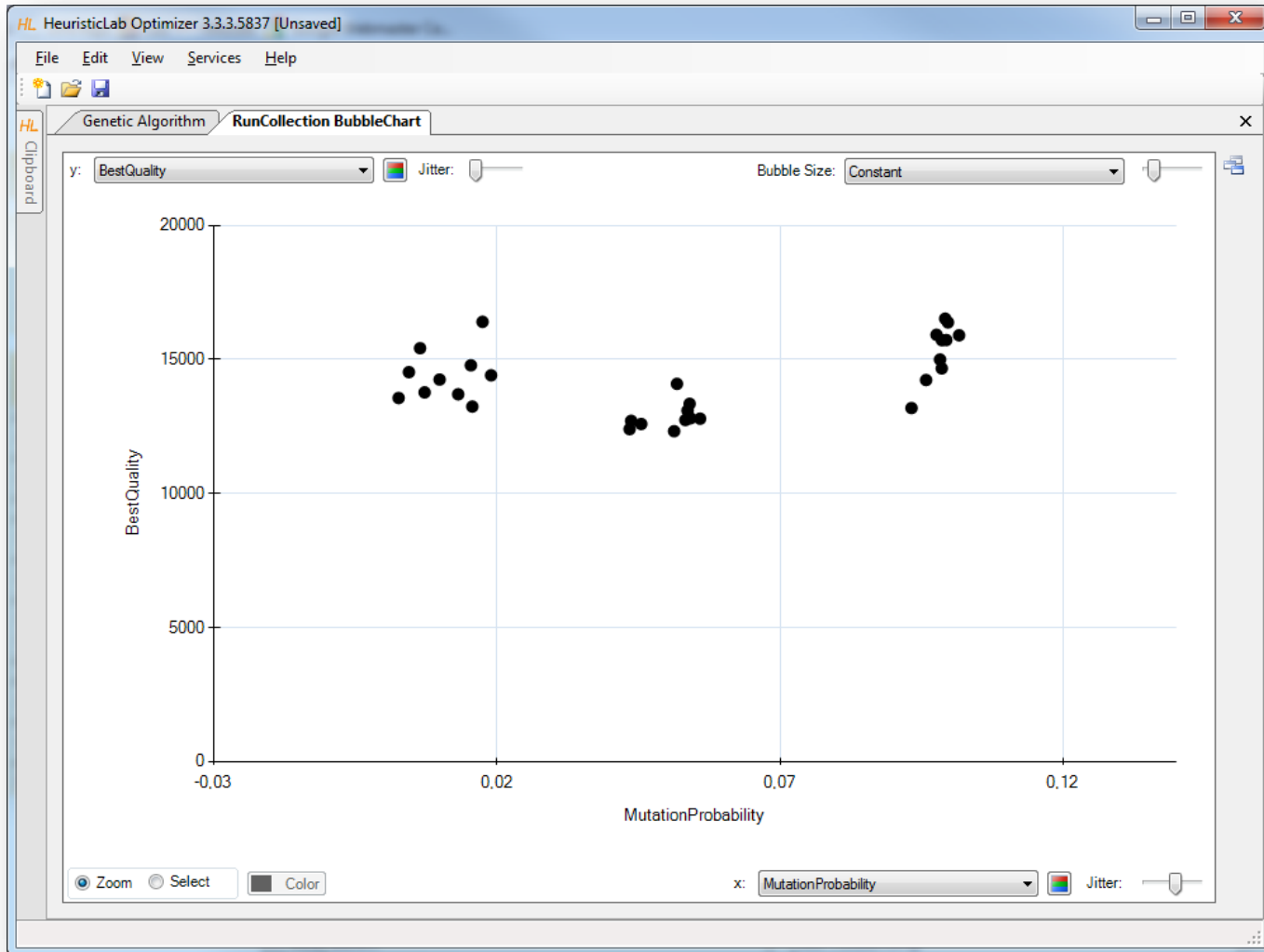
	BestKnownQuality	BestKnownSolution	BestQuality	Coordinates	Crossover	CurrentAverageQuality
▶ Genetic Algorithm (Mutation Rate 1%) Run 13	5110	[0;40;38;116;111;114;...	16405	[[334,5909245...	OrderCrosso...	16543,13
Genetic Algorithm (Mutation Rate 1%) Run 14	5110	[0;40;38;116;111;114;...	14783	[[334,5909245...	OrderCrosso...	15029,02
Genetic Algorithm (Mutation Rate 1%) Run 15	5110	[0;40;38;116;111;114;...	14252	[[334,5909245...	OrderCrosso...	14282,89
Genetic Algorithm (Mutation Rate 1%) Run 16	5110	[0;40;38;116;111;114;...	13243	[[334,5909245...	OrderCrosso...	13245,95
Genetic Algorithm (Mutation Rate 1%) Run 17	5110	[0;40;38;116;111;114;...	13703	[[334,5909245...	OrderCrosso...	13749,98
Genetic Algorithm (Mutation Rate 1%) Run 18	5110	[0;40;38;116;111;114;...	13564	[[334,5909245...	OrderCrosso...	13951,09
Genetic Algorithm (Mutation Rate 1%) Run 19	5110	[0;40;38;116;111;114;...	15421	[[334,5909245...	OrderCrosso...	15431,74
Genetic Algorithm (Mutation Rate 1%) Run 20	5110	[0;40;38;116;111;114;...	14409	[[334,5909245...	OrderCrosso...	15147
Genetic Algorithm (Mutation Rate 1%) Run 21	5110	[0;40;38;116;111;114;...	13771	[[334,5909245...	OrderCrosso...	13954,56
Genetic Algorithm (Mutation Rate 1%) Run 22	5110	[0;40;38;116;111;114;...	14529	[[334,5909245...	OrderCrosso...	14532,3
Genetic Algorithm (Mutation Rate 5%) Run 13	5110	[0;40;38;116;111;114;...	13095	[[334,5909245...	OrderCrosso...	13642,7
Genetic Algorithm (Mutation Rate 5%) Run 14	5110	[0;40;38;116;111;114;...	12403	[[334,5909245...	OrderCrosso...	12818,09
Genetic Algorithm (Mutation Rate 5%) Run 15	5110	[0;40;38;116;111;114;...	14091	[[334,5909245...	OrderCrosso...	14653,98
Genetic Algorithm (Mutation Rate 5%) Run 16	5110	[0;40;38;116;111;114;...	12595	[[334,5909245...	OrderCrosso...	13297,99
Genetic Algorithm (Mutation Rate 5%) Run 17	5110	[0;40;38;116;111;114;...	12792	[[334,5909245...	OrderCrosso...	13264,38
Genetic Algorithm (Mutation Rate 5%) Run 18	5110	[0;40;38;116;111;114;...	12711	[[334,5909245...	OrderCrosso...	13151,19
Genetic Algorithm (Mutation Rate 5%) Run 19	5110	[0;40;38;116;111;114;...	12326	[[334,5909245...	OrderCrosso...	12625,78
Genetic Algorithm (Mutation Rate 5%) Run 20	5110	[0;40;38;116;111;114;...	13346	[[334,5909245...	OrderCrosso...	13777,85
Genetic Algorithm (Mutation Rate 5%) Run 21	5110	[0;40;38;116;111;114;...	12807	[[334,5909245...	OrderCrosso...	13284,81
Genetic Algorithm (Mutation Rate 5%) Run 22	5110	[0;40;38;116;111;114;...	12741	[[334,5909245...	OrderCrosso...	13113,18
Genetic Algorithm (Mutation Rate 10%) Run 13	5110	[0;40;38;116;111;114;...	15921	[[334,5909245...	OrderCrosso...	18084,04
Genetic Algorithm (Mutation Rate 10%) Run 14	5110	[0;40;38;116;111;114;...	16384	[[334,5909245...	OrderCrosso...	19609,36

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

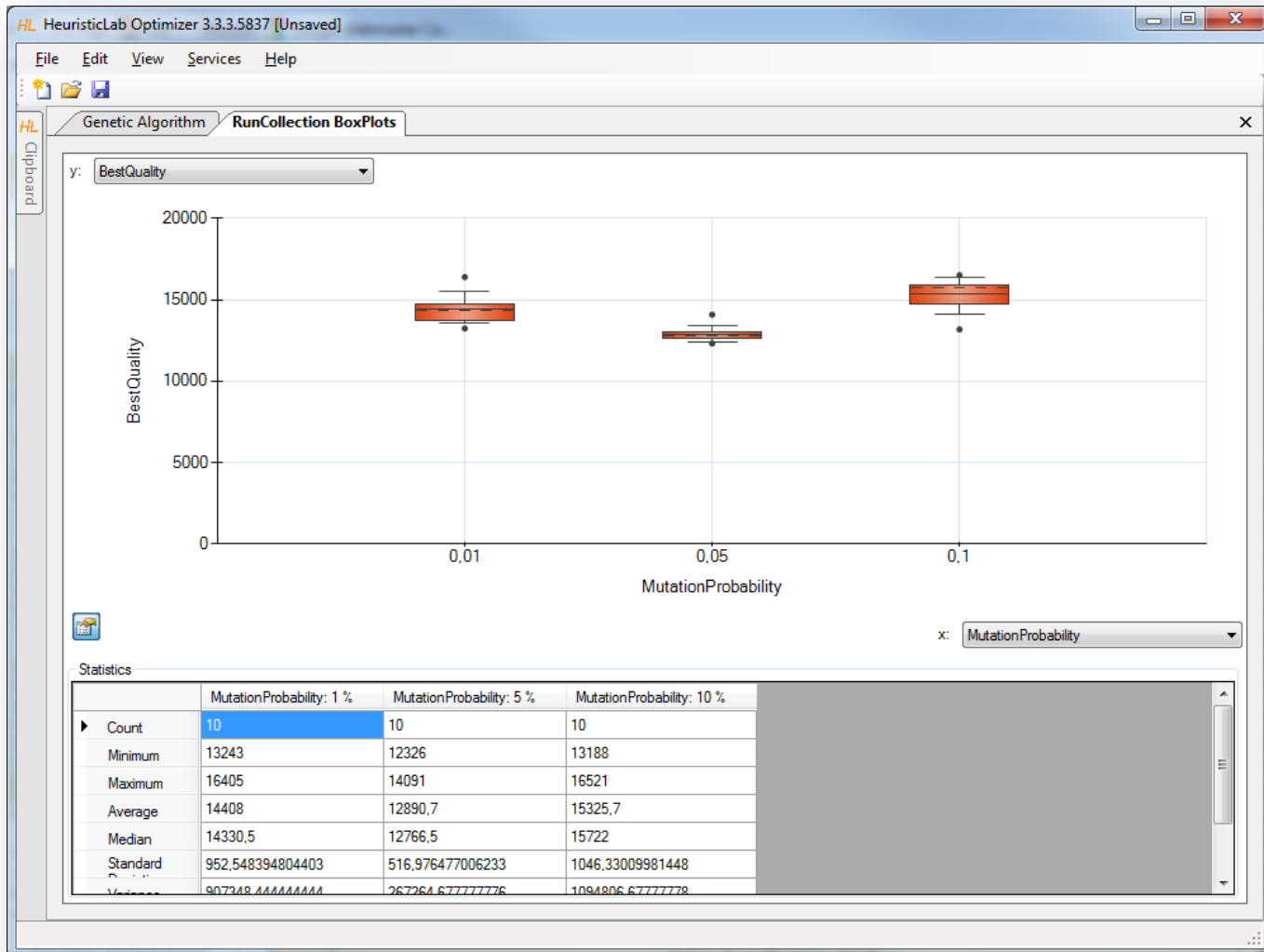
RunCollection BubbleChart



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

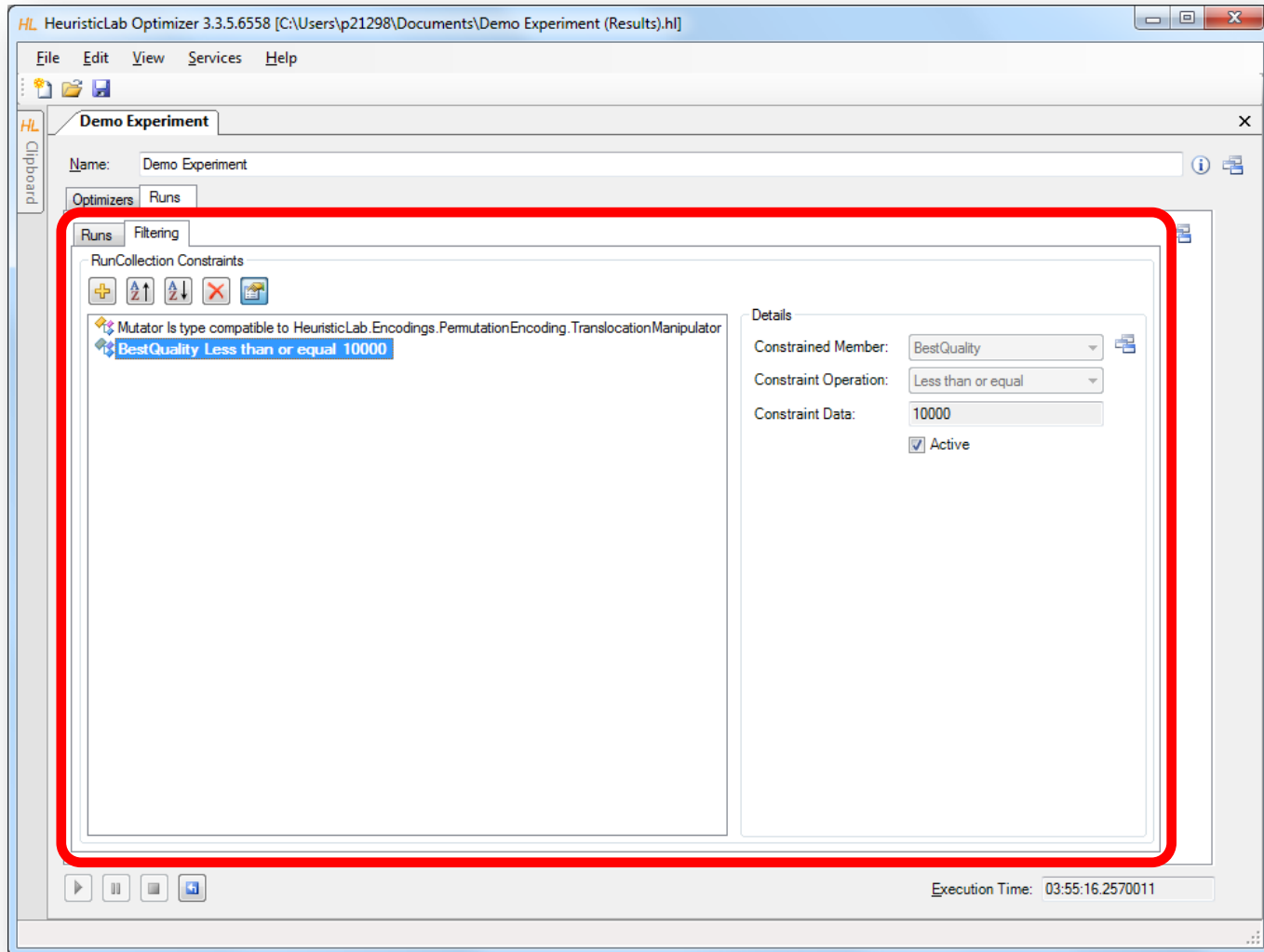
RunCollection BoxPlots



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)

Filter Runs

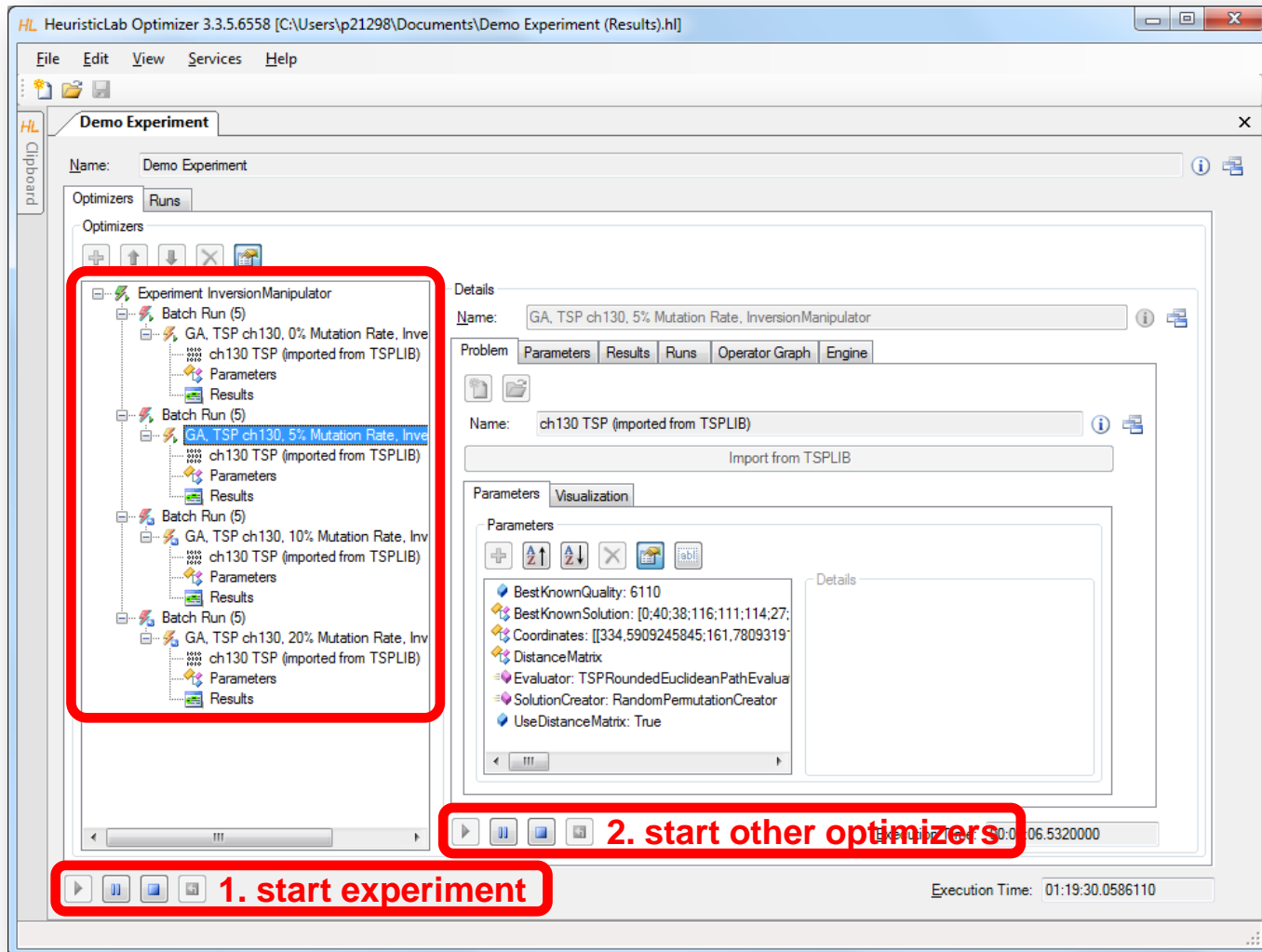


Multi-core CPUs and Parallelization

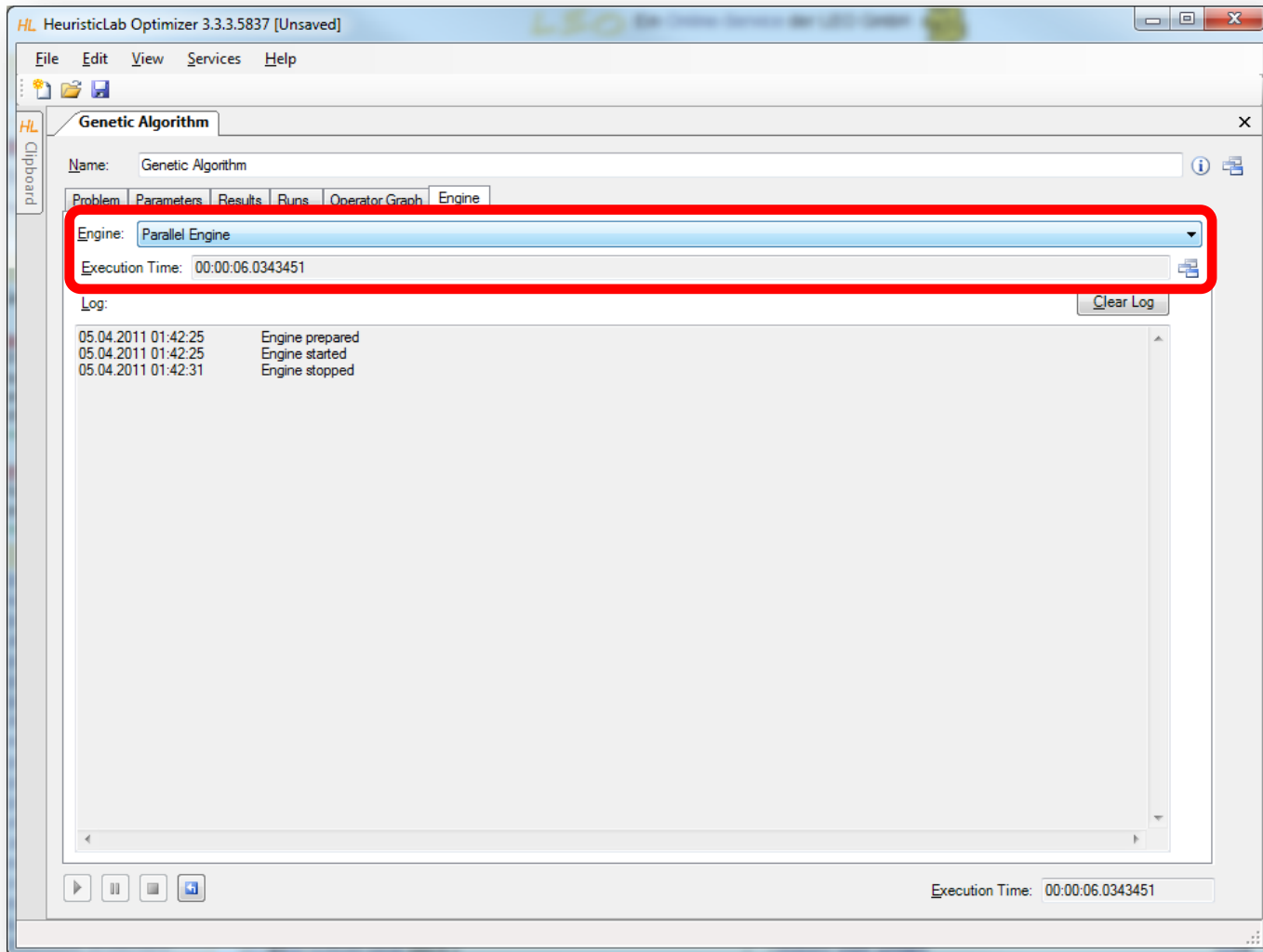


- Parallel execution of optimizers in experiments
 - optimizers in an experiment are executed sequentially from top to bottom per default
 - experiments support parallel execution of their optimizers
 - select a not yet executed optimizer and start it manually to utilize another core
 - execution of 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



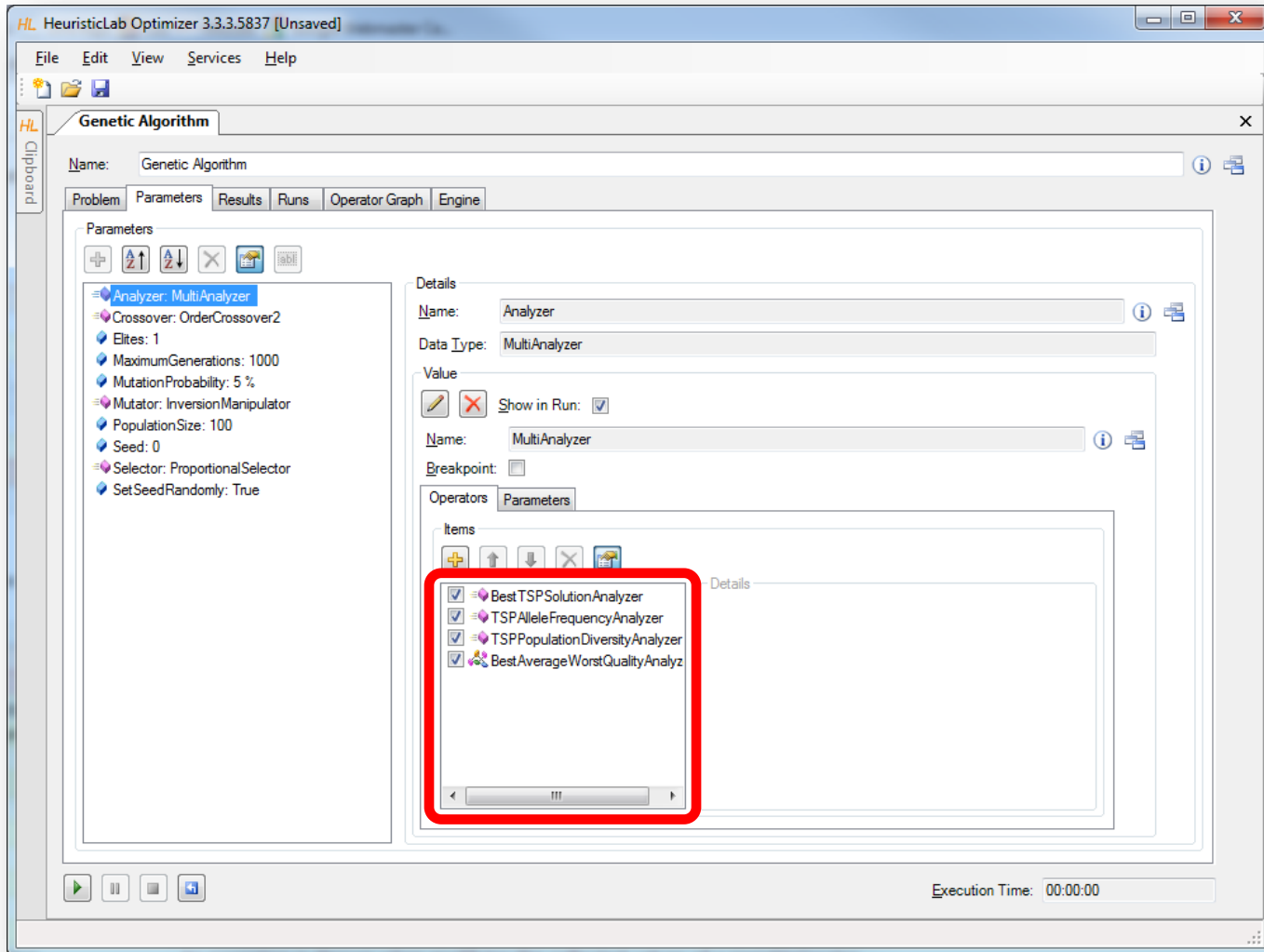
Parallel Execution of Algorithms



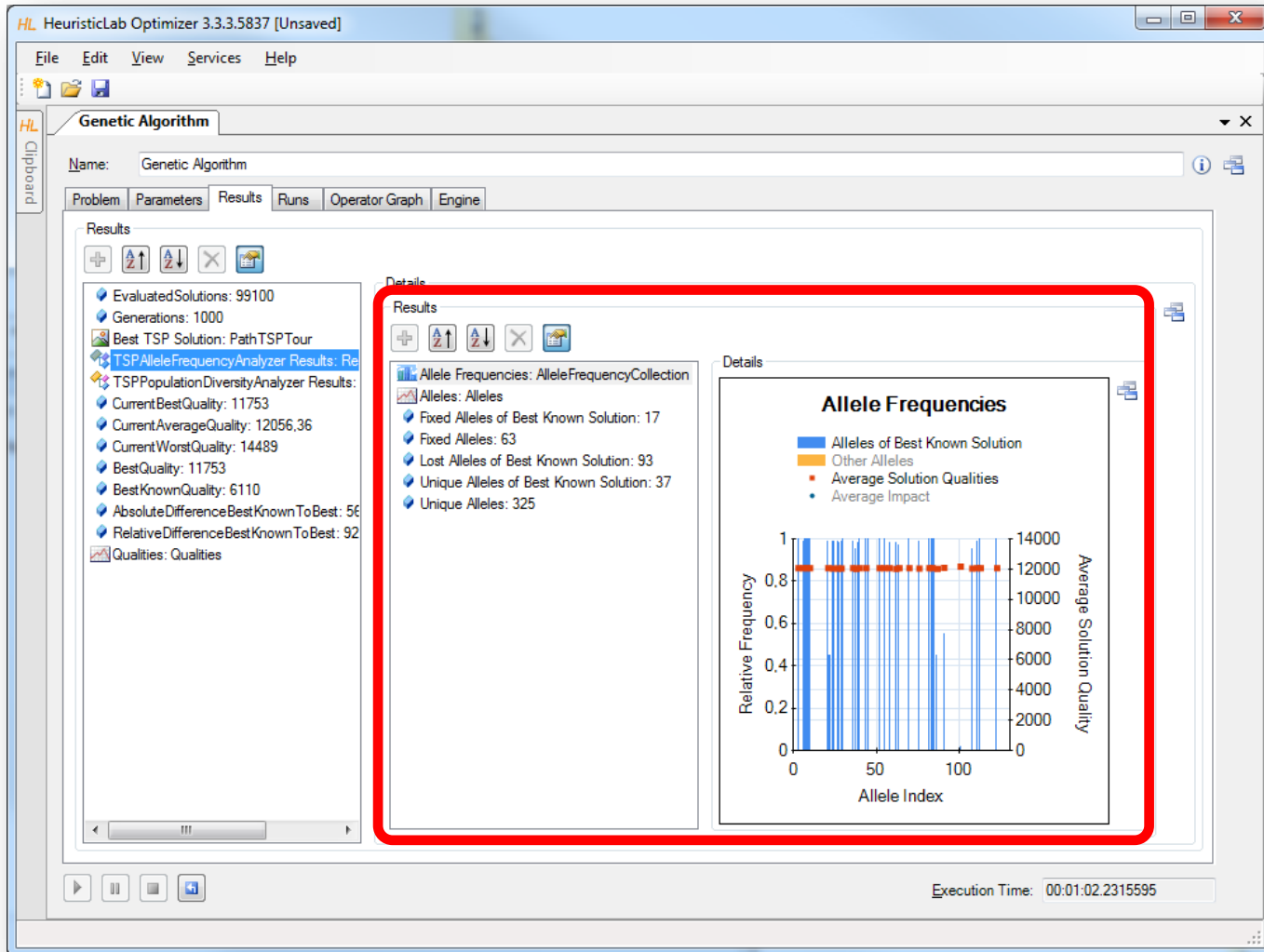
Analyzers

- 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
 - ...

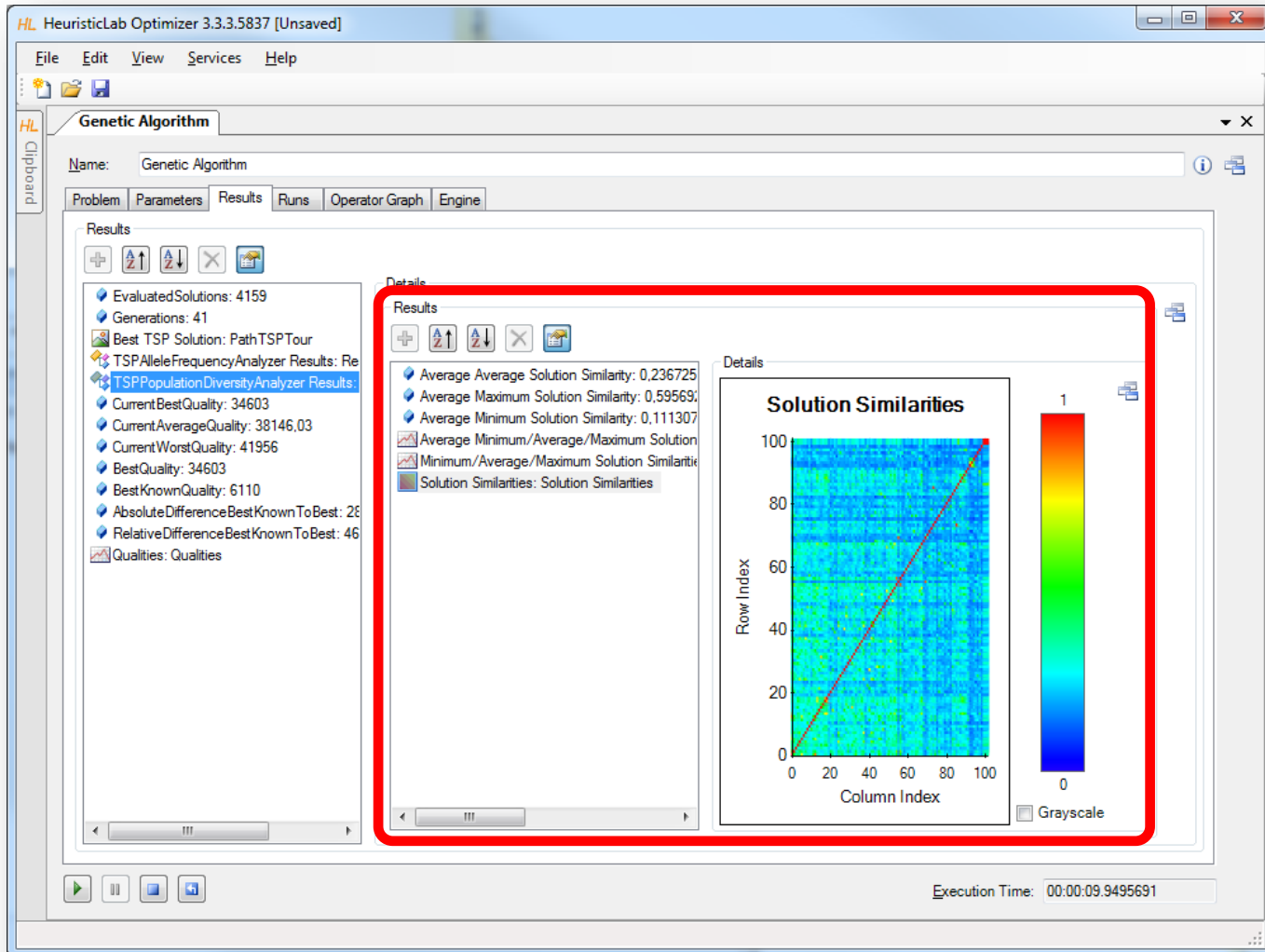
Analyzers



TSPAlleleFrequencyAnalyzer

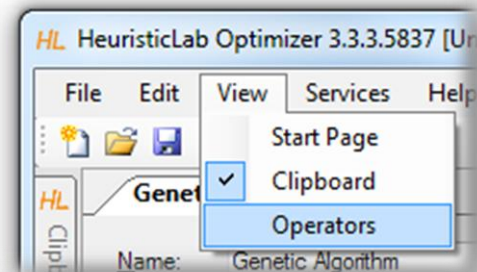
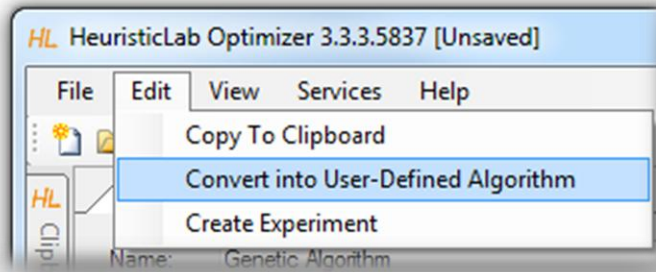


TSP Population Diversity Analyzer



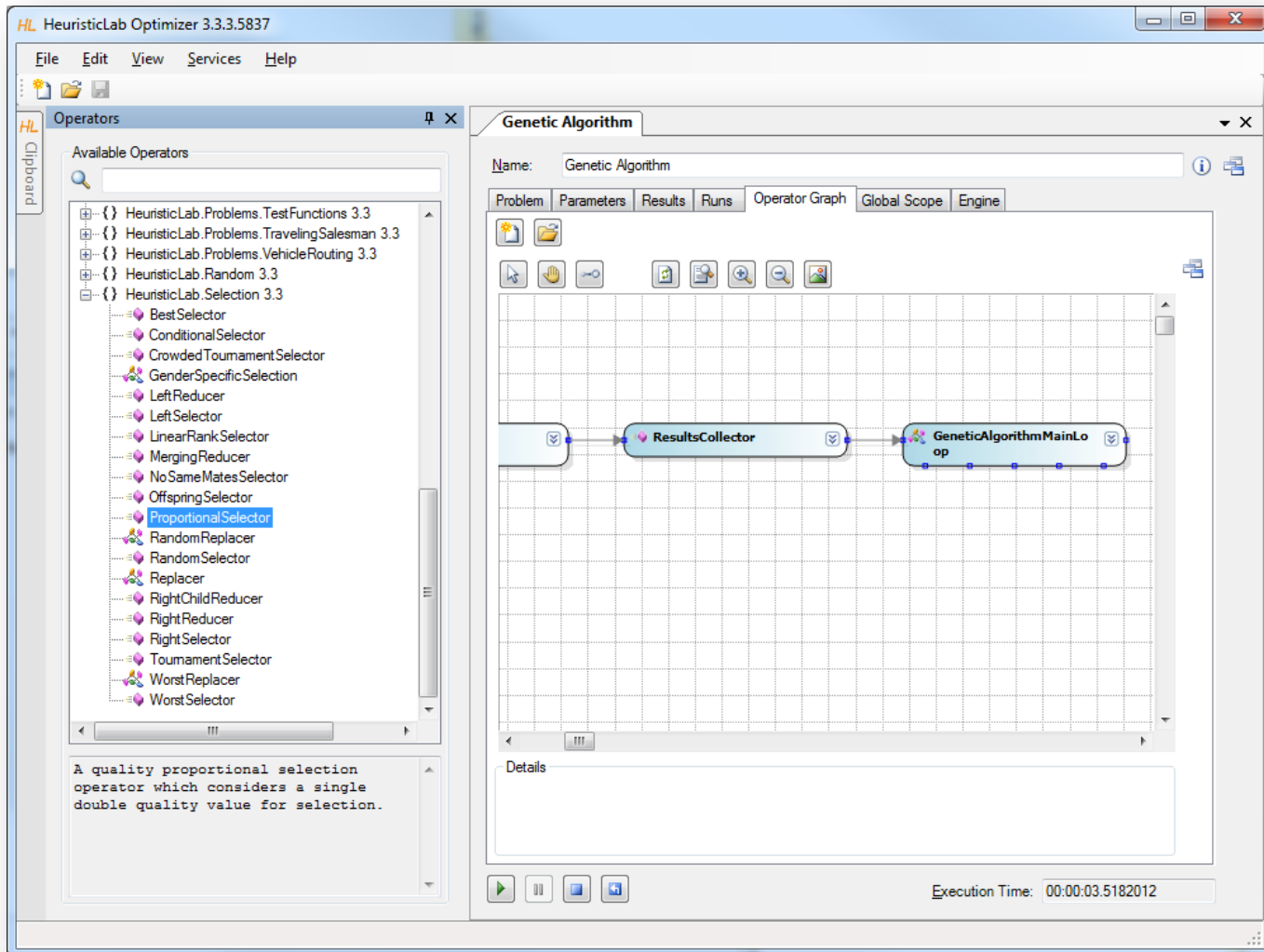
Building User-Defined Algorithms

- 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



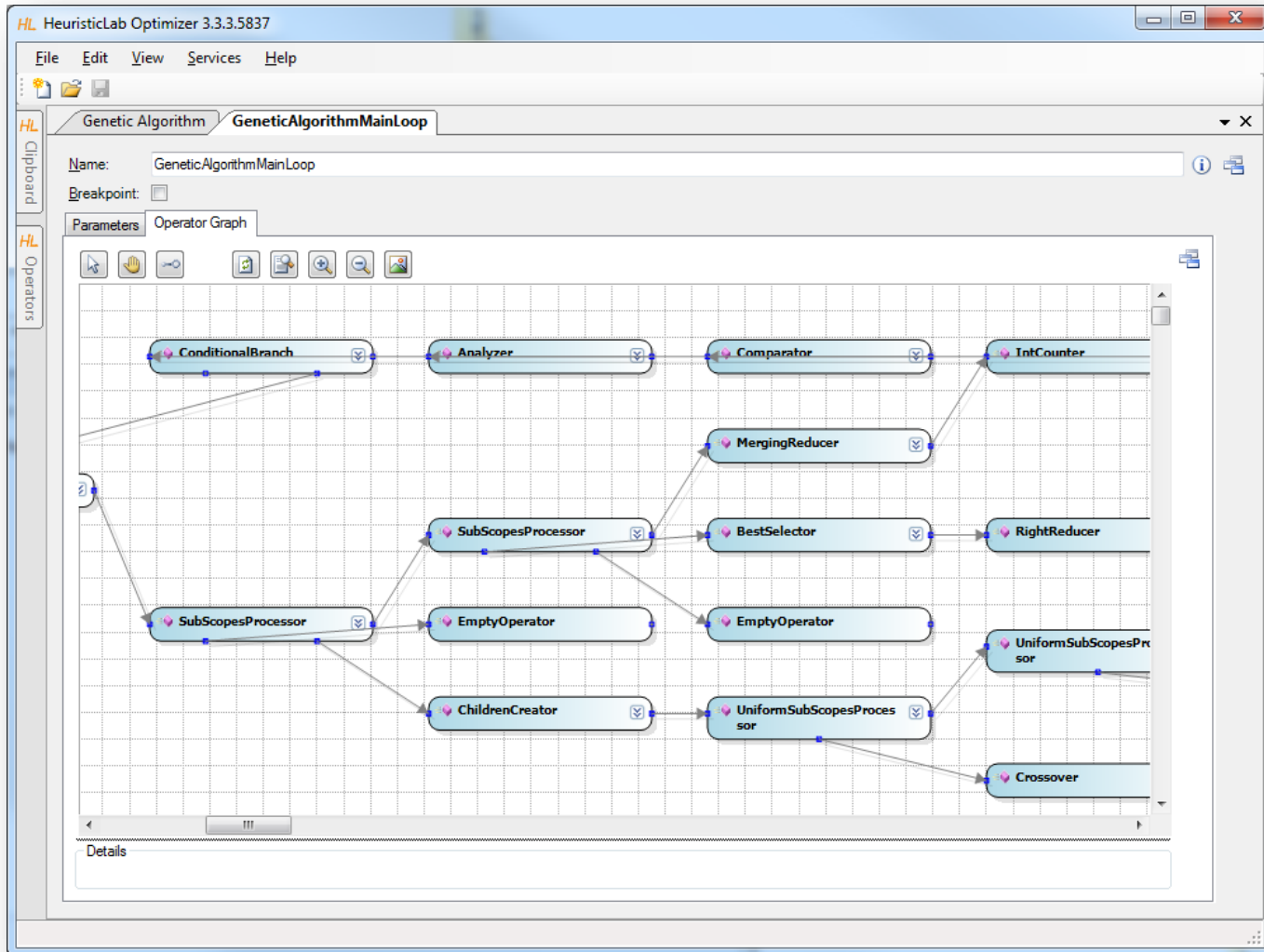
- Operators sidebar
 - drag & drop operators into an operator graph
- Programmable operators
 - add programmable operators in order to implement custom logic in an algorithm
 - no additional development environment needed
- Debug algorithms
 - use the debug engine to obtain detailed information during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

Building User-Defined Algorithms

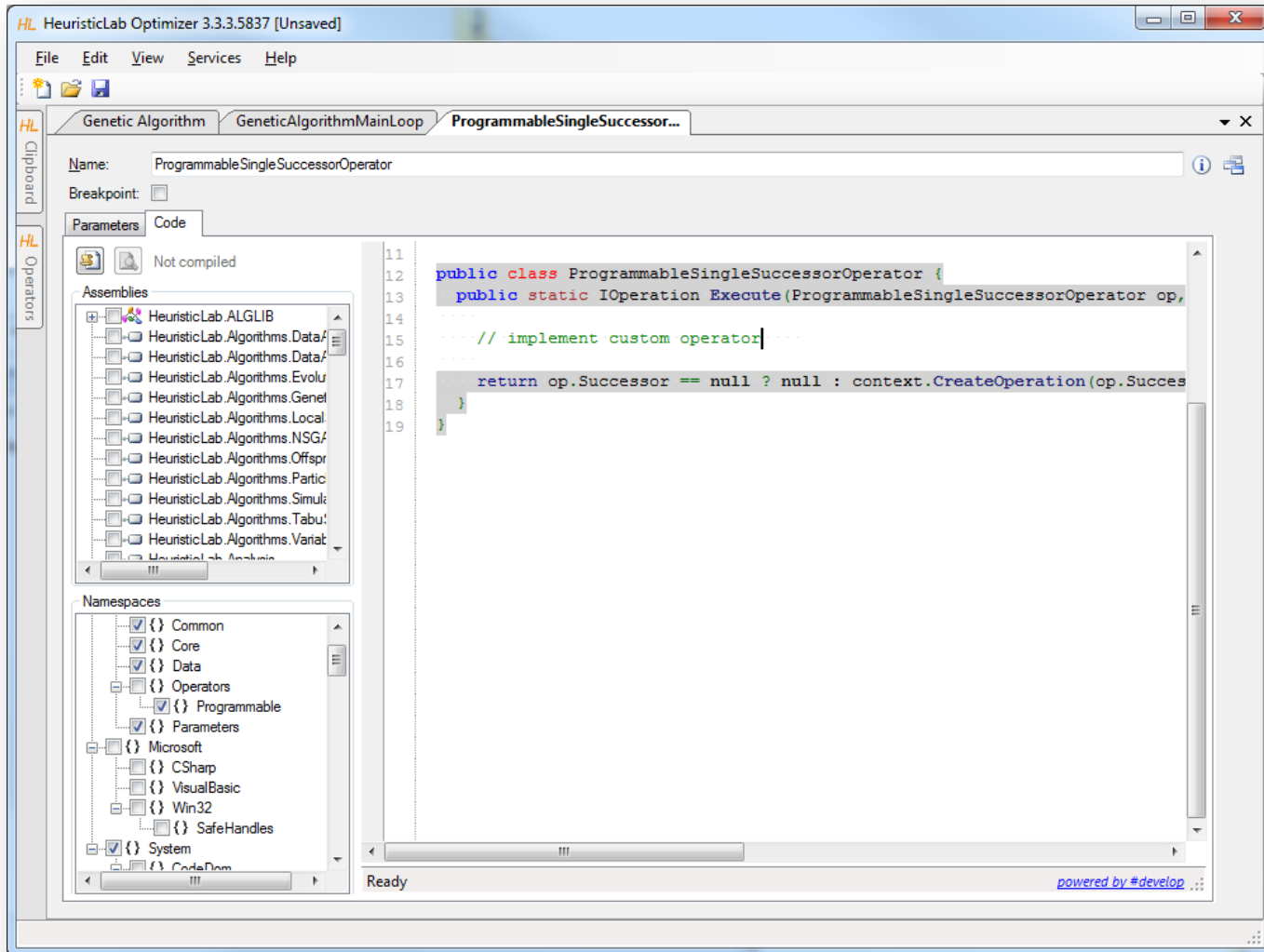


The screenshot displays the HeuristicLab Optimizer interface. On the left, the 'Operators' panel lists various selection and reduction operators, with 'ProportionalSelector' highlighted. Below this list is a description: 'A quality proportional selection operator which considers a single double quality value for selection.' The main workspace shows a 'Genetic Algorithm' configuration with an 'Operator Graph' tab selected. The graph contains two nodes: 'ResultsCollector' and 'GeneticAlgorithmMainLoop', connected by a flow arrow. The 'Details' section at the bottom is empty. The status bar at the bottom right indicates 'Execution Time: 00:00:03.5182012'.

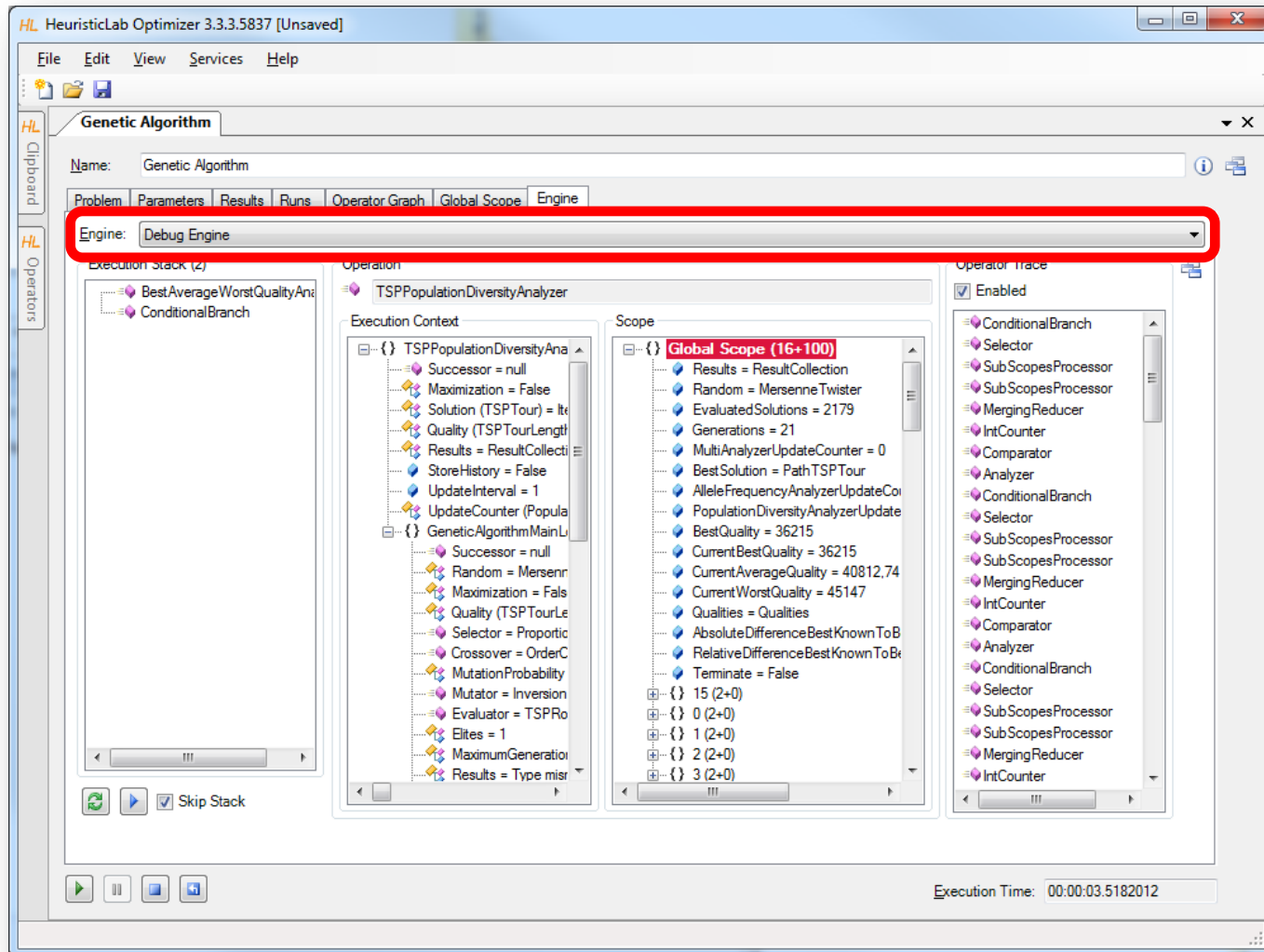
Building User-Defined Algorithms



Programmable Operators



Debugging Algorithms



Agenda



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

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

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

Demonstration Part II: Data-based Modeling



- Introduction
- Regression with HeuristicLab
- Model simplification and export
- Variable relevance analysis
- Classification with HeuristicLab

Introduction to Data-based Modeling



- Dataset: Matrix $(x_{i,j})_{i=1..N, j=1..K}$
 - N observations of K input variables
 - $x_{i,j}$ = i-th observation of j-th variable
 - Additionally: Vector of labels $(y_1 \dots y_N)^T$
- Goal: learn association of input variable values to labels
- Common tasks
 - Regression (real-valued labels)
 - Classification (discrete labels)
 - Clustering (no labels, group similar observations)

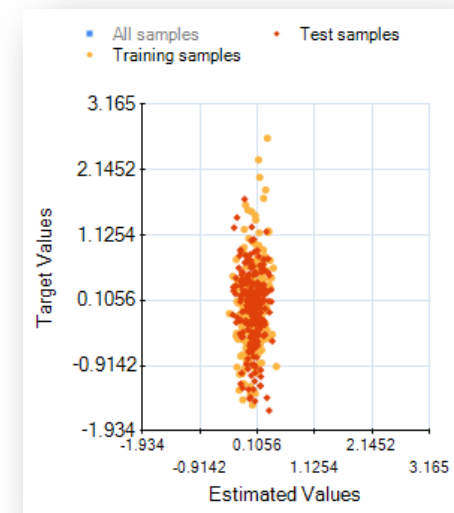
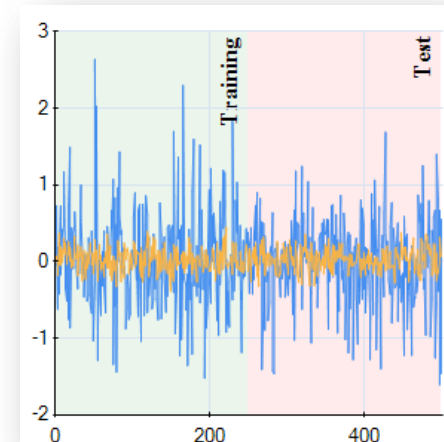
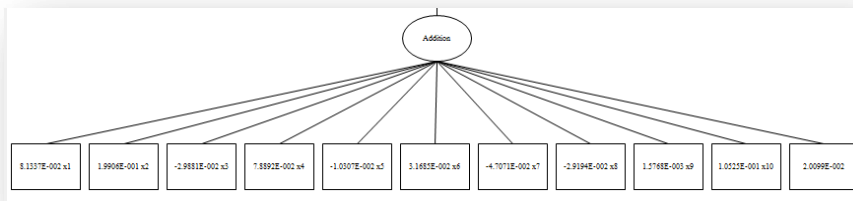
Data-based Modeling Algorithms in HeuristicLab



- 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

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

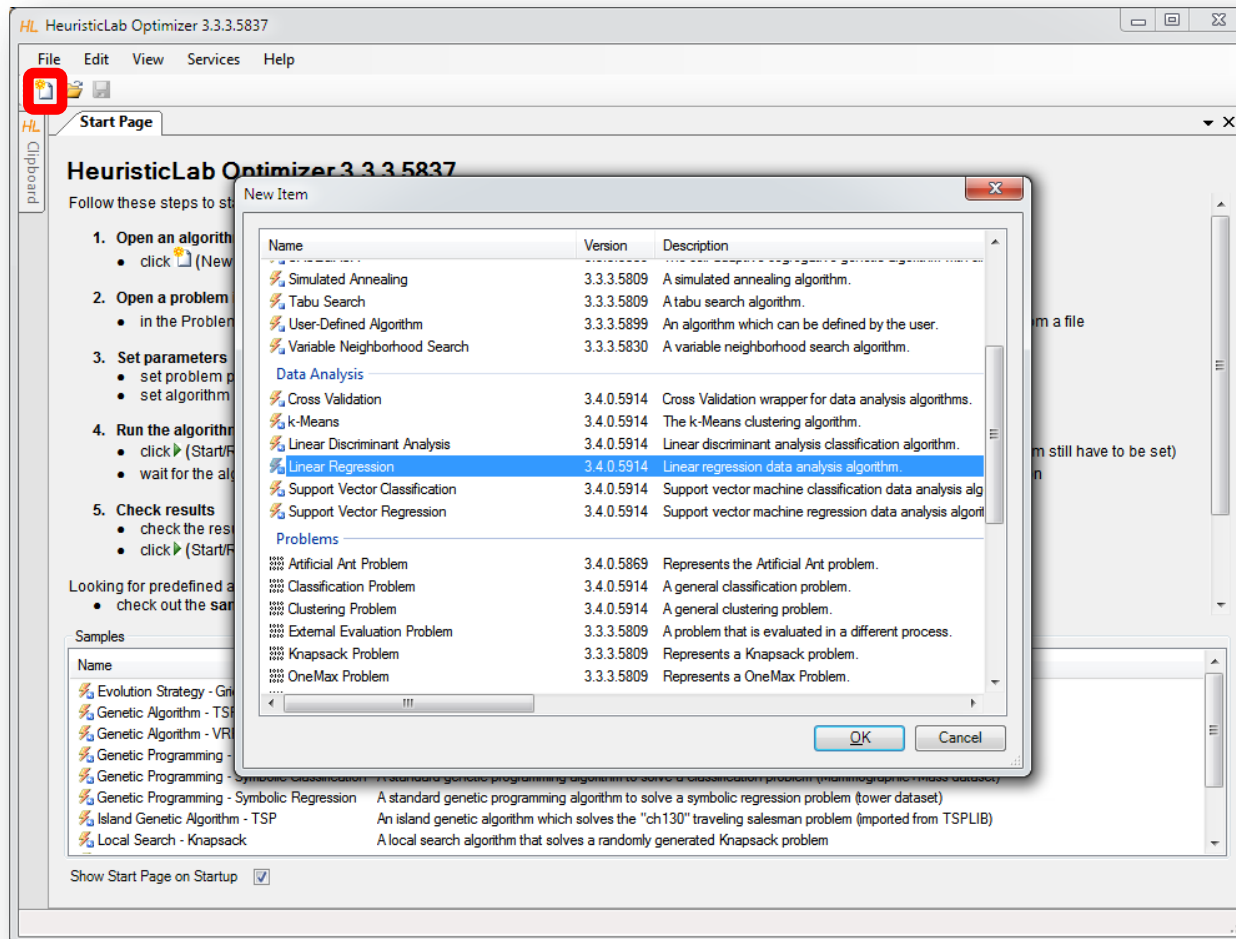


Case Study: Regression

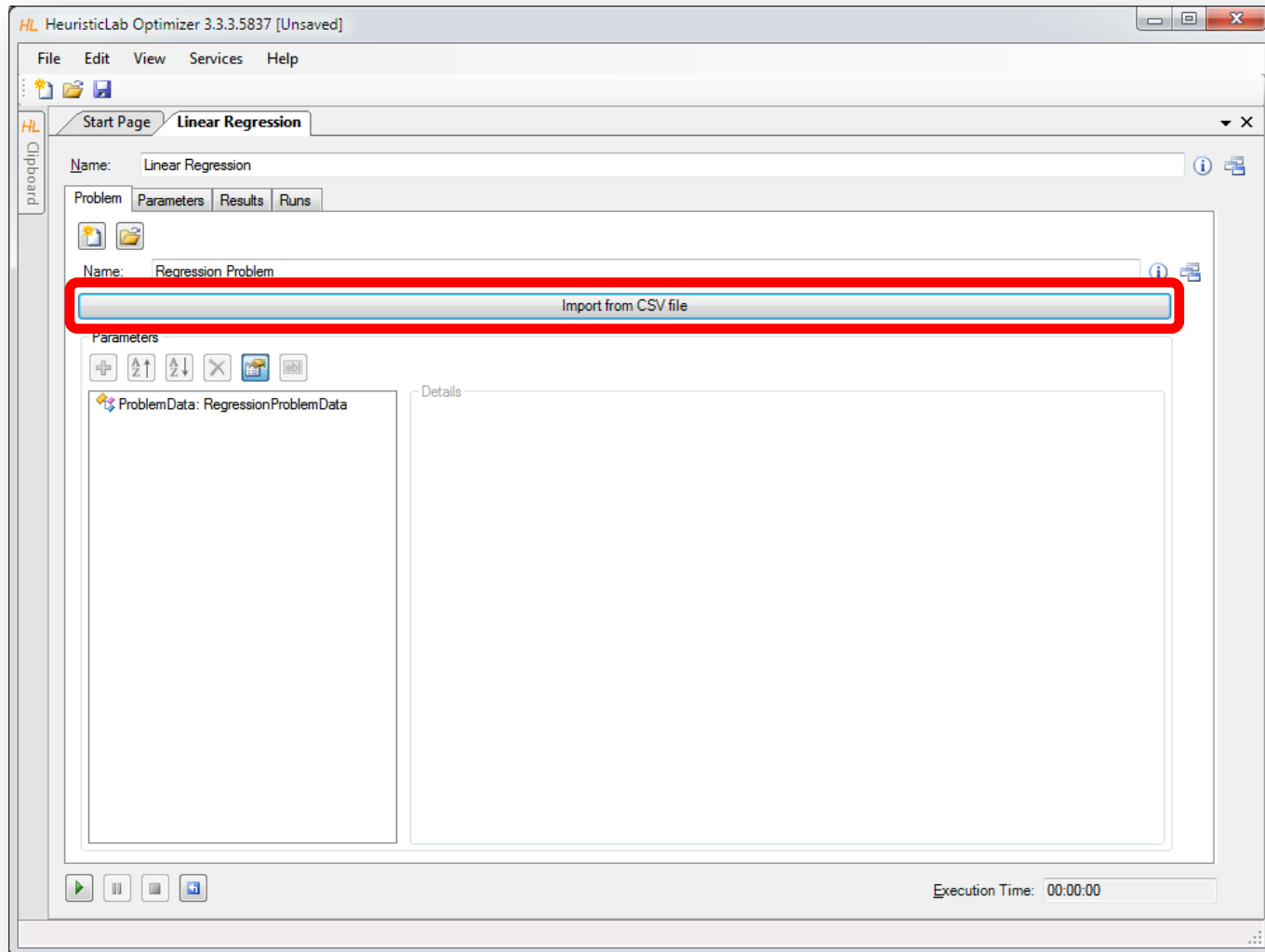
- Poly-10 benchmark problem dataset
 - 10 input variables $x_1 \dots x_{10}$
 - $y = x_1 \cdot x_2 + x_3 \cdot x_4 + x_5 \cdot x_6 + x_1 \cdot x_7 \cdot x_9 + x_3 \cdot x_6 \cdot x_{10}$
 - non-linear modeling approach necessary
 - frequently used in GP literature
 - download
<http://dev.heuristiclab.com/AdditionalMaterial#ICCGI2011>

Linear Regression

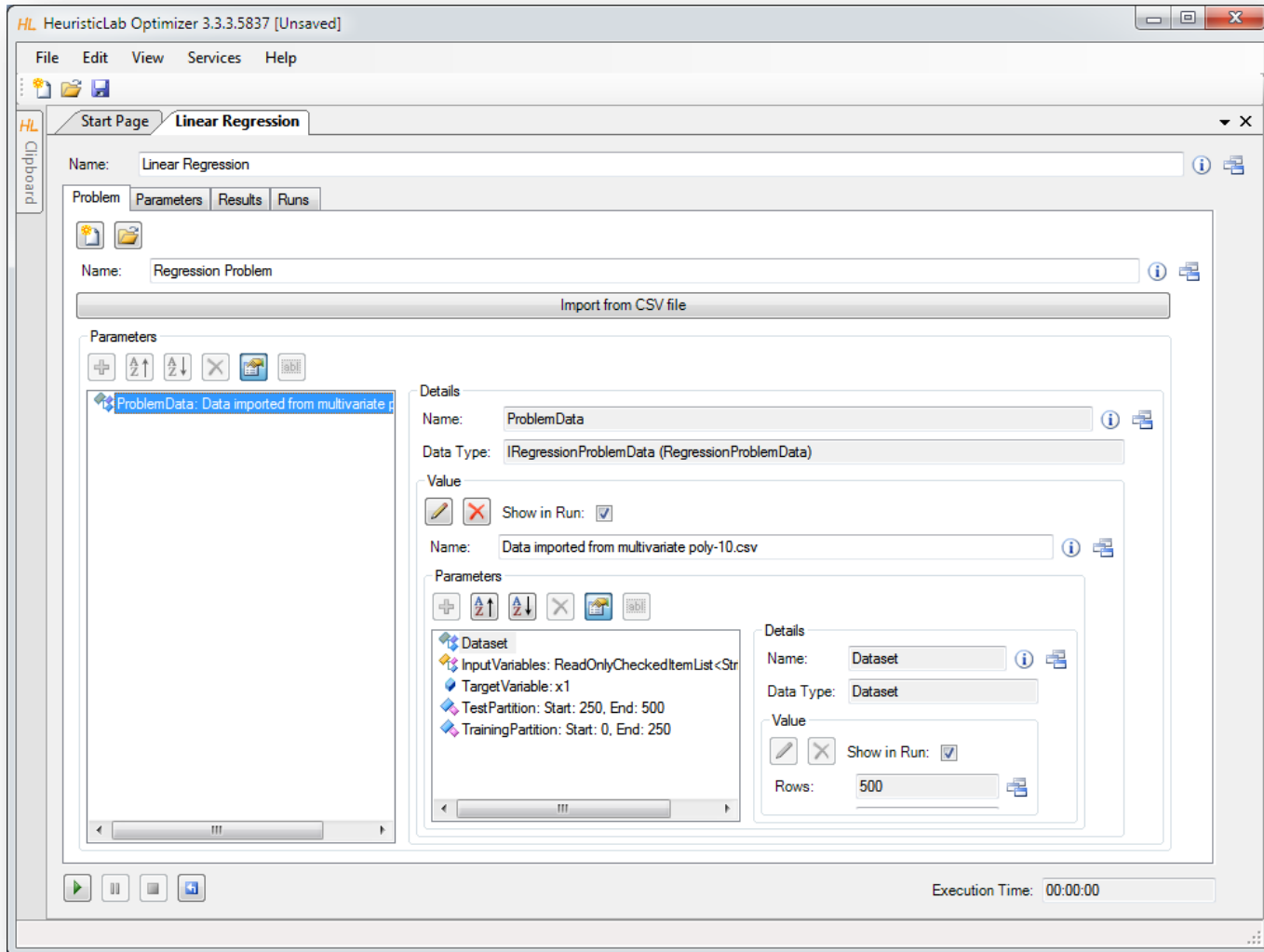
- Create new algorithm



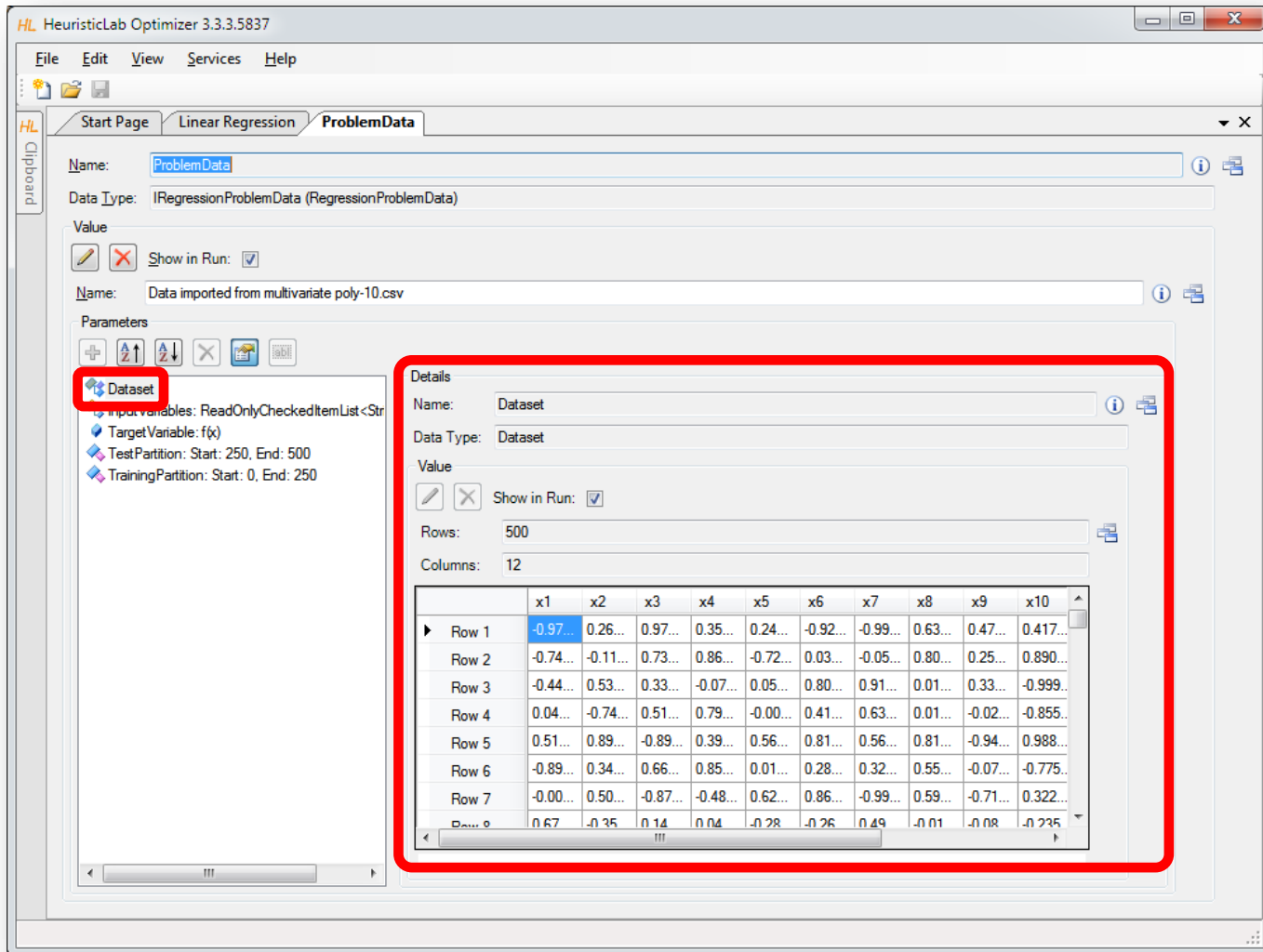
Import Data from CSV-File



Inspect and Configure Dataset



Inspect Imported Data



HL HeuristicLab Optimizer 3.3.3.5837

File Edit View Services Help

Start Page Linear Regression **ProblemData**

Name: ProblemData

Data Type: IRegressionProblemData (RegressionProblemData)

Value

Show in Run:

Name: Data imported from multivariate poly-10.csv

Parameters

Dataset

Input variables: ReadOnlyCheckedItemList<Str

Target Variable: f(x)

TestPartition: Start: 250, End: 500

TrainingPartition: Start: 0, End: 250

Details

Name: Dataset

Data Type: Dataset

Value

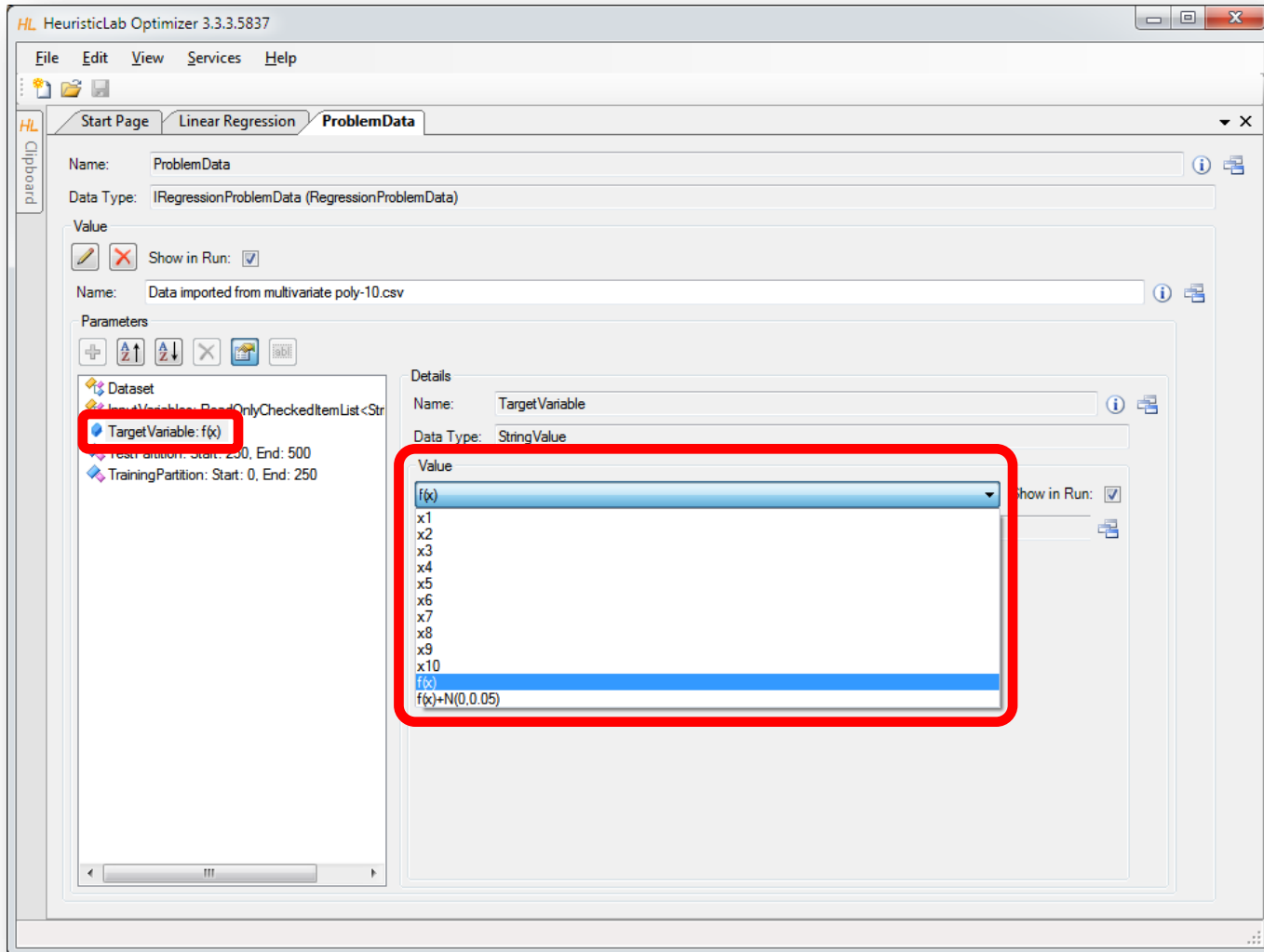
Show in Run:

Rows: 500

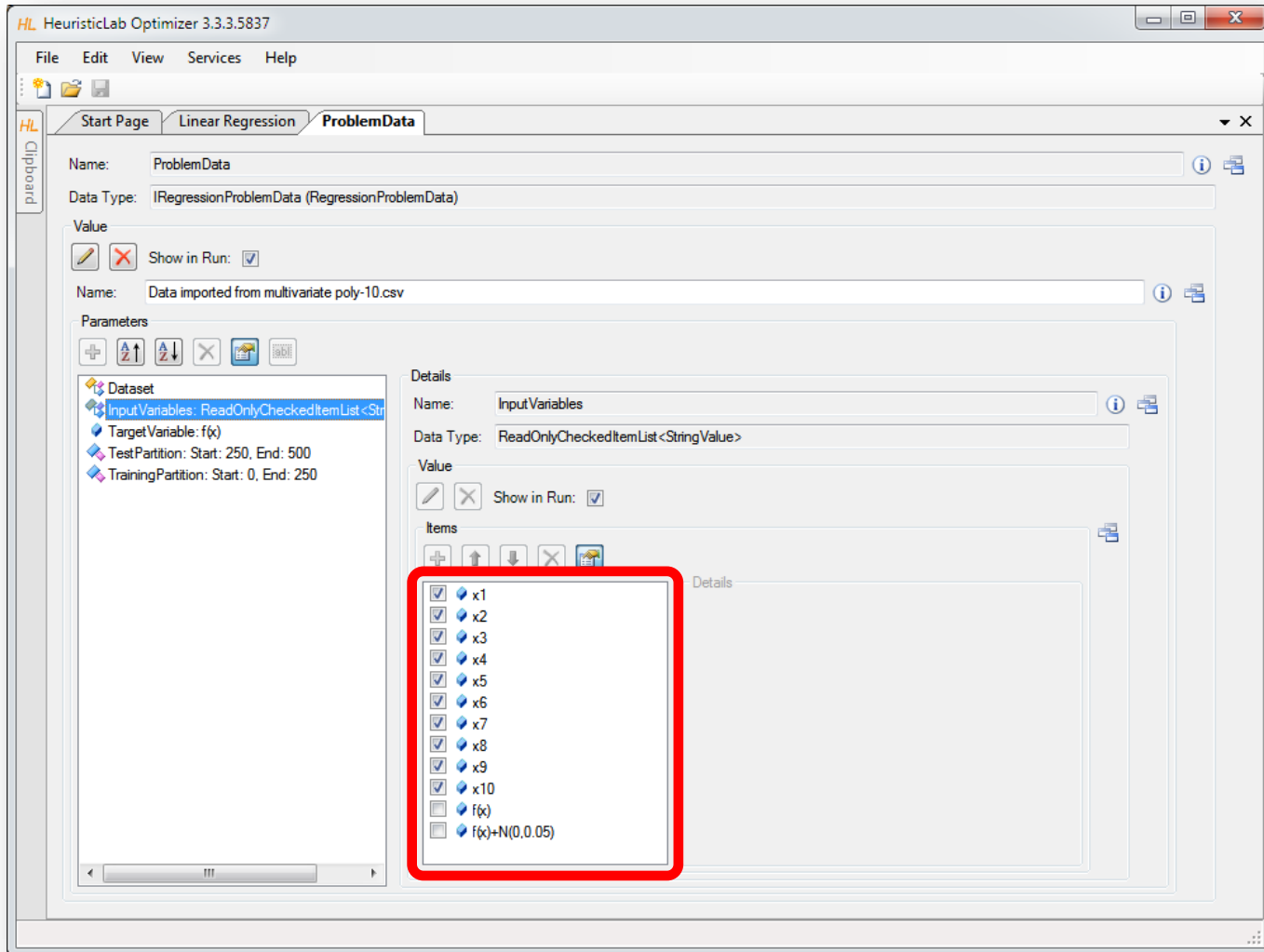
Columns: 12

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
Row 1	-0.97...	0.26...	0.97...	0.35...	0.24...	-0.92...	-0.99...	0.63...	0.47...	0.417...
Row 2	-0.74...	-0.11...	0.73...	0.86...	-0.72...	0.03...	-0.05...	0.80...	0.25...	0.890...
Row 3	-0.44...	0.53...	0.33...	-0.07...	0.05...	0.80...	0.91...	0.01...	0.33...	-0.999...
Row 4	0.04...	-0.74...	0.51...	0.79...	-0.00...	0.41...	0.63...	0.01...	-0.02...	-0.855...
Row 5	0.51...	0.89...	-0.89...	0.39...	0.56...	0.81...	0.56...	0.81...	-0.94...	0.988...
Row 6	-0.89...	0.34...	0.66...	0.85...	0.01...	0.28...	0.32...	0.55...	-0.07...	-0.775...
Row 7	-0.00...	0.50...	-0.87...	-0.48...	0.62...	0.86...	-0.99...	0.59...	-0.71...	0.322...
Row 8	0.67...	-0.35...	0.14...	0.04...	-0.28...	-0.26...	0.49...	-0.01...	-0.08...	-0.235...

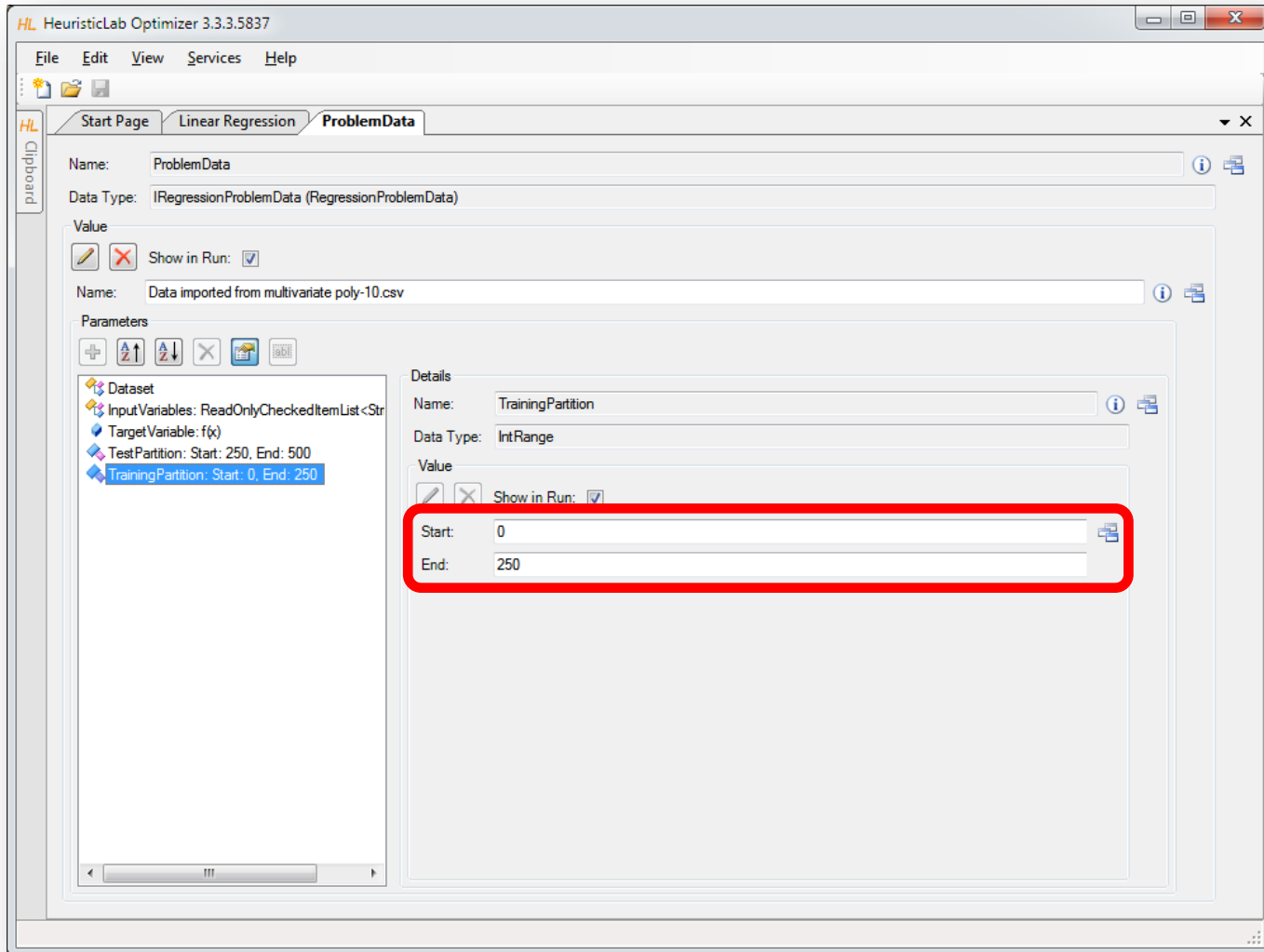
Set Target Variable



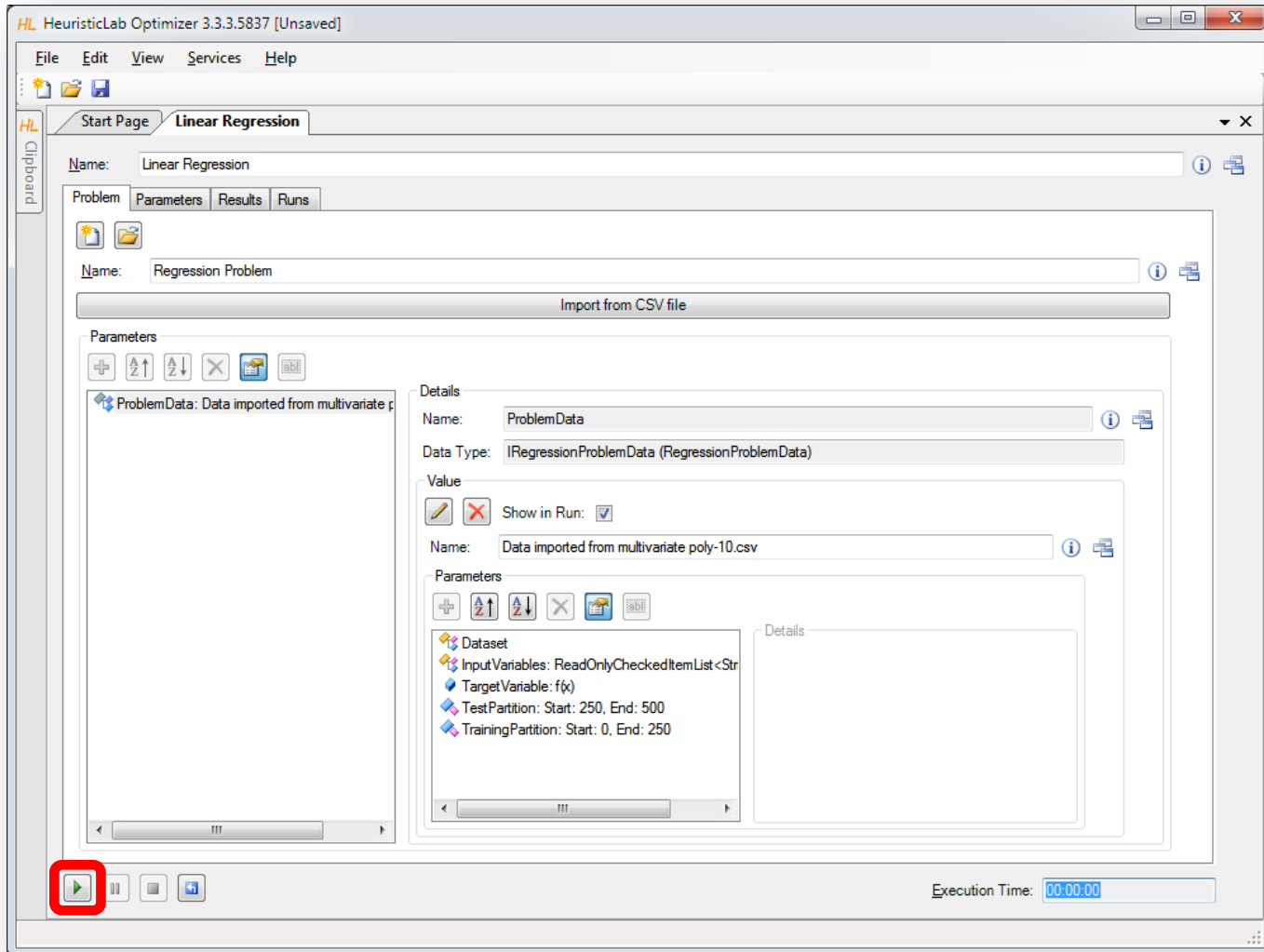
Select Input Variables



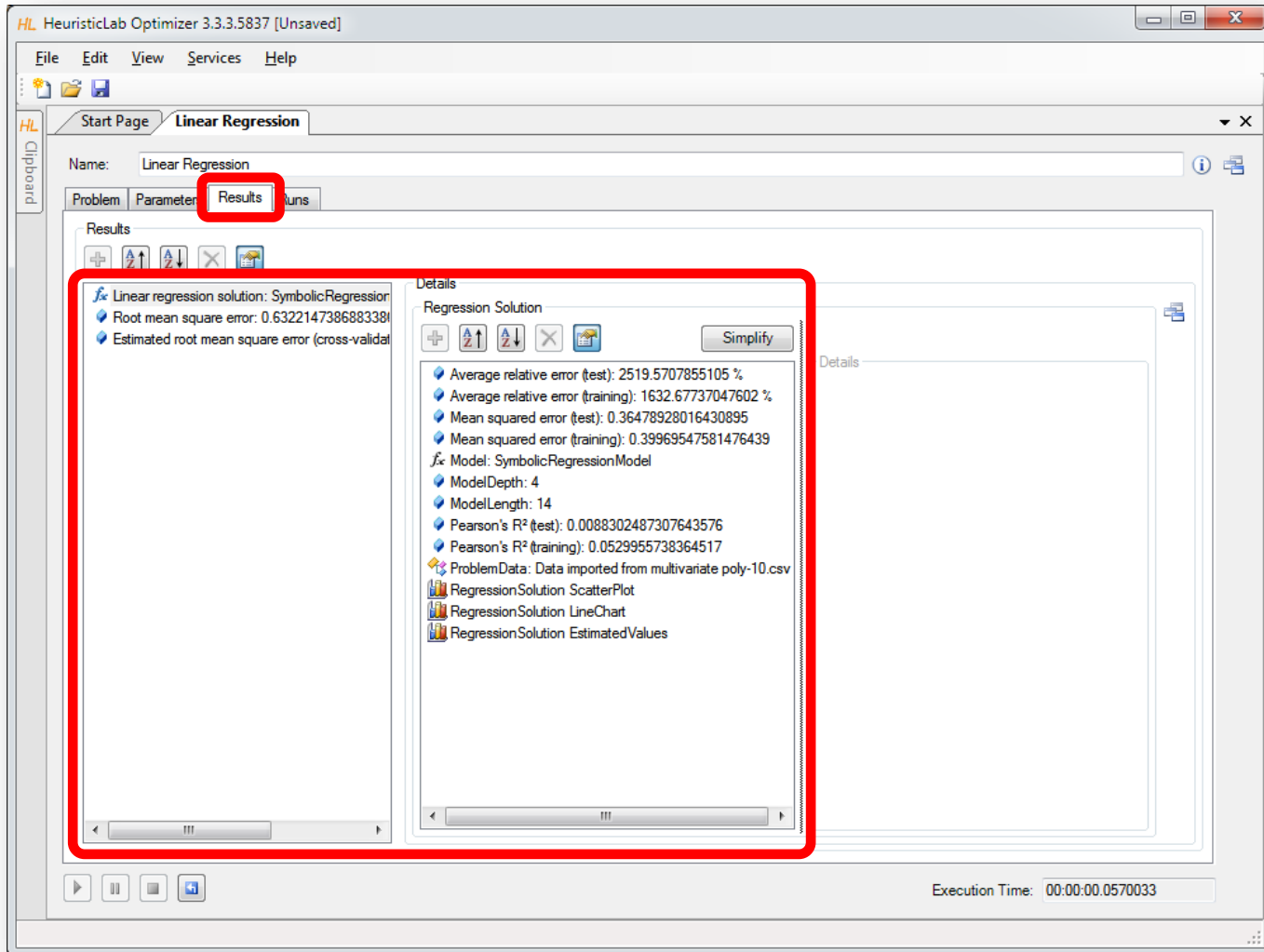
Configure Training and Test Partitions



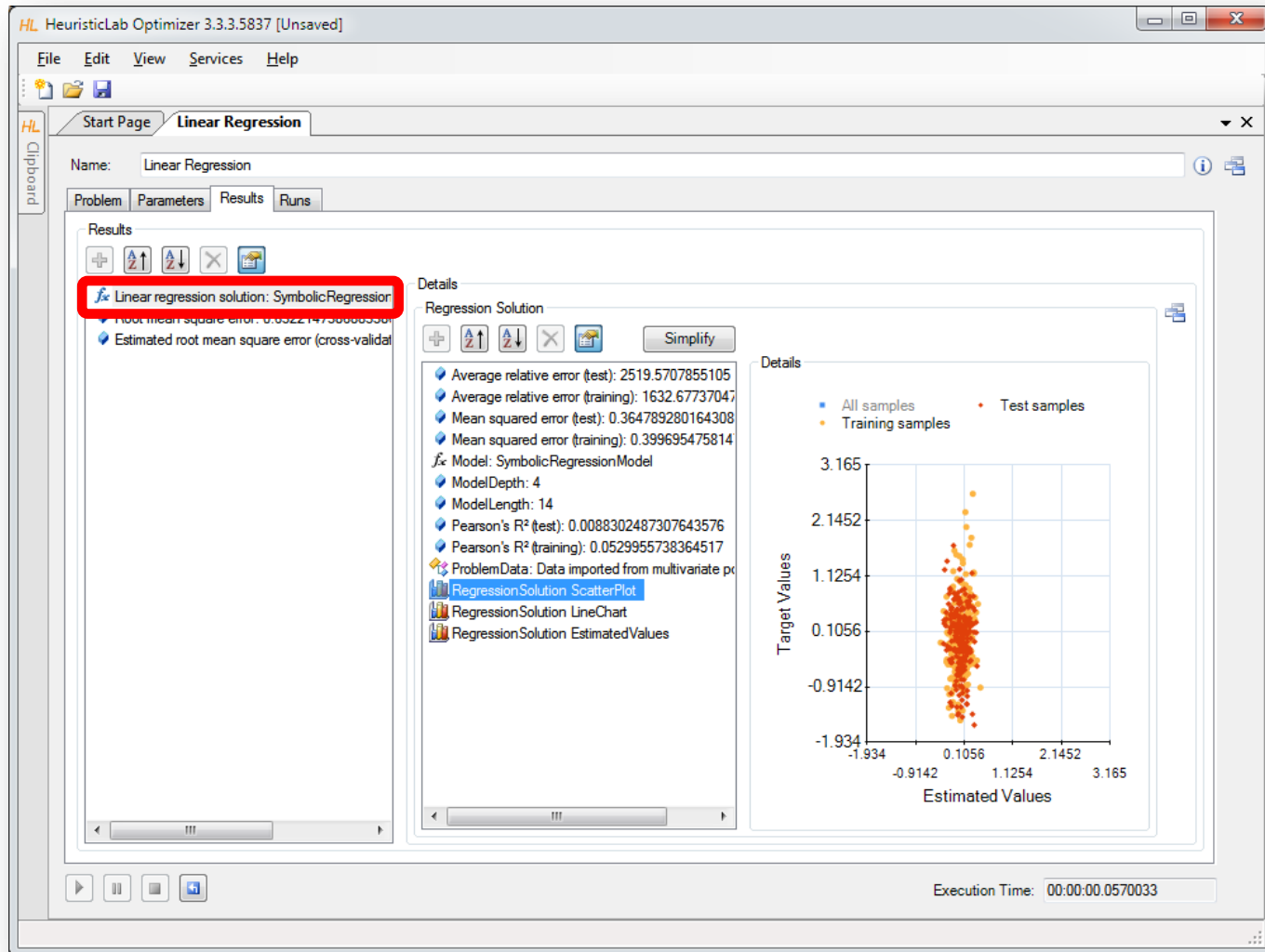
Run Linear Regression



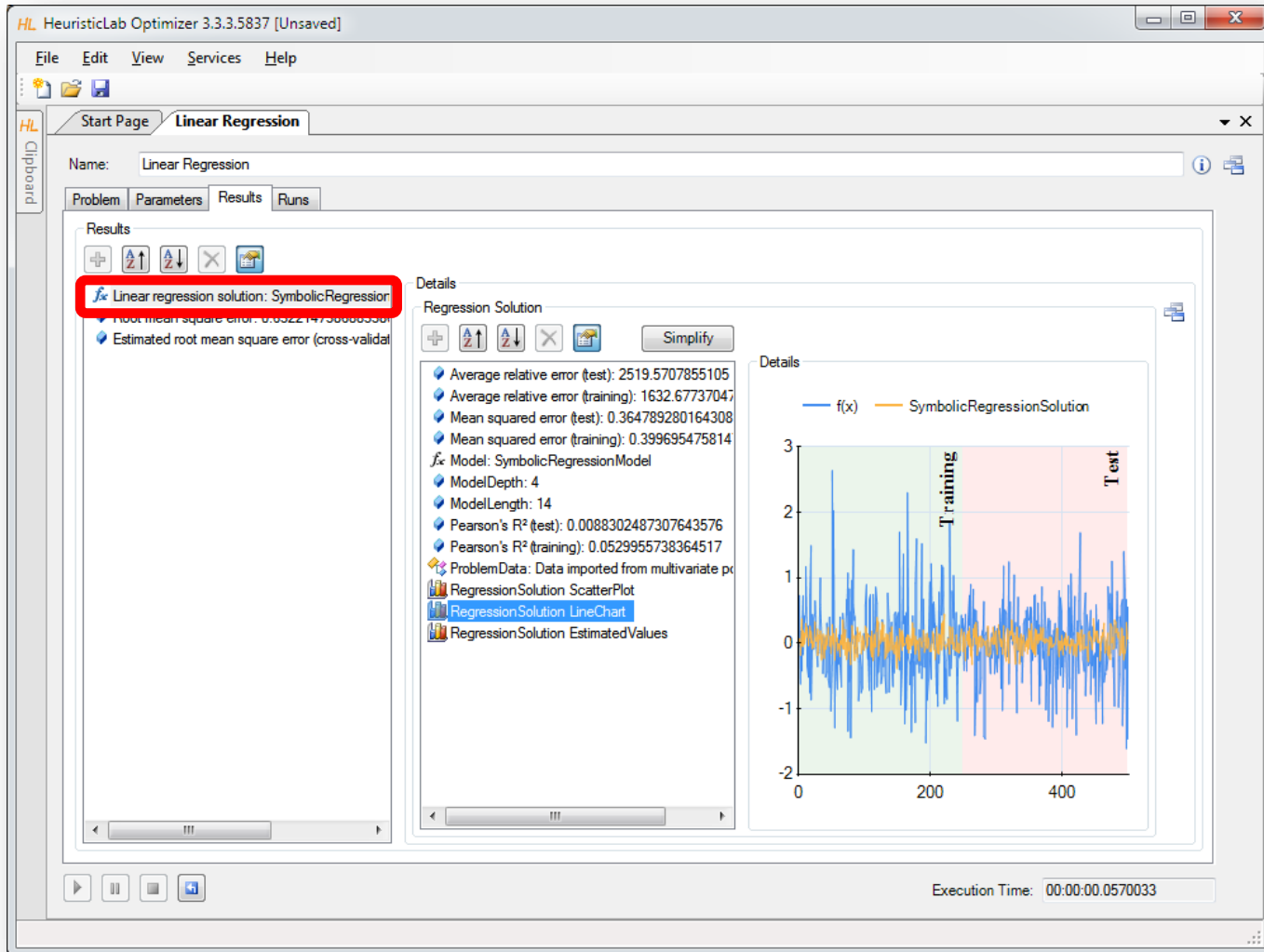
Inspect Results



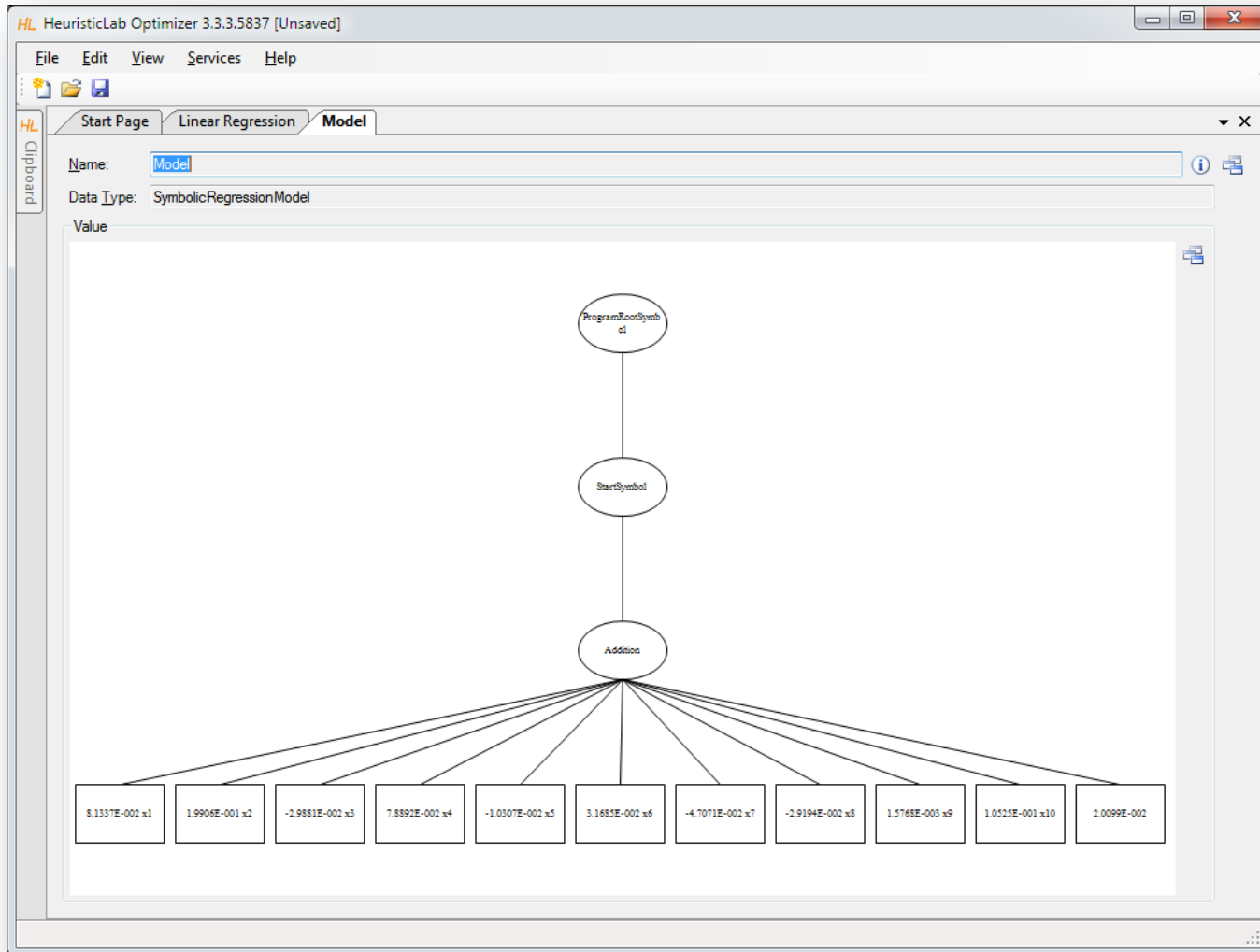
Inspect Scatterplot of Predicted and Target Values



Inspect Linechart

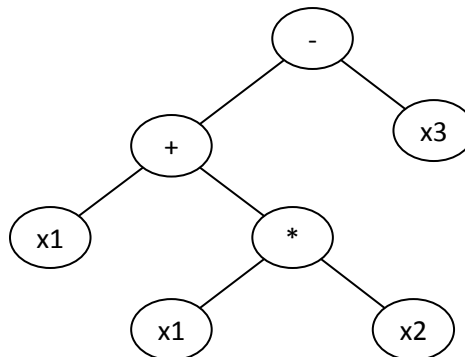


Inspect Graphical Representation of Model



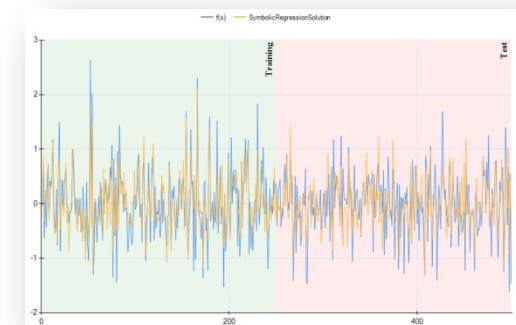
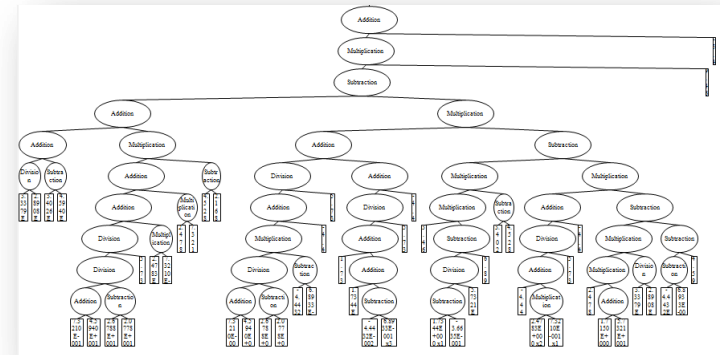
Symbolic Regression with HeuristicLab

- Linear regression produced an inaccurate model.
- Next: produce a nonlinear symbolic regression model using genetic programming
- Genetic programming
 - evolve variable-length models
 - model representation: symbolic expression tree
 - structure and model parameters are evolved side-by-side
 - white-box models

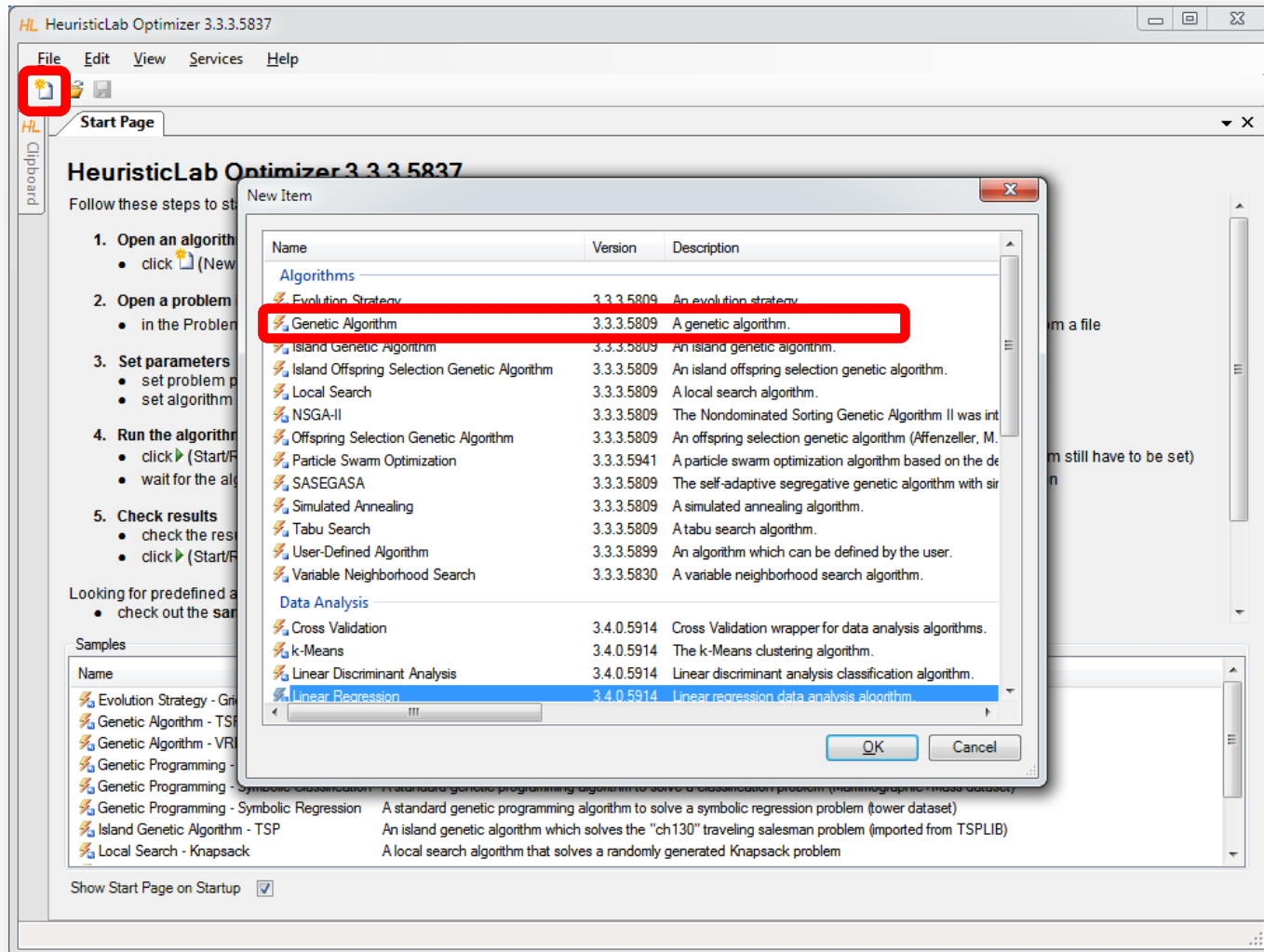


Symbolic Regression with HeuristicLab

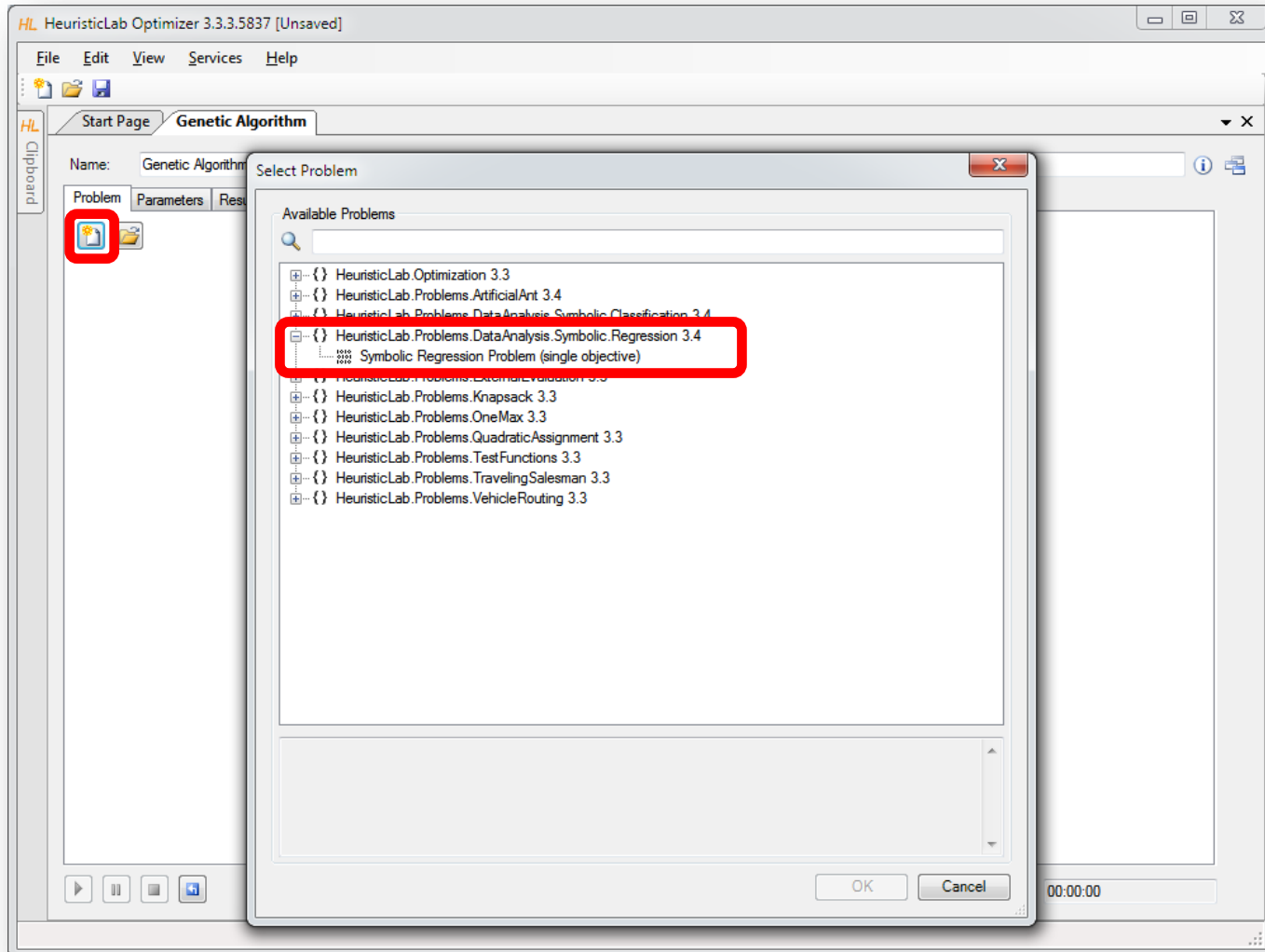
- Demonstration
 - problem configuration
 - function set and terminal set
 - model size constraints
 - Evaluation
- Algorithm configuration
 - selection
 - Mutation
- Analysis of results
 - model accuracy
 - model structure and parameters



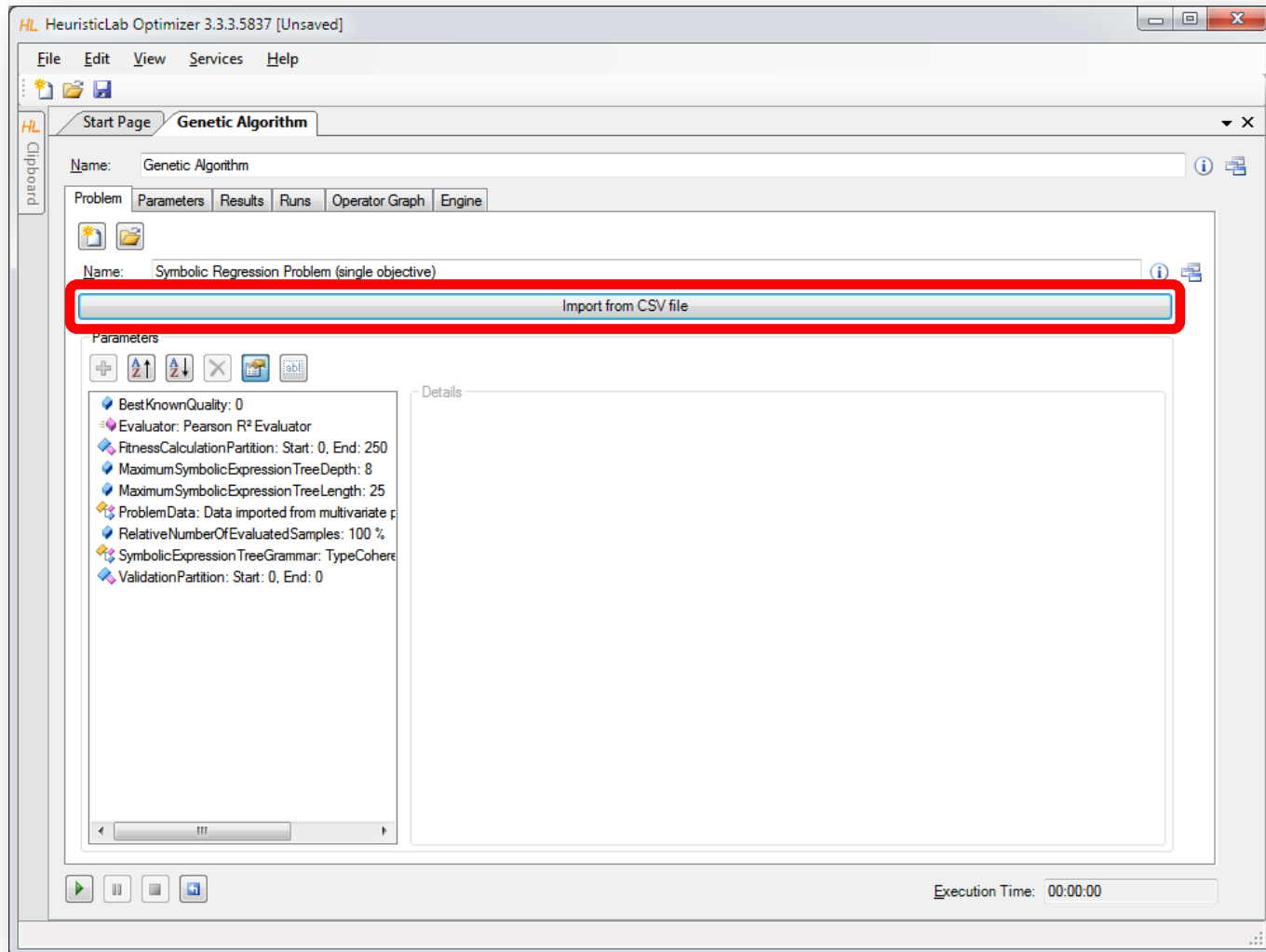
Create New Genetic Algorithm



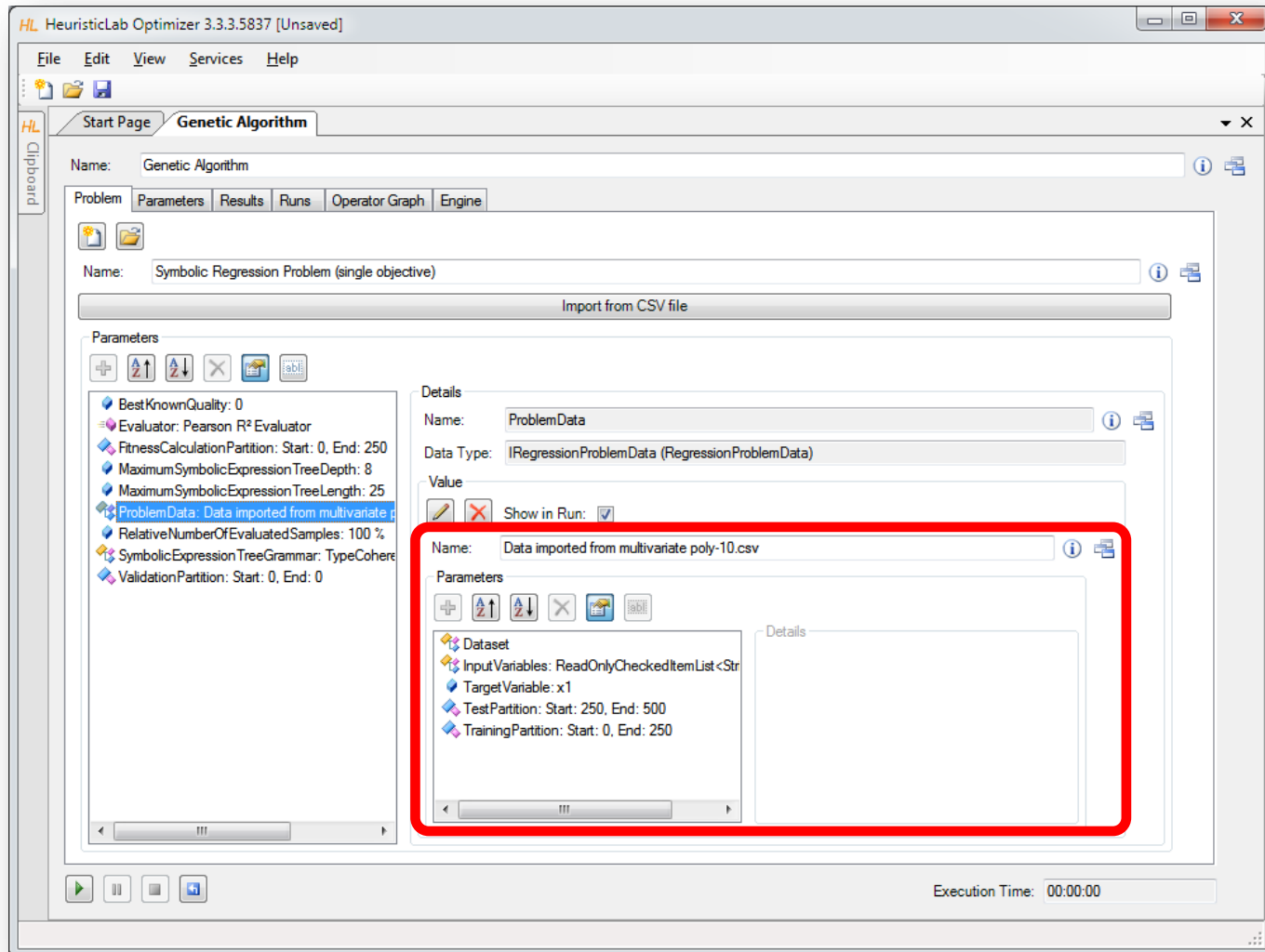
Create New Symbolic Regression Problem



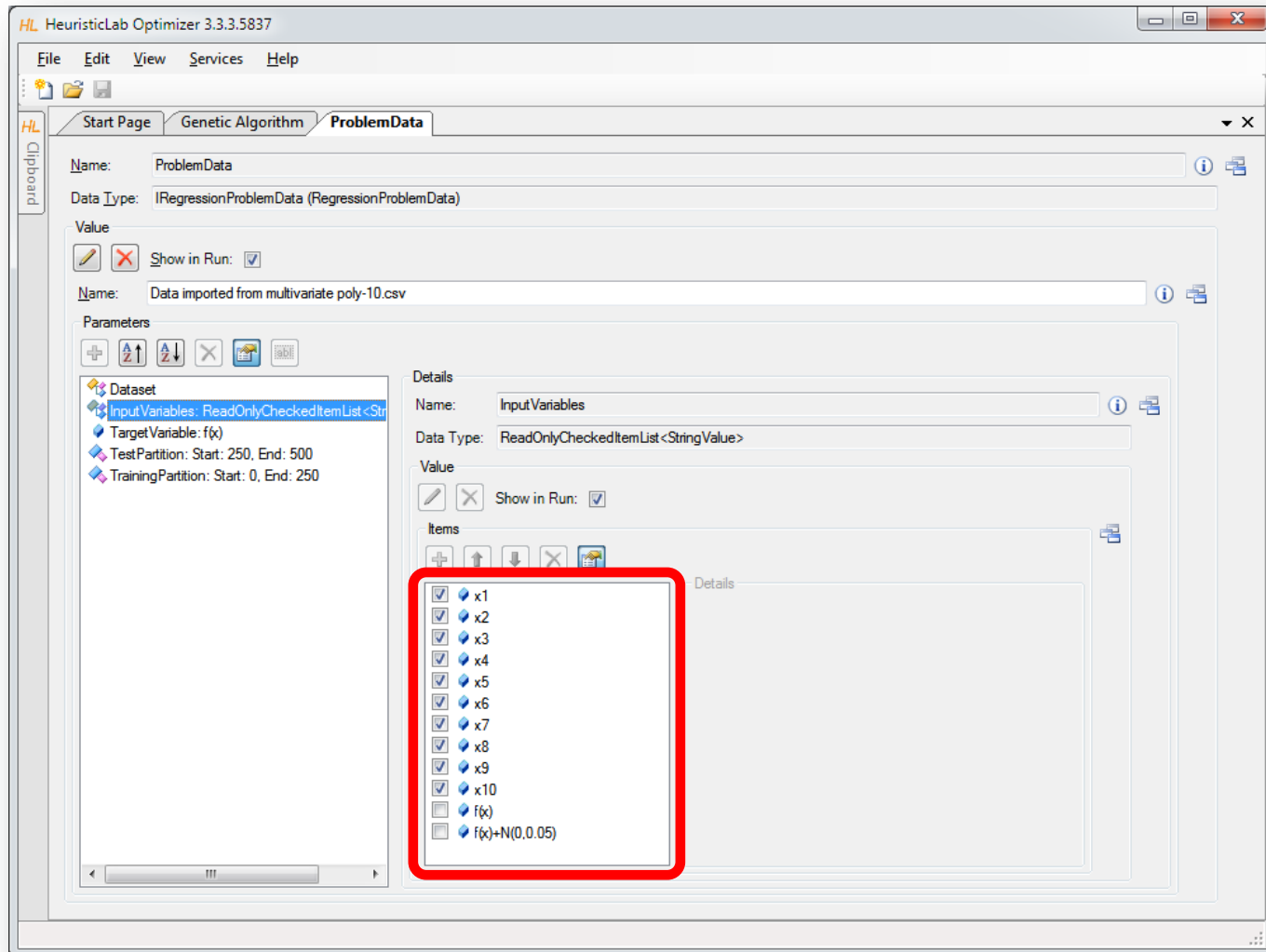
Import Data



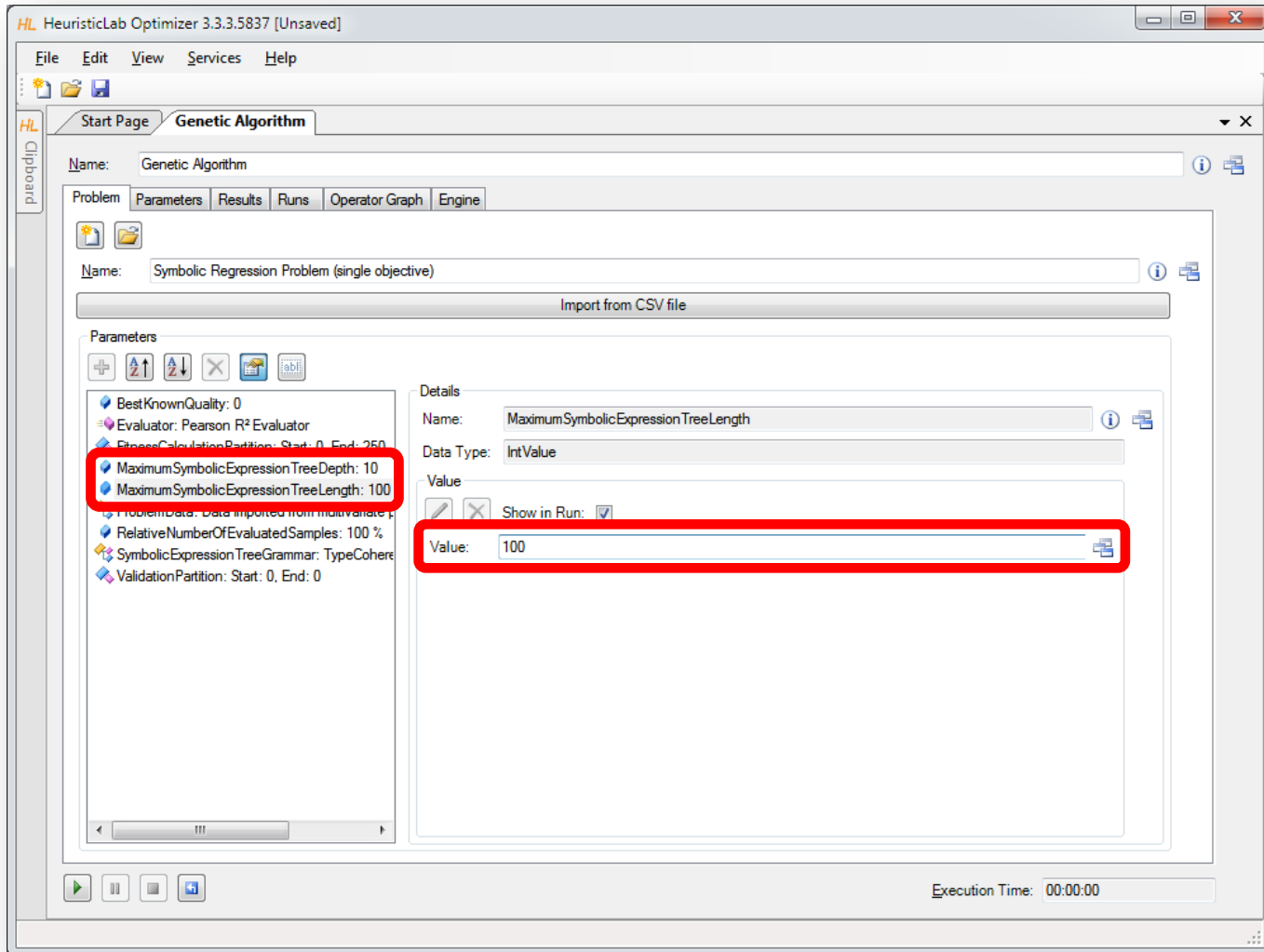
Inspect Data and Configure Dataset



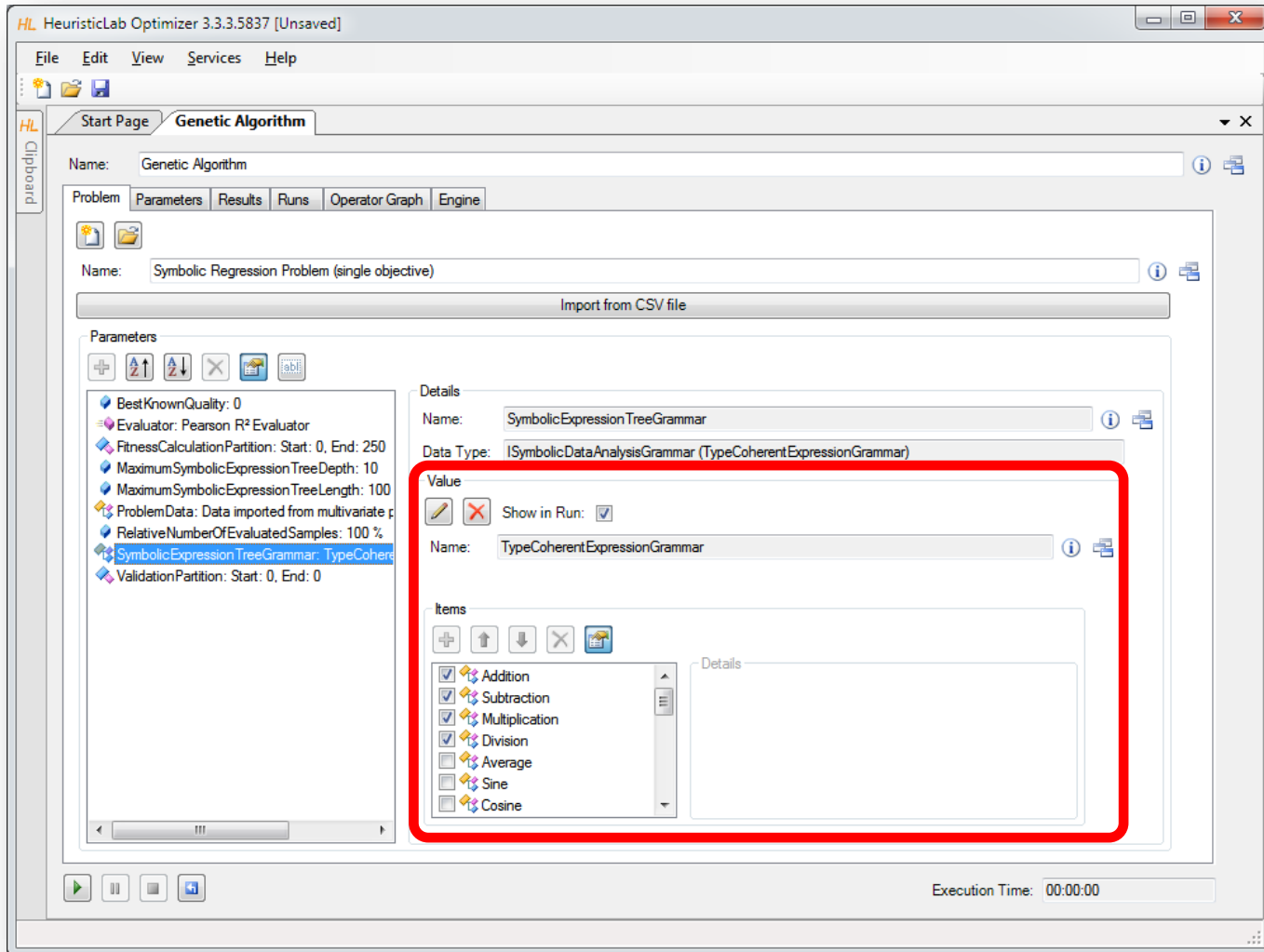
Set Target and Input Variables



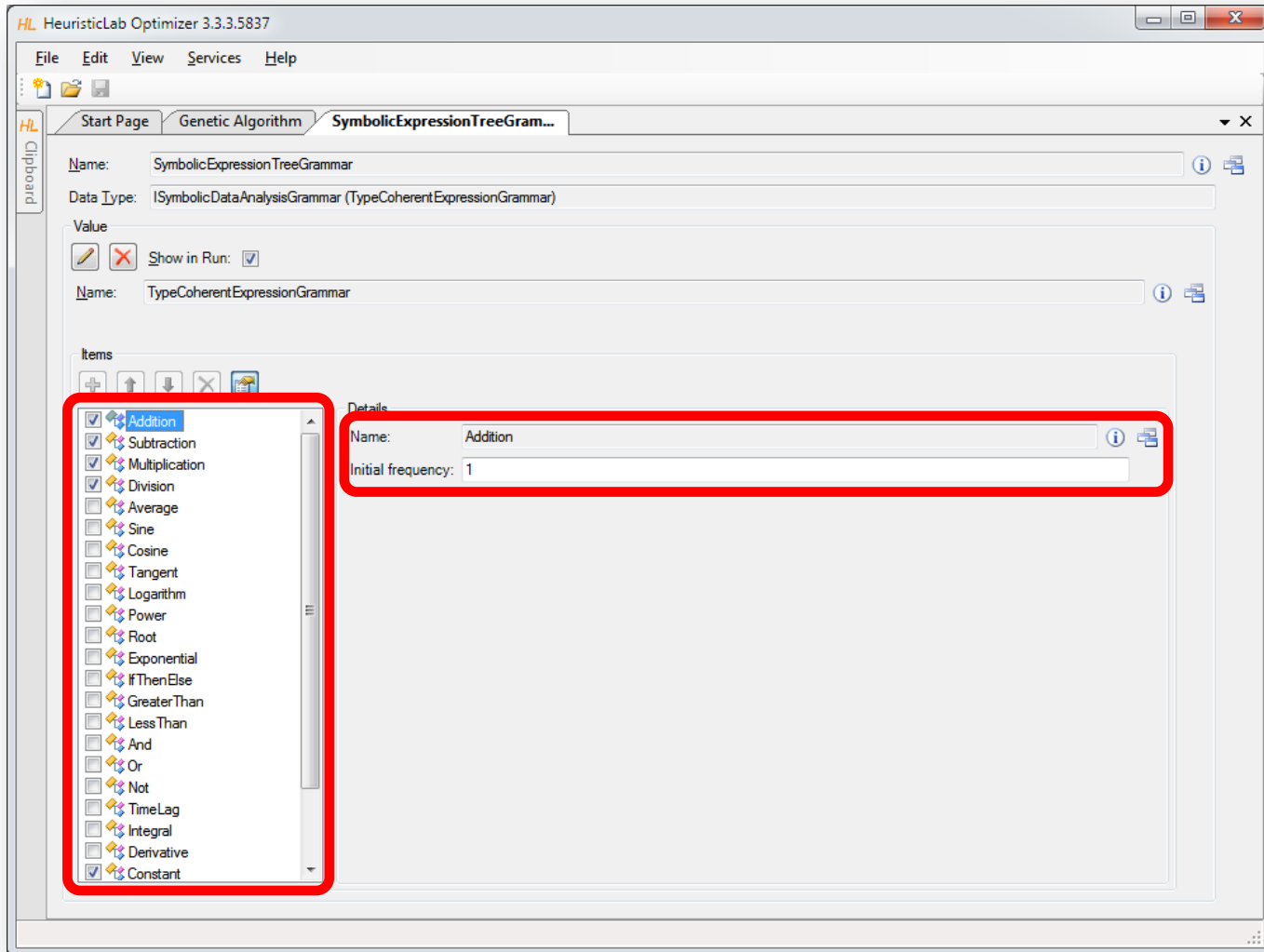
Configure Maximal Model Depth and Length



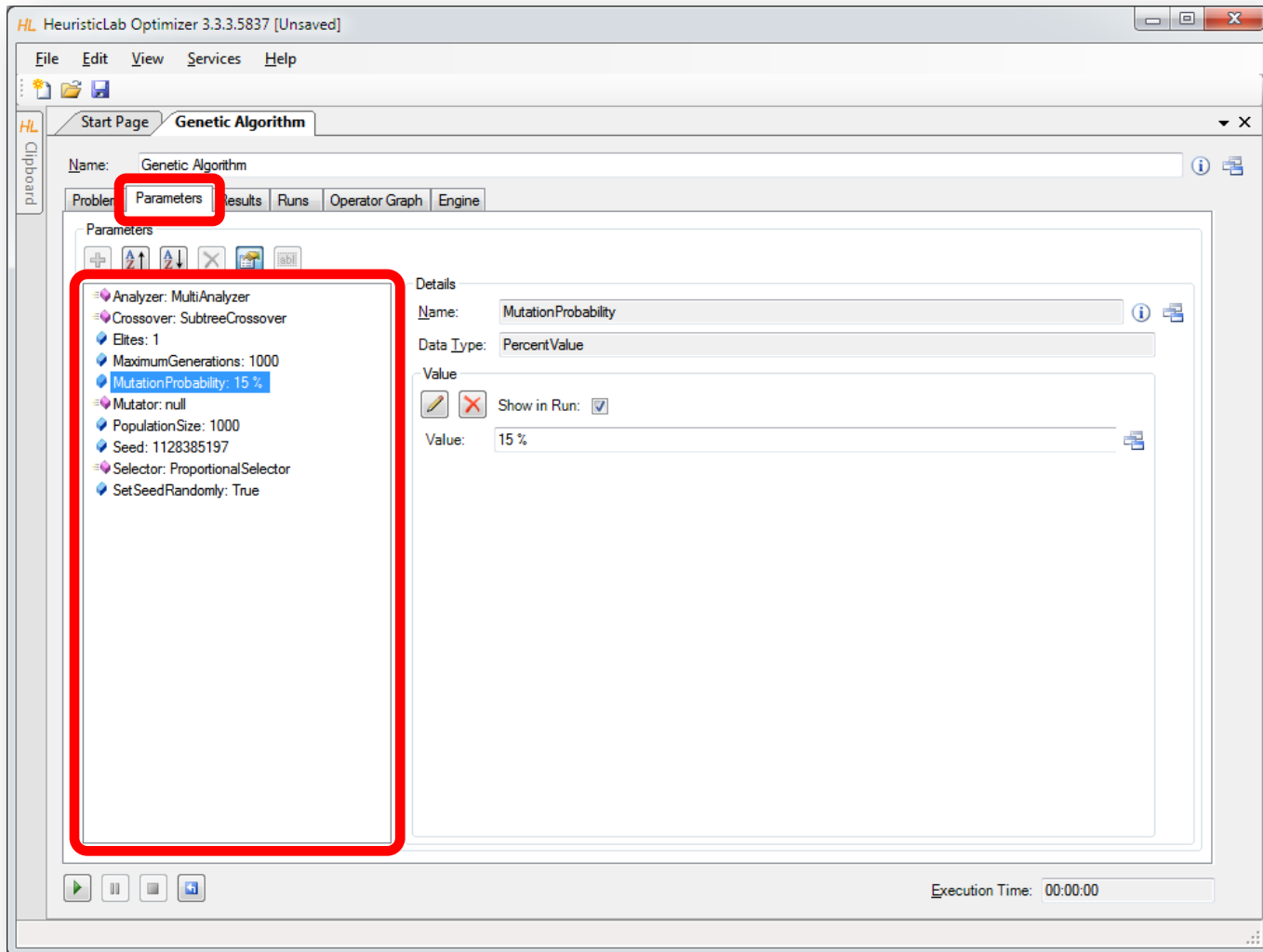
Configure Function Set (Grammar)



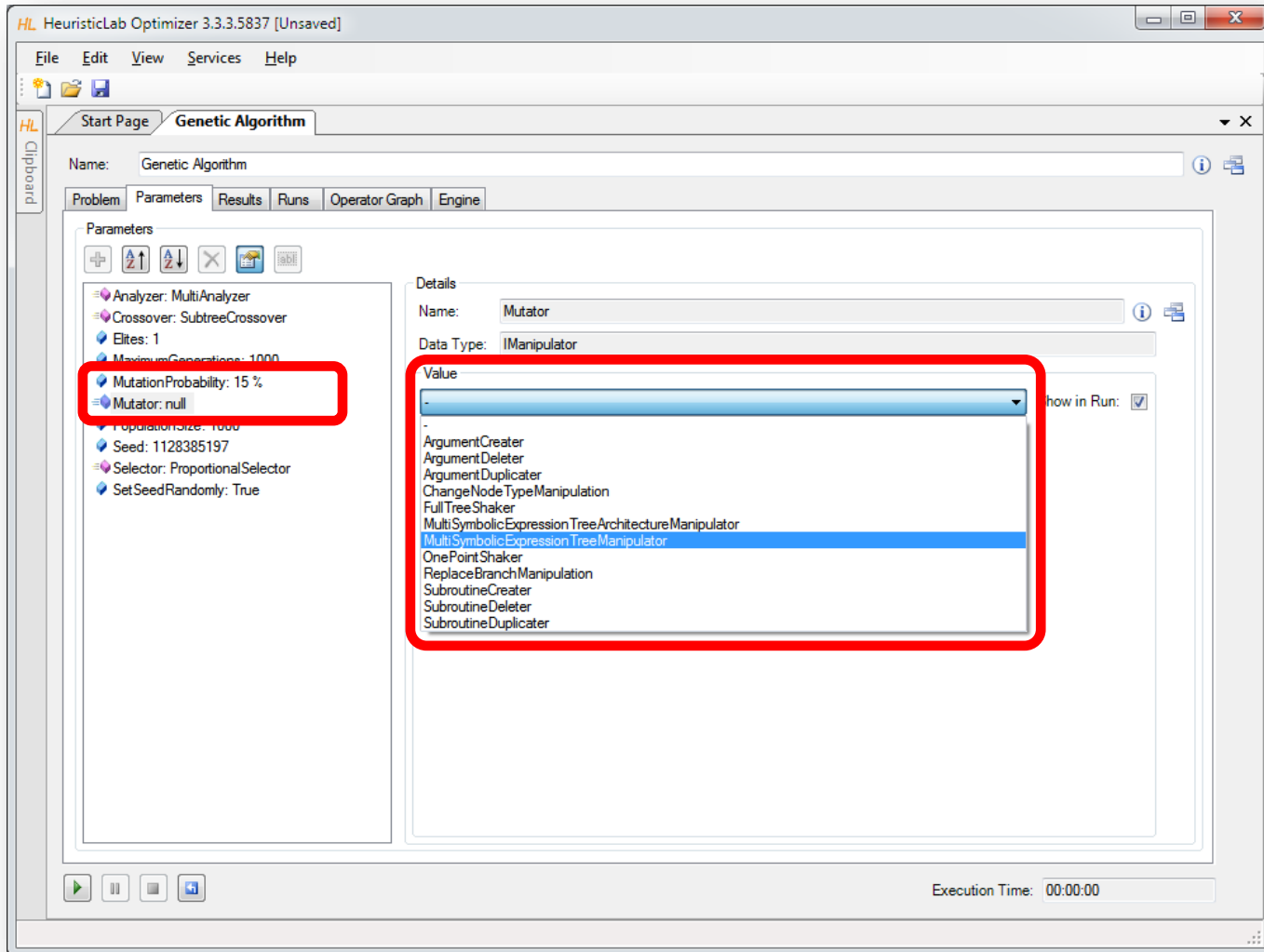
Configure Function Set (Grammar)



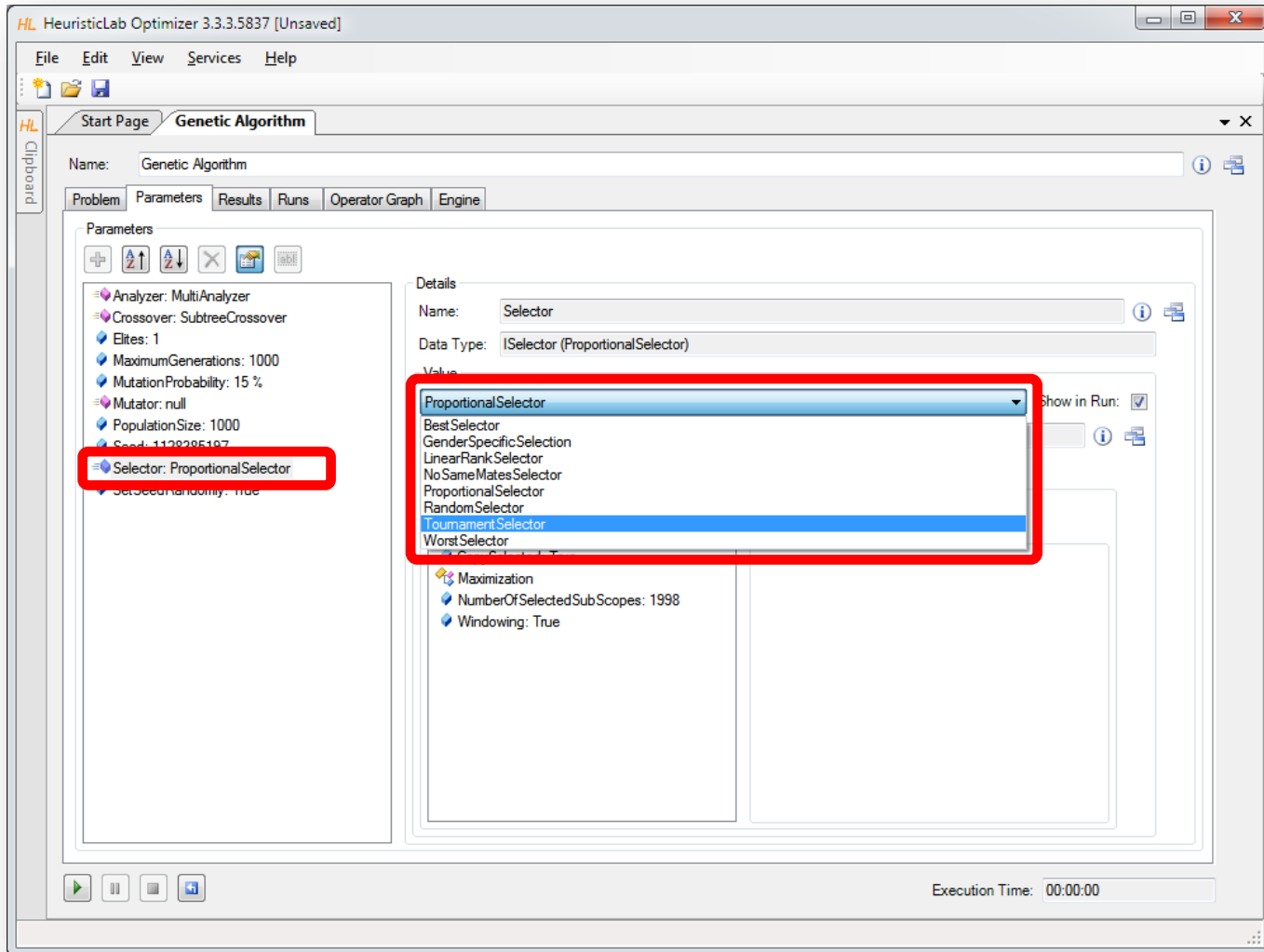
Configure Algorithm Parameters



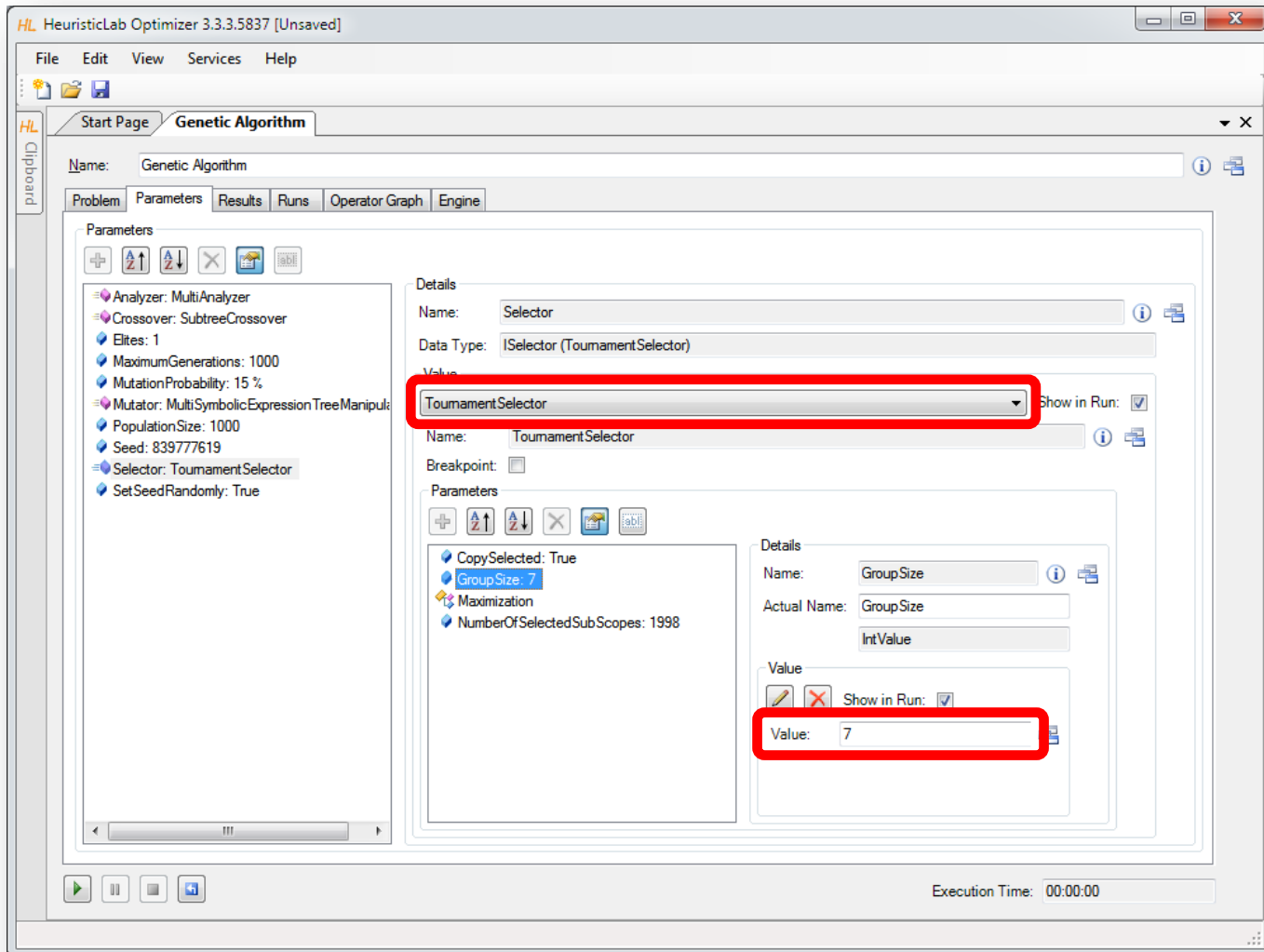
Configure Mutation Operator



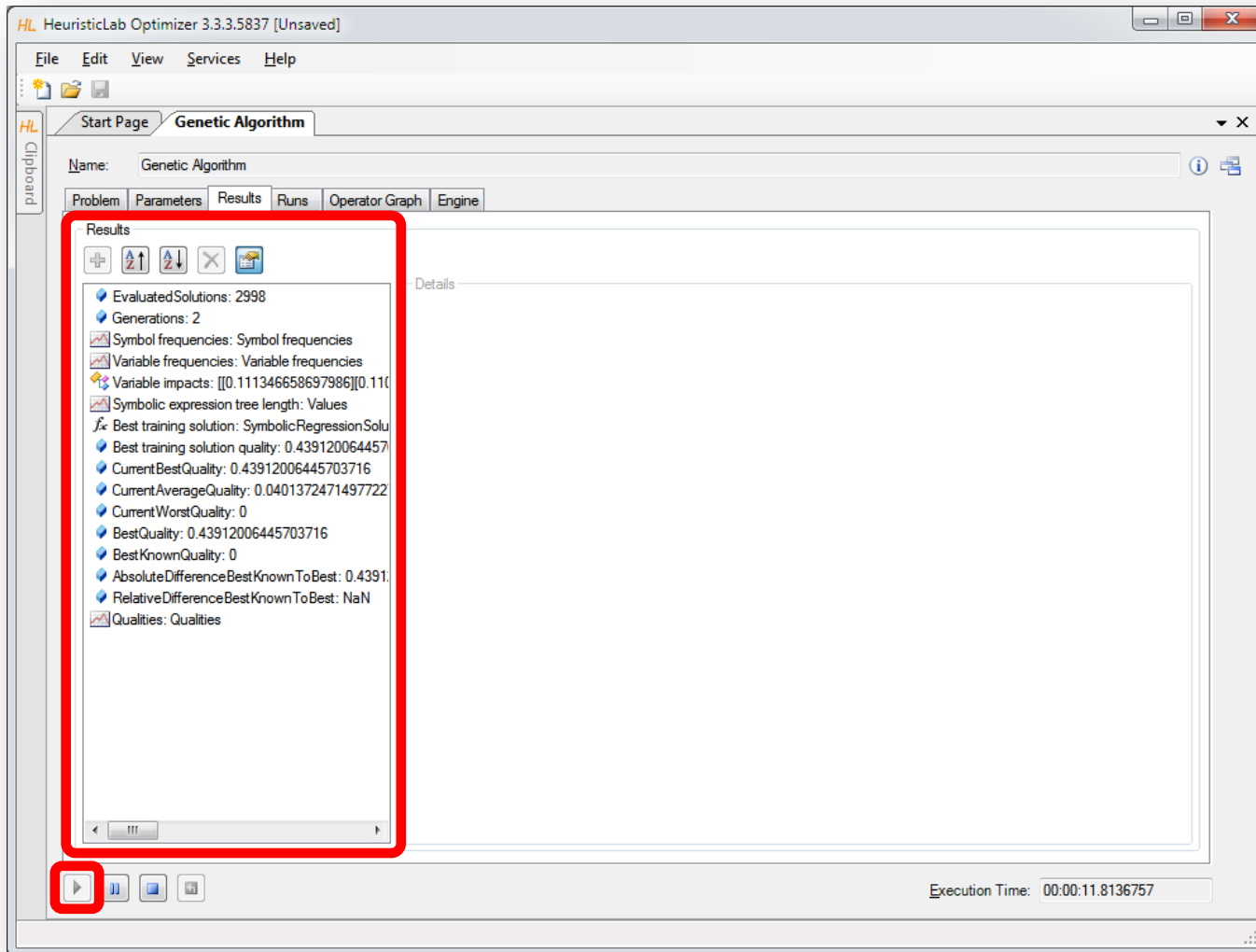
Configure Selection Operator



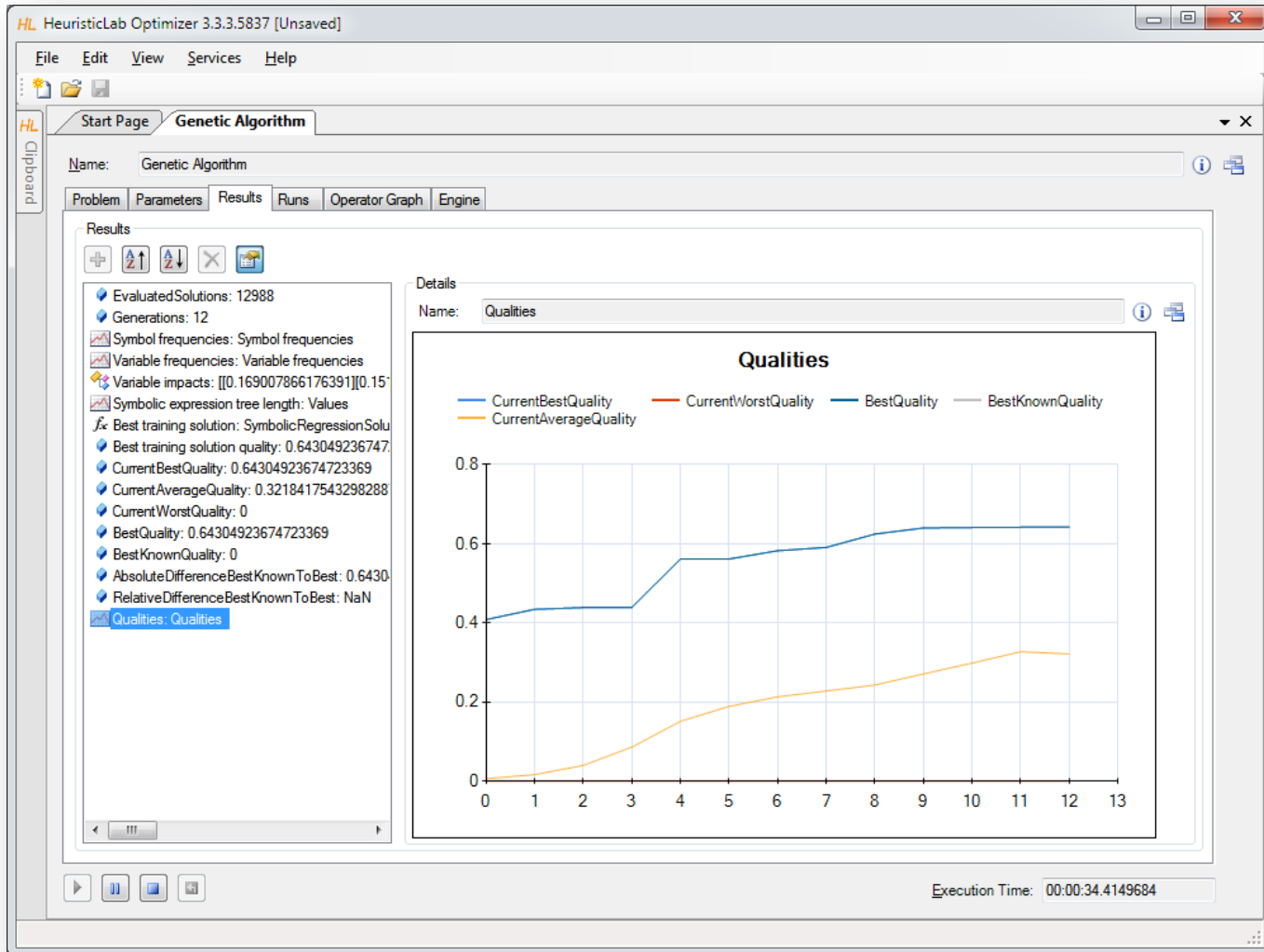
Configure Tournament Group Size



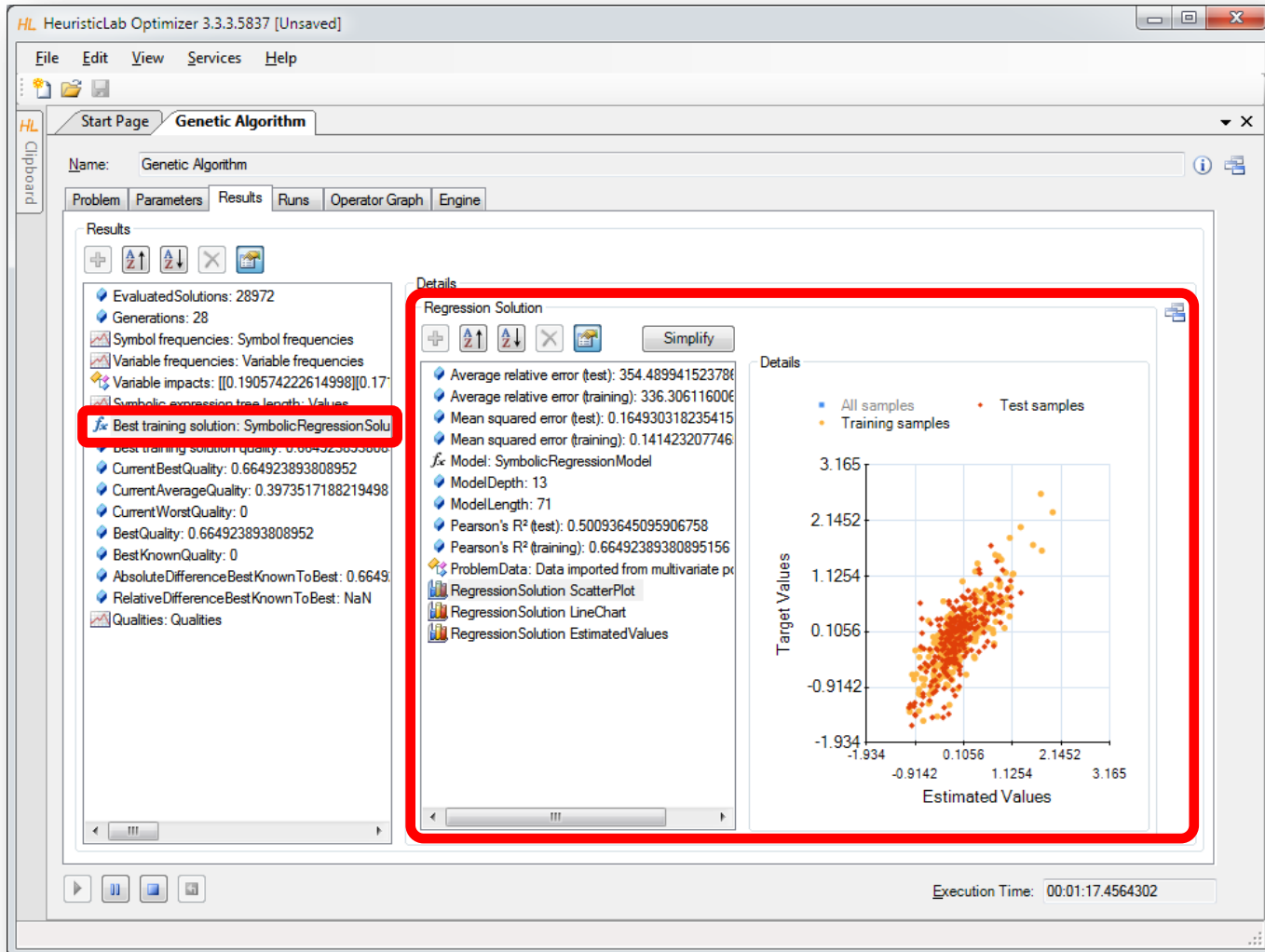
Start Algorithm and Inspect Results



Inspect Quality Chart



Inspect Best Model on Training Partition



The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Genetic Algorithm" tab is active, showing the "Results" section. The "Best training solution" is highlighted in red, indicating the current best model. The "Regression Solution" details are also visible, including a scatter plot of Target Values versus Estimated Values. The scatter plot shows a positive correlation between the estimated and target values, with training samples (orange dots) and test samples (blue dots) plotted. The x-axis is labeled "Estimated Values" and the y-axis is labeled "Target Values".

HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

Start Page Genetic Algorithm

Name: Genetic Algorithm

Problem Parameters Results Runs Operator Graph Engine

Results

- Evaluated Solutions: 28972
- Generations: 28
- Symbol frequencies: Symbol frequencies
- Variable frequencies: Variable frequencies
- Variable impacts: [[0.190574222614998][0.17...
- Symbolic expression tree length: Values
- Best training solution: SymbolicRegressionSolu**
- Best training solution quality: 0.664923893808952
- Current Best Quality: 0.664923893808952
- Current Average Quality: 0.3973517188219498
- Current Worst Quality: 0
- Best Quality: 0.664923893808952
- Best Known Quality: 0
- Absolute Difference Best Known To Best: 0.6649...
- Relative Difference Best Known To Best: NaN
- Qualities: Qualities

Details

Regression Solution

- Average relative error (test): 354.489941523788
- Average relative error (training): 336.306116006
- Mean squared error (test): 0.164930318235415
- Mean squared error (training): 0.141423207746
- Model: SymbolicRegressionModel
- Model Depth: 13
- Model Length: 71
- Pearson's R² (test): 0.50093645095906758
- Pearson's R² (training): 0.66492389380895156
- Problem Data: Data imported from multivariate p...
- Regression Solution ScatterPlot
- Regression Solution LineChart
- Regression Solution EstimatedValues

Details

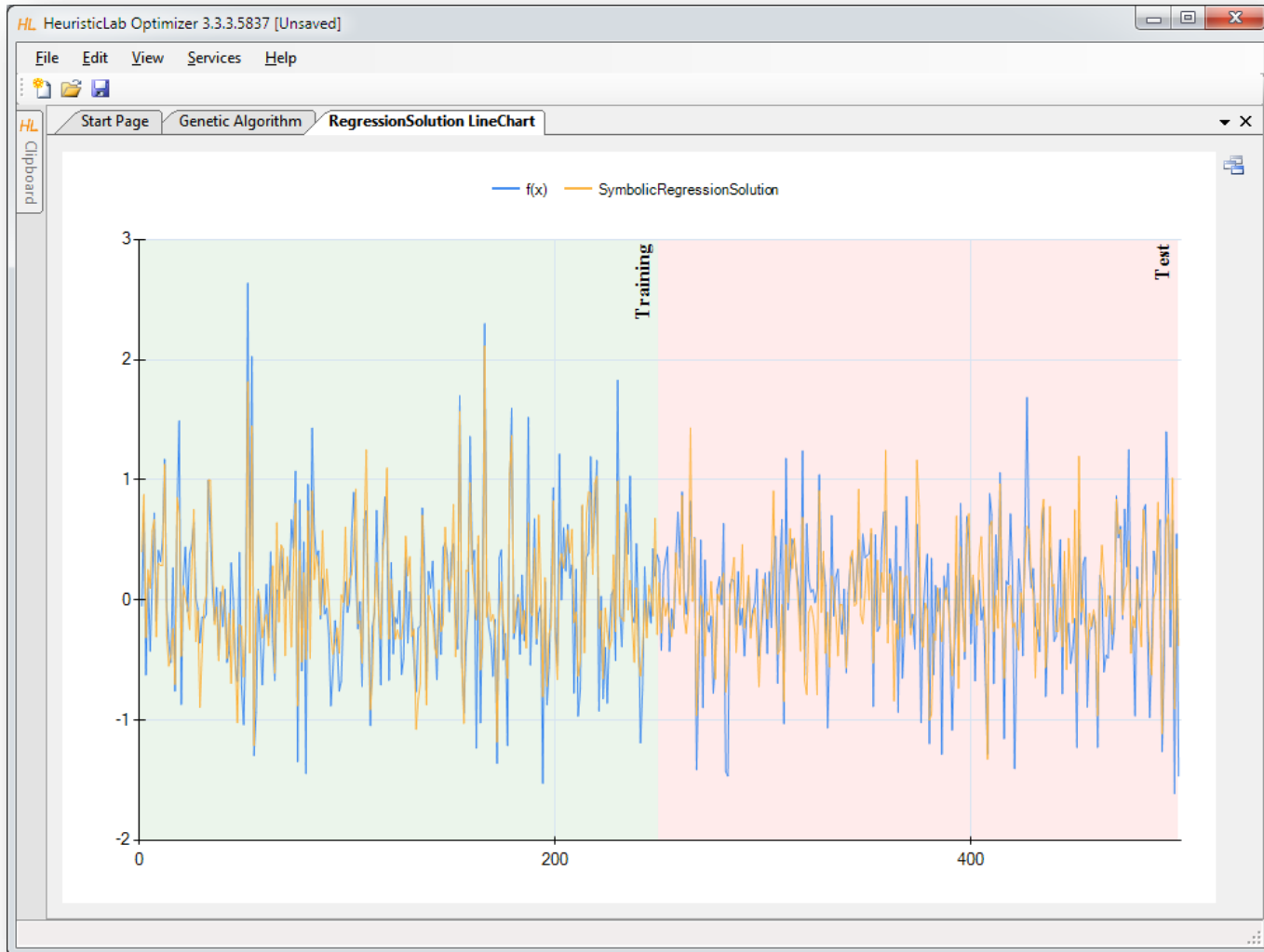
Target Values

Estimated Values

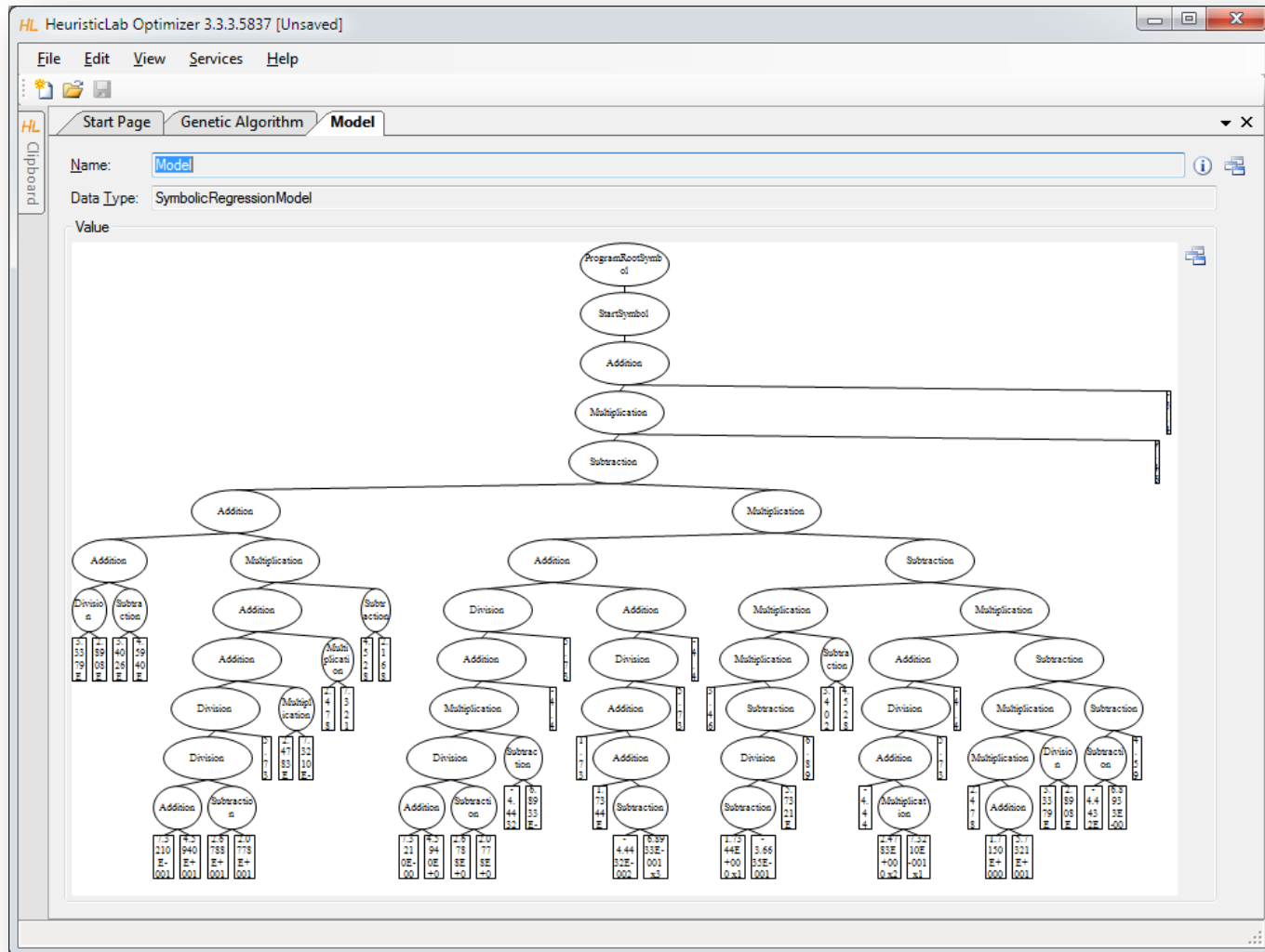
Legend: All samples (blue square), Test samples (red circle), Training samples (orange circle)

Execution Time: 00:01:17.4564302

Inspect Linechart of Best Model on Training Partition

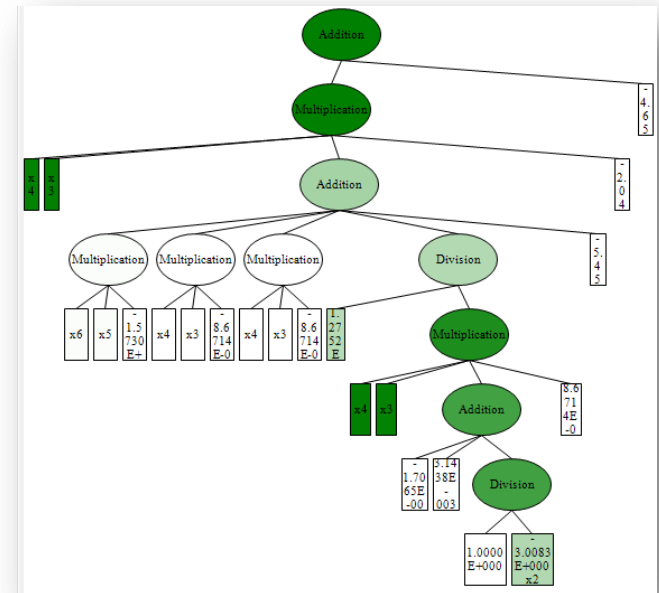


Inspect Structure of Best Model on Training Partition



Model Simplification and Export

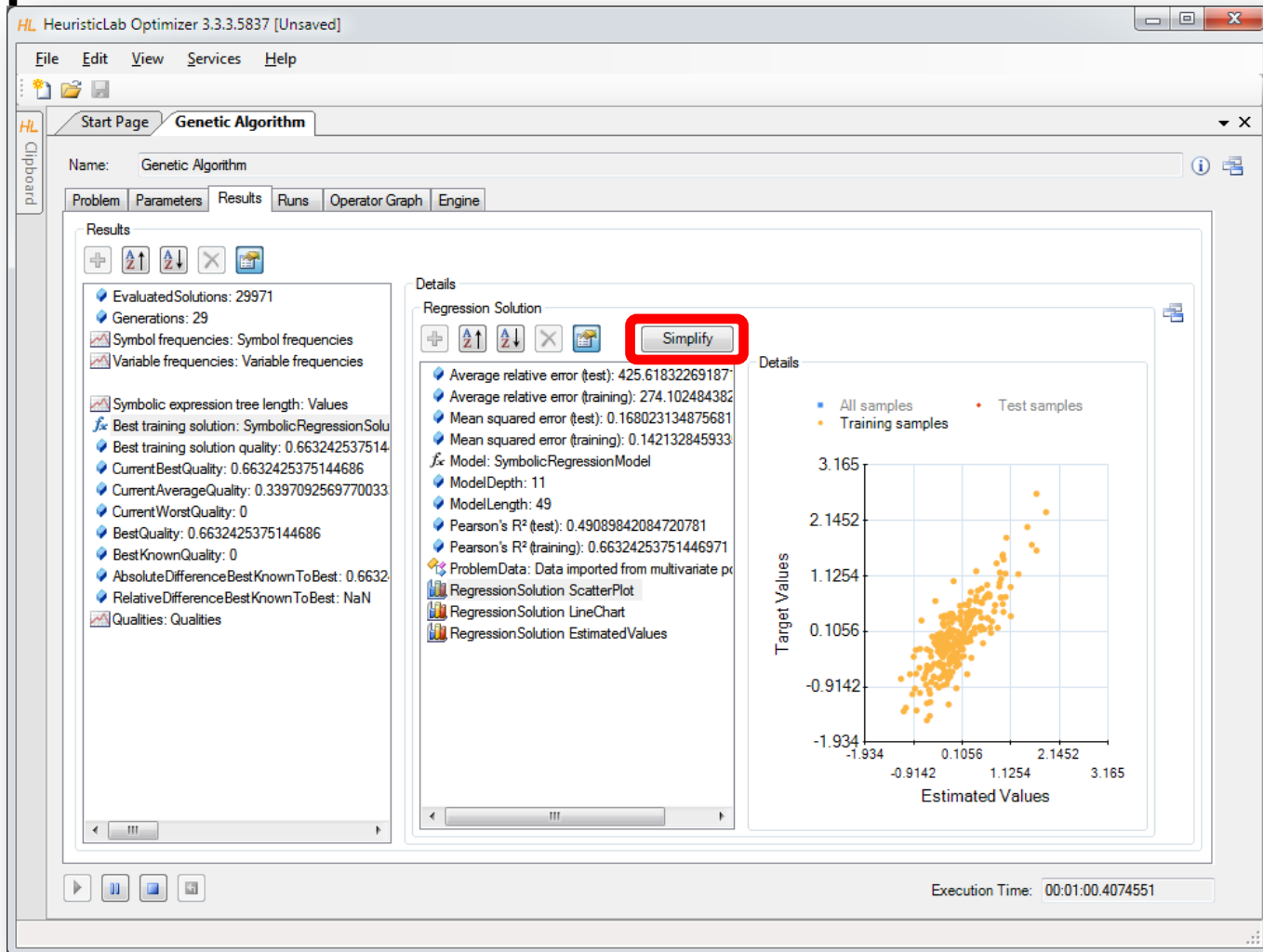
- Demonstration
 - automatic simplification
 - visualization of node impacts
 - manual simplification
 - online update of results
 - model export
 - MATLAB
 - LaTeX



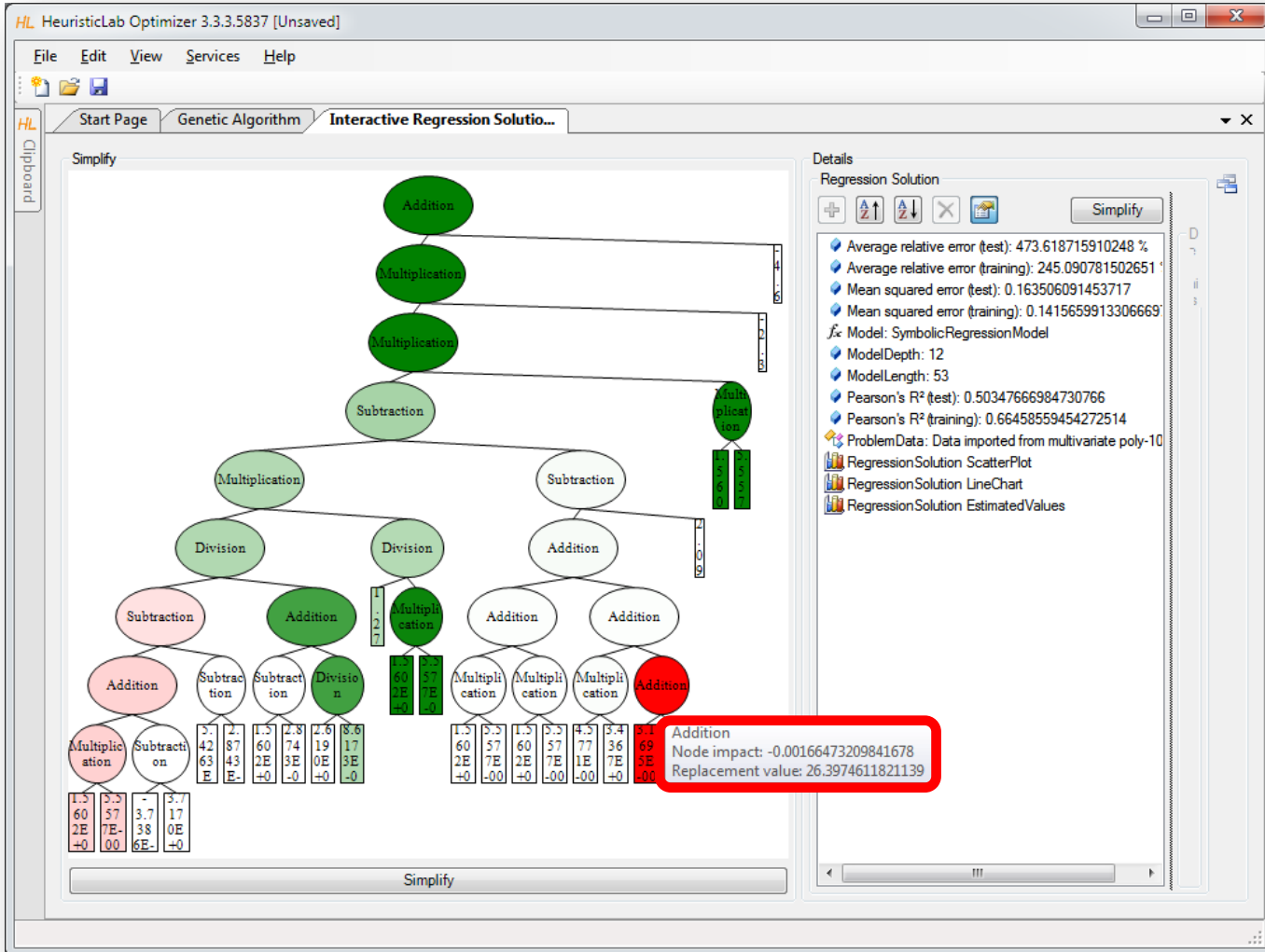
$$Result = x4(t) \cdot x3(t) \cdot c_{20} \quad (13)$$

$$\cdot \left(x6(t) \cdot x5(t) \cdot c_4 + x4(t) \cdot x3(t) \cdot c_7 + x4(t) \cdot x3(t) \cdot c_{10} + \frac{c_{11}x1(t)}{x4(t) \cdot x3(t) \cdot \left(c_{14}x4(t) + c_{15}x5(t) + \frac{1}{c_{17}x2(t)} \right) \cdot c_{18}} + c_{19} \right) + c_{21} \quad (14)$$

Detailed Model Analysis and Simplification



Symbolic Simplification and Node Impacts



The screenshot displays the HeuristicLab Optimizer interface. The main window shows an "Interactive Regression Solution" with a symbolic tree structure. The tree consists of nodes representing mathematical operations: Addition, Multiplication, Subtraction, and Division. The nodes are color-coded: green for operations that are part of the current solution, and pink for operations that are not. A red box highlights a specific node in the tree, which is an "Addition" node. This node is associated with a table of values and a "Node impact" value.

Node Impact Details:

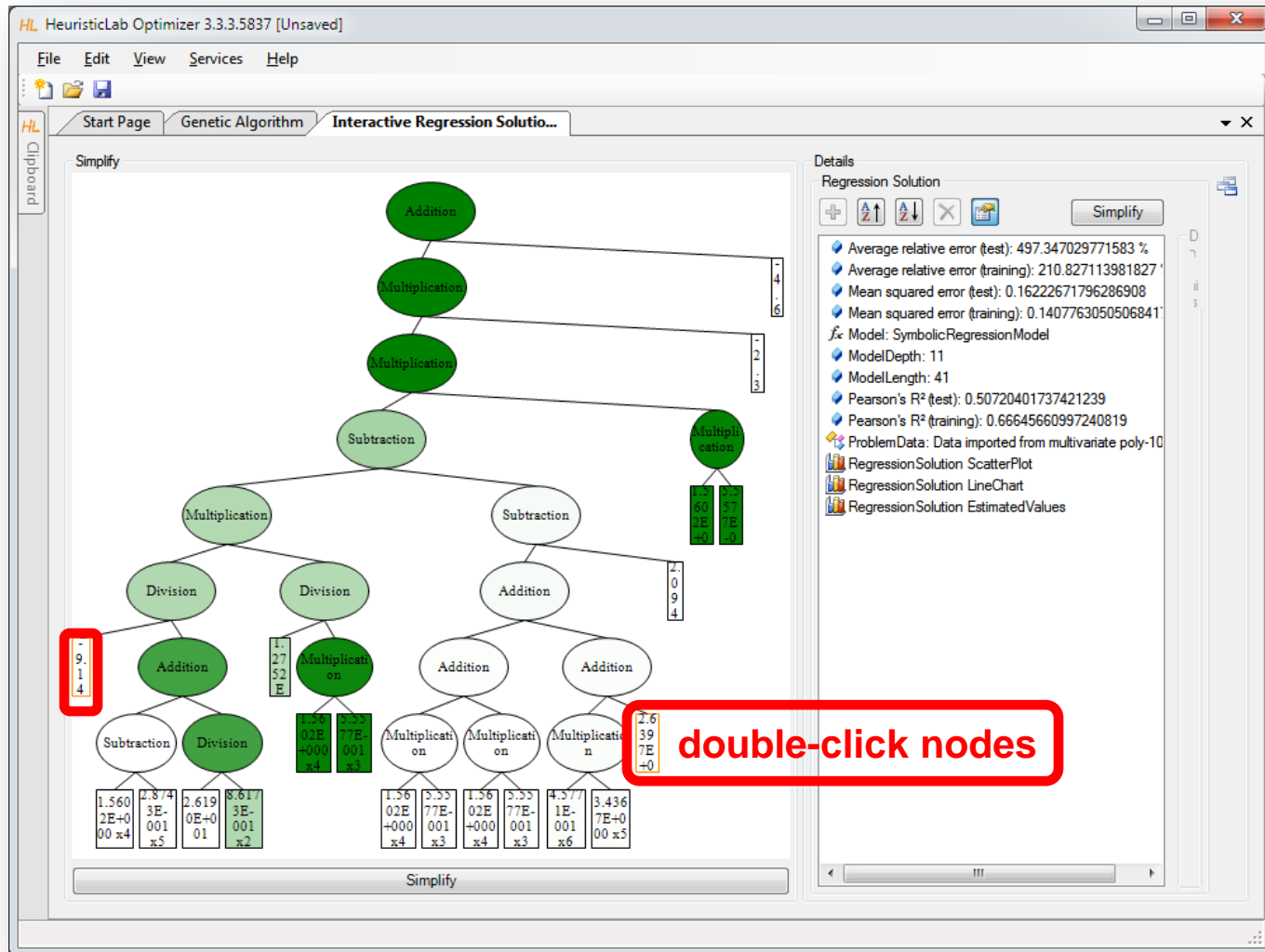
1.5E+00	5.2E+00	-	3.7E+00
6.0E+00	5.7E+00	3.7E+00	1.7E+00
2.2E+00	7.5E+00	3.8E+00	0.0E+00
+0.0E+00	0.0E+00	6.6E+00	+0.0E+00

Node impact: -0.00166473209841678
Replacement value: 26.3974611821139

The "Details" panel on the right provides a "Regression Solution" summary:

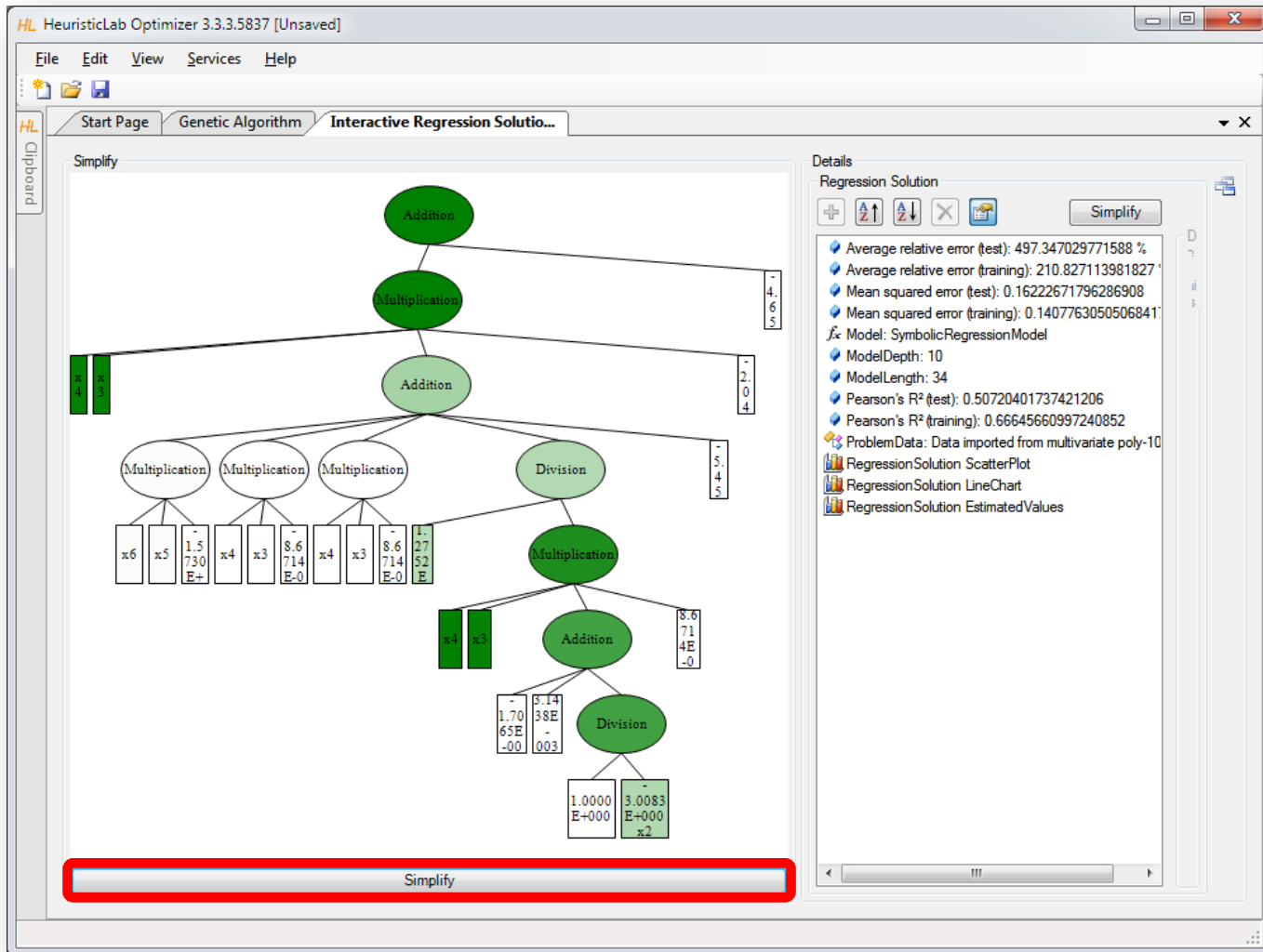
- Average relative error (test): 473.618715910248 %
- Average relative error (training): 245.090781502651 %
- Mean squared error (test): 0.163506091453717
- Mean squared error (training): 0.1415659913306669
- Model: SymbolicRegressionModel
- ModelDepth: 12
- ModelLength: 53
- Pearson's R² (test): 0.50347666984730766
- Pearson's R² (training): 0.66458559454272514
- ProblemData: Data imported from multivariate poly-10
- RegressionSolution ScatterPlot
- RegressionSolution LineChart
- RegressionSolution EstimatedValues

Manual Simplification



The screenshot displays the HeuristicLab Optimizer interface. The main window shows a tree diagram of a regression solution. The tree starts with an 'Addition' node at the top, which branches into 'Multiplication' and 'Subtraction'. The 'Subtraction' node further branches into 'Multiplication' and 'Subtraction'. The 'Multiplication' nodes lead to 'Division' and 'Addition' nodes, which eventually lead to leaf nodes containing numerical values and mathematical expressions like $1.5602E+000 \times 4$ and $4.3771E-001 \times 6$. A red box highlights a node with the value -9.14 . Another red box highlights a node with the value $2.6397E+0$ and the text 'double-click nodes' is written next to it. The 'Details' panel on the right shows the 'Regression Solution' with various metrics: Average relative error (test): 497.347029771583 %, Average relative error (training): 210.827113981827 %, Mean squared error (test): 0.16222671796286908, Mean squared error (training): 0.1407763050506841, Model: SymbolicRegressionModel, ModelDepth: 11, ModelLength: 41, Pearson's R² (test): 0.50720401737421239, and Pearson's R² (training): 0.66645660997240819. The 'Simplify' button is visible in the top right of the details panel.

Automatic Symbolic Simplification



The screenshot displays the HeuristicLab Optimizer interface. The main window shows an "Interactive Regression Solution" with a symbolic regression tree. The tree structure is as follows:

- Root: Addition (Value: -4.65)
- Level 1: Multiplication (Value: -2.04)
- Level 2: Addition (Value: -5.45)
- Level 3: Three Multiplication nodes and one Division node.
- Level 4: Leaf nodes containing numerical values and variables (e.g., x^6 , x^5 , $1.5730E+$, x^4 , x^3 , $8.6714E-0$, x^4 , x^3 , $8.6714E-0$, $1.2752E$).
- Level 5: Multiplication (Value: $8.6714E-0$)
- Level 6: Addition (Value: $1.7065E-00$, $5.1488E-003$)
- Level 7: Division (Value: $1.0000E+000$, $3.0083E+000$, x^2)

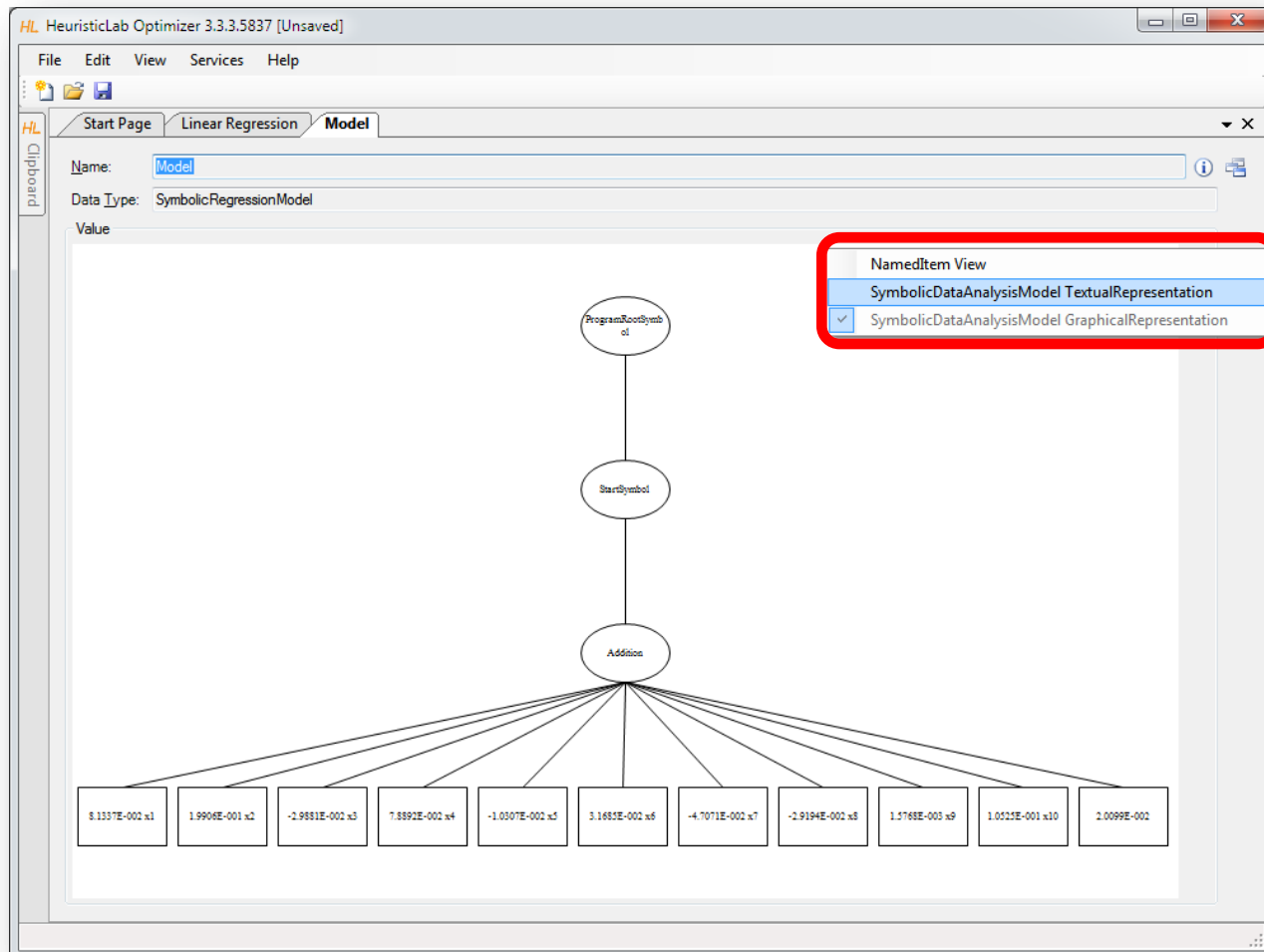
The "Details" panel on the right provides the following information:

- Regression Solution
- Average relative error (test): 497.347029771588 %
- Average relative error (training): 210.827113981827 %
- Mean squared error (test): 0.16222671796286908
- Mean squared error (training): 0.1407763050506841
- Model: SymbolicRegressionModel
- ModelDepth: 10
- ModelLength: 34
- Pearson's R² (test): 0.50720401737421206
- Pearson's R² (training): 0.66645660997240852
- ProblemData: Data imported from multivariate poly-10
- RegressionSolution ScatterPlot
- RegressionSolution LineChart
- RegressionSolution EstimatedValues

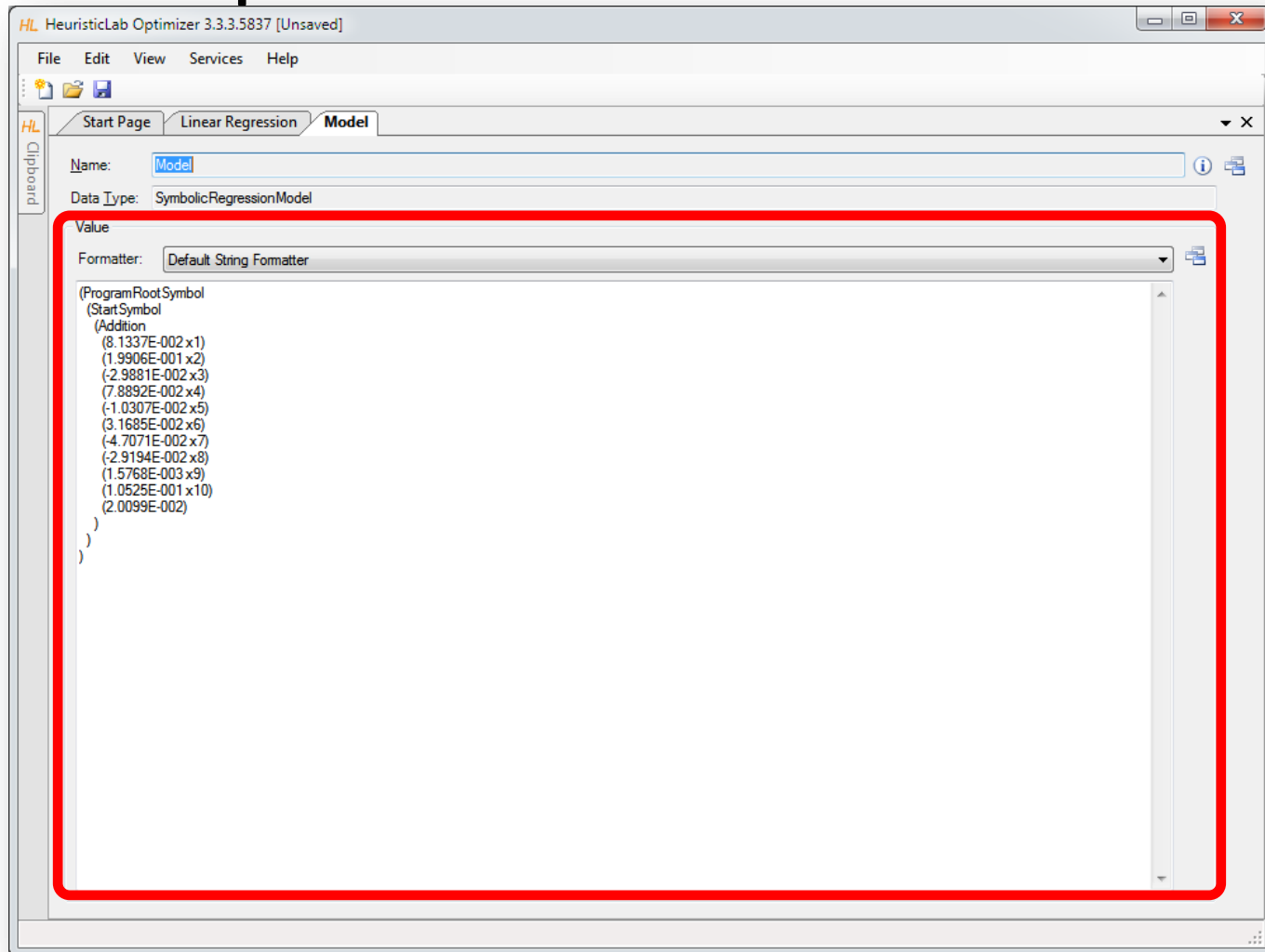
A red box highlights the "Simplify" button at the bottom of the main window.

Textual Representations Are Also Available

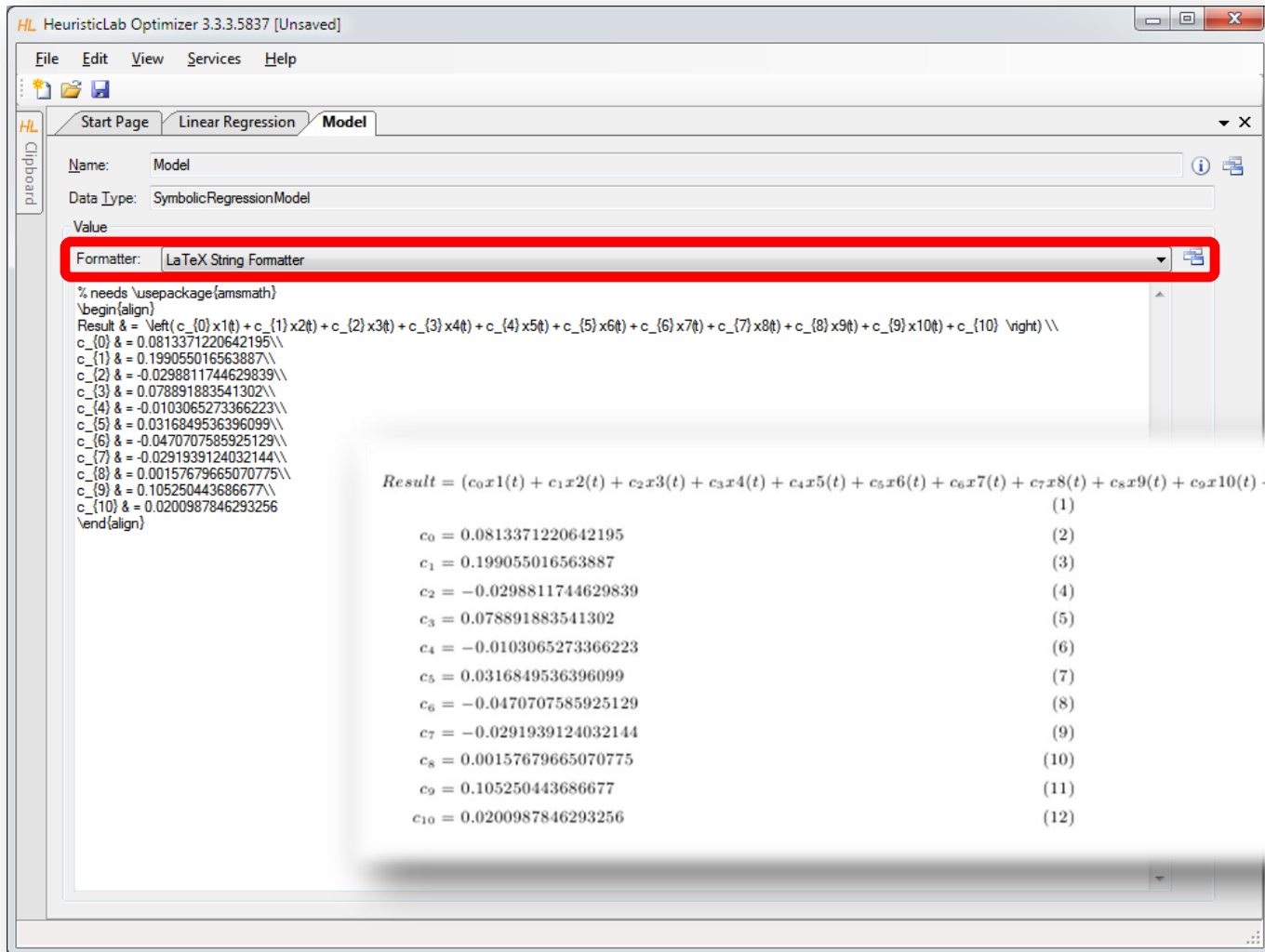
- Use *ViewHost* to switch to textual representation view.



Default Textual Representation for Model Export



Textual Representation for Export to LaTeX



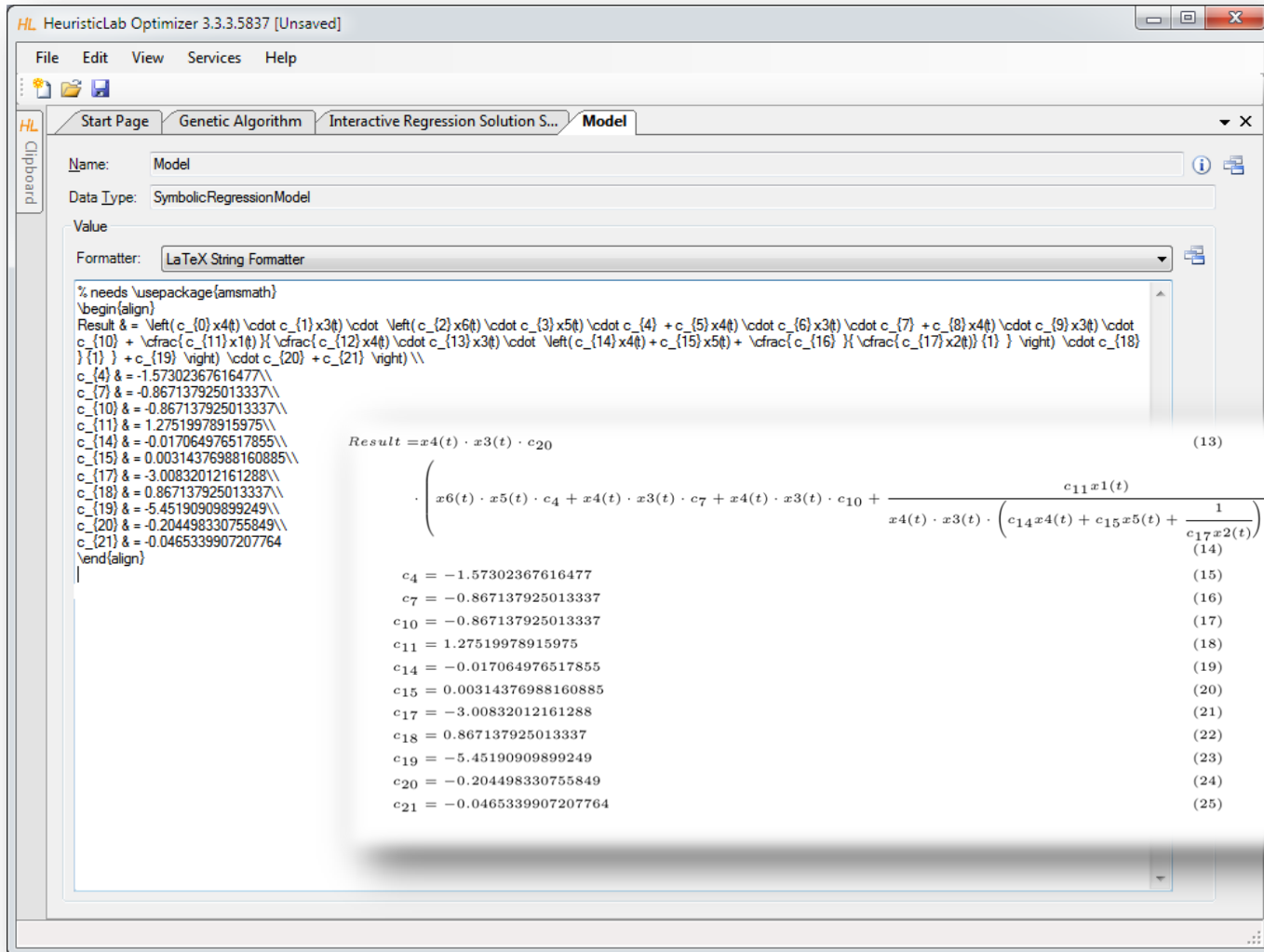
The screenshot shows the HeuristicLab Optimizer interface. The 'Model' tab is active, and the 'Formatter' dropdown menu is highlighted with a red box, showing 'LaTeX String Formatter' selected. The main text area contains LaTeX code for a linear regression model. A callout box displays the resulting LaTeX output, which includes the equation for the result and the values for the coefficients c_0 through c_{10} .

```
% needs \usepackage{amsmath}
\begin{align}
Result &= \text{left}(c_{(0)}x1(t) + c_{(1)}x2(t) + c_{(2)}x3(t) + c_{(3)}x4(t) + c_{(4)}x5(t) + c_{(5)}x6(t) + c_{(6)}x7(t) + c_{(7)}x8(t) + c_{(8)}x9(t) + c_{(9)}x10(t) + c_{(10)} \text{right} \\
c_{(0)} &= 0.0813371220642195 \\
c_{(1)} &= 0.199055016563887 \\
c_{(2)} &= -0.0298811744629839 \\
c_{(3)} &= 0.078891883541302 \\
c_{(4)} &= -0.0103065273366223 \\
c_{(5)} &= 0.0316849536396099 \\
c_{(6)} &= -0.0470707585925129 \\
c_{(7)} &= -0.0291939124032144 \\
c_{(8)} &= 0.00157679665070775 \\
c_{(9)} &= 0.105250443686677 \\
c_{(10)} &= 0.0200987846293256
\end{align}
```

$$Result = (c_0x1(t) + c_1x2(t) + c_2x3(t) + c_3x4(t) + c_4x5(t) + c_5x6(t) + c_6x7(t) + c_7x8(t) + c_8x9(t) + c_9x10(t) + c_{10})$$

$c_0 = 0.0813371220642195$	(1)
$c_1 = 0.199055016563887$	(2)
$c_2 = -0.0298811744629839$	(3)
$c_3 = 0.078891883541302$	(4)
$c_4 = -0.0103065273366223$	(5)
$c_5 = 0.0316849536396099$	(6)
$c_6 = -0.0470707585925129$	(7)
$c_7 = -0.0291939124032144$	(8)
$c_8 = 0.00157679665070775$	(9)
$c_9 = 0.105250443686677$	(10)
$c_{10} = 0.0200987846293256$	(11)

LaTeX Export



The screenshot shows the HeuristicLab Optimizer interface with a model named 'Model' of type 'SymbolicRegressionModel'. The 'Value' field is set to 'LaTeX String Formatter'. The output shows the LaTeX code for the model and its numerical coefficients.

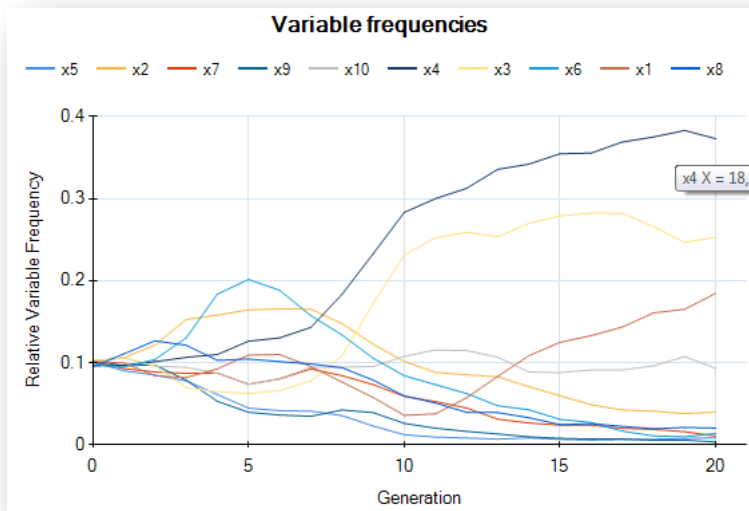
```

% needs \usepackage{amsmath}
\begin{align}
Result &= \left( c_4 x_4(t) + c_7 x_7(t) + c_{10} x_{10}(t) + c_{11} x_{11}(t) + c_{14} x_{14}(t) + c_{15} x_{15}(t) + c_{17} x_{17}(t) + c_{18} x_{18}(t) + c_{19} x_{19}(t) + c_{20} x_{20}(t) + c_{21} x_{21}(t) \right) \\
&+ \frac{c_{11} x_1(t)}{x_4(t) \cdot x_3(t) \cdot \left( c_{14} x_4(t) + c_{15} x_5(t) + \frac{1}{c_{17} x_2(t)} \right) \cdot c_{18}} + c_{19} + c_{21} \\
c_4 &= -1.57302367616477 \\
c_7 &= -0.867137925013337 \\
c_{10} &= -0.867137925013337 \\
c_{11} &= 1.27519978915975 \\
c_{14} &= -0.017064976517855 \\
c_{15} &= 0.00314376988160885 \\
c_{17} &= -3.00832012161288 \\
c_{18} &= 0.867137925013337 \\
c_{19} &= -5.45190909899249 \\
c_{20} &= -0.204498330755849 \\
c_{21} &= -0.0465339907207764
\end{align}

```

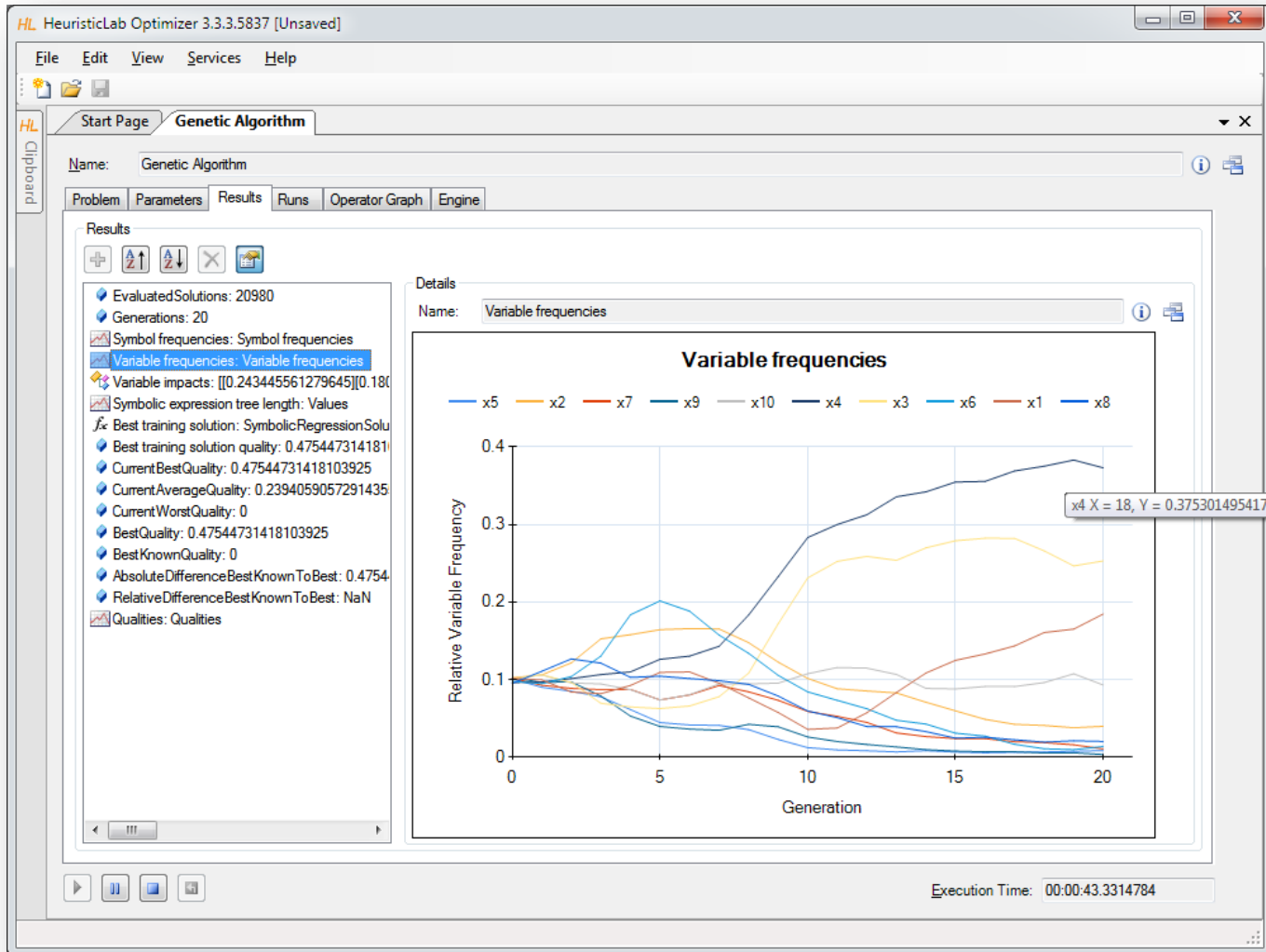
Variable Relevance Analysis

- Which variables are important to predict classes correctly?
- Demonstration
 - Variable frequency analyzer
 - symbol frequency analyzer
 - variable impacts

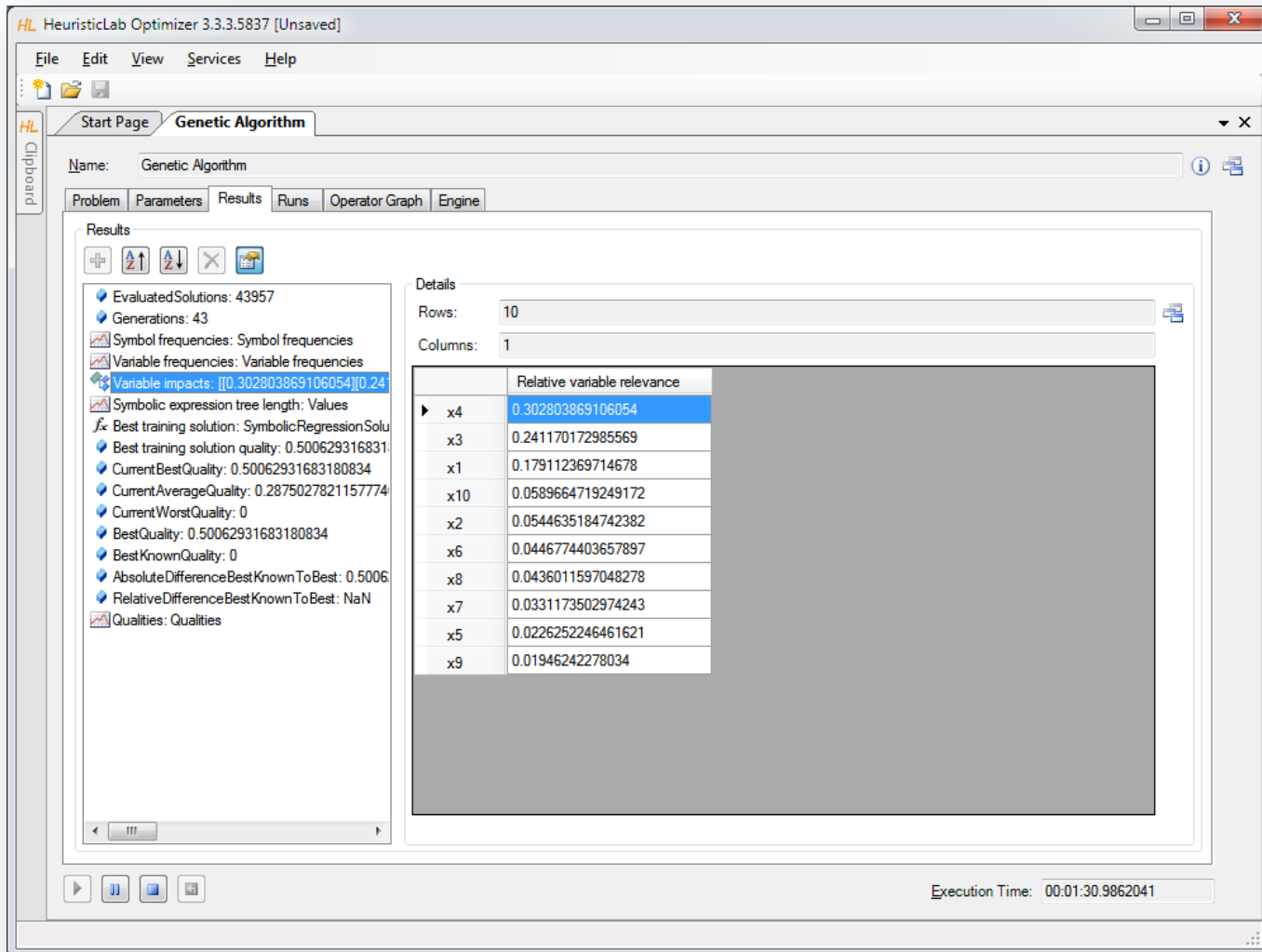


	Relative variable relevance
x4	0.302803869106054
x3	0.241170172985569
x1	0.179112369714678
x10	0.0589664719249172
x2	0.0544635184742382
x6	0.0446774403657897
x8	0.0436011597048278
x7	0.0331173502974243
x5	0.0226252246461621
x9	0.01946242278034

Inspect Variable Frequency Chart



Inspect Variable Impacts

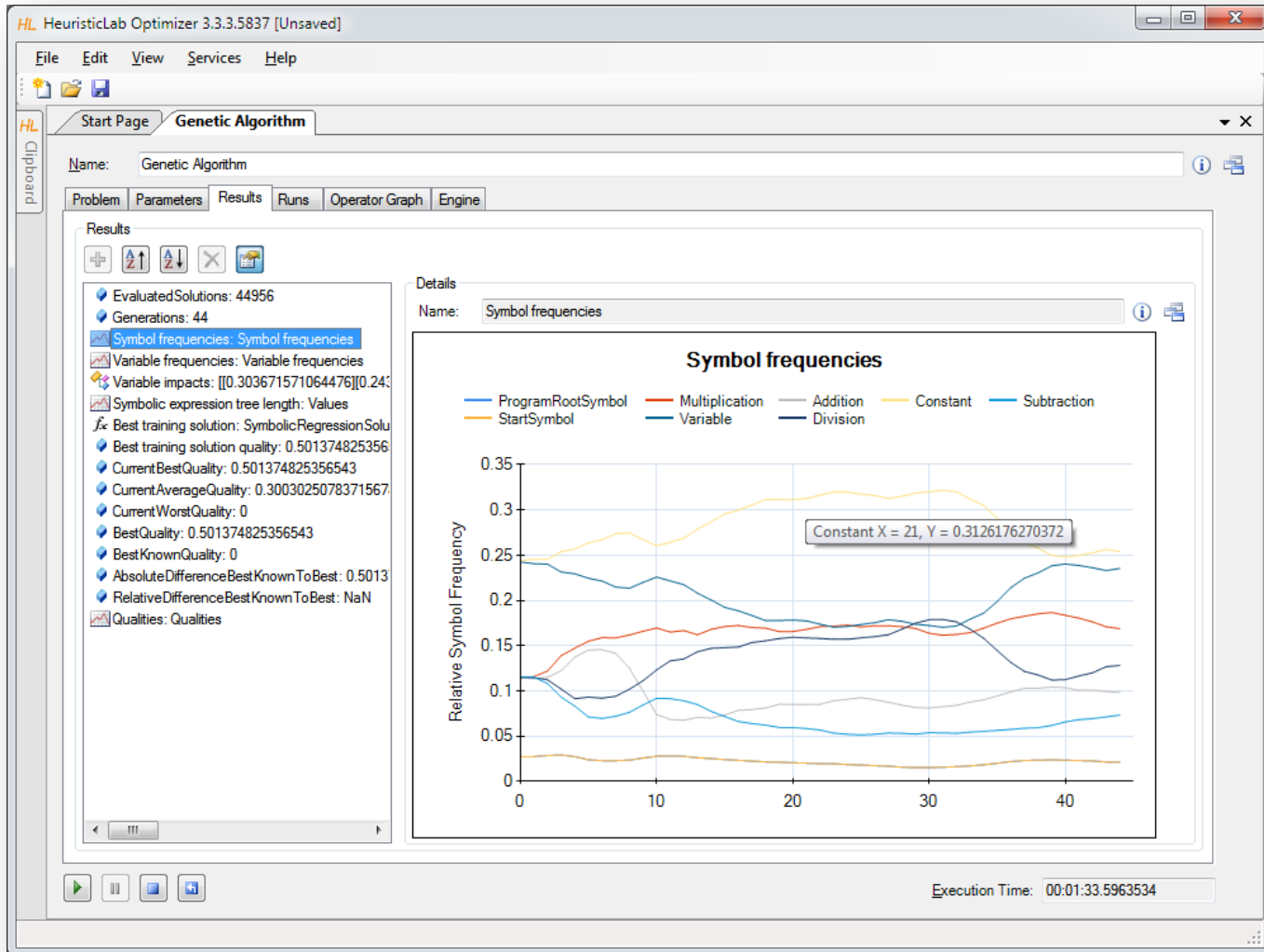


The screenshot shows the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Genetic Algorithm" tab is active, and the "Results" sub-tab is selected. The "Results" panel on the left lists various metrics, with "Variable impacts: [[0.302803869106054]][0.241170172985569]" highlighted. The "Details" panel on the right shows a table of relative variable relevance for 10 rows and 1 column. The table data is as follows:

	Relative variable relevance
x4	0.302803869106054
x3	0.241170172985569
x1	0.179112369714678
x10	0.0589664719249172
x2	0.0544635184742382
x6	0.0446774403657897
x8	0.0436011597048278
x7	0.0331173502974243
x5	0.0226252246461621
x9	0.01946242278034

At the bottom right of the window, the "Execution Time" is displayed as 00:01:30.9862041.

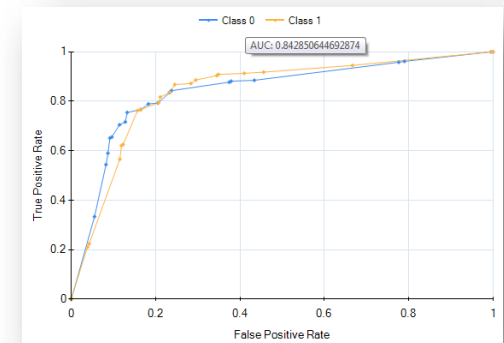
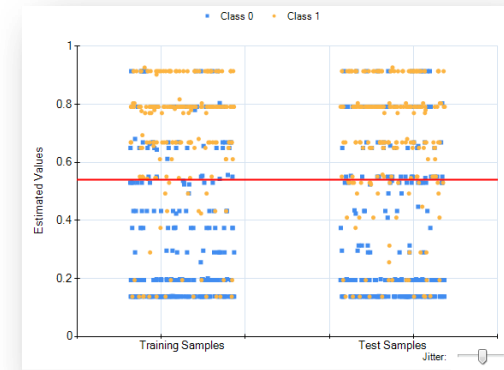
Inspect Symbol Frequencies



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

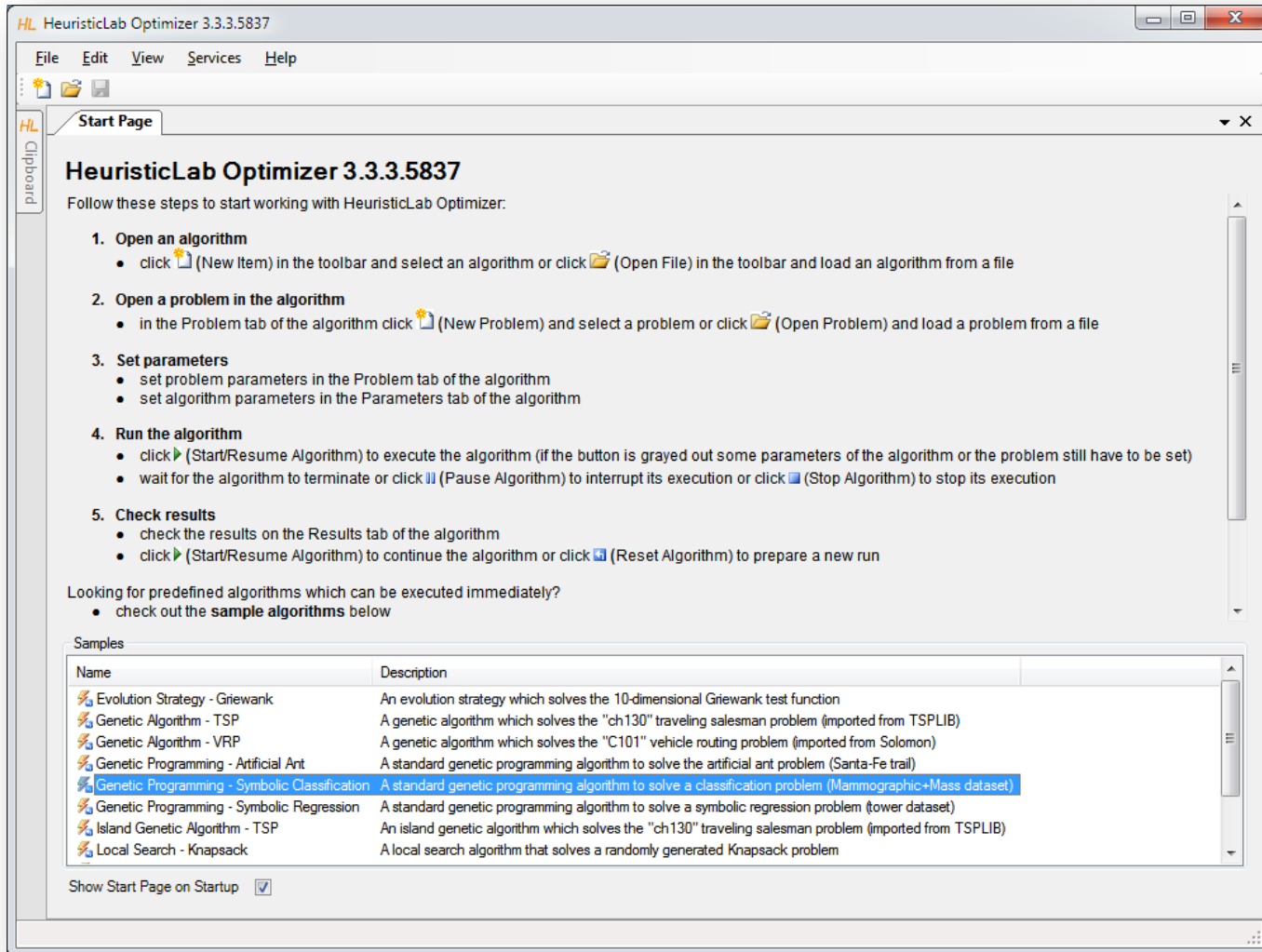


	Actual Class 0	Actual Class 1
Predicted Class 0	197	29
Predicted Class 1	64	190

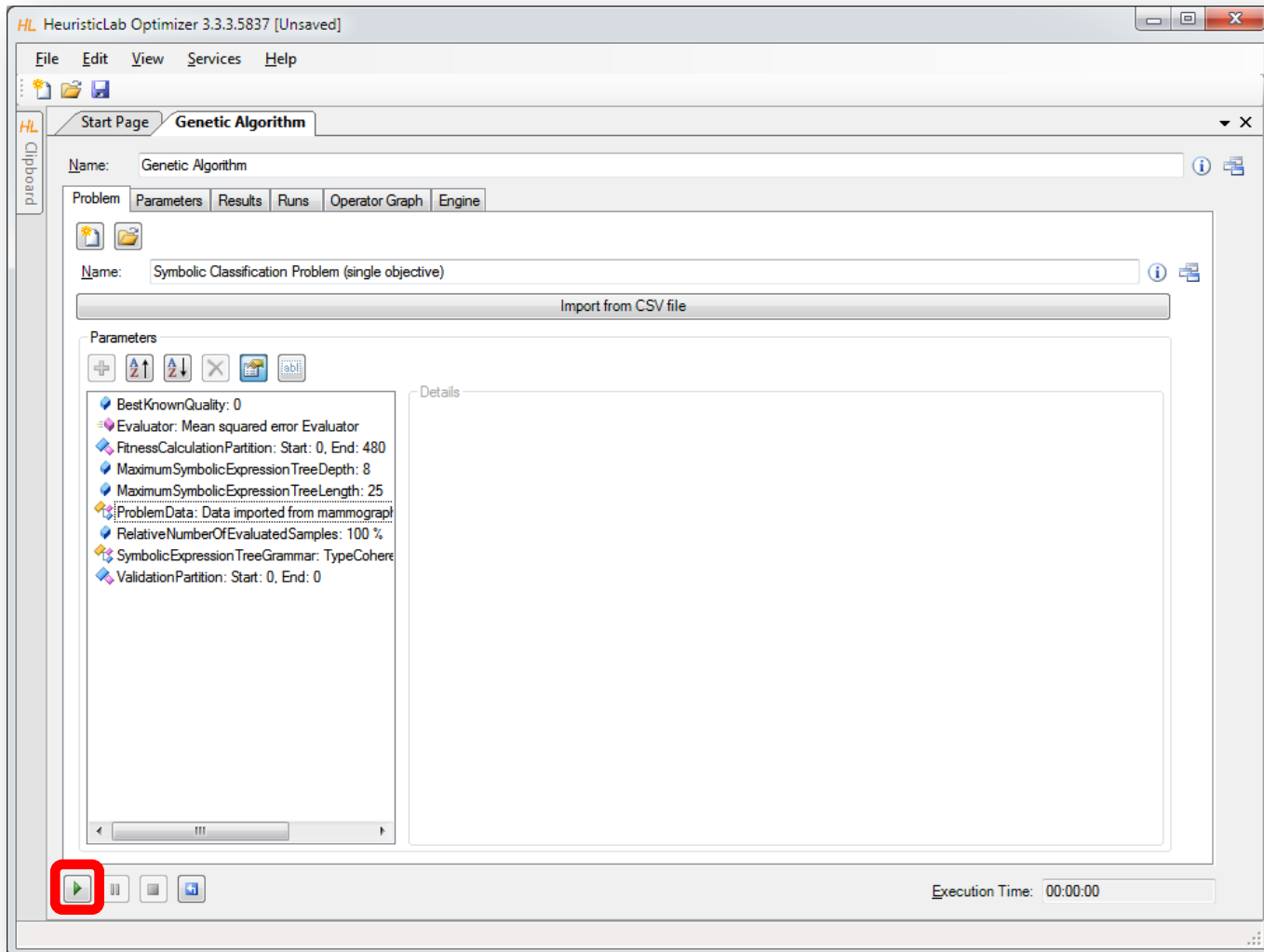
Case Study: Classification

- 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#ICCGI2011>

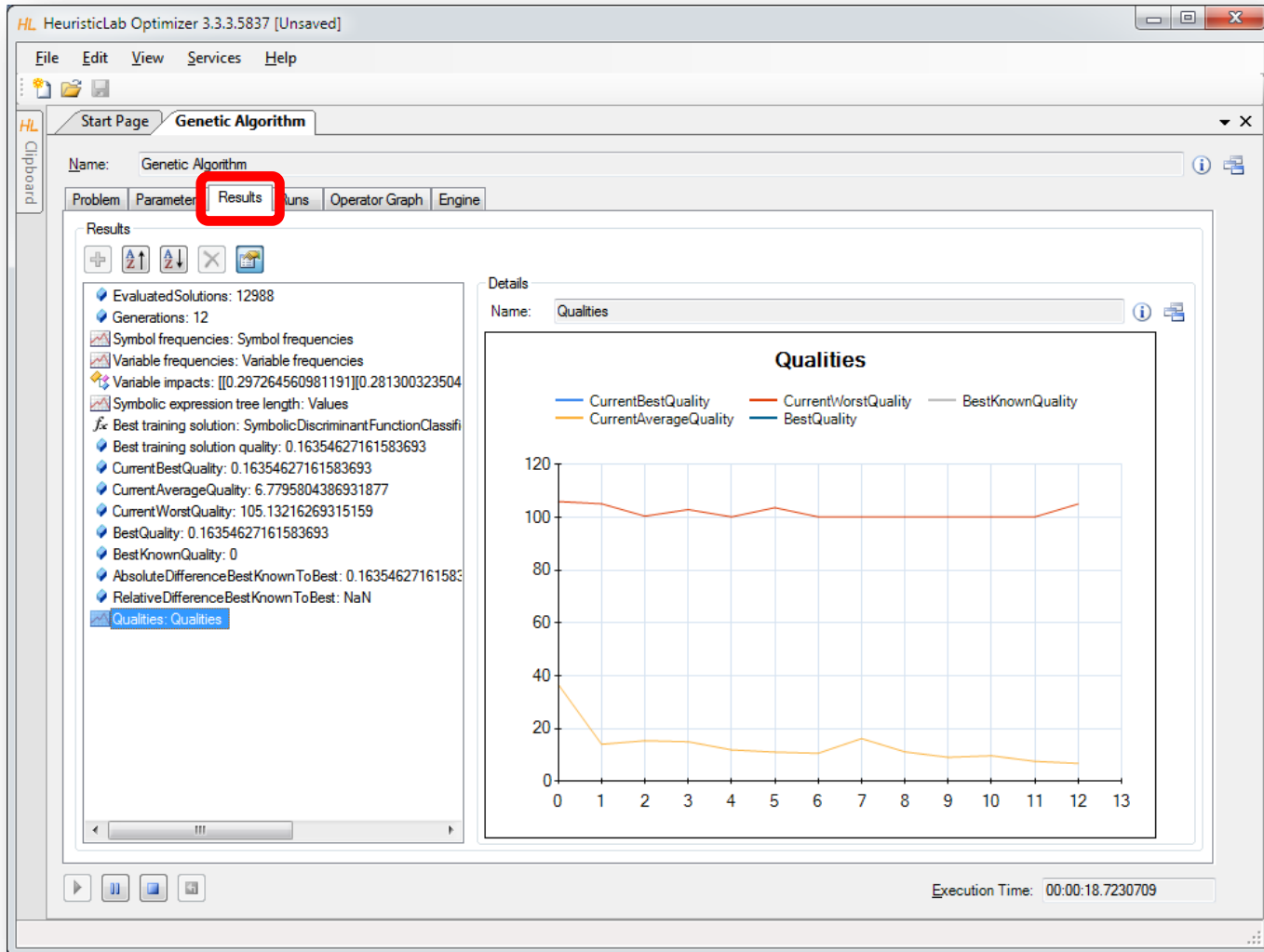
Open Sample



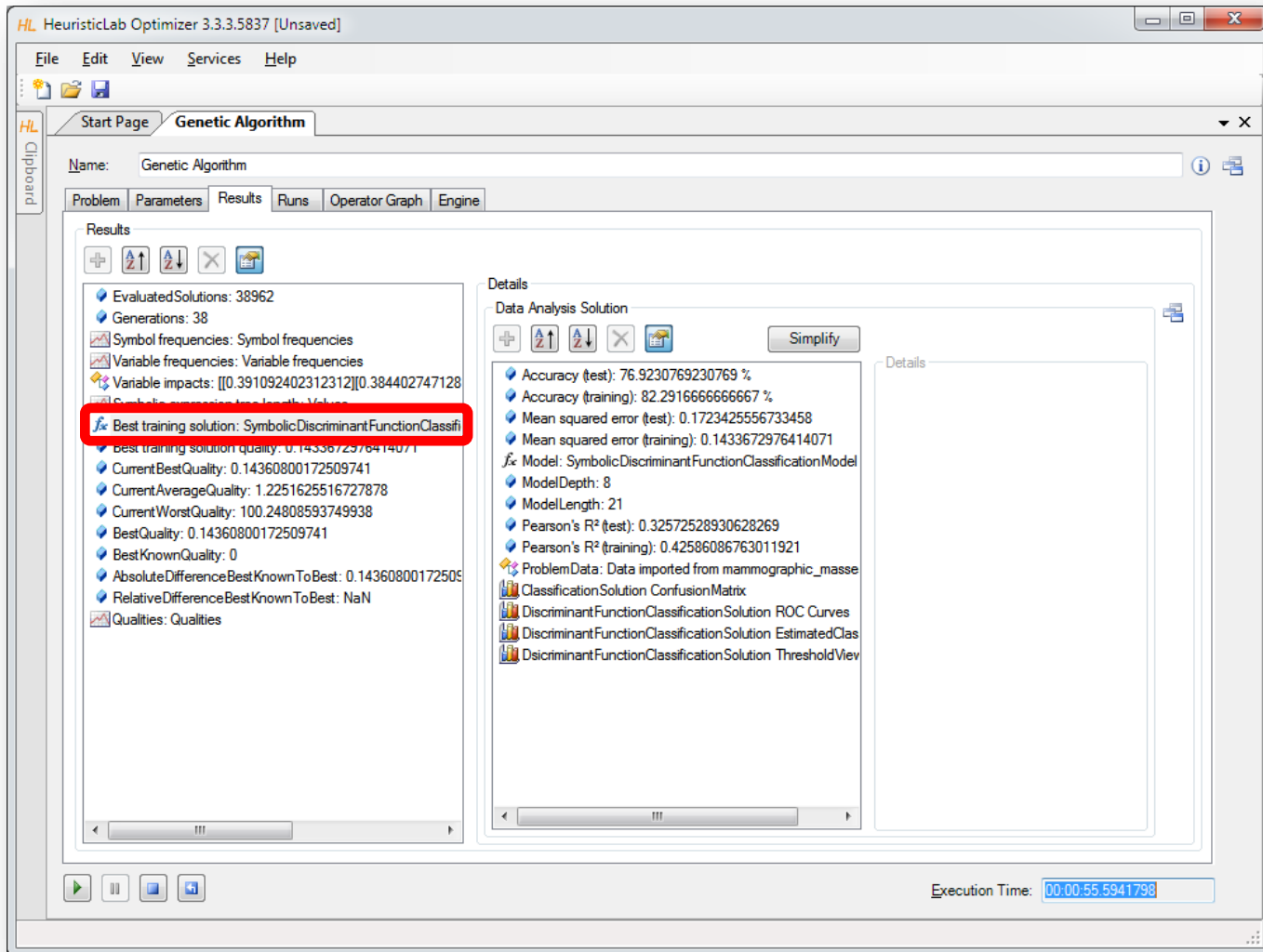
Configure and Run Algorithm



Inspect Quality Linechart



Inspect Best Training Solution



The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Genetic Algorithm" is selected in the "Start Page" tab. The "Results" tab is active, showing a list of metrics. The "Best training solution: SymbolicDiscriminantFunctionClassifi" is highlighted with a red box. The "Details" tab is also active, showing the "Data Analysis Solution" for the selected best training solution. The "Execution Time" is displayed as 00:00:55.5941798.

Results

- Evaluated Solutions: 38962
- Generations: 38
- Symbol frequencies: Symbol frequencies
- Variable frequencies: Variable frequencies
- Variable impacts: [[0.391092402312312][0.384402747128
- Symbol frequencies: Symbol frequencies
- Best training solution: SymbolicDiscriminantFunctionClassifi**
- Best training solution quality: 0.14360800172509741
- CurrentBestQuality: 0.14360800172509741
- CurrentAverageQuality: 1.2251625516727878
- CurrentWorstQuality: 100.24808593749938
- BestQuality: 0.14360800172509741
- BestKnownQuality: 0
- AbsoluteDifferenceBestKnownToBest: 0.14360800172509741
- RelativeDifferenceBestKnownToBest: NaN
- Qualities: Qualities

Details

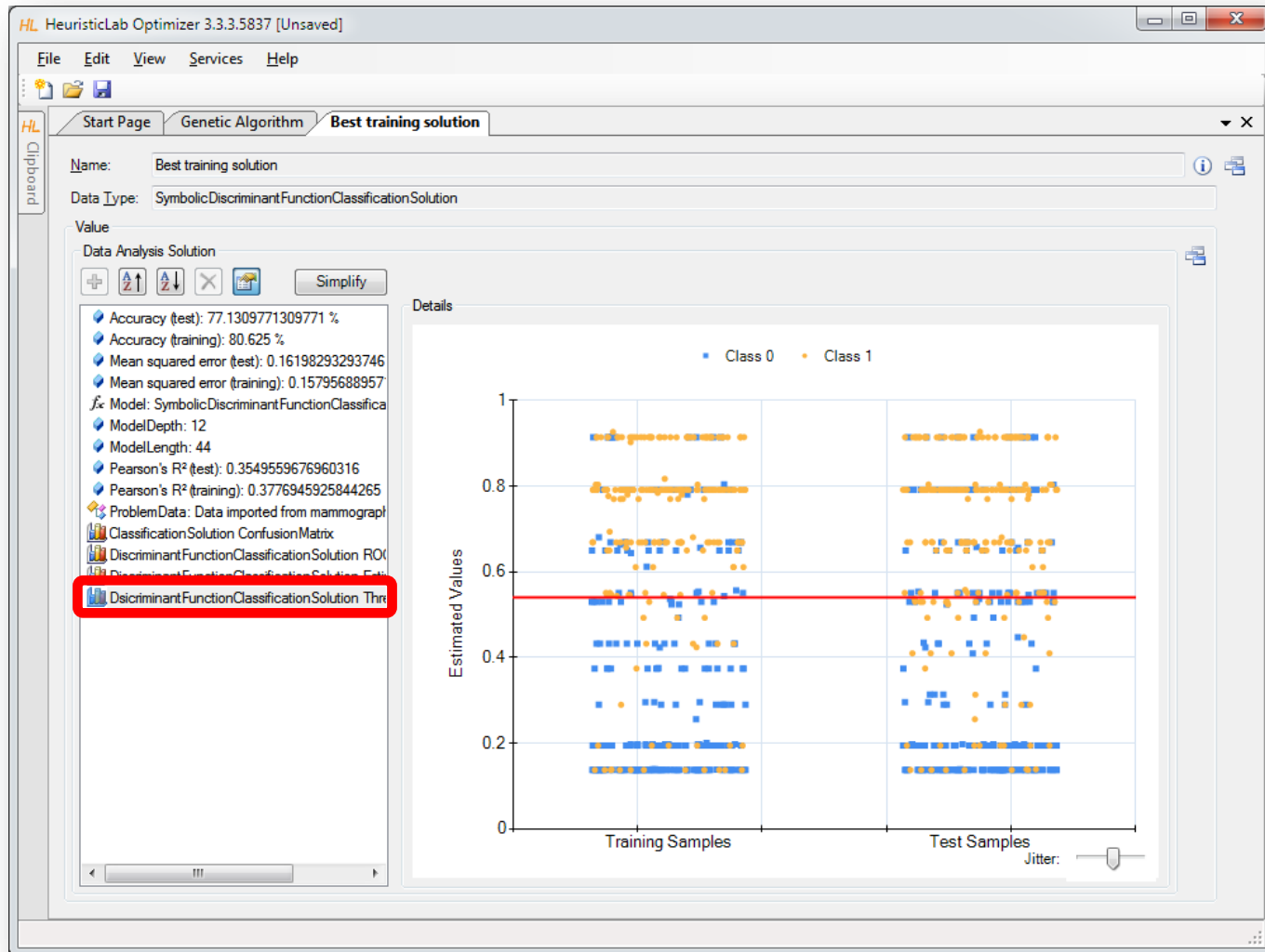
Data Analysis Solution

Simplify

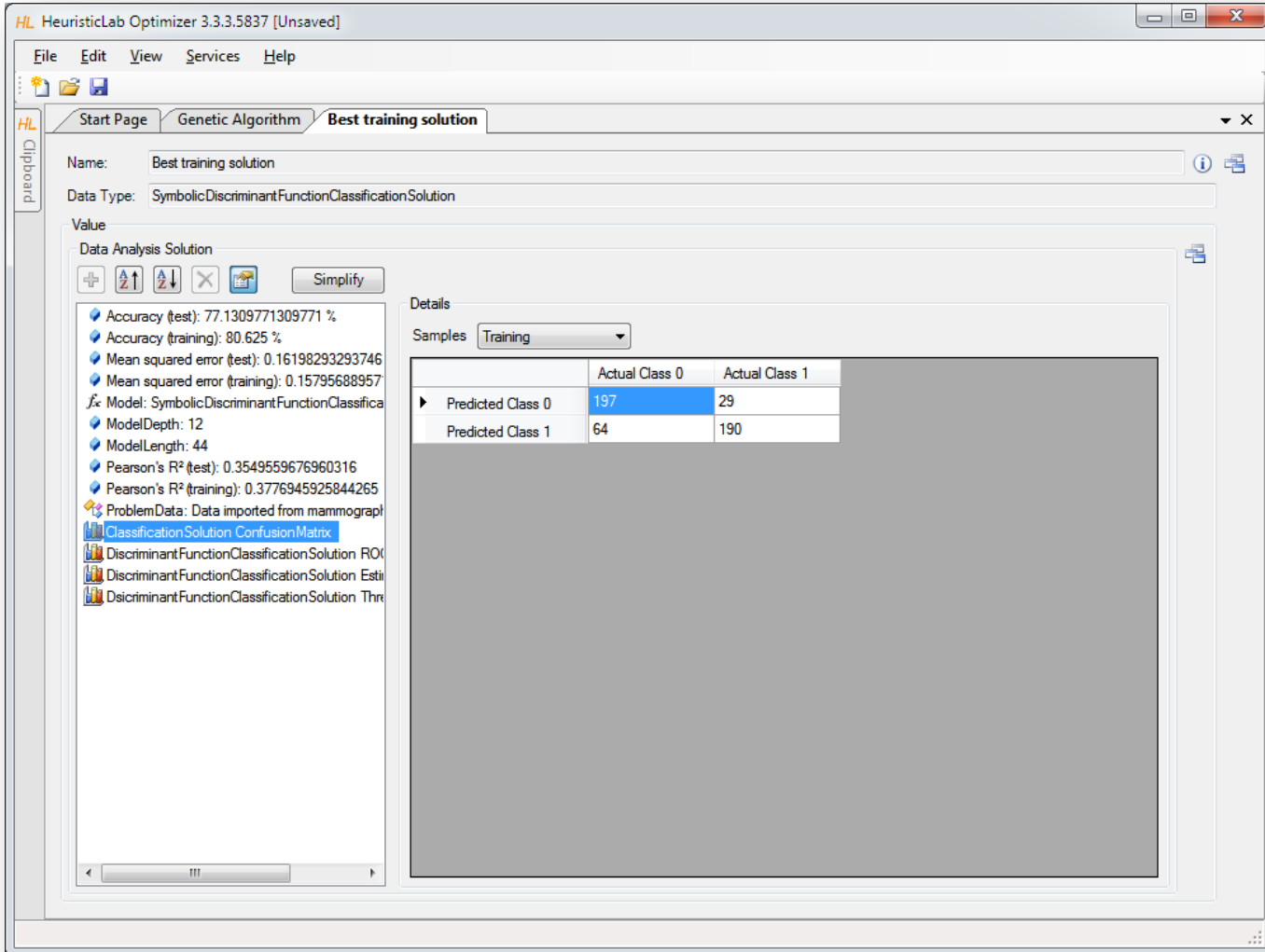
- Accuracy (test): 76.9230769230769 %
- Accuracy (training): 82.2916666666667 %
- Mean squared error (test): 0.1723425556733458
- Mean squared error (training): 0.1433672976414071
- Model: SymbolicDiscriminantFunctionClassificationModel
- ModelDepth: 8
- ModelLength: 21
- Pearson's R² (test): 0.32572528930628269
- Pearson's R² (training): 0.42586086763011921
- ProblemData: Data imported from mammographic_masse
- ClassificationSolution ConfusionMatrix
- DiscriminantFunctionClassificationSolution ROC Curves
- DiscriminantFunctionClassificationSolution EstimatedClass
- DiscriminantFunctionClassificationSolution ThresholdView

Execution Time: 00:00:55.5941798

Inspect Model Output and Thresholds



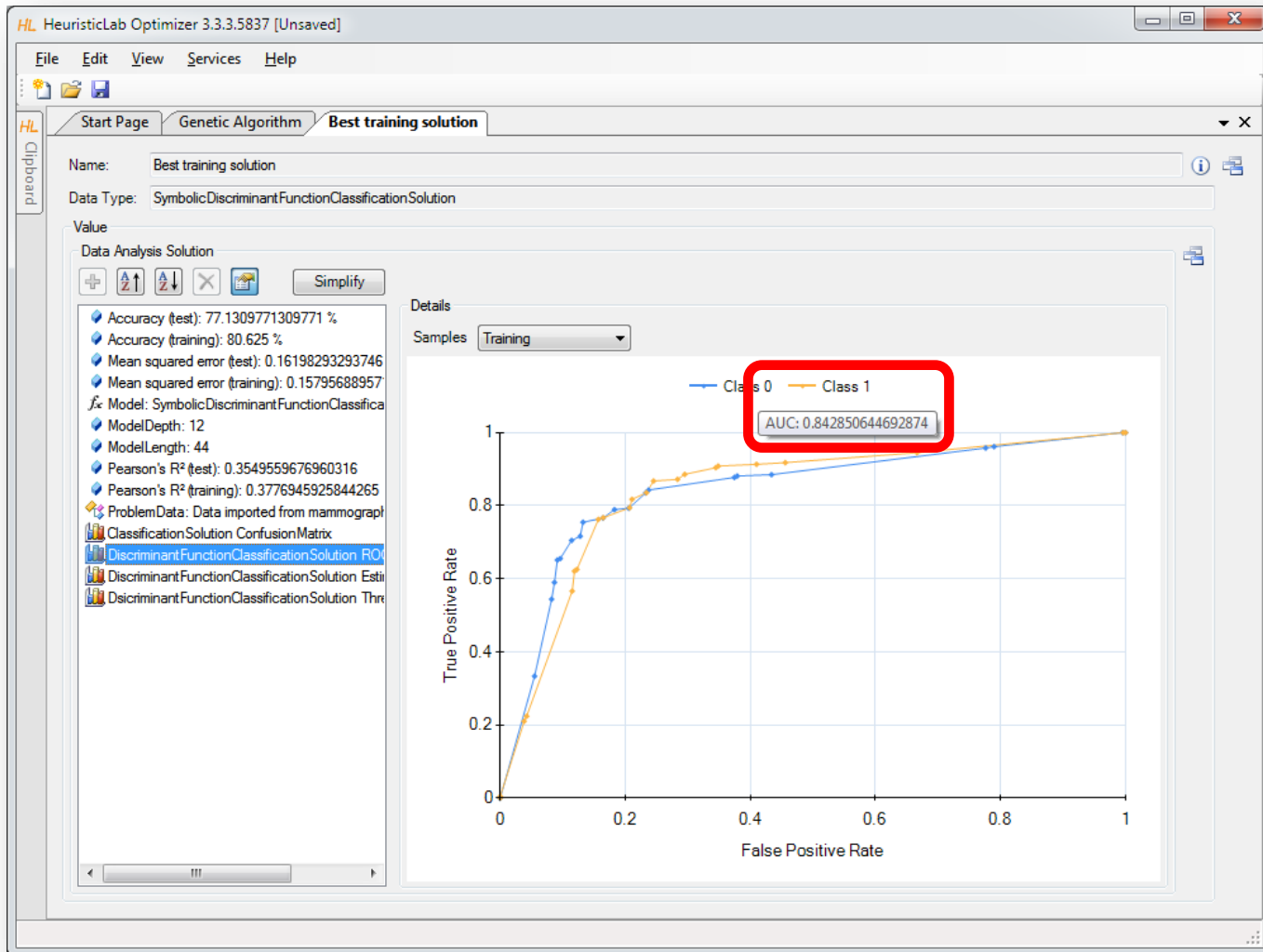
Inspect Confusion Matrix



The screenshot shows the HeuristicLab Optimizer interface. The main window displays the 'Best training solution' for a 'SymbolicDiscriminantFunctionClassificationSolution'. The 'Value' section shows a 'Data Analysis Solution' with various performance metrics and a list of items, including 'ClassificationSolution ConfusionMatrix'. The 'Details' section shows a 'Samples' dropdown set to 'Training' and a confusion matrix table.

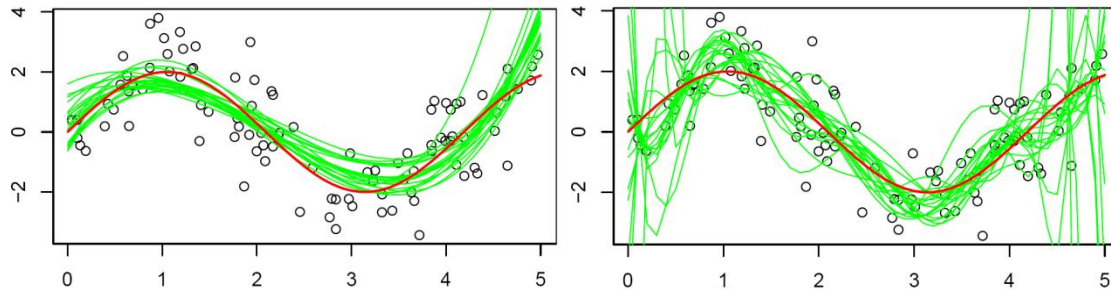
	Actual Class 0	Actual Class 1
Predicted Class 0	197	29
Predicted Class 1	64	190

Inspect ROC Curve



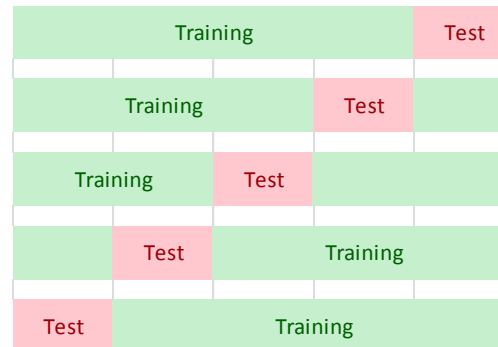
Validation of Results

- Overfitting = memorizing data



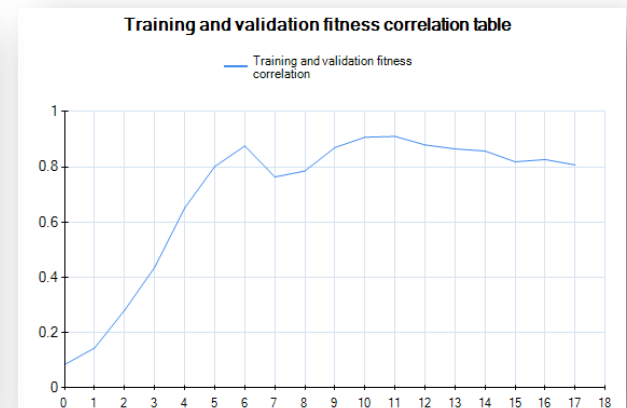
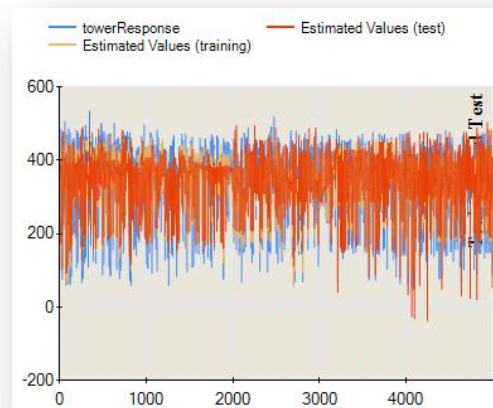
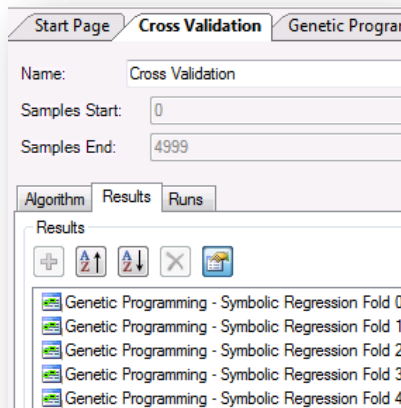
- Strategies to reduce overfitting

- validation partition
- cross-validation

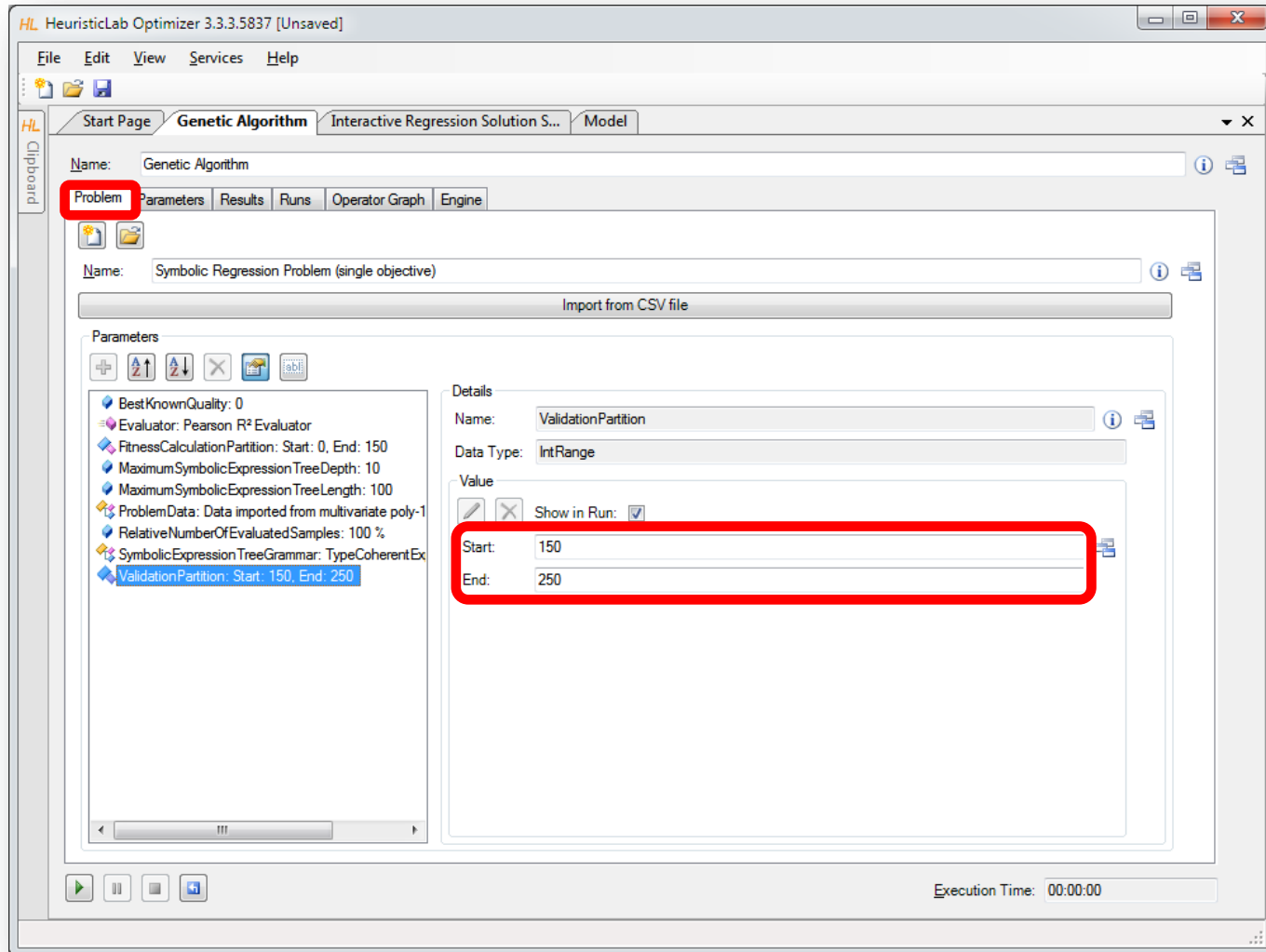


Validation of Results

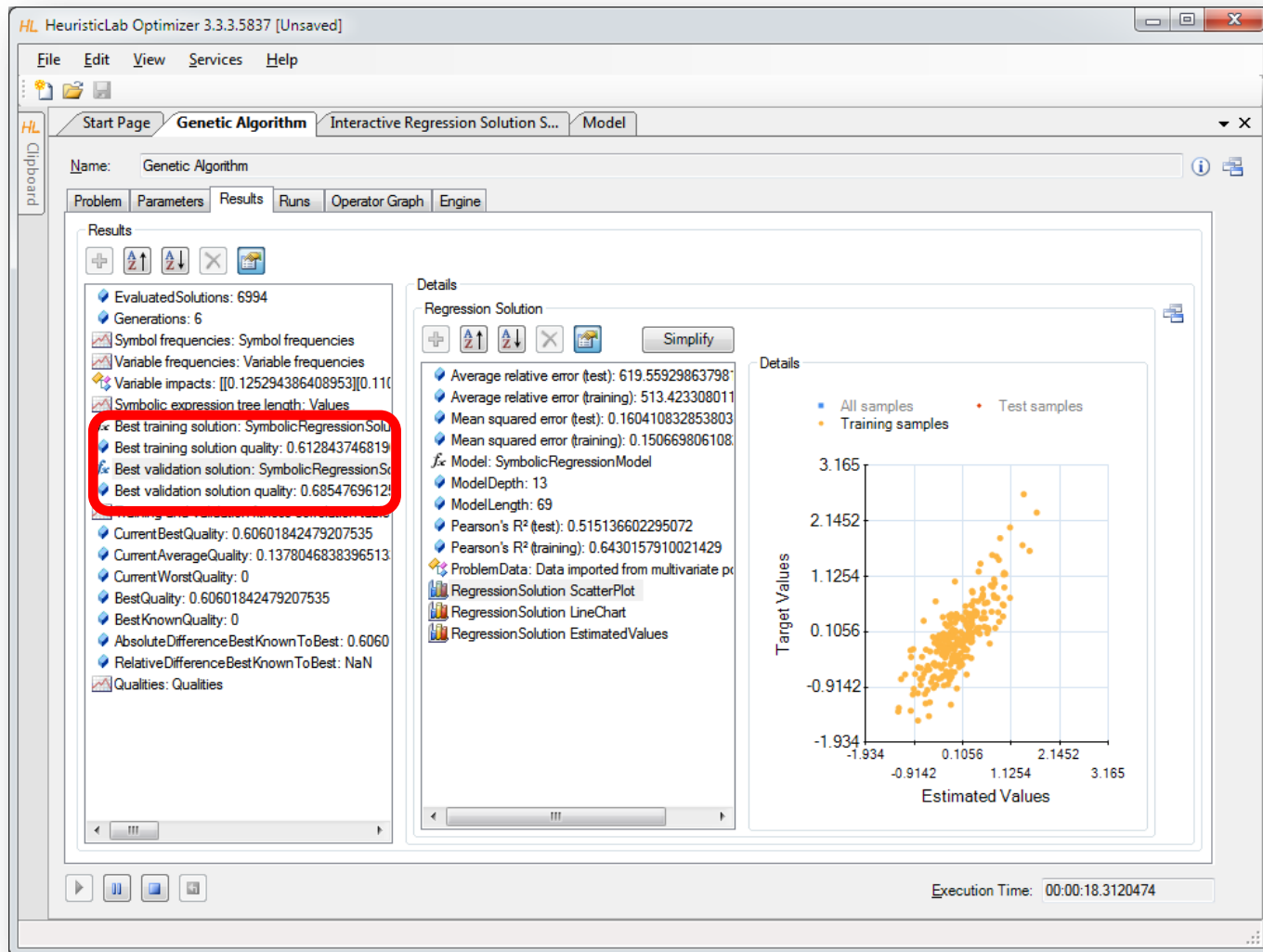
- Demonstration
 - Configuration of a validation set
 - Inspection of best solution on validation set
 - Analysis of training- and validation fitness correlation
 - Cross-validation
 - Configuration
 - Analysis of results



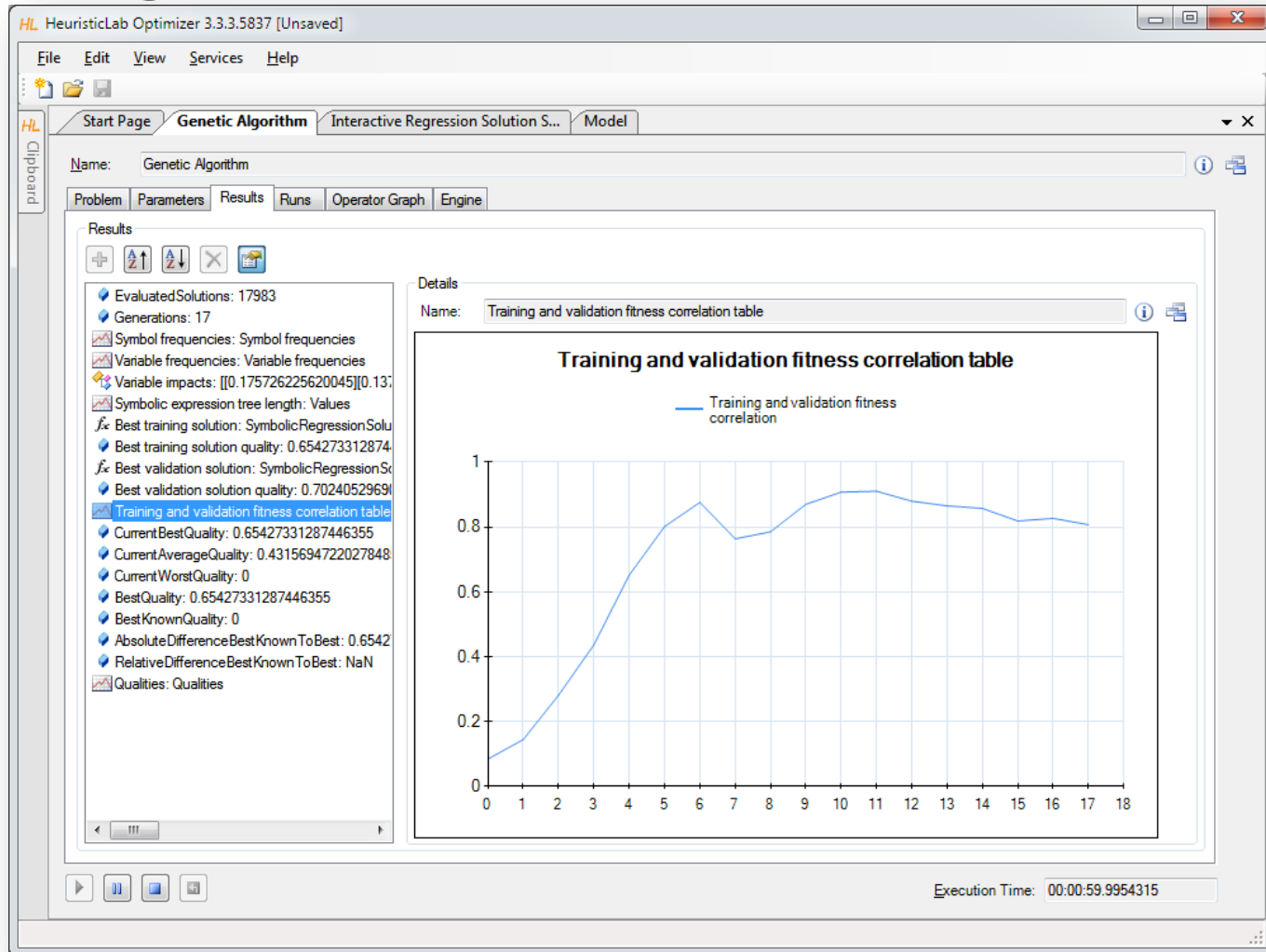
Configuration of Validation Partition



Inspect Best Model on Validation Partition



Inspect Linechart of Correlation of Training and Validation Fitness



Agenda



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

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

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

Some Additional Features

- HeuristicLab Hive
 - parallel and distributed execution of algorithms and experiments on many computers in a network
- Optimization Knowledge Base (OKB)
 - database to store algorithms, problems, parameters and results
 - open to the public
 - open for other frameworks
 - analyze and store characteristics of problem instances and problem classes
- External solution evaluation and simulation-based optimization
 - interface to couple HeuristicLab with other applications (MatLab, 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

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>



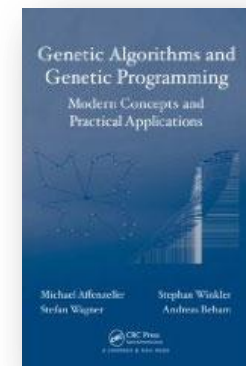
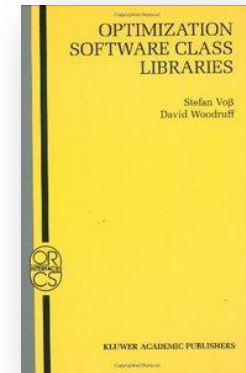
HEAL

Heuristic and Evolutionary
Algorithms Laboratory



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>

heuristiclab@googlegroups.com