



# 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



**Heuristic  
Optimization in  
Production and  
Logistics**

# Instructor Biographies

- Stefan Wagner
  - Full professor for complex software systems (since 2009)  
University of Applied Sciences Upper Austria
  - Co-founder of the HEAL research group
  - Project manager and chief architect of HeuristicLab
  - PhD in technical sciences (2009)  
Johannes Kepler University Linz, Austria
  - Associate professor (2005 – 2009)  
University of Applied Sciences Upper Austria
  - <http://heal.heuristiclab.com/team/wagner>
- Gabriel Kronberger
  - Full professor for business intelligence (since 2011)  
University of Applied Sciences Upper Austria
  - Member of the HEAL research group
  - Architect of HeuristicLab
  - PhD in technical sciences (2010)  
Johannes Kepler University Linz, Austria
  - Research assistant (2005 – 2011)  
University of Applied Sciences Upper Austria
  - <http://heal.heuristiclab.com/team/kronberger>



# Agenda



- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
  
- **Demonstration Part I: Working with HeuristicLab**
- **Demonstration Part II: Data-based Modeling**
  
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

# Objectives of the Tutorial



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

# Introduction

- Motivation and Goals
  - graphical user interface
  - paradigm independence
  - multiple algorithms and problems
  - large scale experiments and analyses
  - parallelization
  - extensibility, flexibility and reusability
  - visual and interactive algorithm development
  - multiple layers of abstraction
- Facts
  - development of HeuristicLab started in 2002
  - based on Microsoft .NET and C#
  - used in research and education
  - second place at the *Microsoft Innovation Award 2009*
  - open source (GNU General Public License)
  - version 3.3.0 released on May 18th, 2010
  - latest version 3.3.13 "Windischgarsten"



# Where to get HeuristicLab?

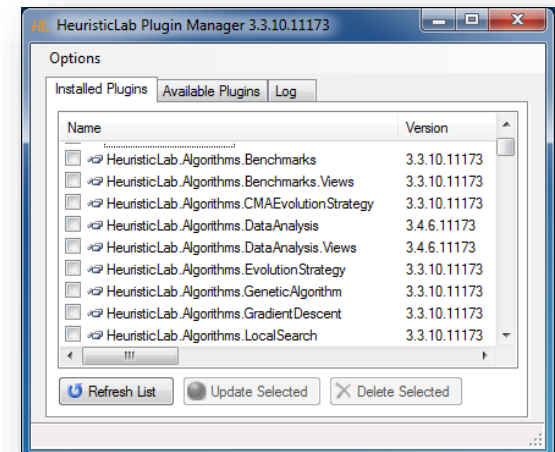
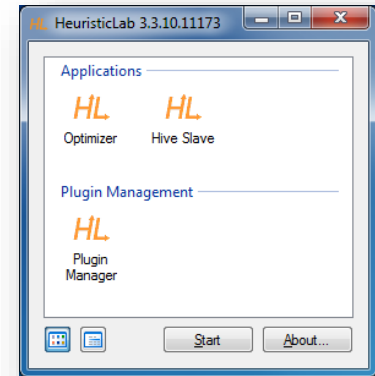


- Download binaries
  - deployed as ZIP archives
  - latest stable version 3.3.13 “Windischgarsten”
    - released on November 20th, 2015
  - daily trunk builds
  - <http://dev.heuristiclab.com/download>
- Check out sources
  - SVN repository
  - HeuristicLab 3.3.13 tag
    - <http://svn.heuristiclab.com/svn/core/tags/3.3.13>
  - Stable development version
    - <http://svn.heuristiclab.com/svn/core/stable>
- License
  - GNU General Public License (Version 3)
- System requirements
  - Microsoft .NET Framework 4.5
  - enough RAM and CPU power ;-)

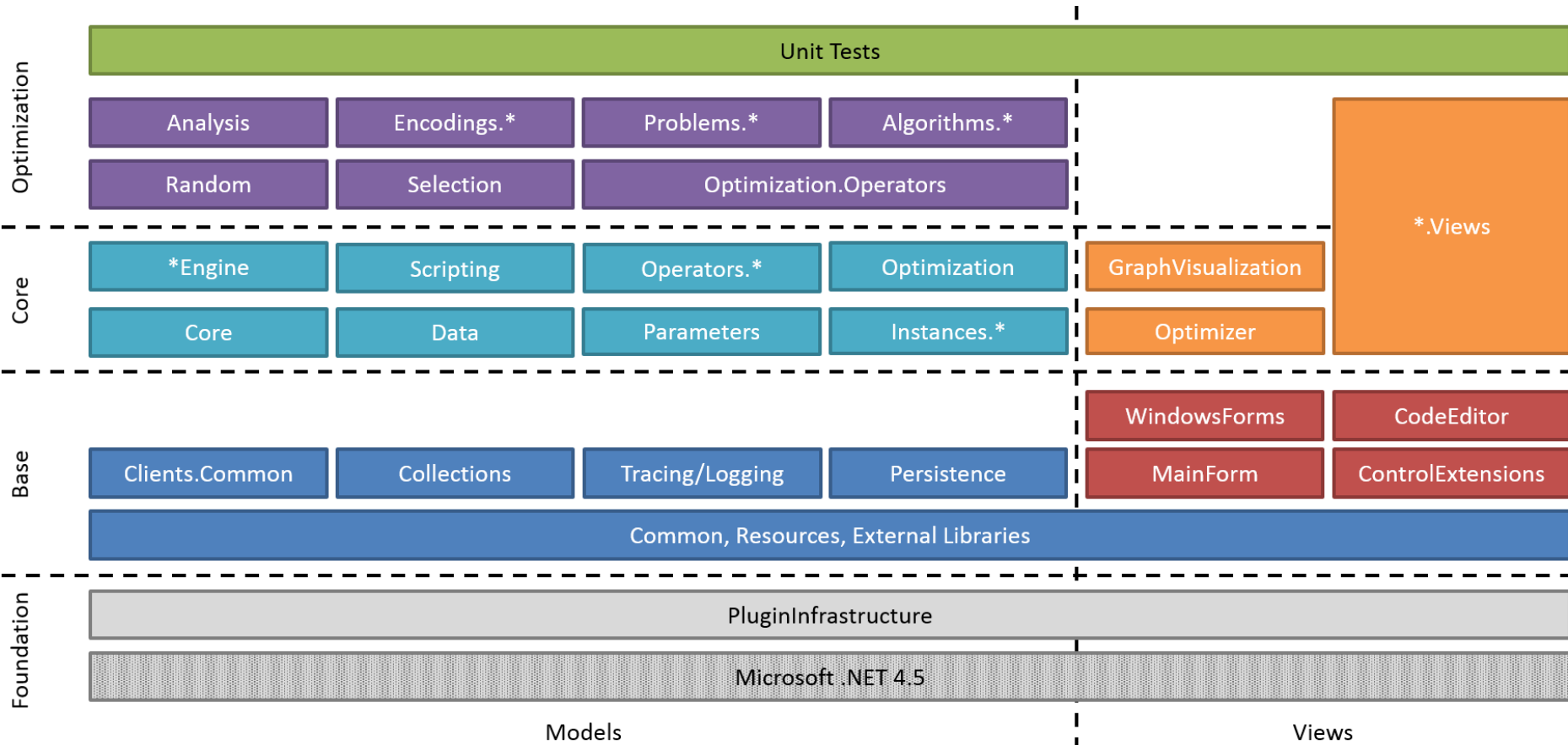
A screenshot of the HeuristicLab website. The page has a white background with a navigation bar at the top containing links for Home, News, Download, Features, Documentation, Support, and Search. Below the navigation bar is a main content area with a heading 'HeuristicLab' and a sub-heading 'A Paradigm-Independent and Extensible Environment for Heuristic Optimization'. The main content area contains a paragraph of text, a video player titled 'HeuristicLab Tour', a list of features, a download button for 'HeuristicLab 3.3', and a section for 'Research &amp; Publications'. At the bottom of the page, there is a 'Thank you!' section with a logo for ReSharper and a 'Download in other formats' section.

# Plugin Infrastructure

- HeuristicLab consists of many assemblies
  - 150+ plugins in HeuristicLab 3.3.13
  - plugins can be loaded or unloaded at runtime
  - plugins can be updated via internet
  - application plugins provide GUI frontends
- Extensibility
  - developing and deploying new plugins is easy
  - dependencies are explicitly defined, automatically checked and resolved
  - automatic discovery of interface implementations (service locator pattern)
- Plugin Manager
  - GUI to check, install, update or delete plugins



# Plugin Architecture



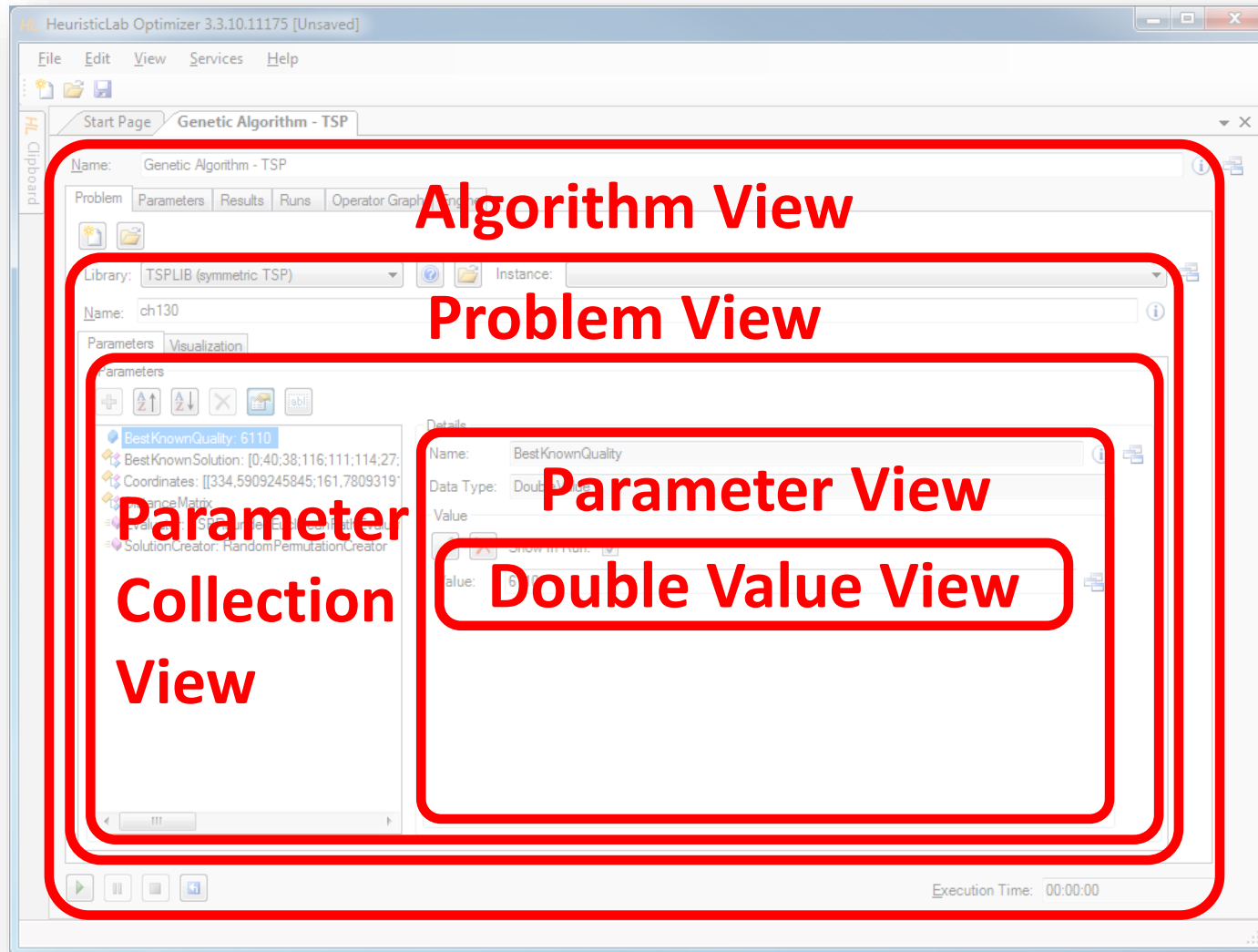


# Graphical User Interface



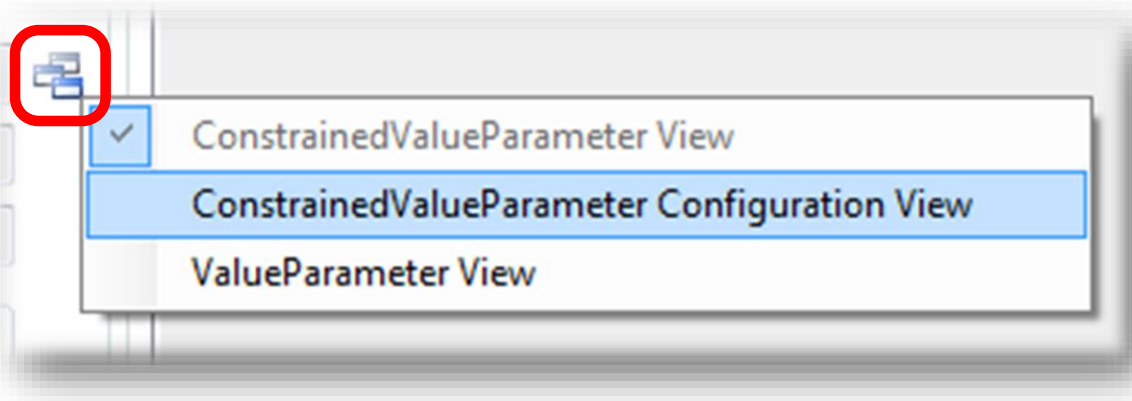
- HeuristicLab GUI is made up of views
  - views are visual representations of content objects
  - views are composed in the same way as their content
  - views and content objects are loosely coupled
  - multiple different views may exist for the same content
- Drag & Drop
  - views support drag & drop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

# Graphical User Interface



# Graphical User Interface

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



# Available Algorithms

## Population-based

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

## Trajectory-based

- Local Search
- Tabu Search
- Robust Taboo Search
- Variable Neighborhood Search
- Simulated Annealing

## Data Analysis

- Linear Discriminant Analysis
- Linear Regression
- Multinomial Logit Classification
- k-Nearest Neighbor
- k-Means
- Neighbourhood Component Analysis
- Artificial Neural Networks
- Random Forests
- Support Vector Machines
- Gaussian Processes

## Additional Algorithms

- User-defined Algorithm
- Performance Benchmarks
- Hungarian Algorithm
- Cross Validation
- LM-BFGS

# Available Problems

## Combinatorial Problems

- Traveling Salesman
- Vehicle Routing
- Knapsack
- NK[P,Q]
- Job Shop Scheduling
- Linear Assignment
- Quadratic Assignment
- OneMax
- Orienteering
- Deceptive trap
- Deceptive trap step
- HIFF

## Genetic Programming Problems

- Symbolic Classification
- Symbolic Regression
- Symbolic Time-Series Prognosis
- Artificial Ant
- Lawn Mower

## Additional Problems

- Single-Objective Test Function
- User-defined Problem
- Programmable Problem
- External Evaluation Problem (Anylogic, Scilab, MATLAB)
- Regression, Classification, Clustering
- Trading
- Grammatical Evolution

# Agenda



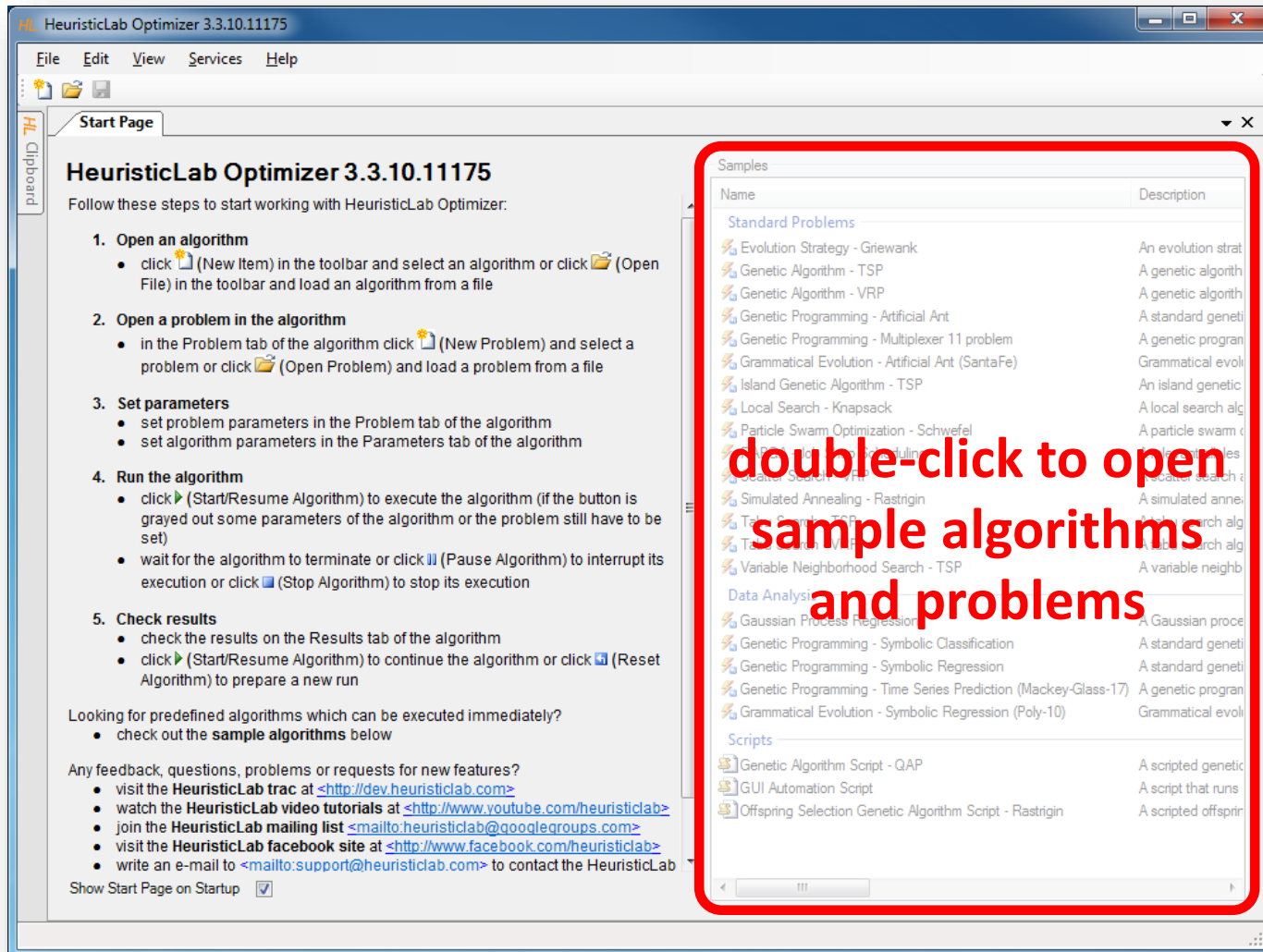
- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
  
- **Demonstration Part I: Working with HeuristicLab**
- **Demonstration Part II: Data-based Modeling**
  
- Some Additional Features
- Planned 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



The screenshot shows the HeuristicLab Optimizer 3.3.10.11175 application window. The main area displays a 'Start Page' with instructions for getting started. A red box highlights the 'Samples' panel on the right, which contains a table of predefined algorithms and problems. A red text overlay reads 'double-click to open sample algorithms and problems'.

**HeuristicLab Optimizer 3.3.10.11175**

Follow these steps to start working with HeuristicLab Optimizer:

- 1. Open an algorithm**
  - click (New Item) in the toolbar and select an algorithm or click (Open File) in the toolbar and load an algorithm from a file
- 2. Open a problem in the algorithm**
  - in the Problem tab of the algorithm click (New Problem) and select a problem or click (Open Problem) and load a problem from a file
- 3. Set parameters**
  - set problem parameters in the Problem tab of the algorithm
  - set algorithm parameters in the Parameters tab of the algorithm
- 4. Run the algorithm**
  - click (Start/Resume Algorithm) to execute the algorithm (if the button is grayed out some parameters of the algorithm or the problem still have to be set)
  - wait for the algorithm to terminate or click (Pause Algorithm) to interrupt its execution or click (Stop Algorithm) to stop its execution
- 5. Check results**
  - check the results on the Results tab of the algorithm
  - click (Start/Resume Algorithm) to continue the algorithm or click (Reset Algorithm) to prepare a new run

Looking for predefined algorithms which can be executed immediately?

- check out the **sample algorithms** below

Any feedback, questions, problems or requests for new features?

- visit the **HeuristicLab trac** at <http://dev.heuristiclab.com>
- watch the **HeuristicLab video tutorials** at <http://www.youtube.com/heuristiclab>
- join the **HeuristicLab mailing list** <mailto:heuristiclab@googlegroups.com>
- visit the **HeuristicLab facebook site** at <http://www.facebook.com/heuristiclab>
- write an e-mail to <mailto:support@heuristiclab.com> to contact the HeuristicLab

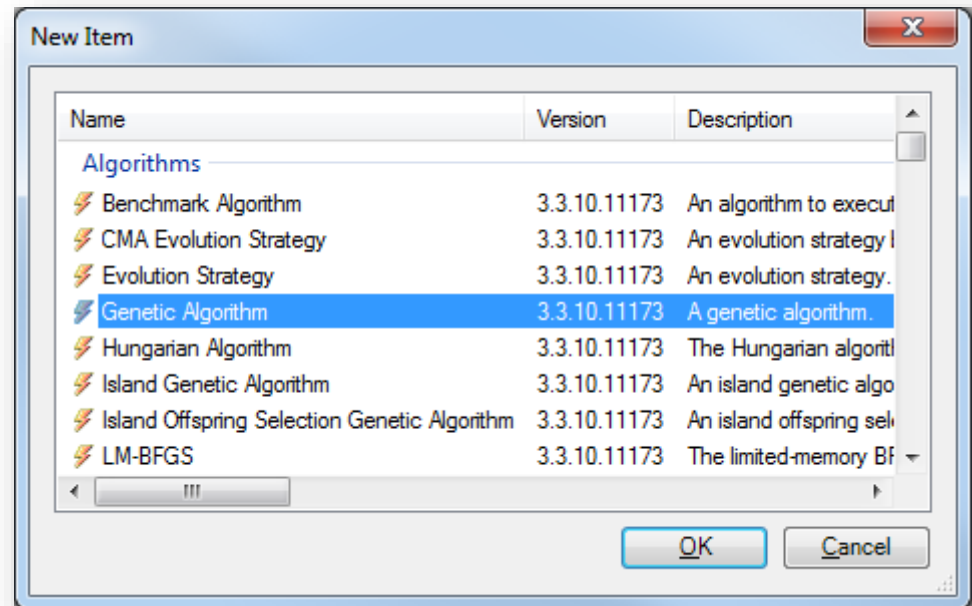
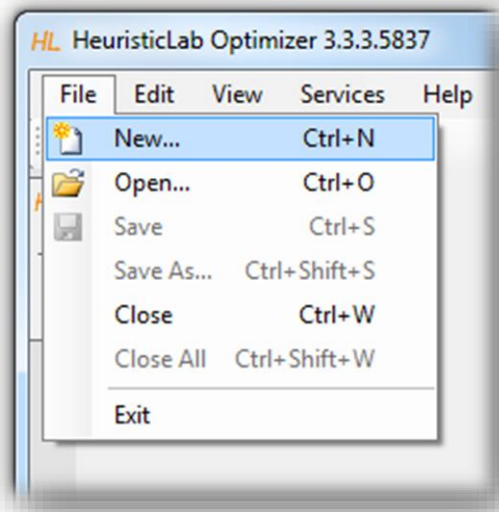
Show Start Page on Startup

| Name   | Description         |
|--|---------------------|
| <b>Standard Problems</b>                                       |                     |
| Evolution Strategy - Griewank                                  | An evolution strat  |
| Genetic Algorithm - TSP  | A genetic algorith  |
| Genetic Algorithm - VRP  | A genetic algorith  |
| Genetic Programming - Artificial Ant                           | A standard geneti   |
| Genetic Programming - Multiplexer 11 problem                   | A genetic program   |
| Grammatical Evolution - Artificial Ant (SantaFe)               | Grammatical evoli   |
| Island Genetic Algorithm - TSP                                 | An island genetic   |
| Local Search - Knapsack  | A local search alg  |
| Particle Swarm Optimization - Schwefel                         | A particle swarm c  |
| Tabu Search - Traveling Salesman Problem                       | A tabu search alg   |
| Tabu Search - TSP  | A tabu search alg   |
| Simulated Annealing - Rastrigin                                | A simulated anneal  |
| Tabu Search - TSP  | A tabu search alg   |
| Tabu Search - TSP  | A tabu search alg   |
| Variable Neighborhood Search - TSP                             | A variable neighb   |
| <b>Data Analysis</b>   |                     |
| Gaussian Process Regression                                    | A Gaussian proces   |
| Genetic Programming - Symbolic Classification                  | A standard geneti   |
| Genetic Programming - Symbolic Regression                      | A standard geneti   |
| Genetic Programming - Time Series Prediction (Mackey-Glass-17) | A genetic program   |
| Grammatical Evolution - Symbolic Regression (Poly-10)          | Grammatical evoli   |
| <b>Scripts</b>   |                     |
| Genetic Algorithm Script - QAP                                 | A scripted genetic  |
| GUI Automation Script  | A script that runs  |
| Offspring Selection Genetic Algorithm Script - Rastrigin       | A scripted offsprir |

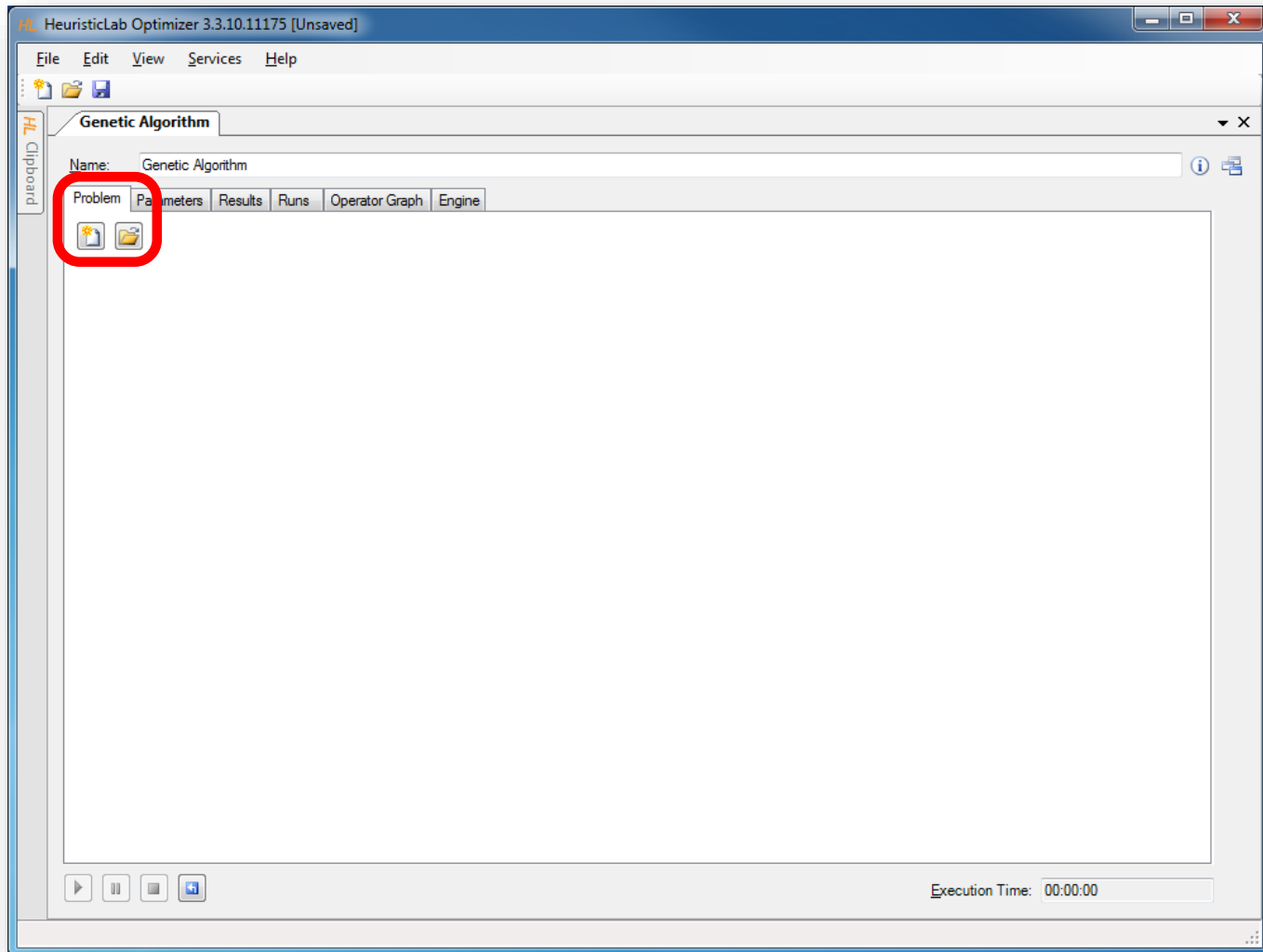
double-click to open  
sample algorithms  
and problems



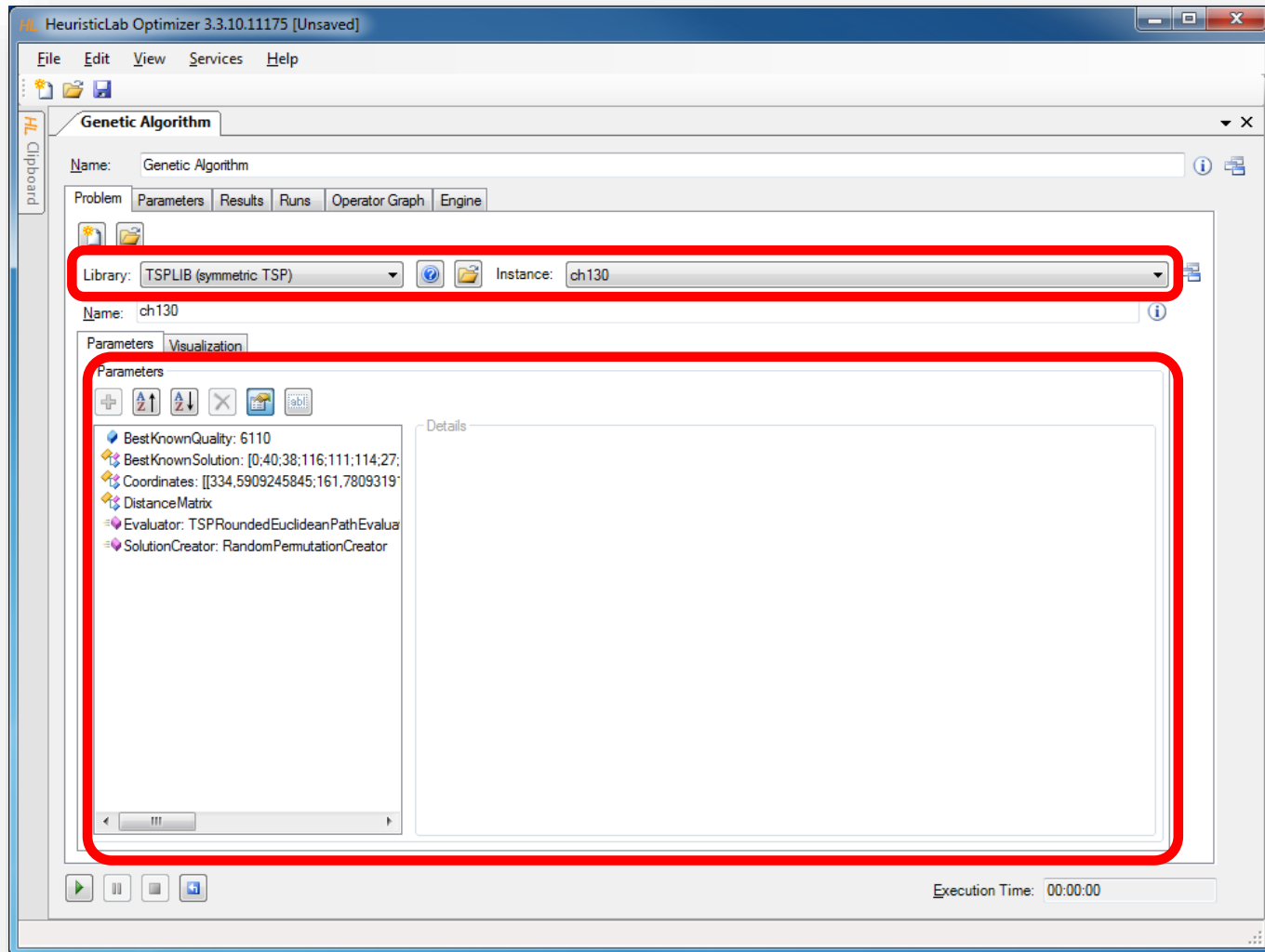
# Create Algorithm



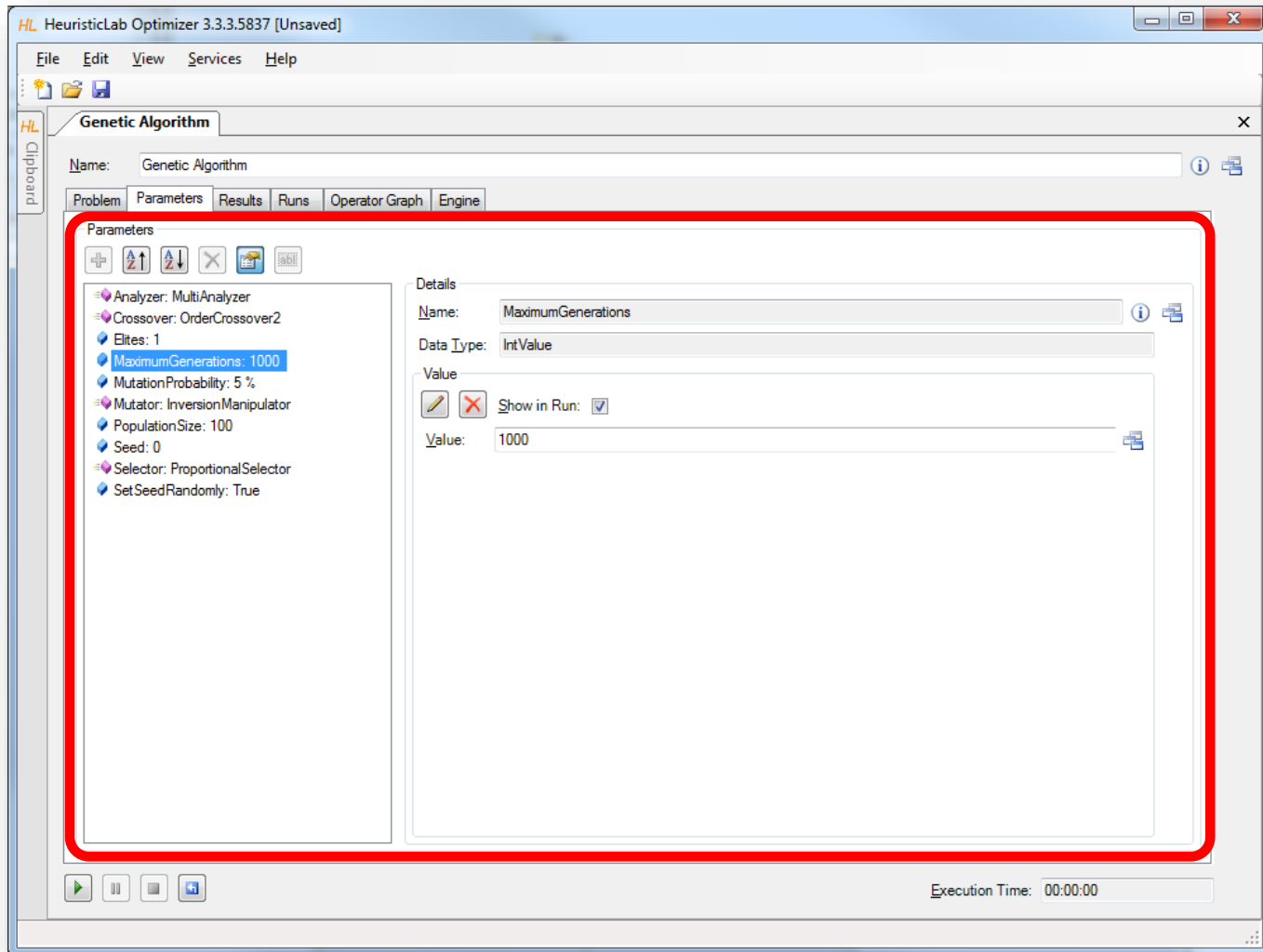
# Create or Load Problem



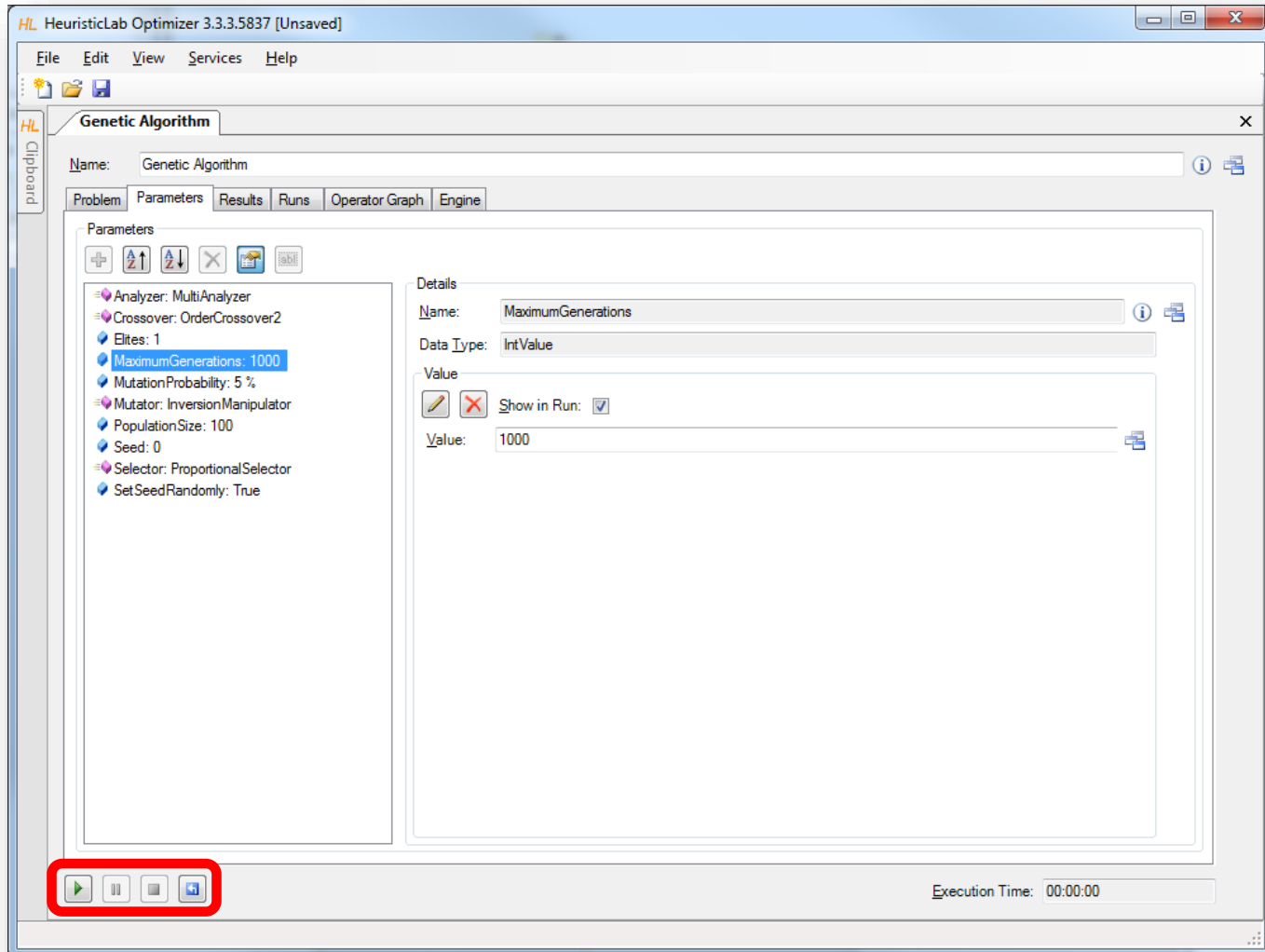
# Import or Parameterize Problem Data



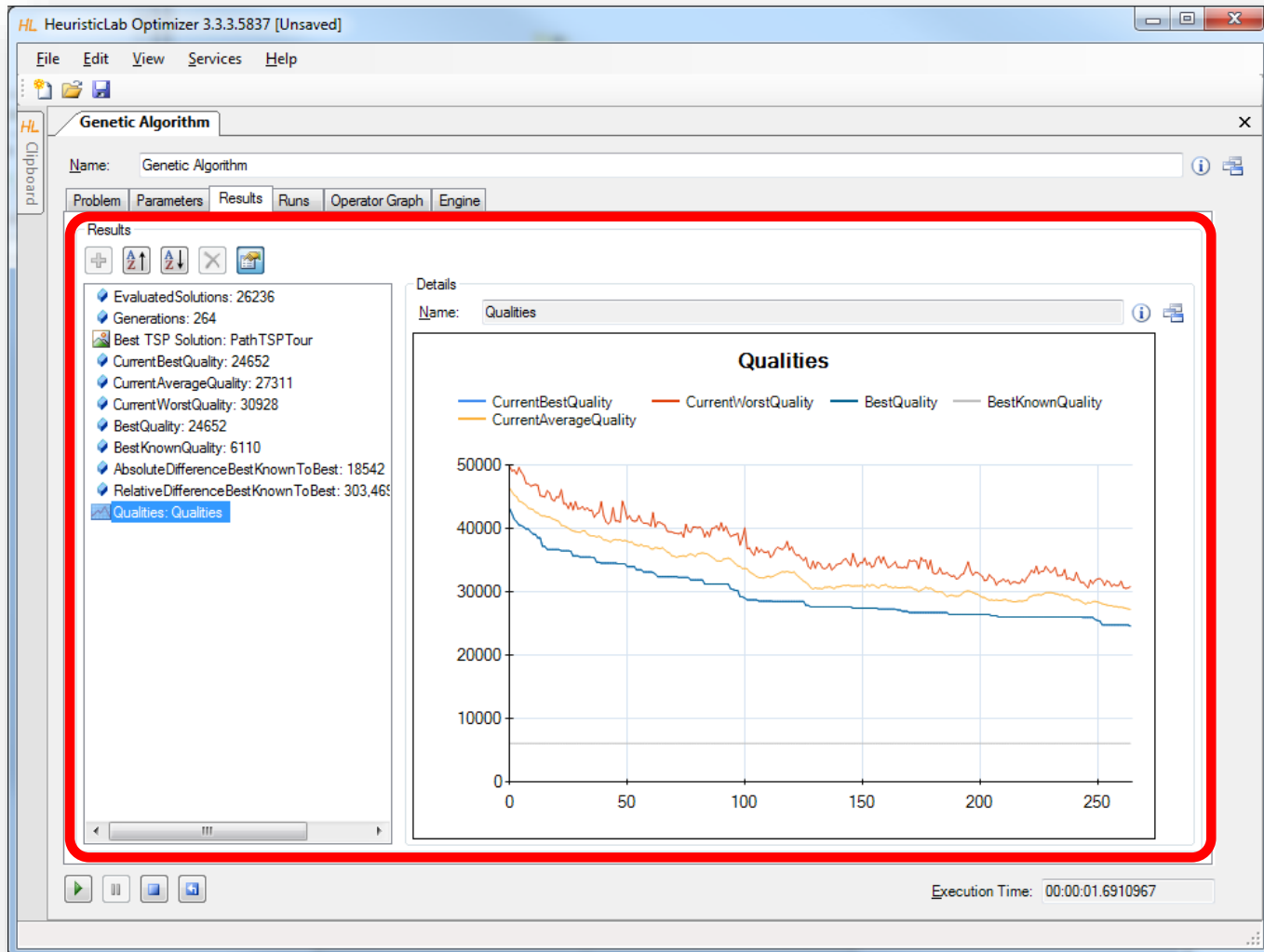
# Parameterize Algorithm



# Start, Pause, Resume, Stop and Reset

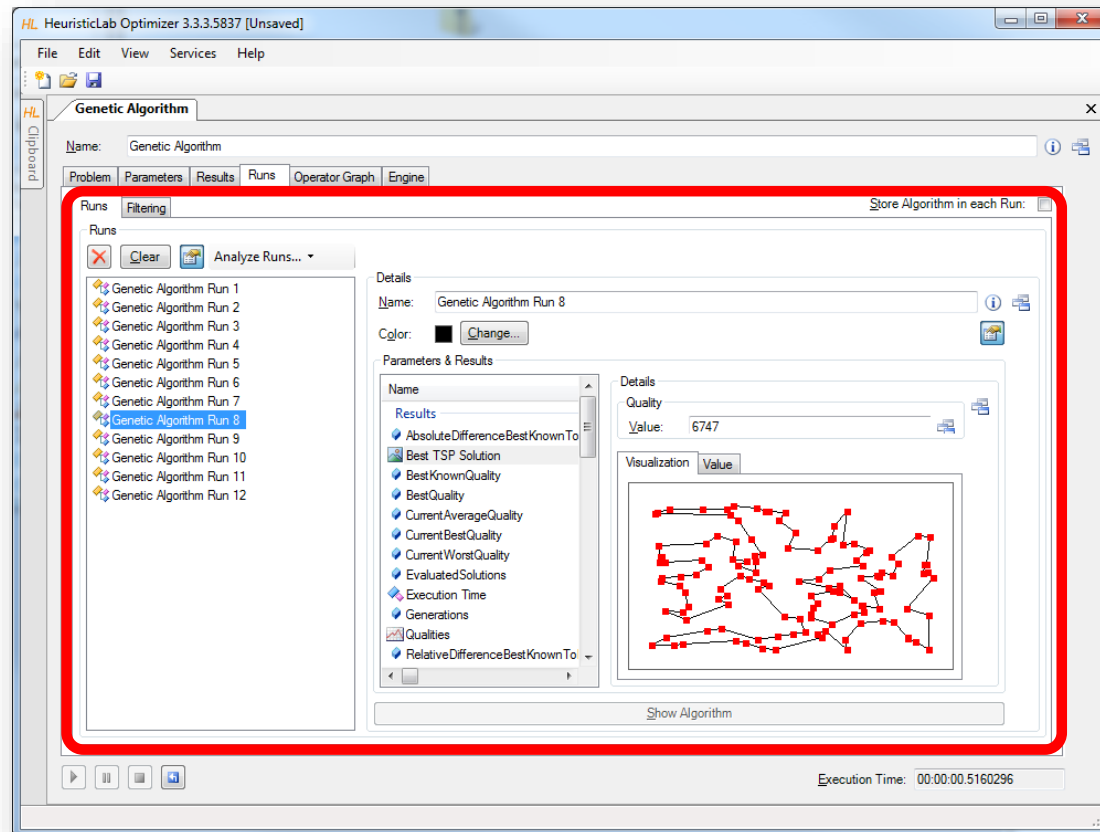


# Inspect Results



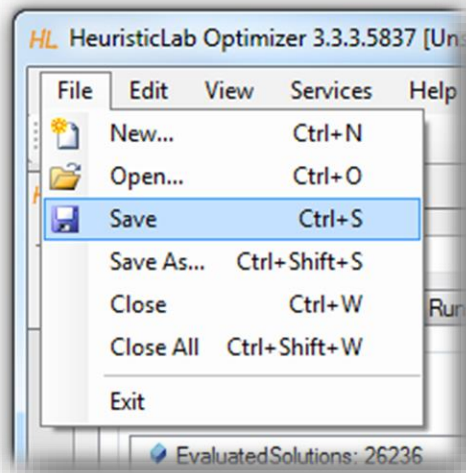
# Compare Runs

- 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

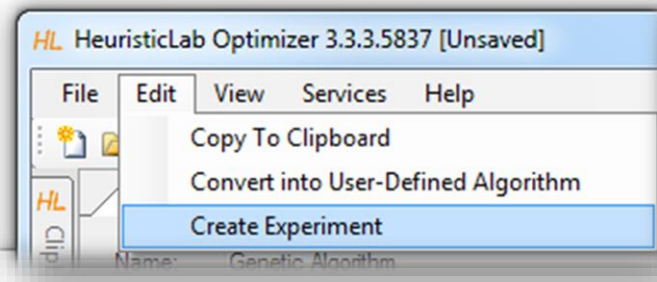
- Save to and load from disk
  - HeuristicLab items (i.e., algorithms, problems, experiments, ...) can be saved to and loaded from a file
  - algorithms can be paused, saved, loaded and resumed
  - data format is custom compressed XML
  - saving and loading files might take several minutes
  - saving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



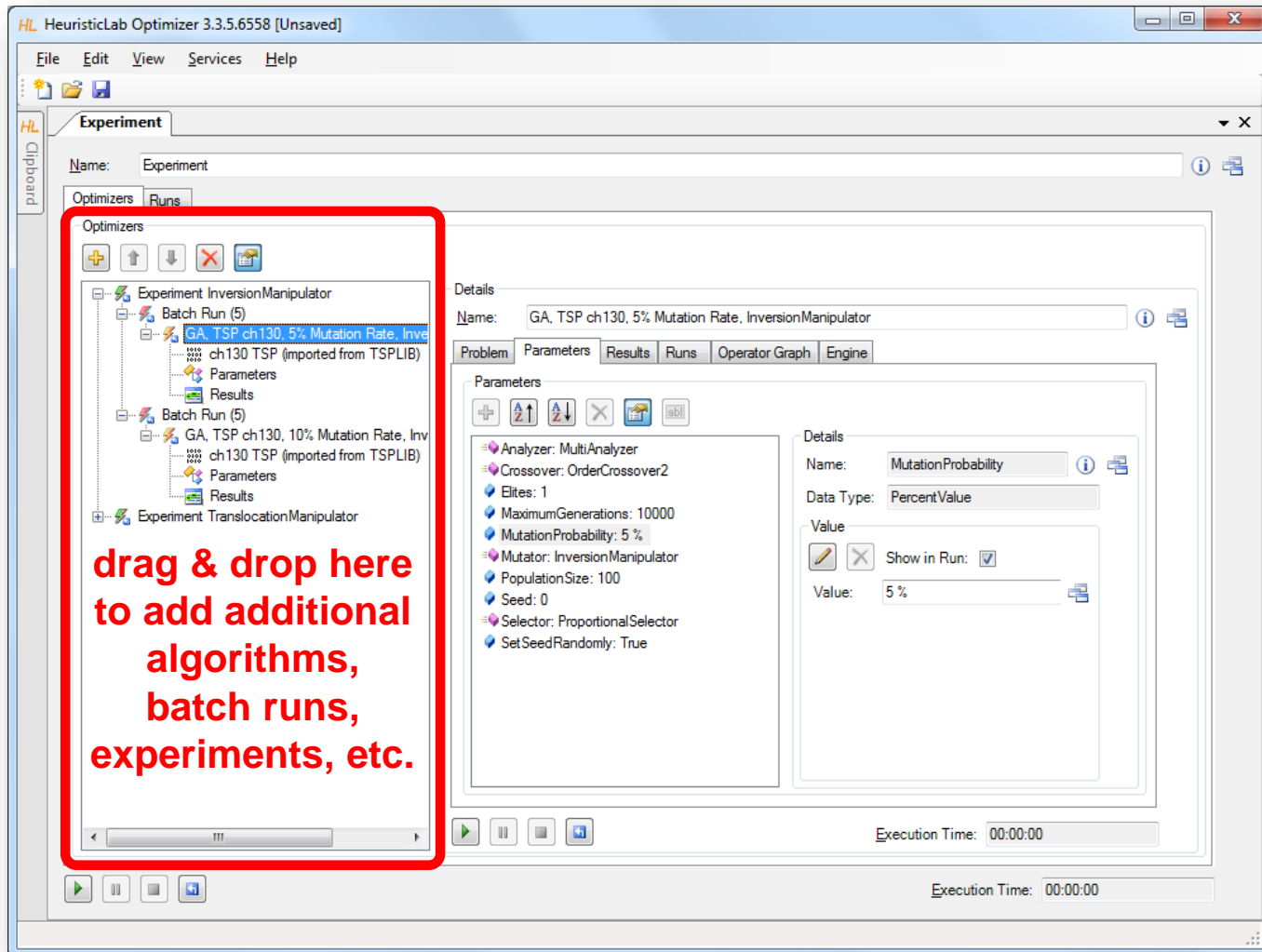


# Create Batch Runs and Experiments

- 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

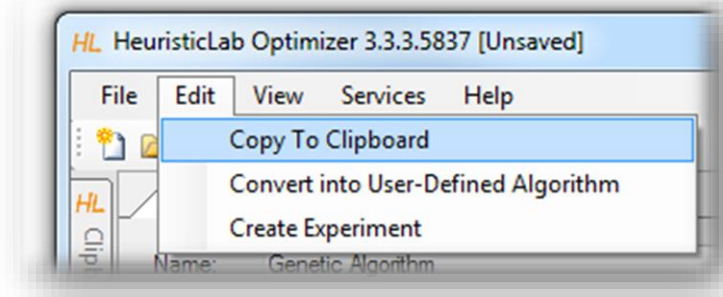


# Create Batch Runs and Experiments



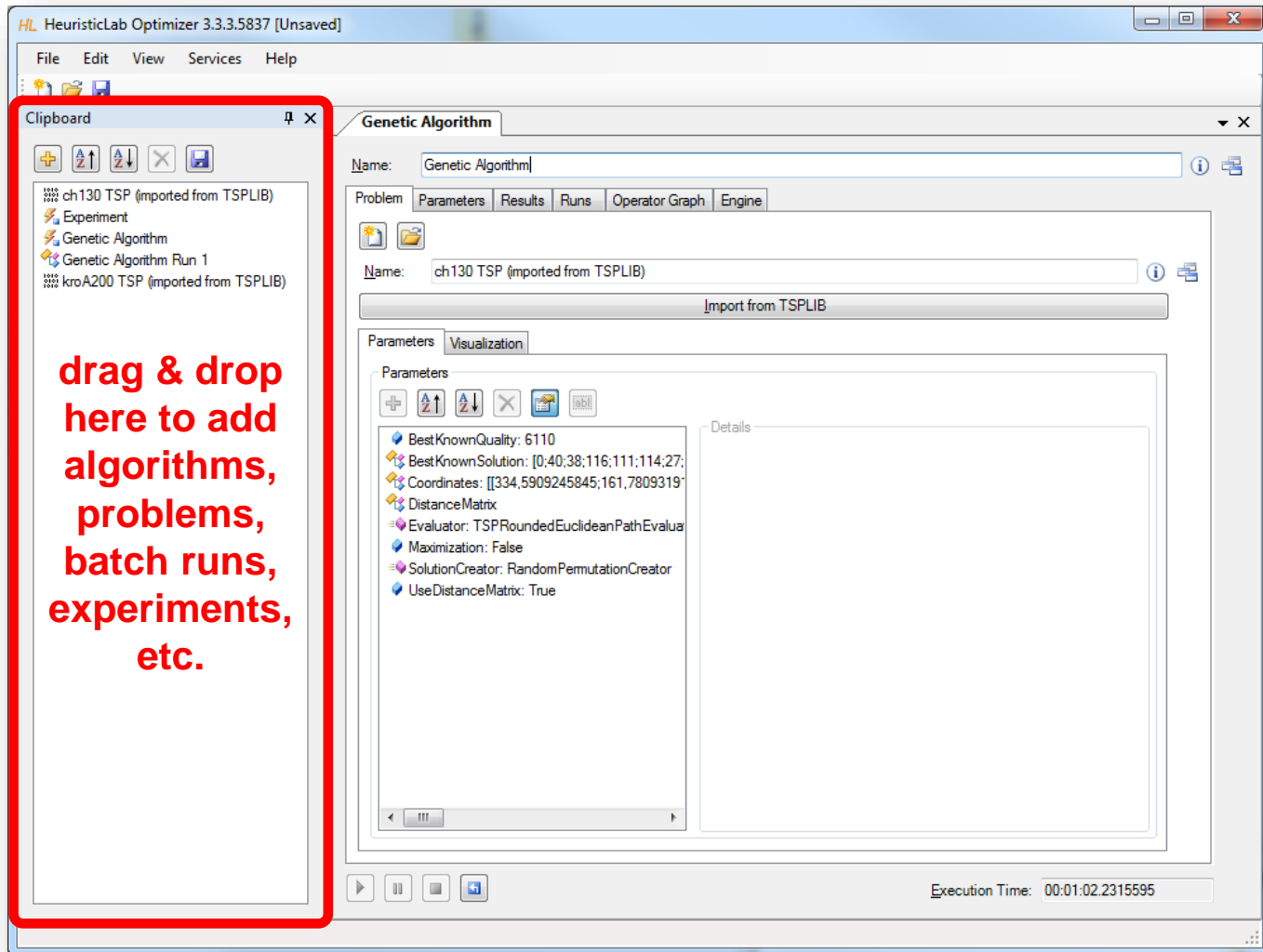
# Clipboard

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

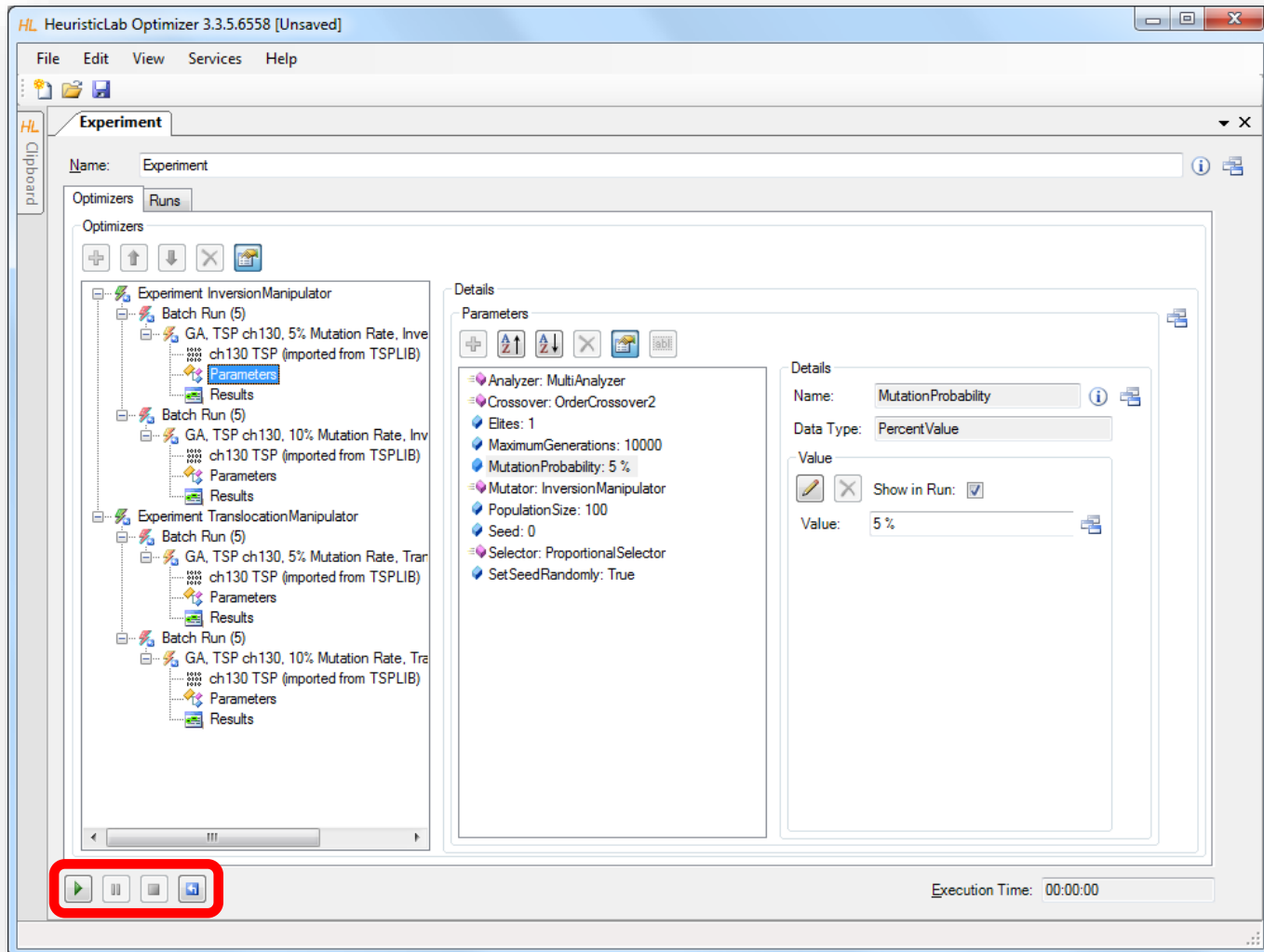


- Show items
  - double-click on an item in the clipboard to show its view
- Save and restore clipboard content
  - click on the save button to write the clipboard content to disk
  - clipboard is automatically restored when HeuristicLab is started the next time

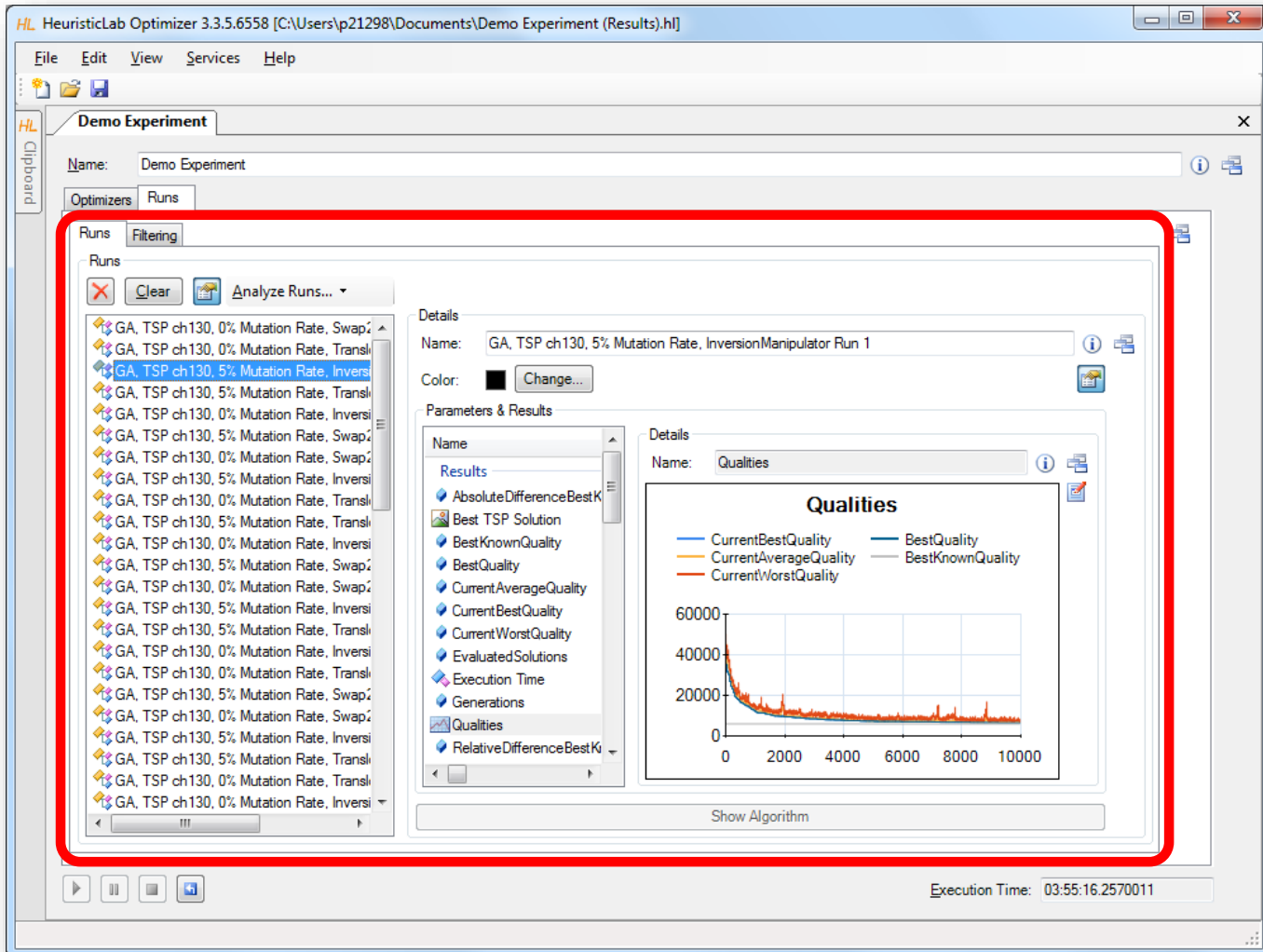
# Clipboard



# Start, Pause, Resume, Stop, Reset



# Compare Runs



The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "Demo Experiment" and shows a list of runs under the "Runs" tab. A red box highlights the "Runs" list and the "Details" panel for a selected run. The "Details" panel shows the name "GA, TSP ch130, 5% Mutation Rate, InversionManipulator Run 1" and a "Qualities" graph. The graph plots "CurrentBestQuality", "CurrentAverageQuality", "CurrentWorstQuality", "BestQuality", and "BestKnownQuality" against "Generations" (0 to 10000). The "CurrentBestQuality" (blue line) starts at approximately 60000 and rapidly descends to about 10000 within the first 1000 generations, then remains stable. The "CurrentAverageQuality" (orange line) and "CurrentWorstQuality" (red line) also show a sharp initial drop and then stabilize around 10000. The "BestQuality" (green line) and "BestKnownQuality" (grey line) are constant at approximately 10000.

HL HeuristicLab Optimizer 3.3.5.6558 [C:\Users\p21298\Documents\Demo Experiment (Results).hl]

File Edit View Services Help

Demo Experiment

Name: Demo Experiment

Optimizers Runs

Runs Filtering

Runs

Clear Analyze Runs...

GA, TSP ch130, 0% Mutation Rate, Swap; GA, TSP ch130, 0% Mutation Rate, Transl; GA, TSP ch130, 5% Mutation Rate, Invers; GA, TSP ch130, 5% Mutation Rate, Transl; GA, TSP ch130, 0% Mutation Rate, Invers; GA, TSP ch130, 5% Mutation Rate, Swap; GA, TSP ch130, 0% Mutation Rate, Swap; GA, TSP ch130, 5% Mutation Rate, Invers; GA, TSP ch130, 0% Mutation Rate, Transl; GA, TSP ch130, 5% Mutation Rate, Swap; GA, TSP ch130, 0% Mutation Rate, Swap; GA, TSP ch130, 5% Mutation Rate, Invers; GA, TSP ch130, 5% Mutation Rate, Transl; GA, TSP ch130, 0% Mutation Rate, Invers; GA, TSP ch130, 5% Mutation Rate, Transl; GA, TSP ch130, 0% Mutation Rate, Invers; GA, TSP ch130, 5% Mutation Rate, Swap; GA, TSP ch130, 0% Mutation Rate, Swap; GA, TSP ch130, 5% Mutation Rate, Invers; GA, TSP ch130, 5% Mutation Rate, Transl; GA, TSP ch130, 0% Mutation Rate, Transl; GA, TSP ch130, 0% Mutation Rate, Invers

Details

Name: GA, TSP ch130, 5% Mutation Rate, InversionManipulator Run 1

Color: Change...

Parameters & Results

Name

Results

- AbsoluteDifferenceBestK
- Best TSP Solution
- BestKnownQuality
- BestQuality
- CurrentAverageQuality
- CurrentBestQuality
- CurrentWorstQuality
- EvaluatedSolutions
- Execution Time
- Generations
- Qualities
- RelativeDifferenceBestK

Details

Name: Qualities

Qualities

- CurrentBestQuality
- CurrentAverageQuality
- CurrentWorstQuality
- BestQuality
- BestKnownQuality

60000  
40000  
20000  
0

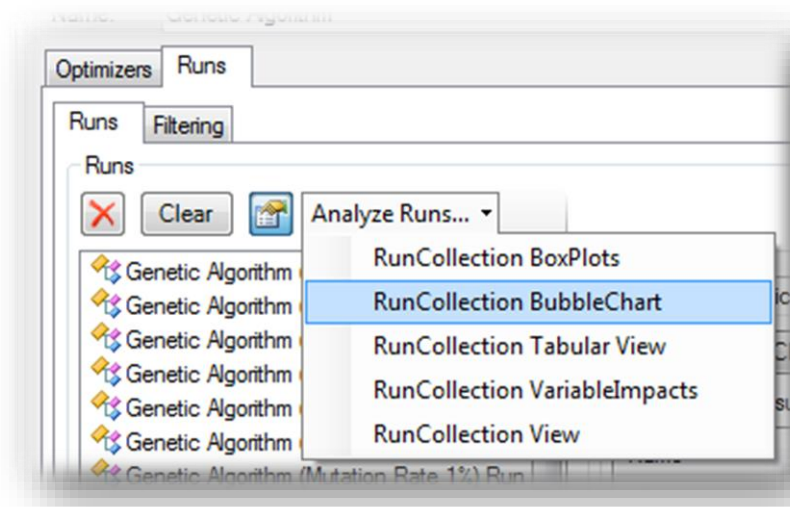
0 2000 4000 6000 8000 10000

Show Algorithm

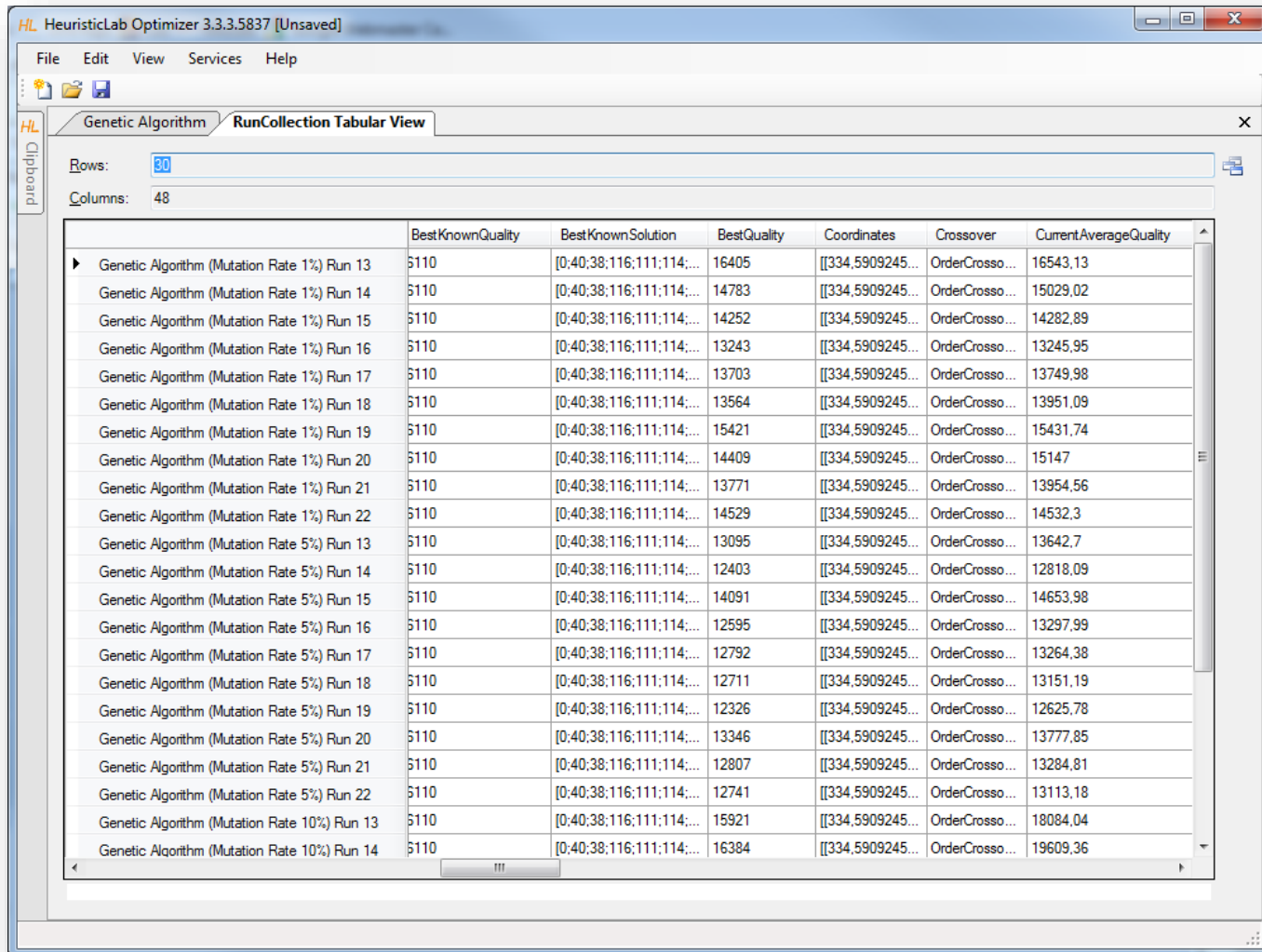
Execution Time: 03:55:16.2570011

# 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



# Runs – Tabular View



|   | 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              |

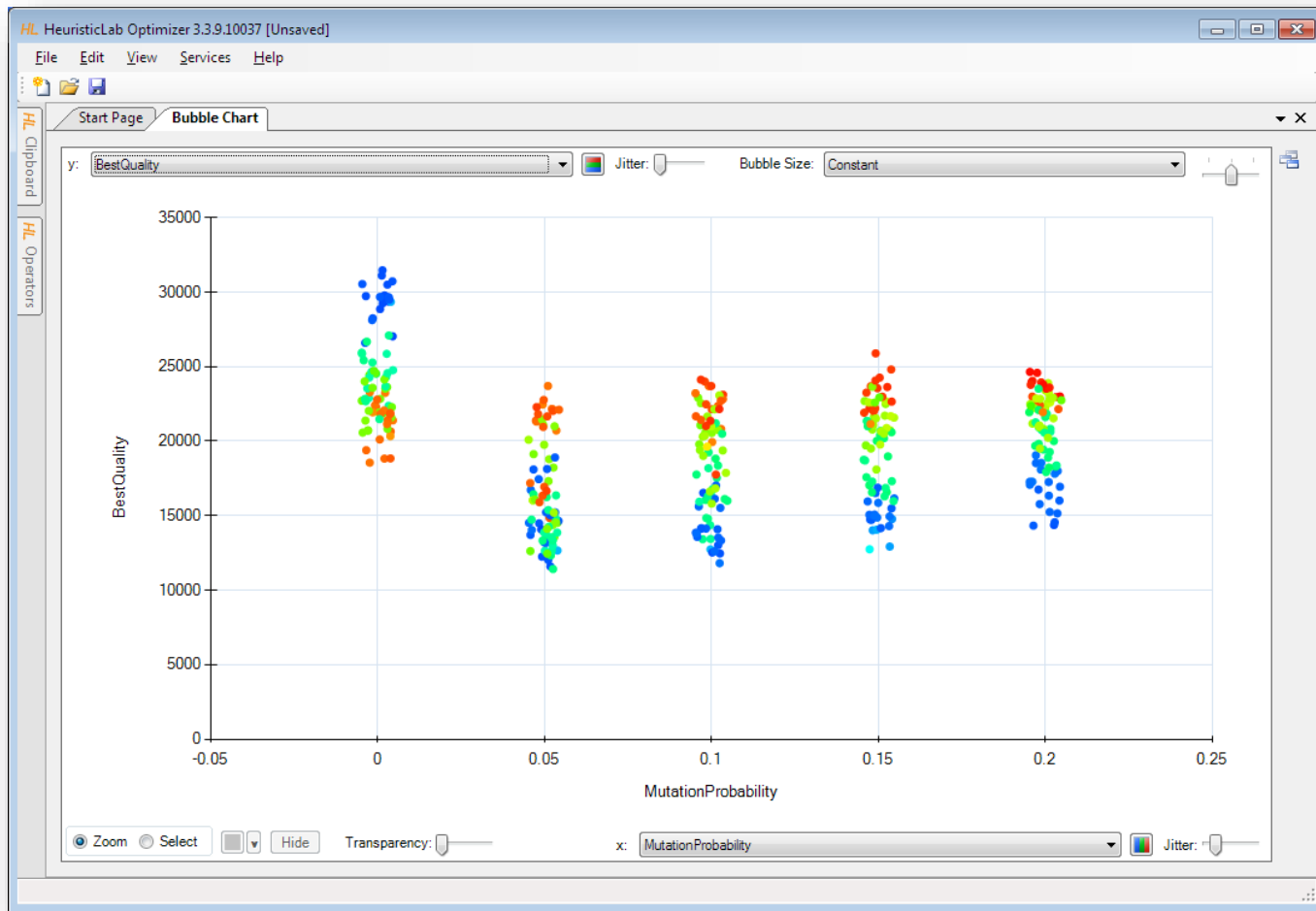


# Runs – Tabular View



- Sort columns
  - click on column header to sort column
  - Ctrl-click on column header to sort multiple columns
- Show or hide columns
  - right-click on table to open dialog to show or hide columns
- Compute statistical values
  - select multiple numerical values to see count, sum, minimum, maximum, average and standard deviation
- Select, copy and paste into other applications

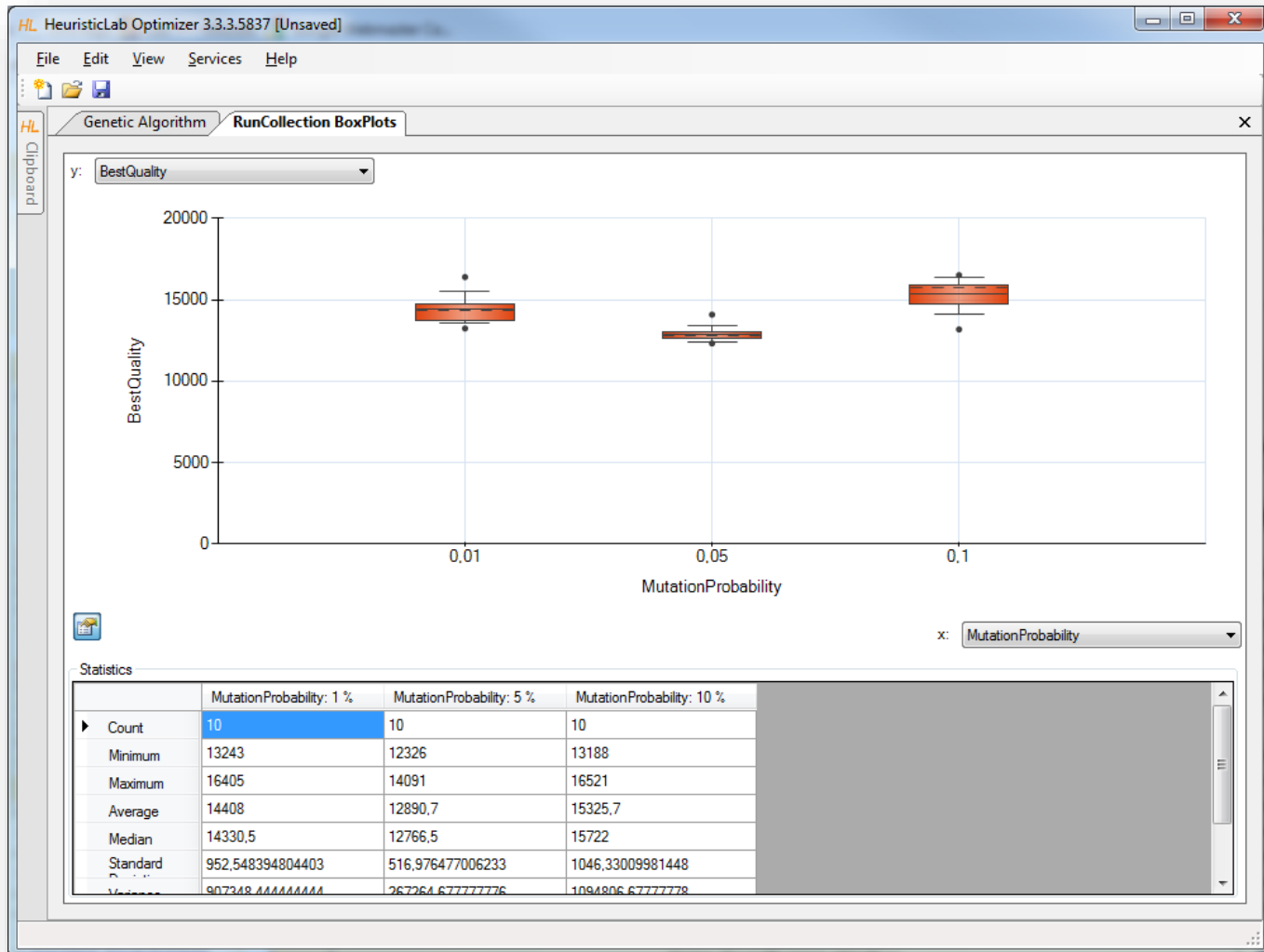
# Runs – BubbleChart



# Runs – BubbleChart

- Choose values to plot
  - choose which values to show on the x-axis, the y-axis and as bubble size
  - possible values are all parameter settings and results
- Add jitter
  - add jitter to separate overlapping bubbles
- Zoom in and out
  - click on Zoom and click and drag in the chart area to zoom in
  - double click on the chart area background or on the circle buttons beside the scroll bars to zoom out
- Color bubbles
  - click on Select, choose a color and click and drag in the chart area to select and color bubbles
  - apply coloring automatically by clicking on the axis coloring buttons
- Show runs
  - double click on a bubble to open its run
- Export image
  - right-click to open context menu to copy or save image
  - save image as pixel (BMP, JPG, PNG, GIF, TIF) or vector graphics (EMF)
- Show box plots
  - right-click to open context menu to show box plots view

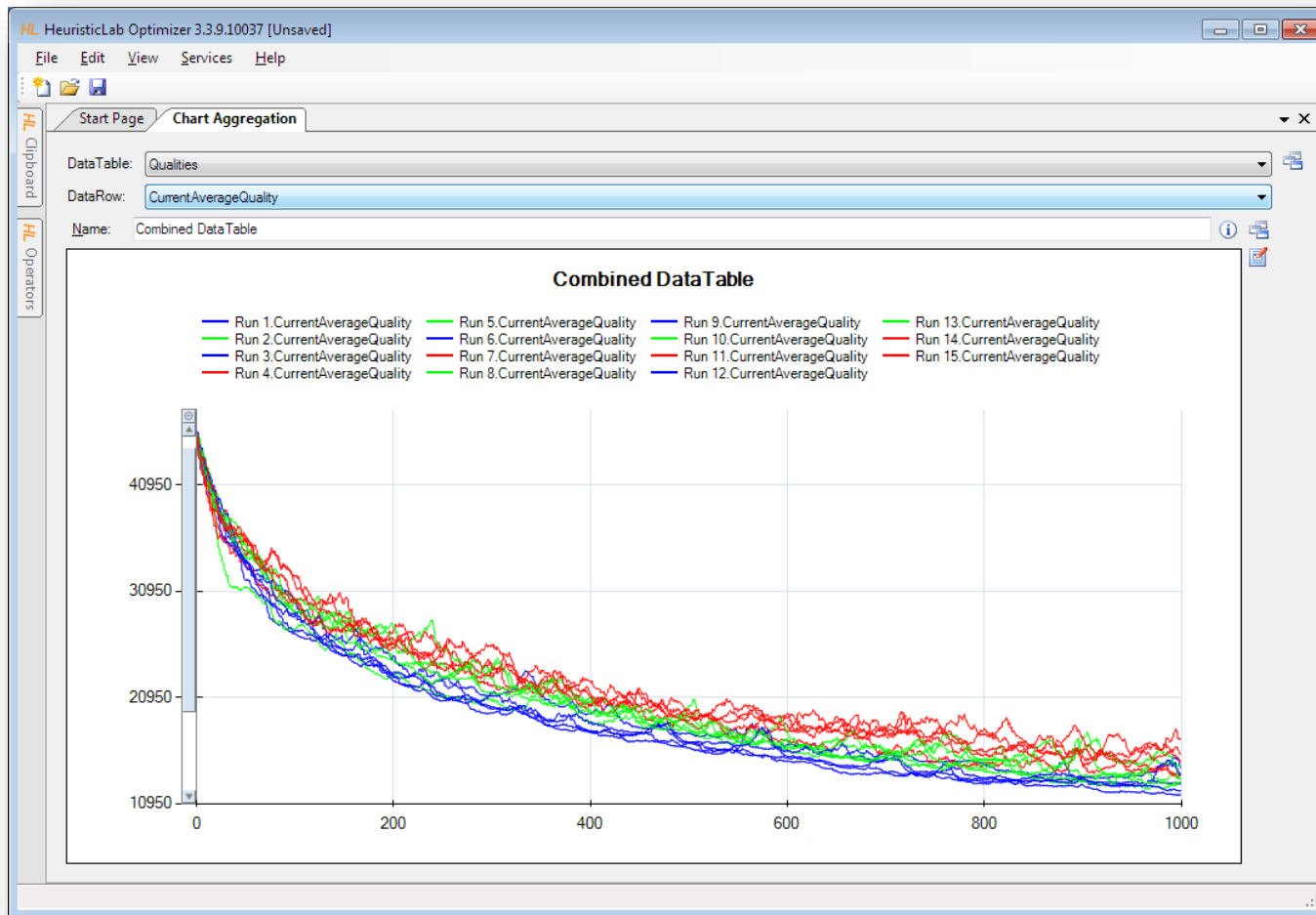
# Runs – BoxPlots



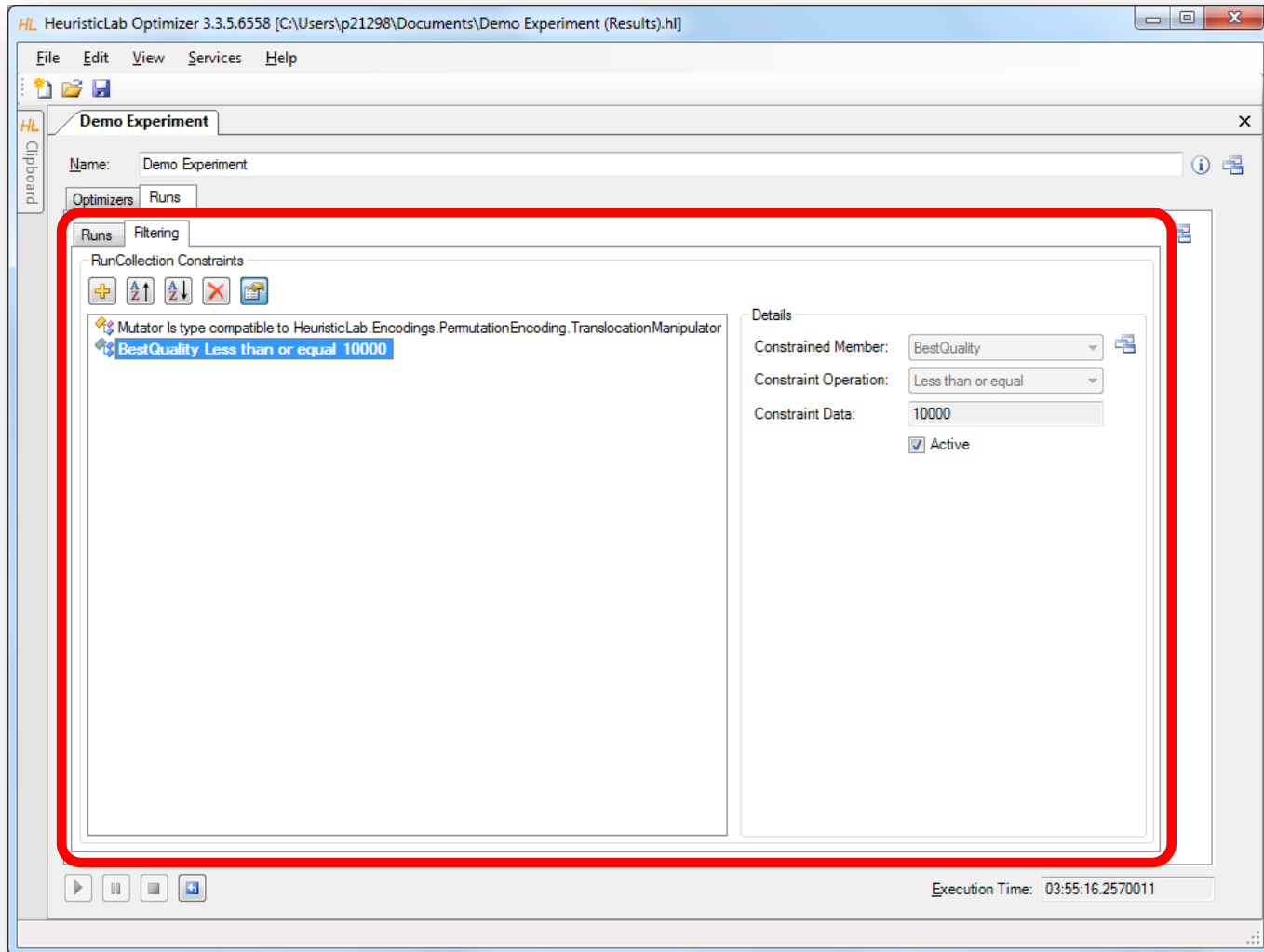
# Runs – BoxPlots

- Choose values to plot
  - choose which values to show on the x-axis and y-axis
  - possible values are all parameter settings and results
- Zoom in and out
  - click on Zoom and click and drag in the chart area to zoom in
  - double click on the chart area background or on the circle buttons beside the scroll bars to zoom out
- Show or hide statistical values
  - click on the lower left button to show or hide statistical values
- Export image
  - right-click to open context menu to copy or save image
  - save image as pixel (BMP, JPG, PNG, GIF, TIF) or vector graphics (EMF)

# Runs – Multi-Line Chart



# Filter Runs



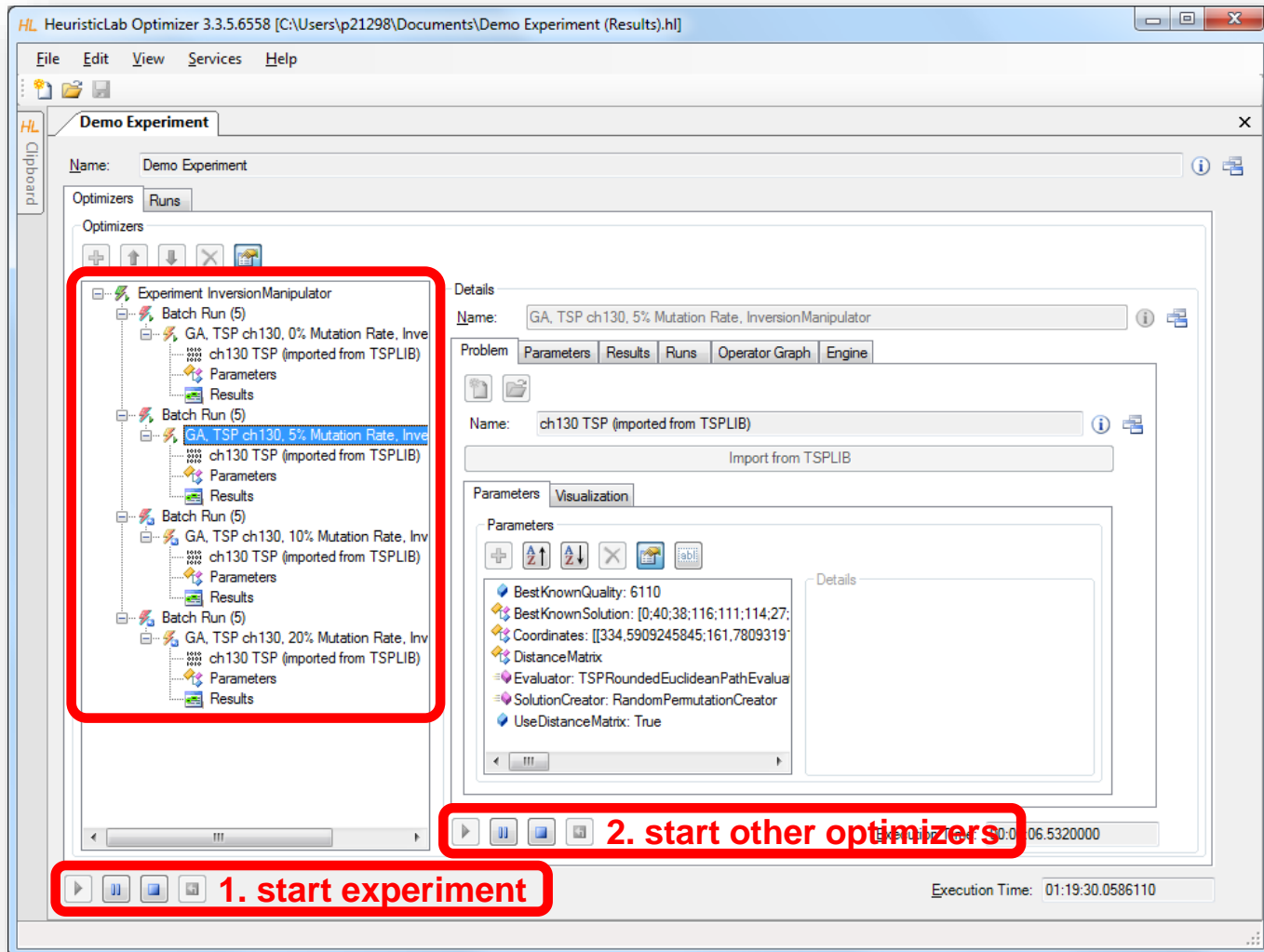
# Multi-core CPUs and Parallelization



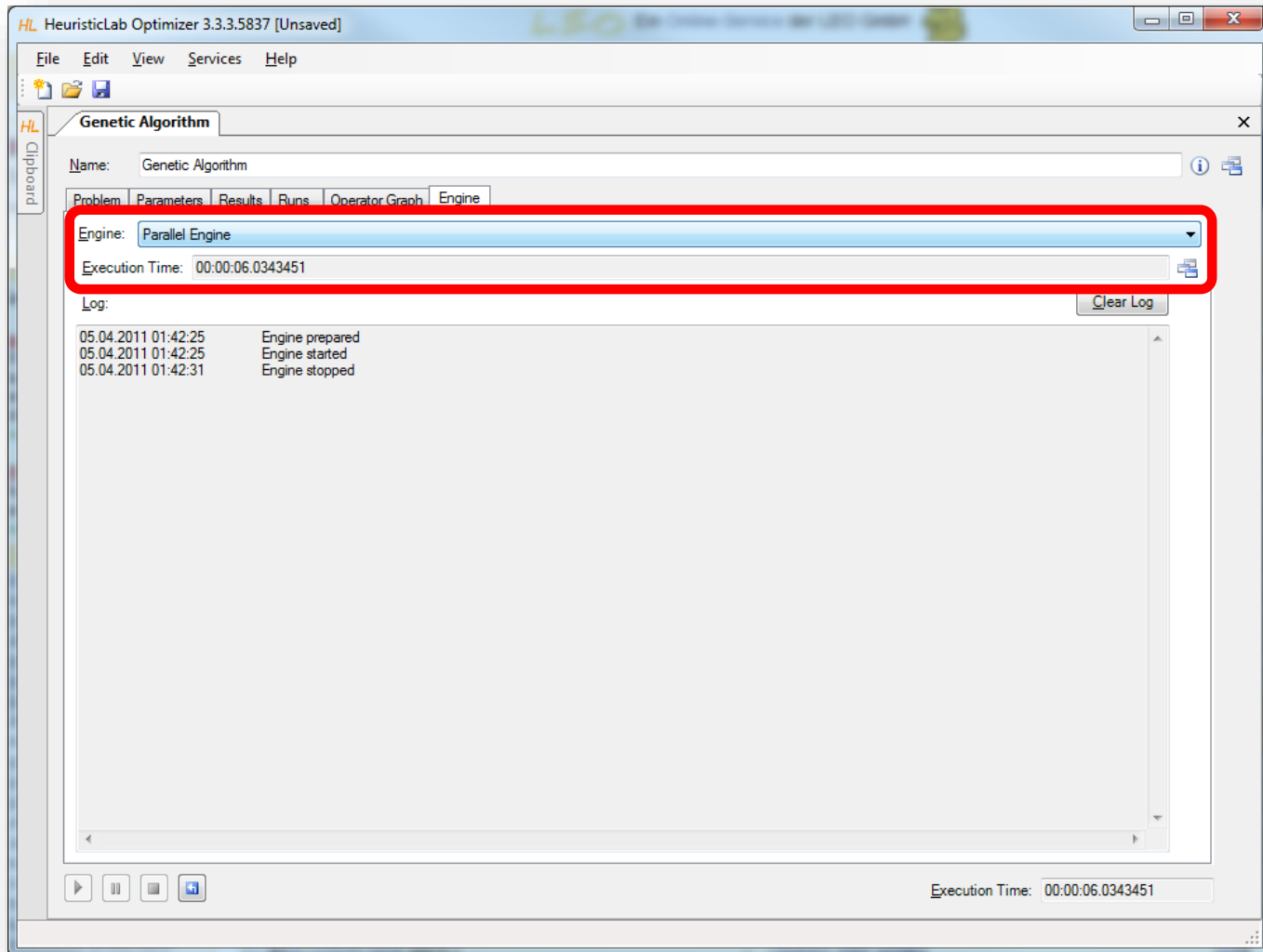
- Parallel execution of optimizers in experiments
  - optimizers in an experiment are executed sequentially from top to bottom per default
  - experiments support parallel execution of their optimizers
  - select a not yet executed optimizer and start it manually to utilize another core
  - execution of one of the next optimizers is started automatically after an optimizer is finished
- Parallel execution of algorithms
  - HeuristicLab provides special operators for parallelization
  - engines decide how to execute parallel operations
  - sequential engine executes everything sequentially
  - parallel engine executes parallel operations on multiple cores
  - Hive engine (under development) executes parallel operations on multiple computers
  - all implemented algorithms support parallel solution evaluation



# Parallel Execution of Experiments



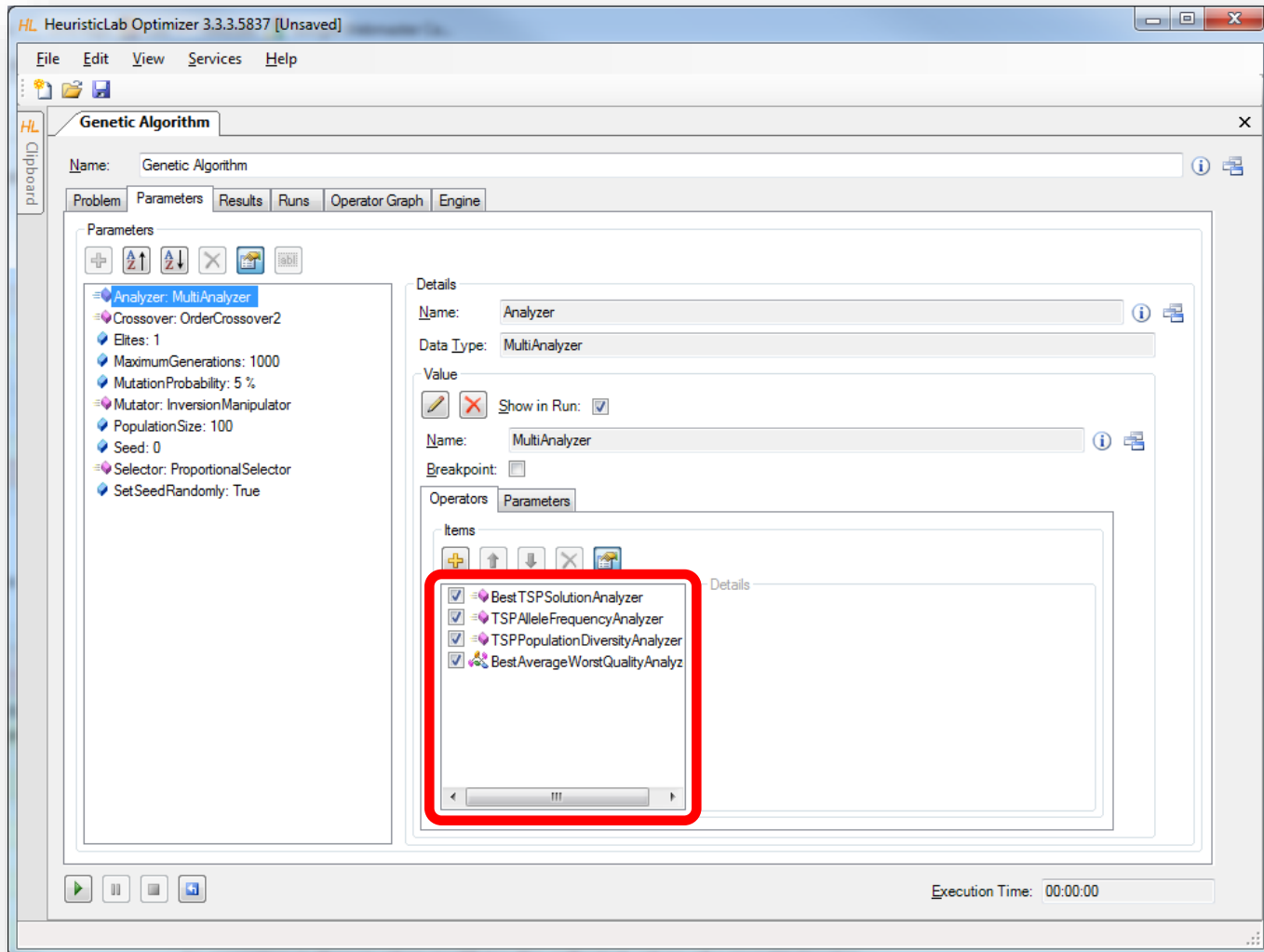
# Parallel Execution of Algorithms



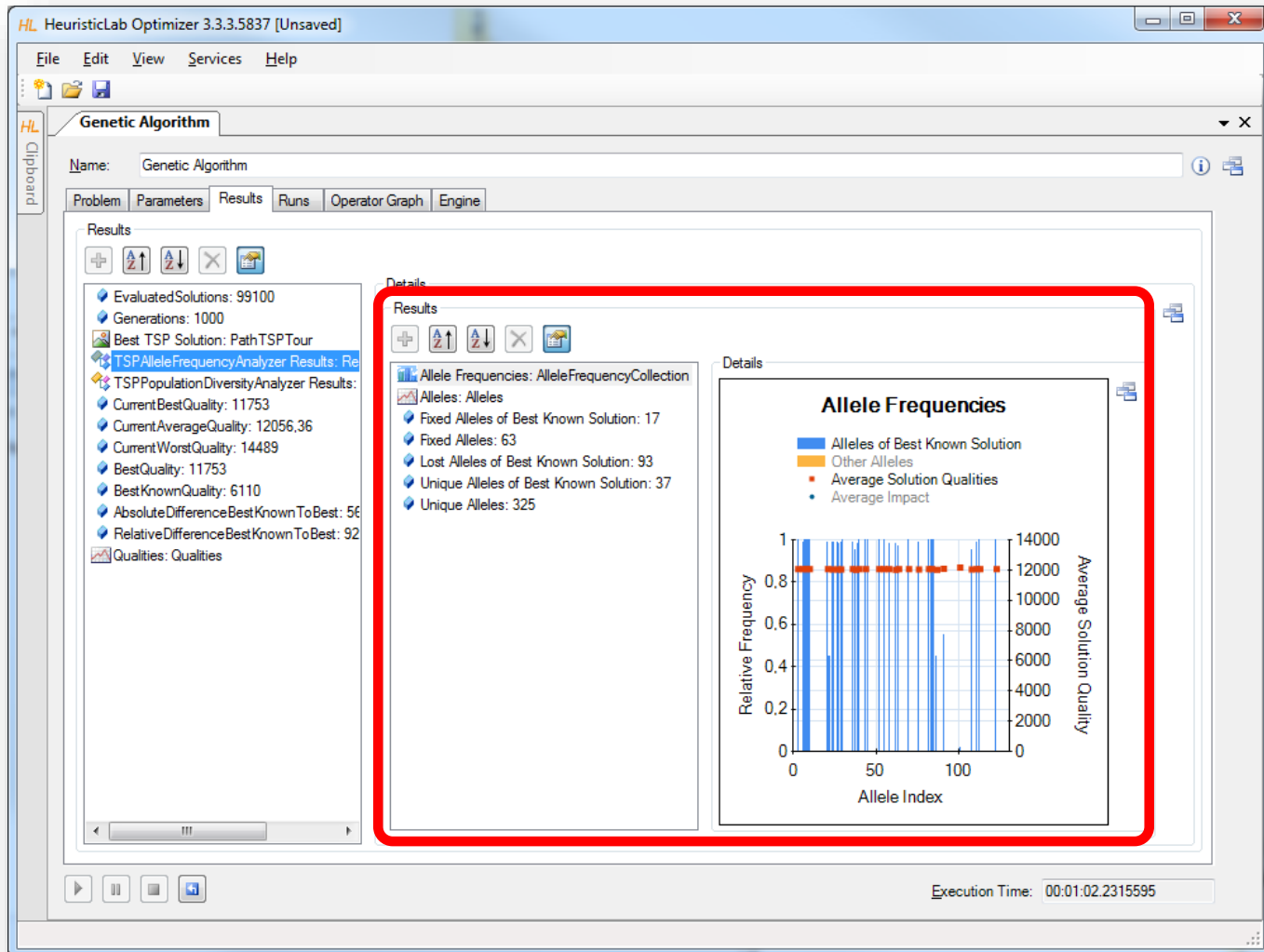
# Analyzers

- 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
  - ...

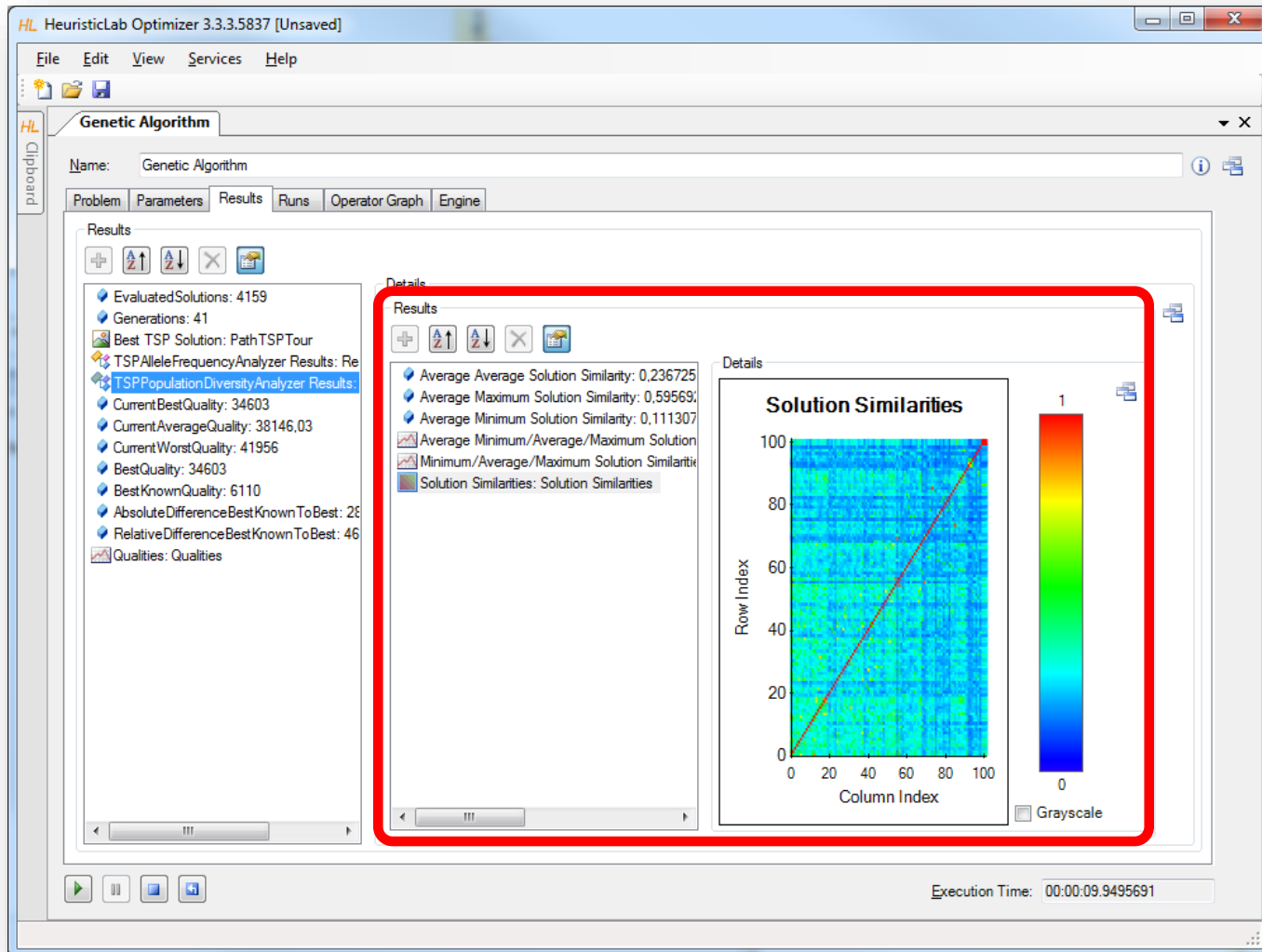
# Analyzers



# TSPAlleleFrequencyAnalyzer

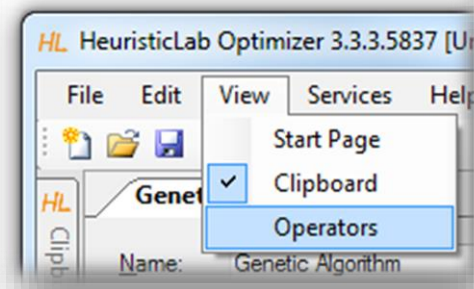
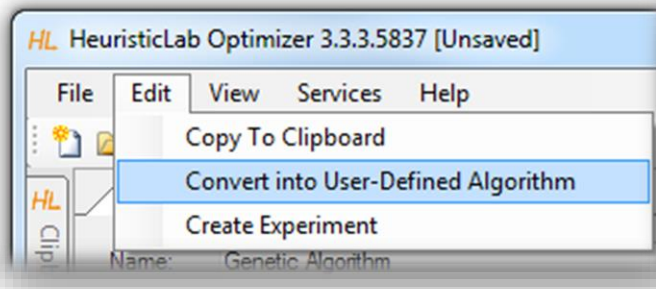


# TSP Population Diversity Analyzer



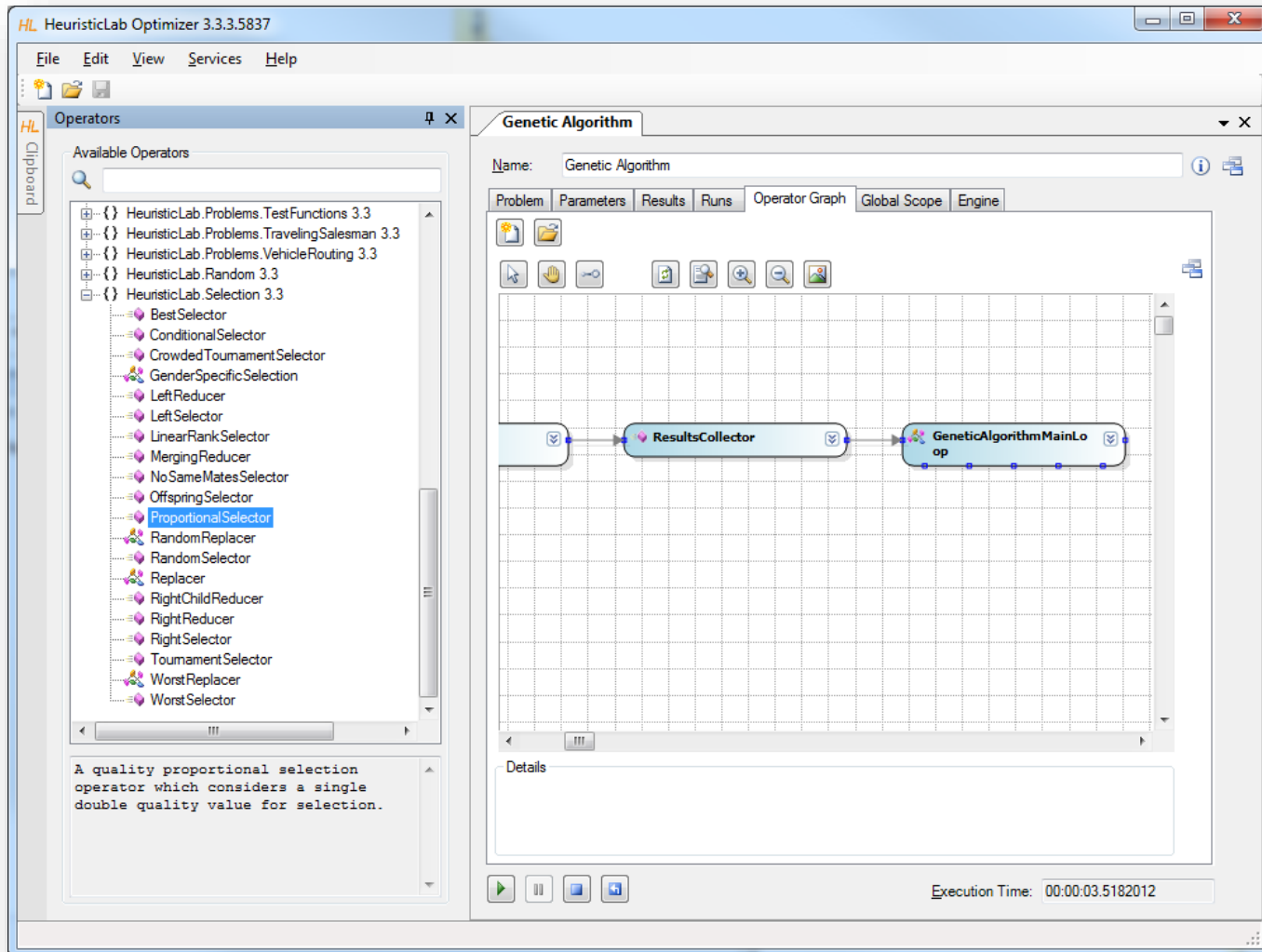
# Building User-Defined Algorithms

- 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

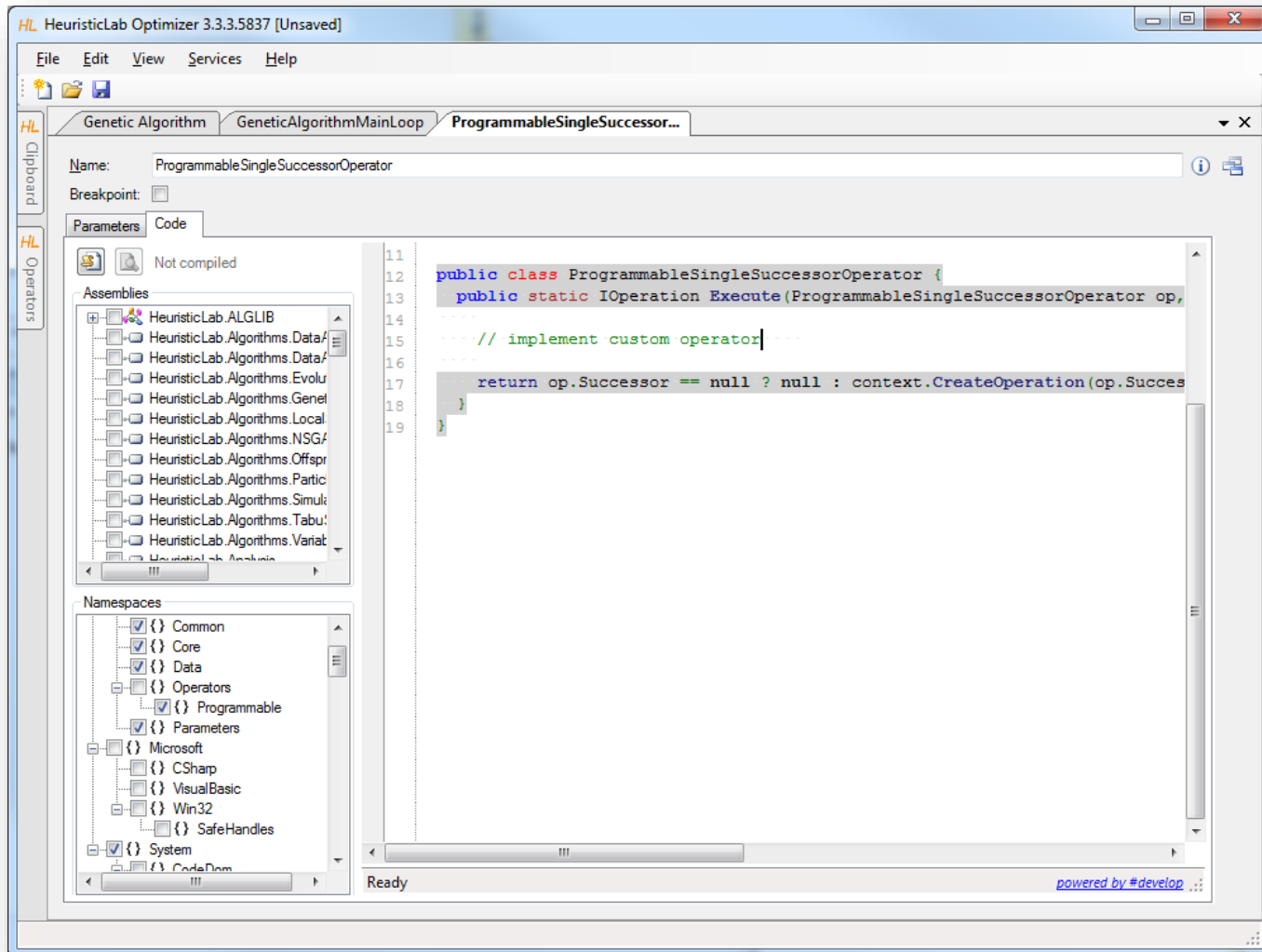
# Building User-Defined Algorithms





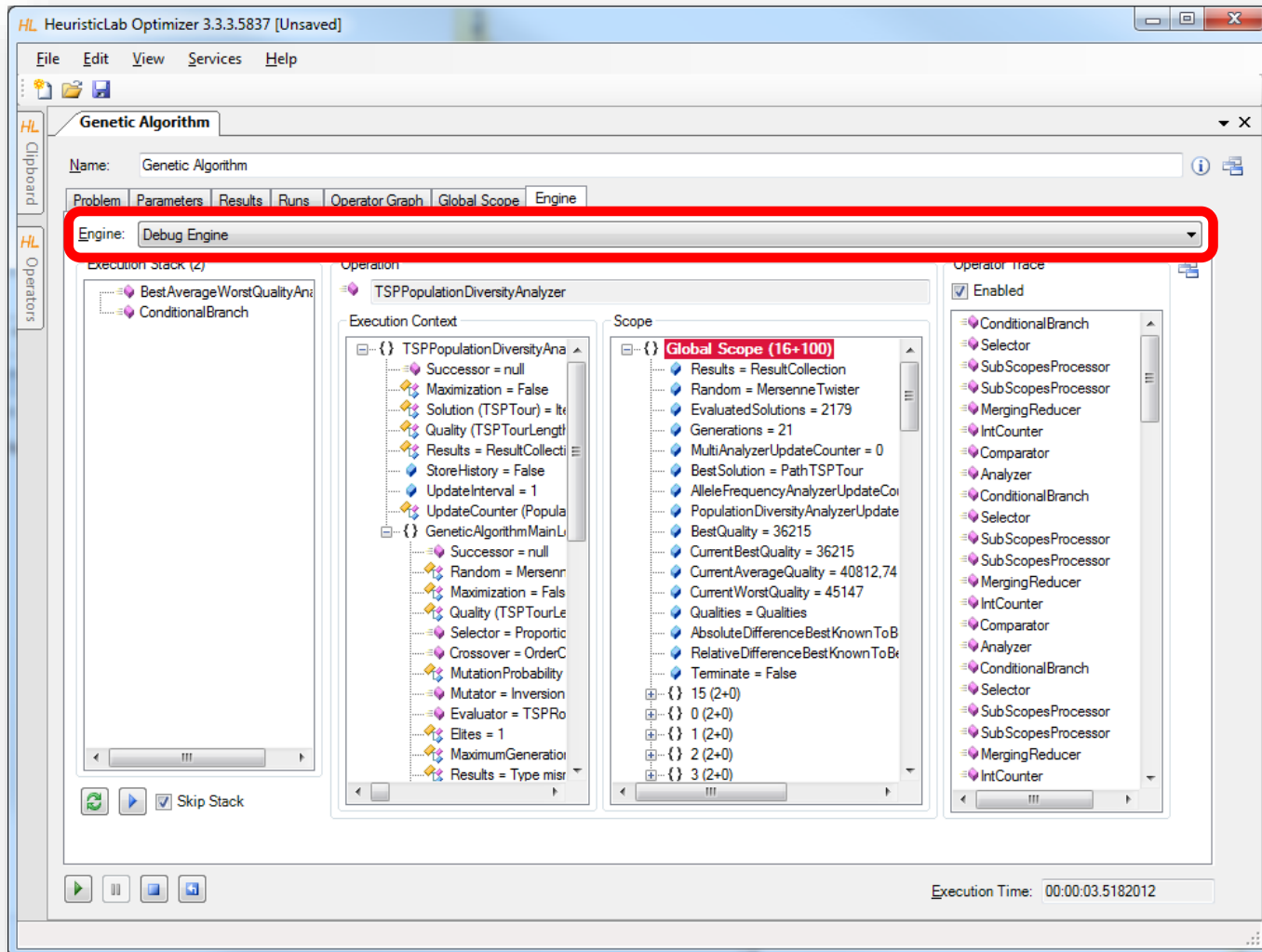


# Programmable Operators





# 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

# Data Analysis in HeuristicLab



- Symbolic regression and classification using genetic programming
- External Libraries:
  - Linear Regression, Logistic Regression,
  - k-Nearest Neighbours, k-Means,
  - Random Forest, Support Vector Machines, Neural Networks, Gaussian Processes

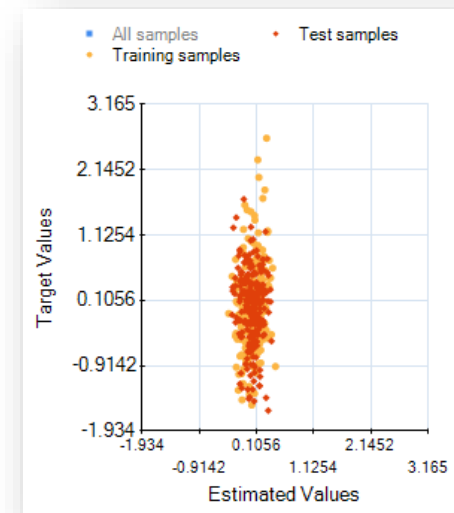
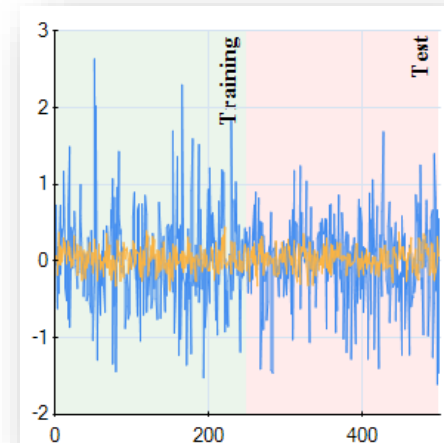


# Case Study: Regression

- Poly-10 benchmark problem dataset
  - 10 input variables  $x_1 \dots x_{10}$
  - $y = x_1x_2 + x_3x_4 + x_5x_6 + x_1x_7x_9 + x_3x_6x_{10}$
  - non-linear modeling approach necessary
  - frequently used in GP literature
  - available as benchmark problem instance in HeuristicLab

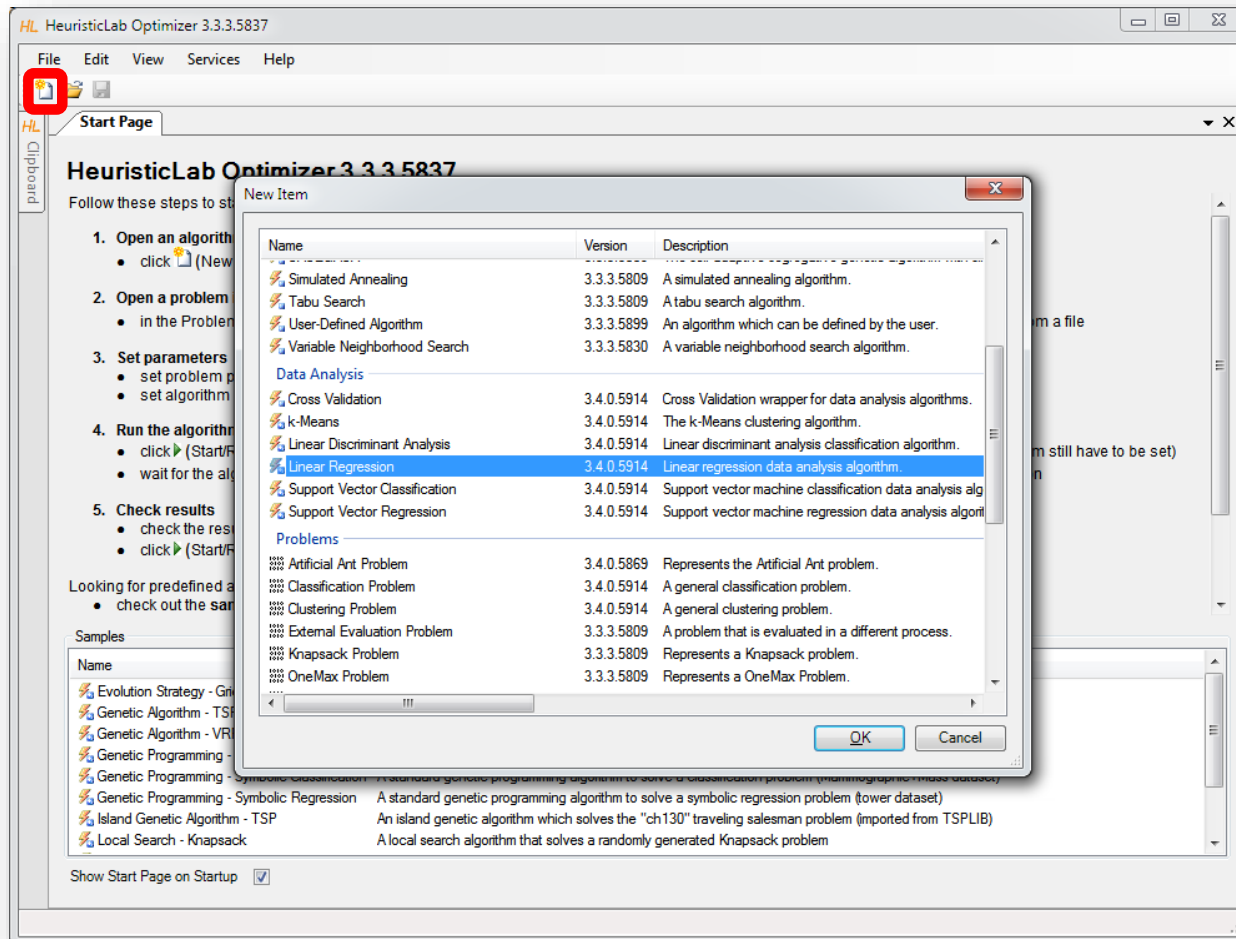
# Demonstration

- problem configuration
  - data import
  - target and input variables
  - data partitions (training and test)
- algorithm configuration
- analysis of results
  - accuracy metrics
  - visualization of model output

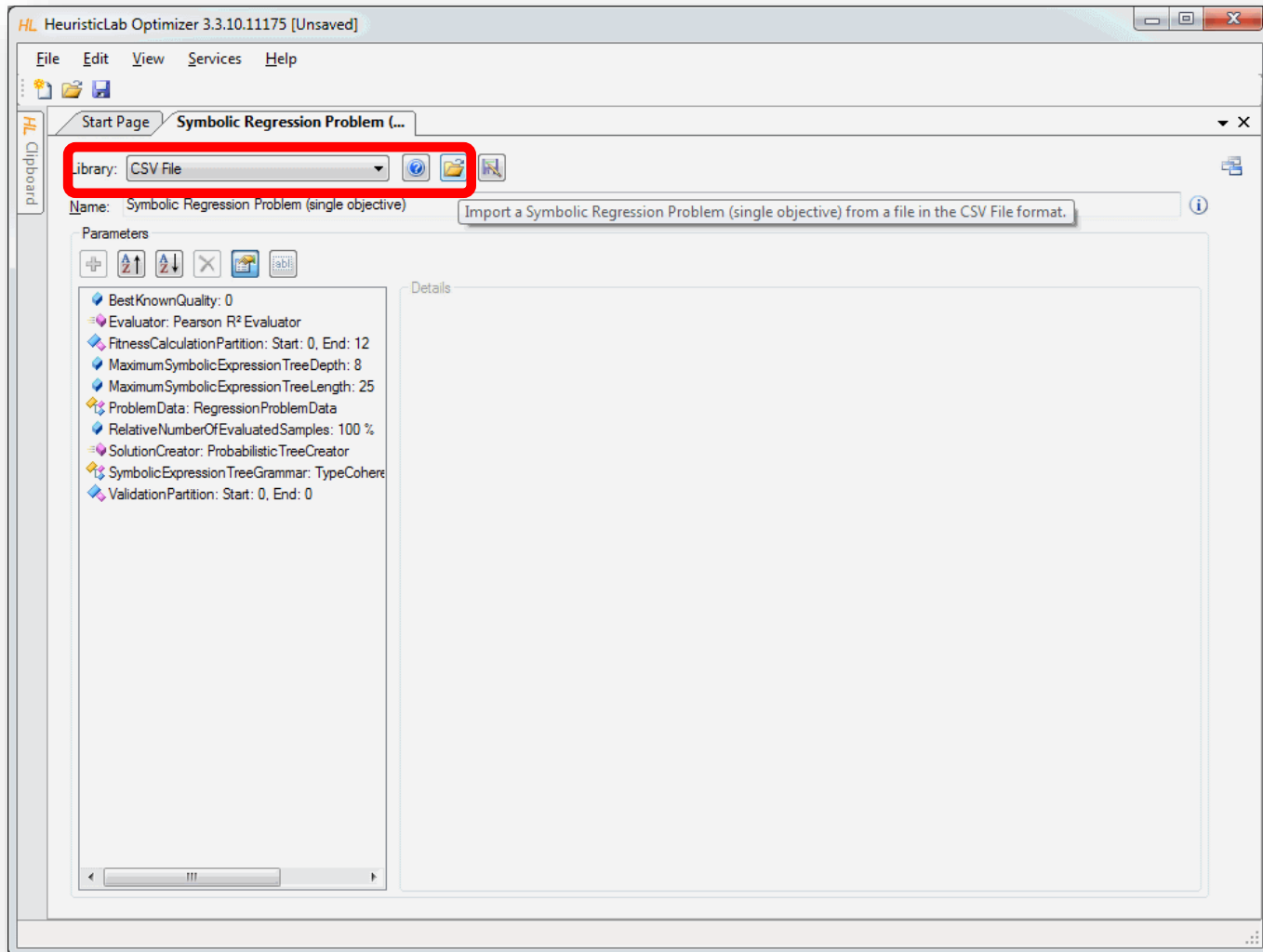


# Linear Regression

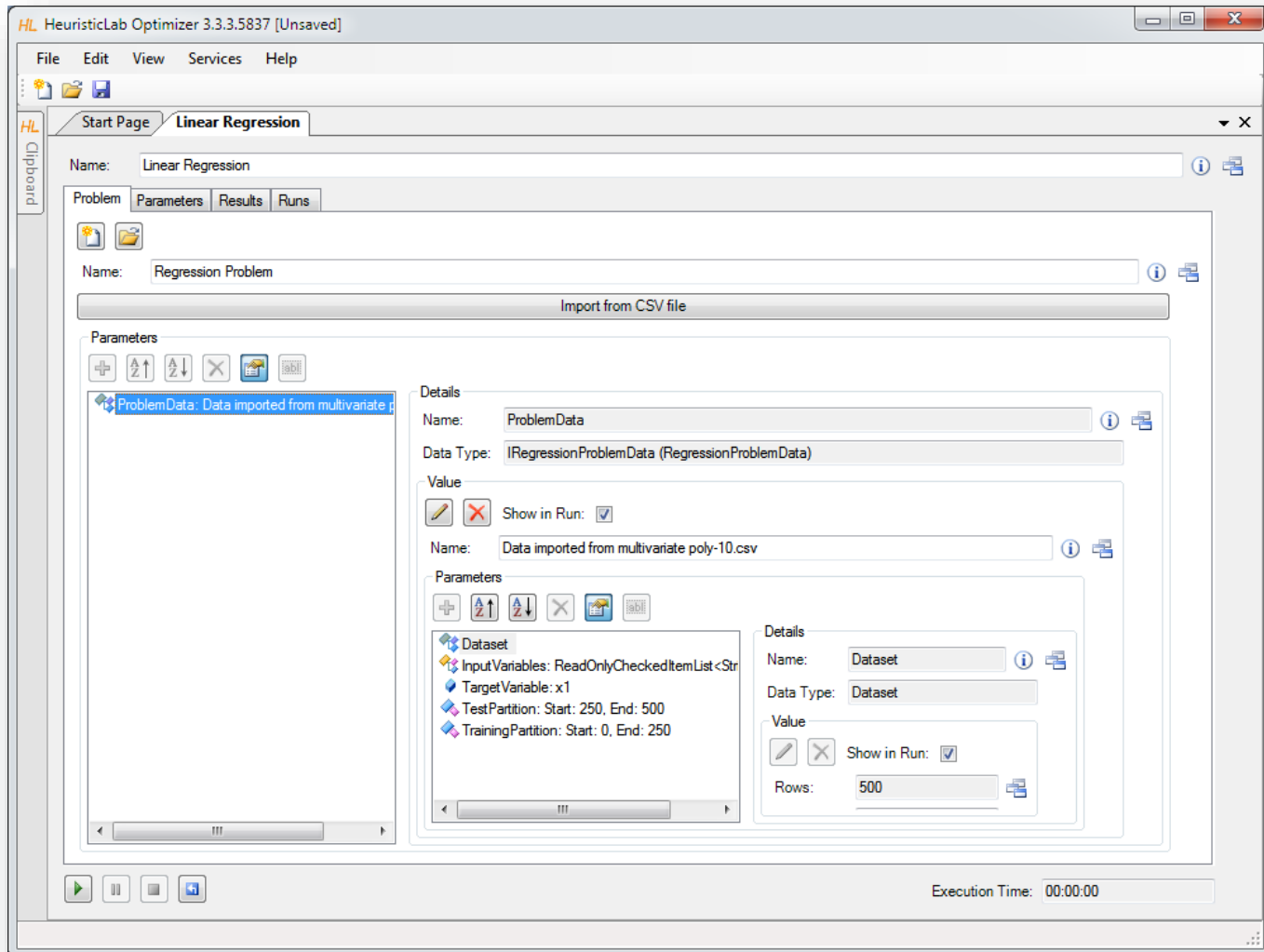
- Create new algorithm



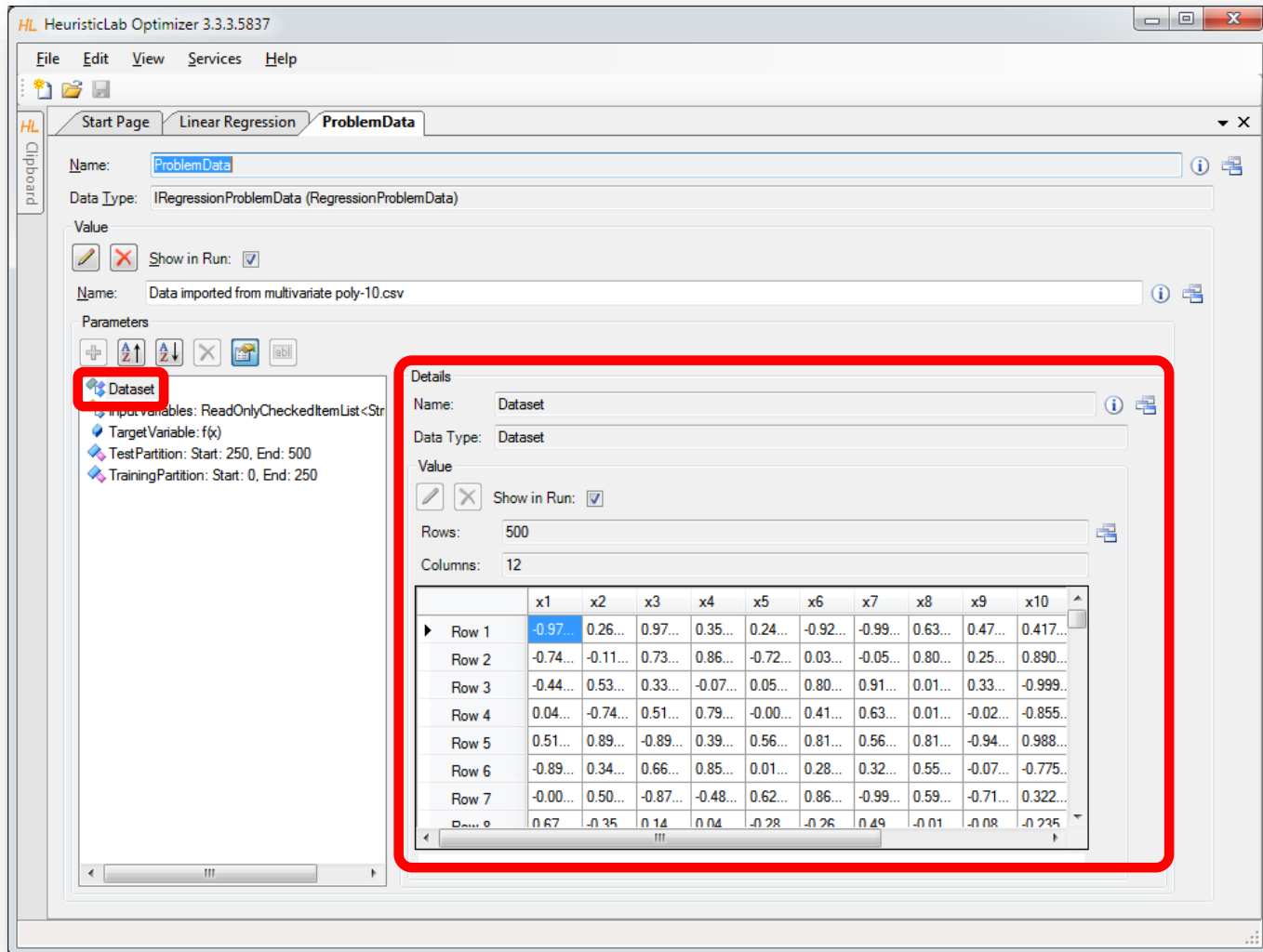
# Import Data from CSV-File



# Inspect and Configure Dataset



# Inspect Imported Data



The screenshot shows the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837" and has a menu bar with "File", "Edit", "View", "Services", and "Help". The "ProblemData" tab is active, showing a configuration for a linear regression problem. The "Name" field is "ProblemData" and the "Data Type" is "IRegressionProblemData (RegressionProblemData)". The "Value" section shows a "Show in Run" checkbox checked. The "Name" field for the data source is "Data imported from multivariate poly-10.csv".

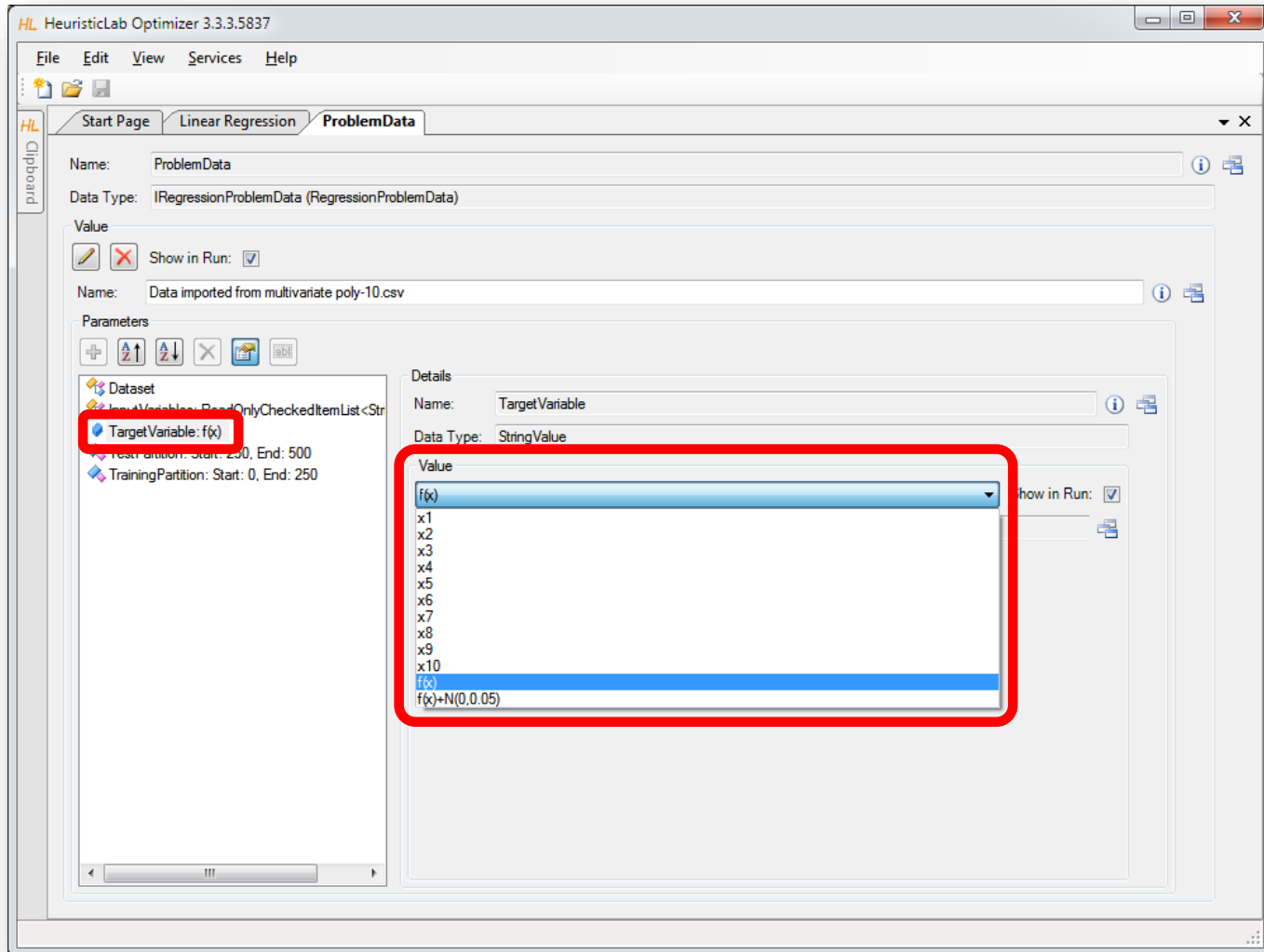
The "Parameters" section is expanded, and the "Dataset" parameter is highlighted with a red box. The "Dataset" details window shows the following information:

- Name: Dataset
- Data Type: Dataset
- Value: Show in Run:
- Rows: 500
- Columns: 12

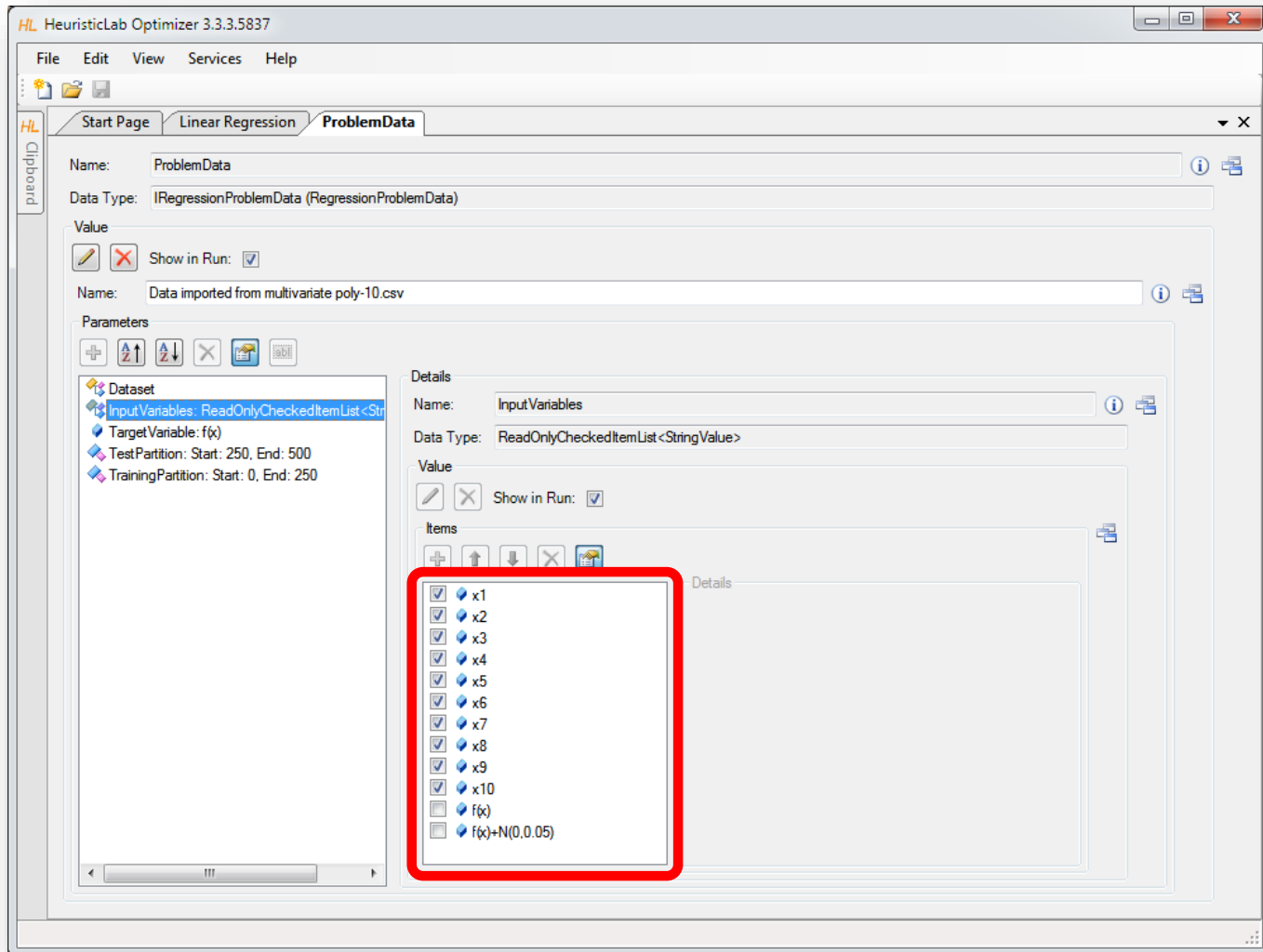
The "Dataset" details window also displays a table of data with 10 columns (x1 to x10) and 7 rows (Row 1 to Row 7). The first row is highlighted in blue.

|       | 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...  |

# Set Target Variable

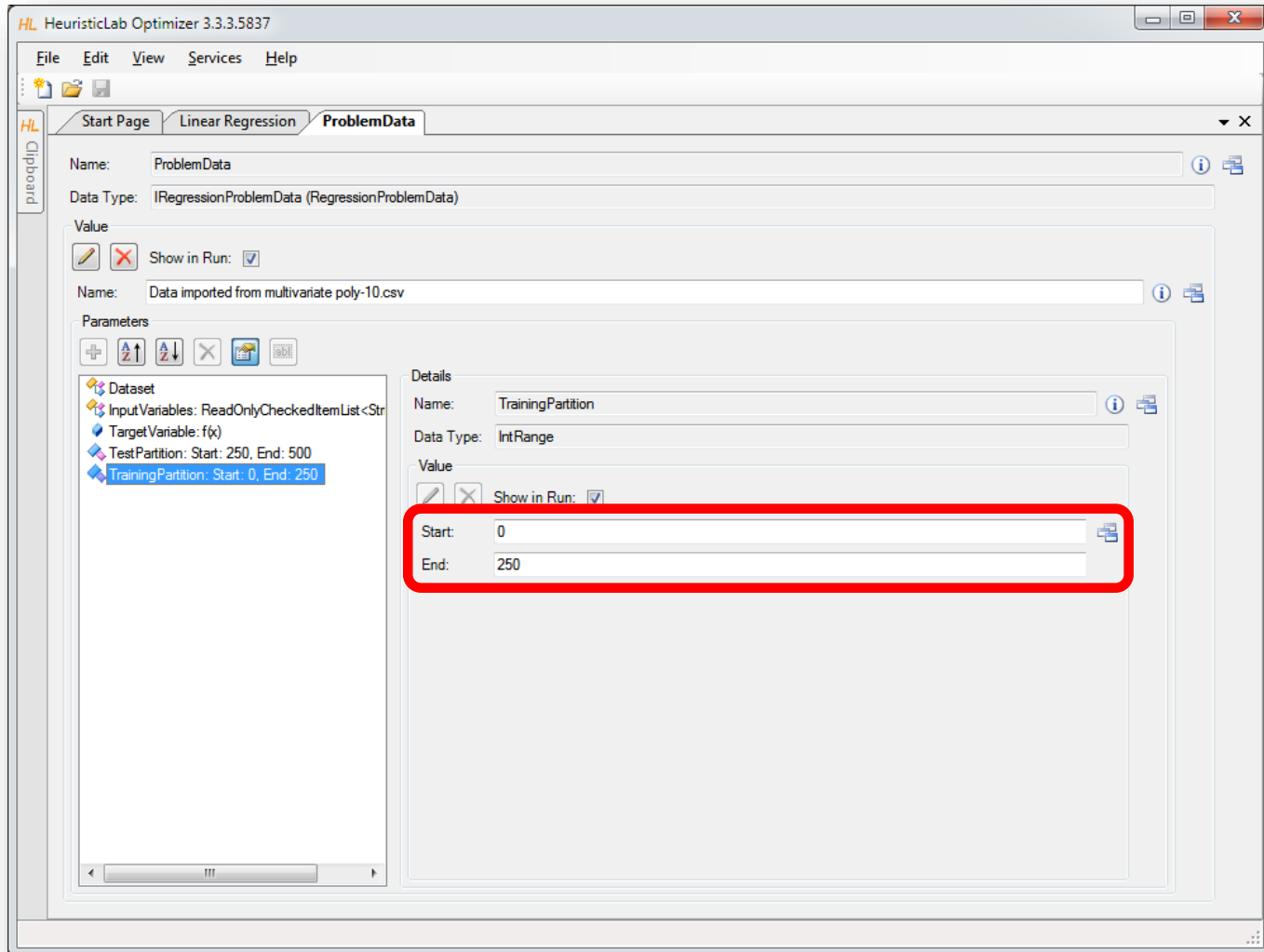


# Select Input Variables

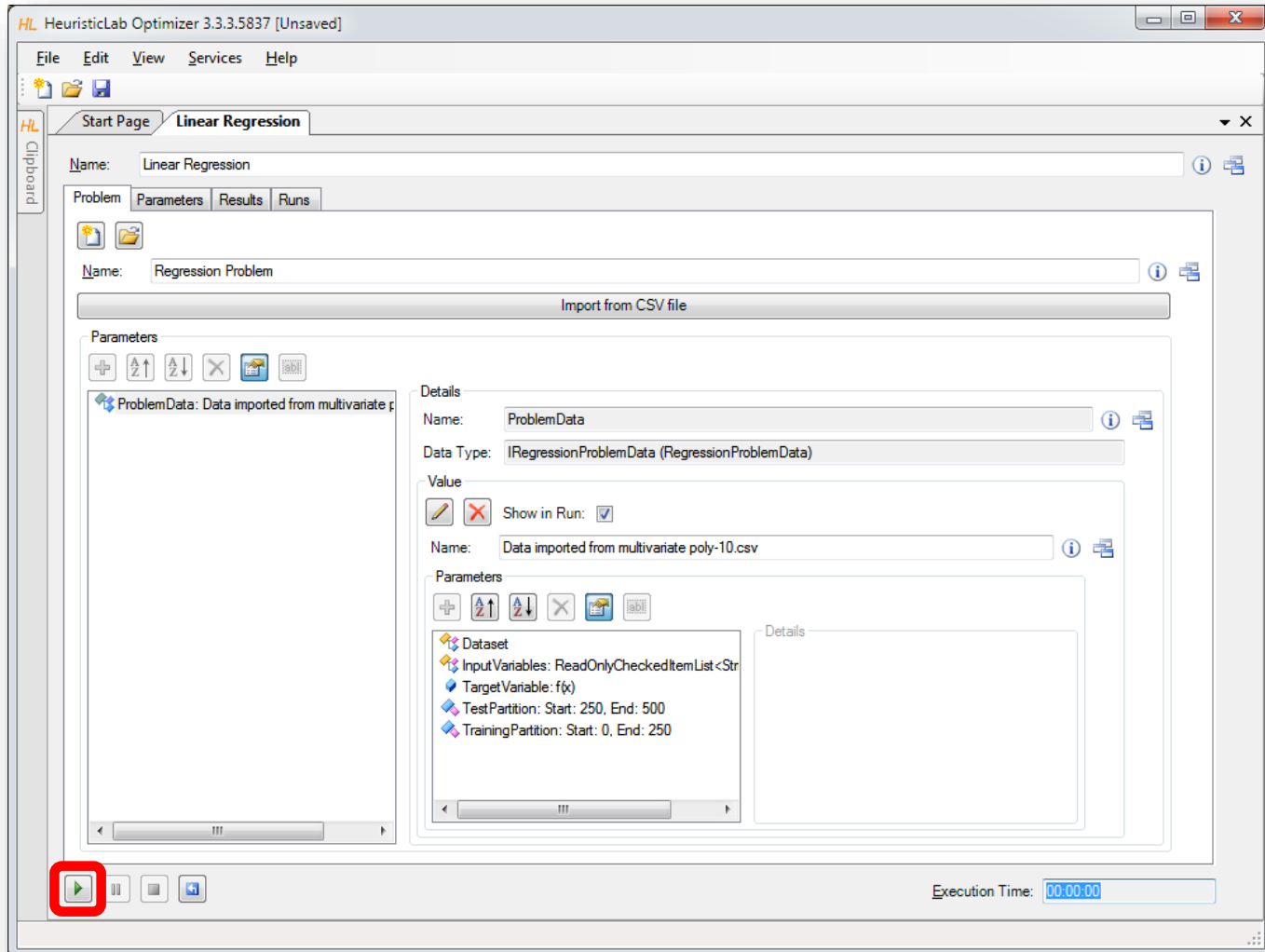




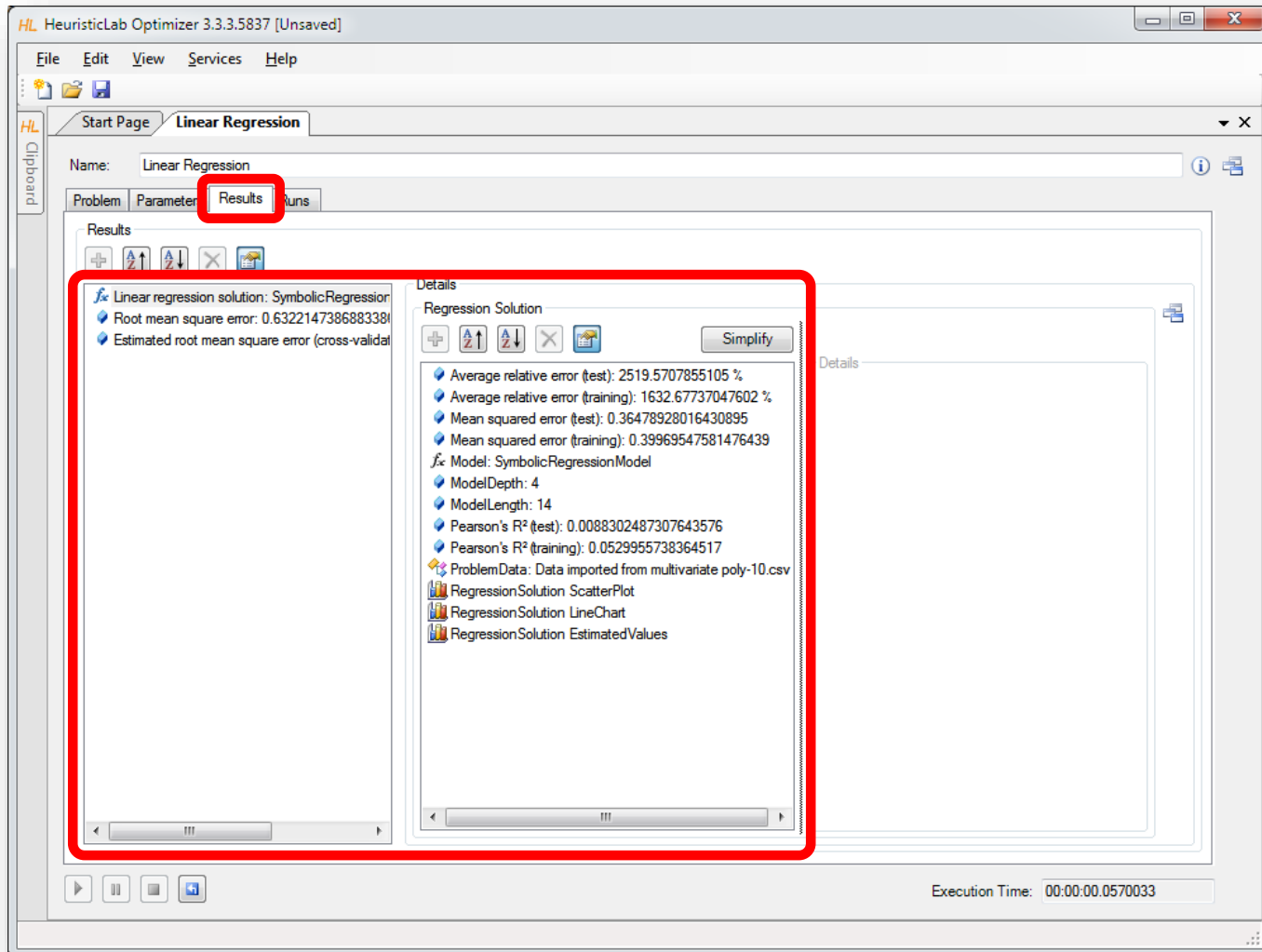
# Configure Training and Test Partitions



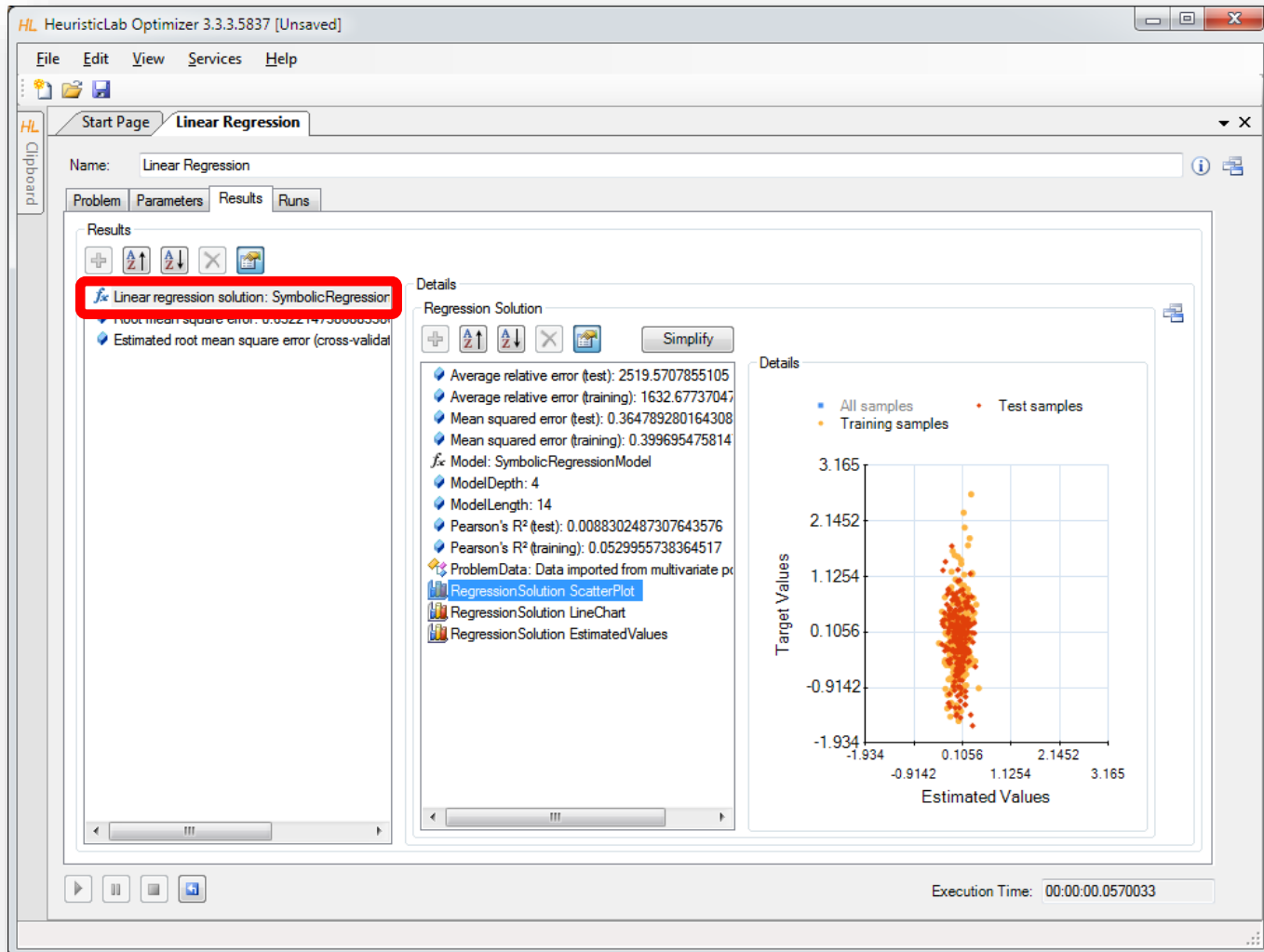
# Run Linear Regression



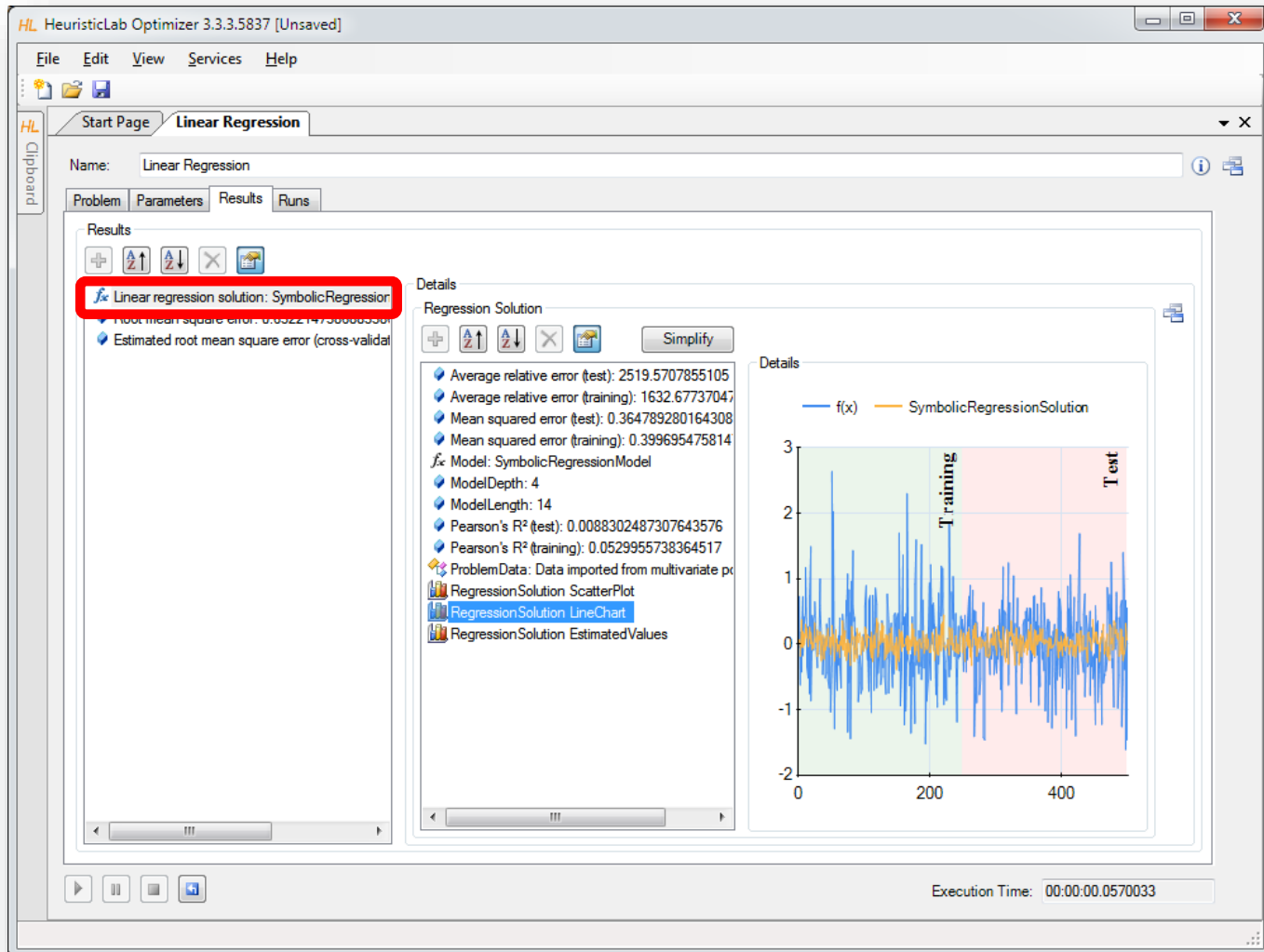
# Inspect Results



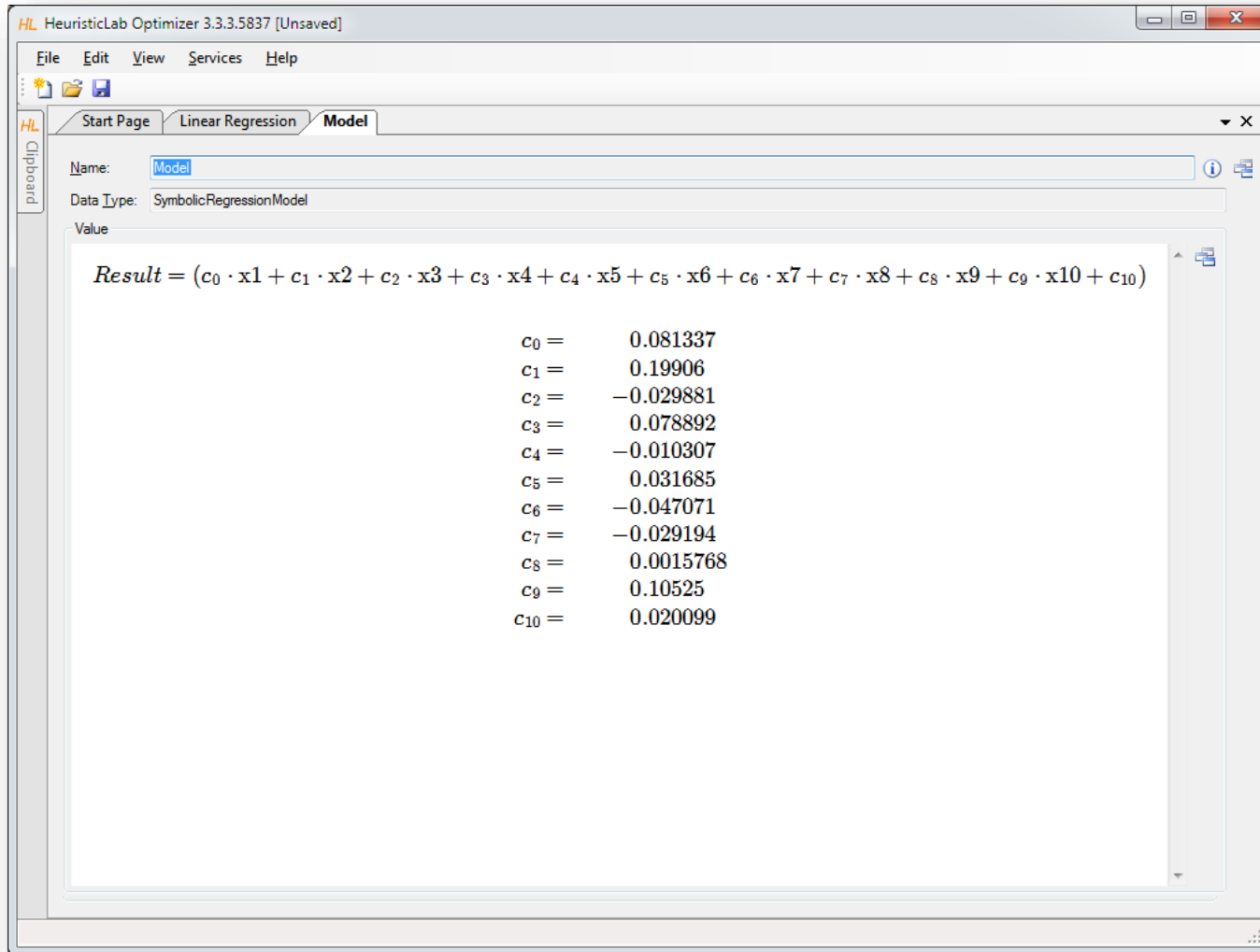
# Inspect Scatterplot of Predicted and Target Values



# Inspect Linechart



# Inspect the Model



HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

HL Start Page Linear Regression Model

Name: Model

Data Type: SymbolicRegressionModel

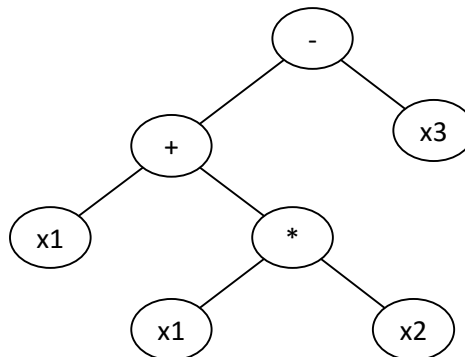
Value

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

|            |           |
|------------|-----------|
| $c_0 =$    | 0.081337  |
| $c_1 =$    | 0.19906   |
| $c_2 =$    | -0.029881 |
| $c_3 =$    | 0.078892  |
| $c_4 =$    | -0.010307 |
| $c_5 =$    | 0.031685  |
| $c_6 =$    | -0.047071 |
| $c_7 =$    | -0.029194 |
| $c_8 =$    | 0.0015768 |
| $c_9 =$    | 0.10525   |
| $c_{10} =$ | 0.020099  |

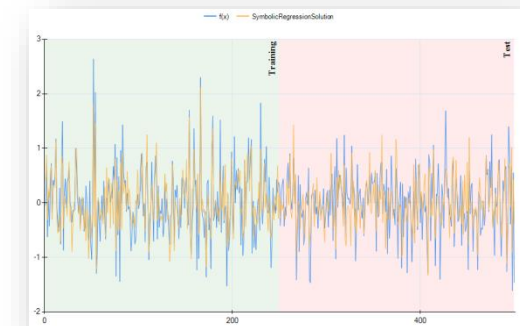
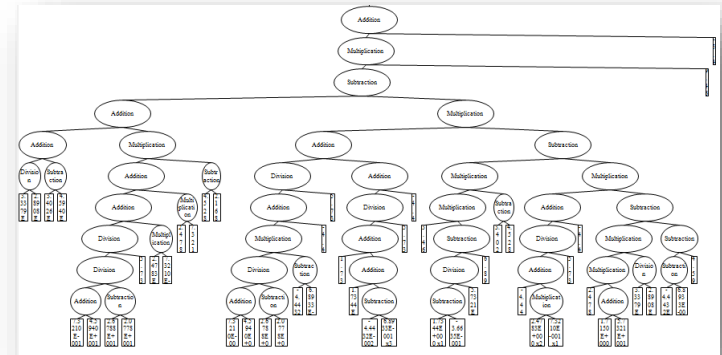
# Symbolic Regression with HeuristicLab

- 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



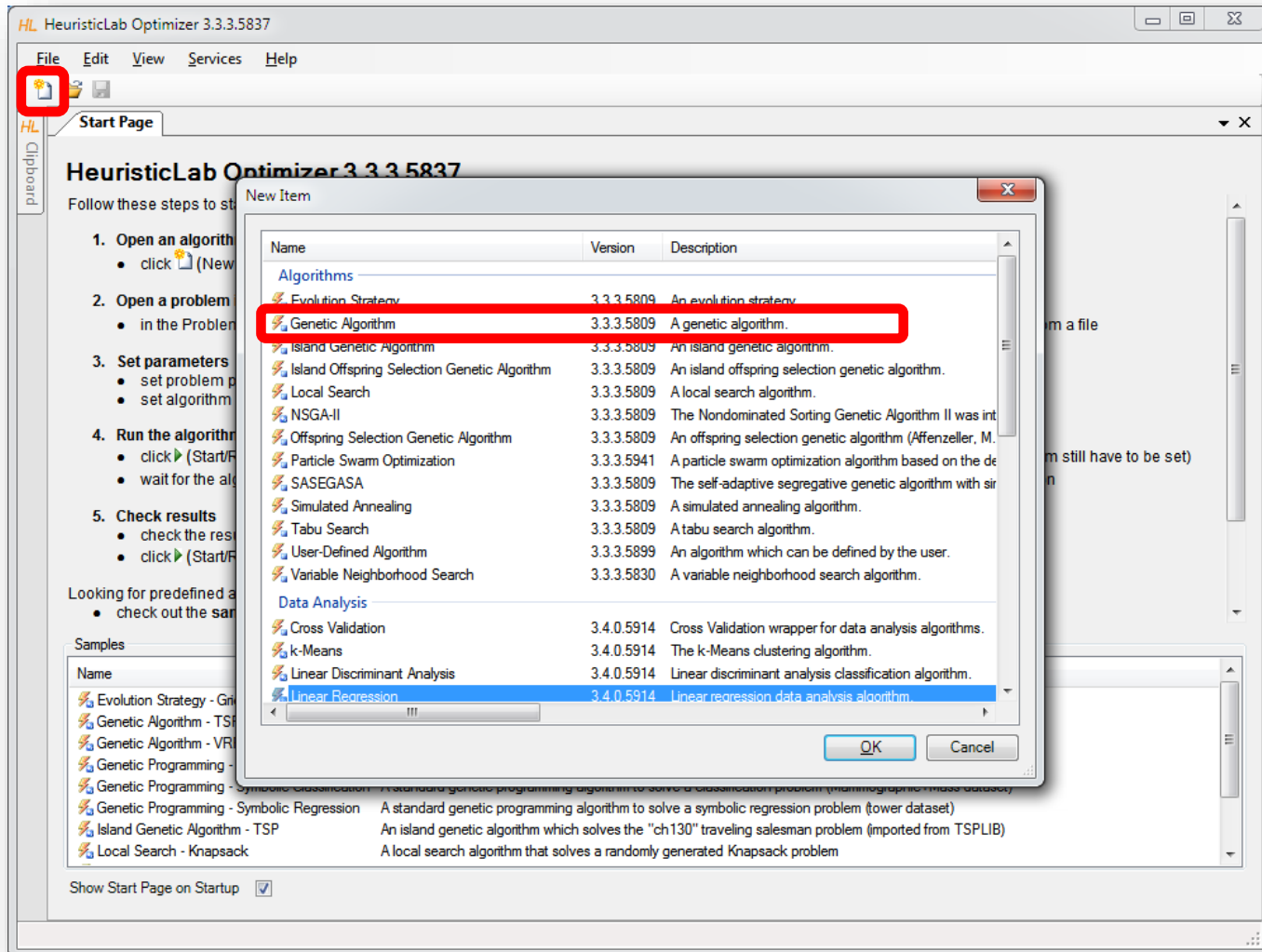
# Symbolic Regression with HeuristicLab

- 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

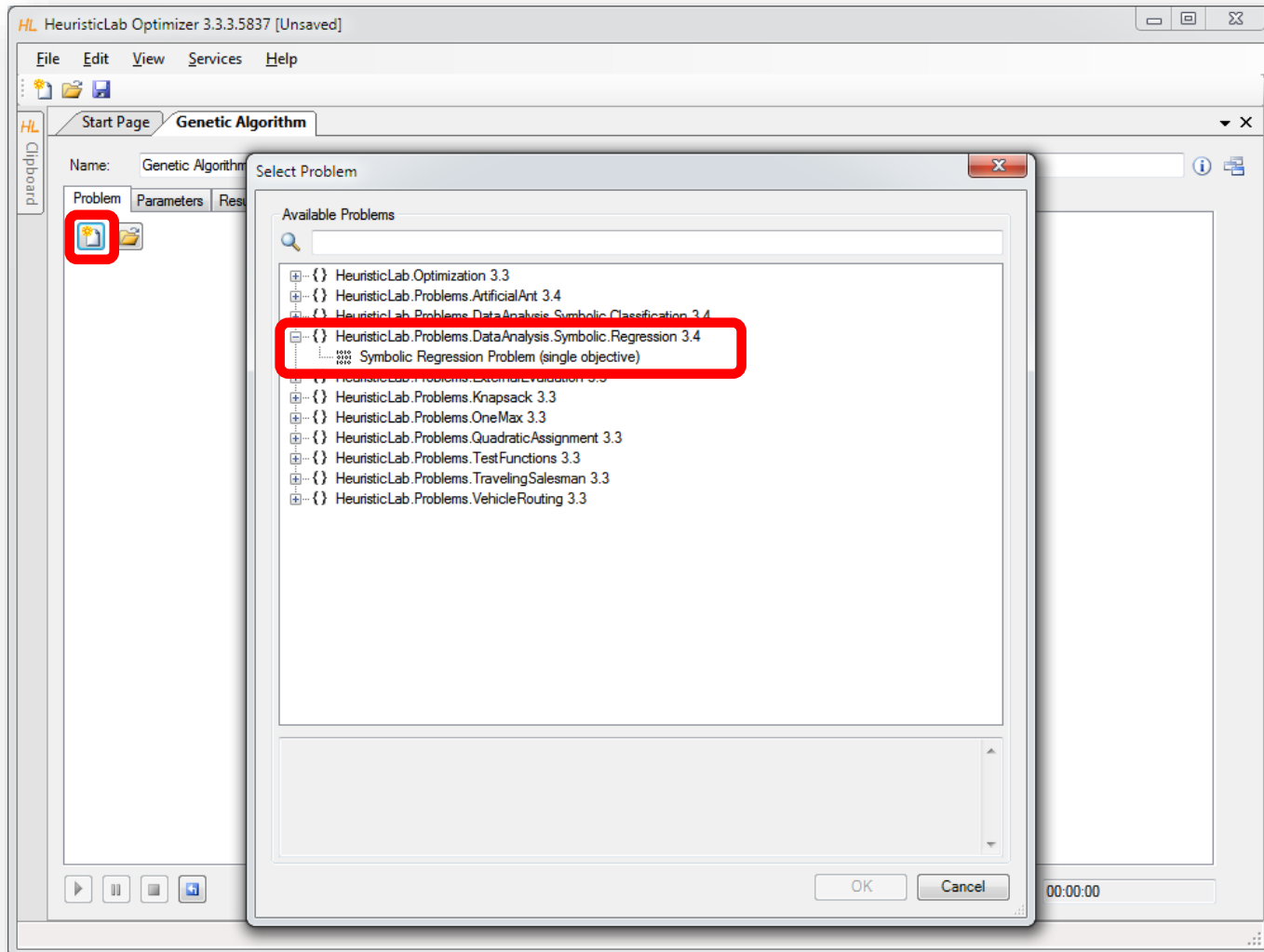




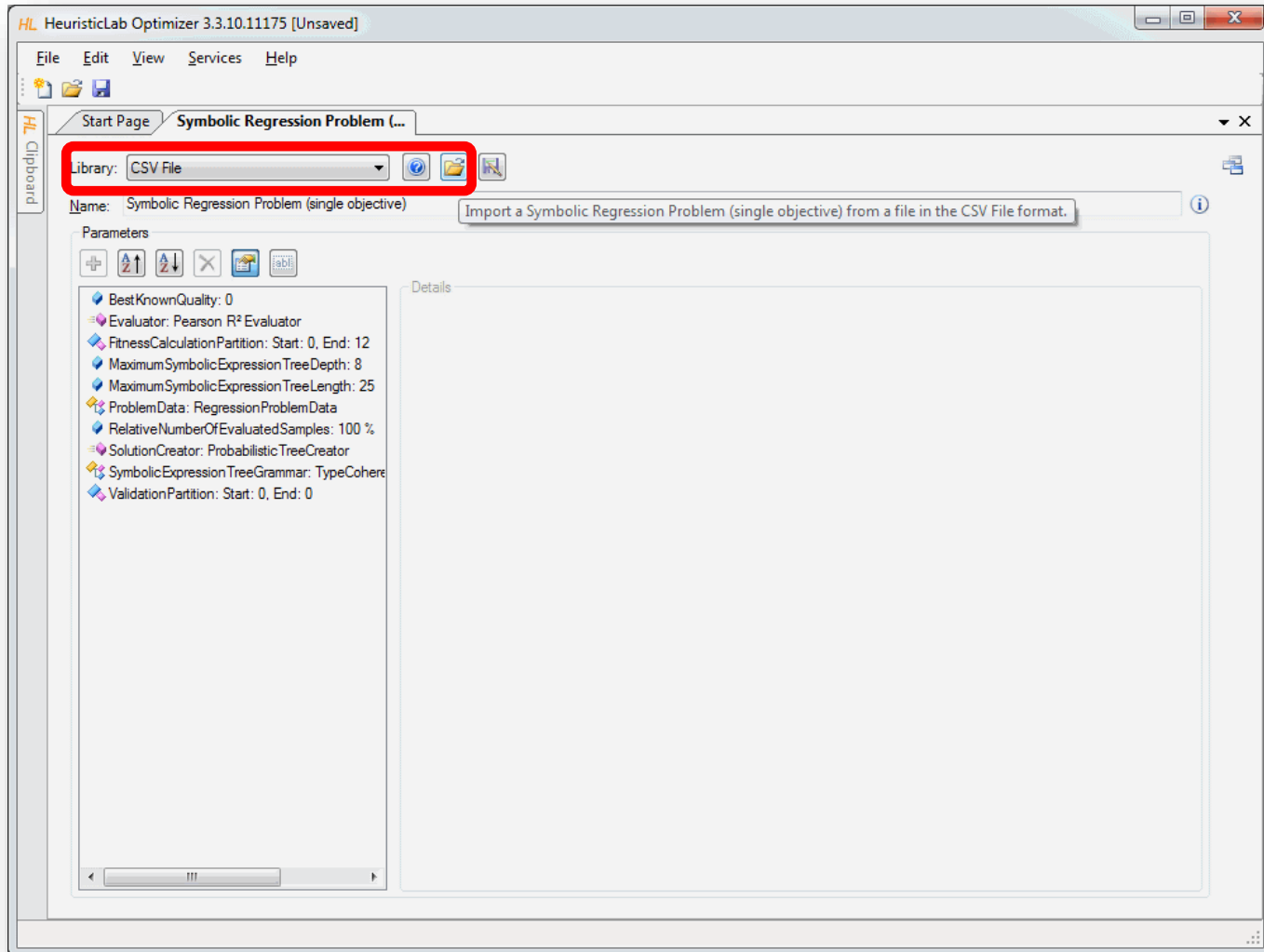
# Create New Genetic Algorithm



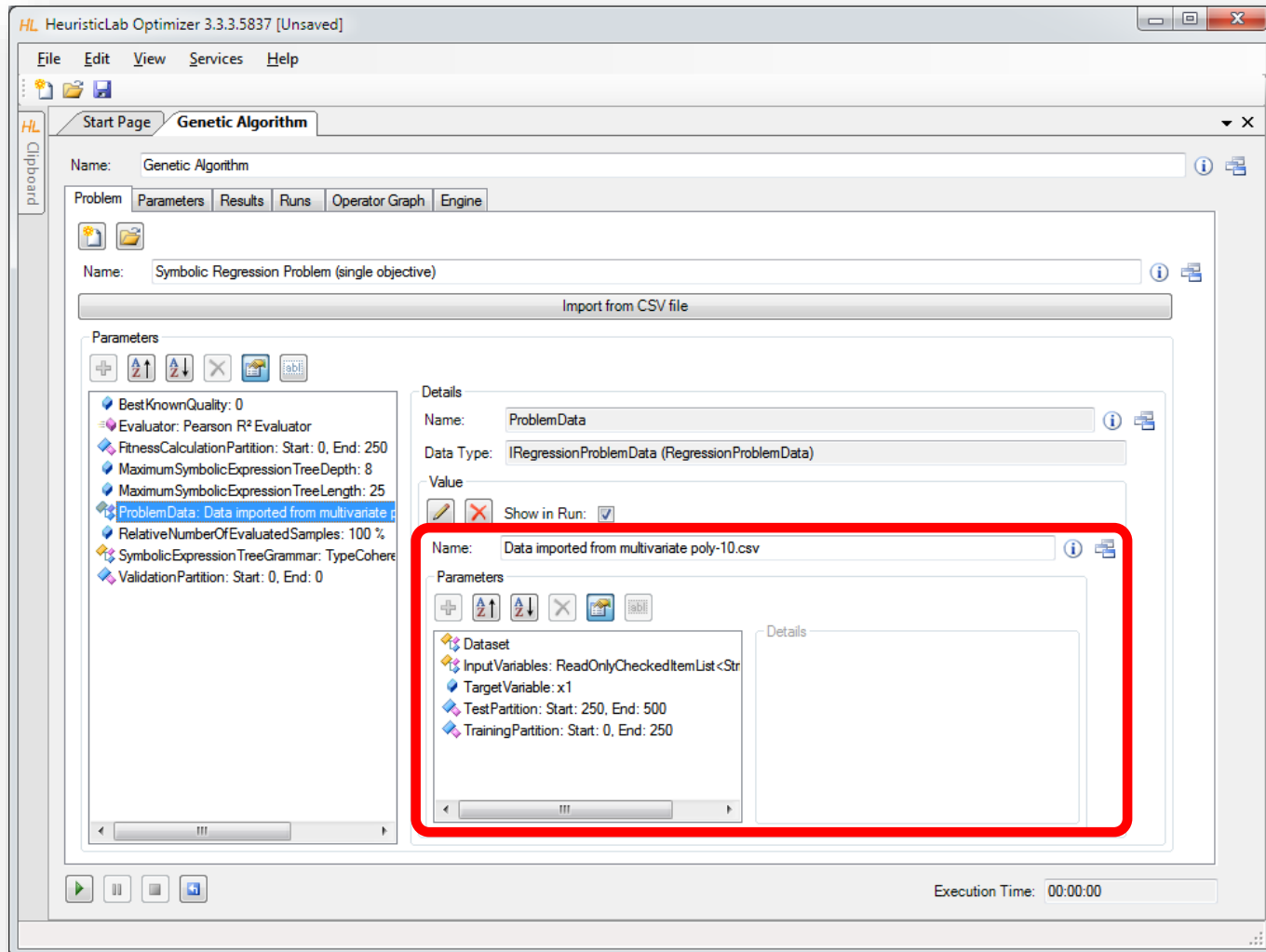
# Create New Symbolic Regression Problem



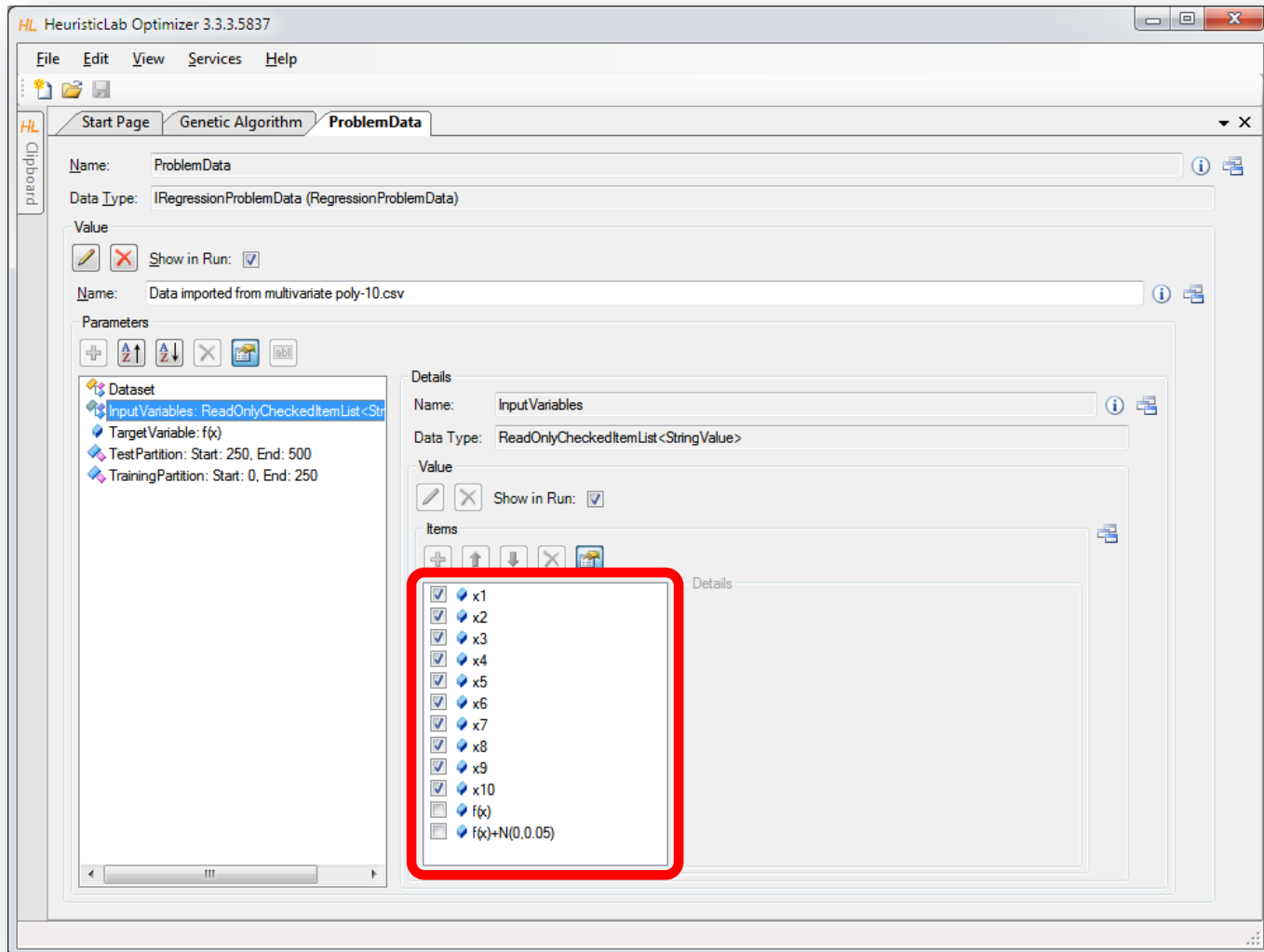
# Import Data



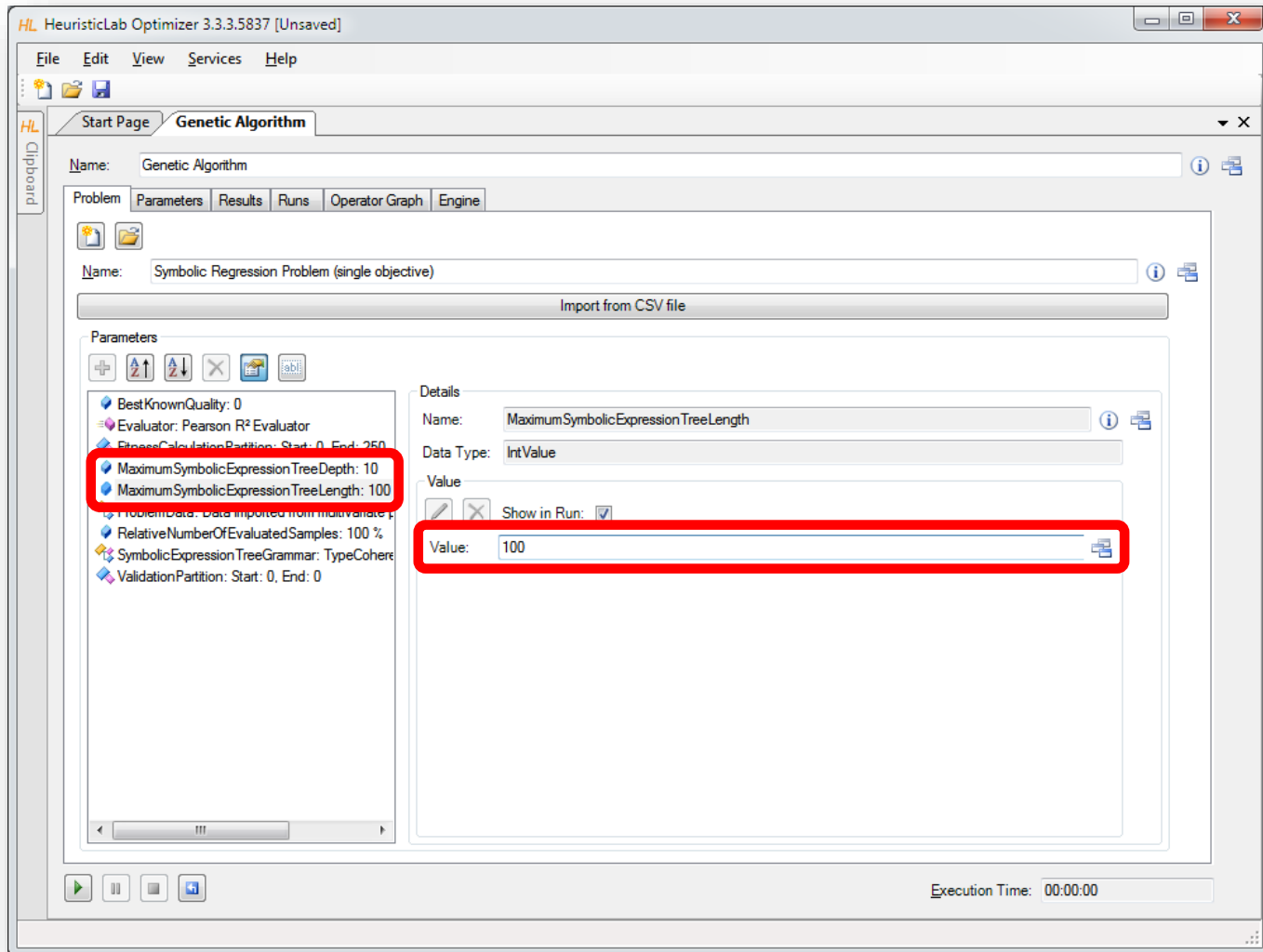
# Inspect Data and Configure Dataset



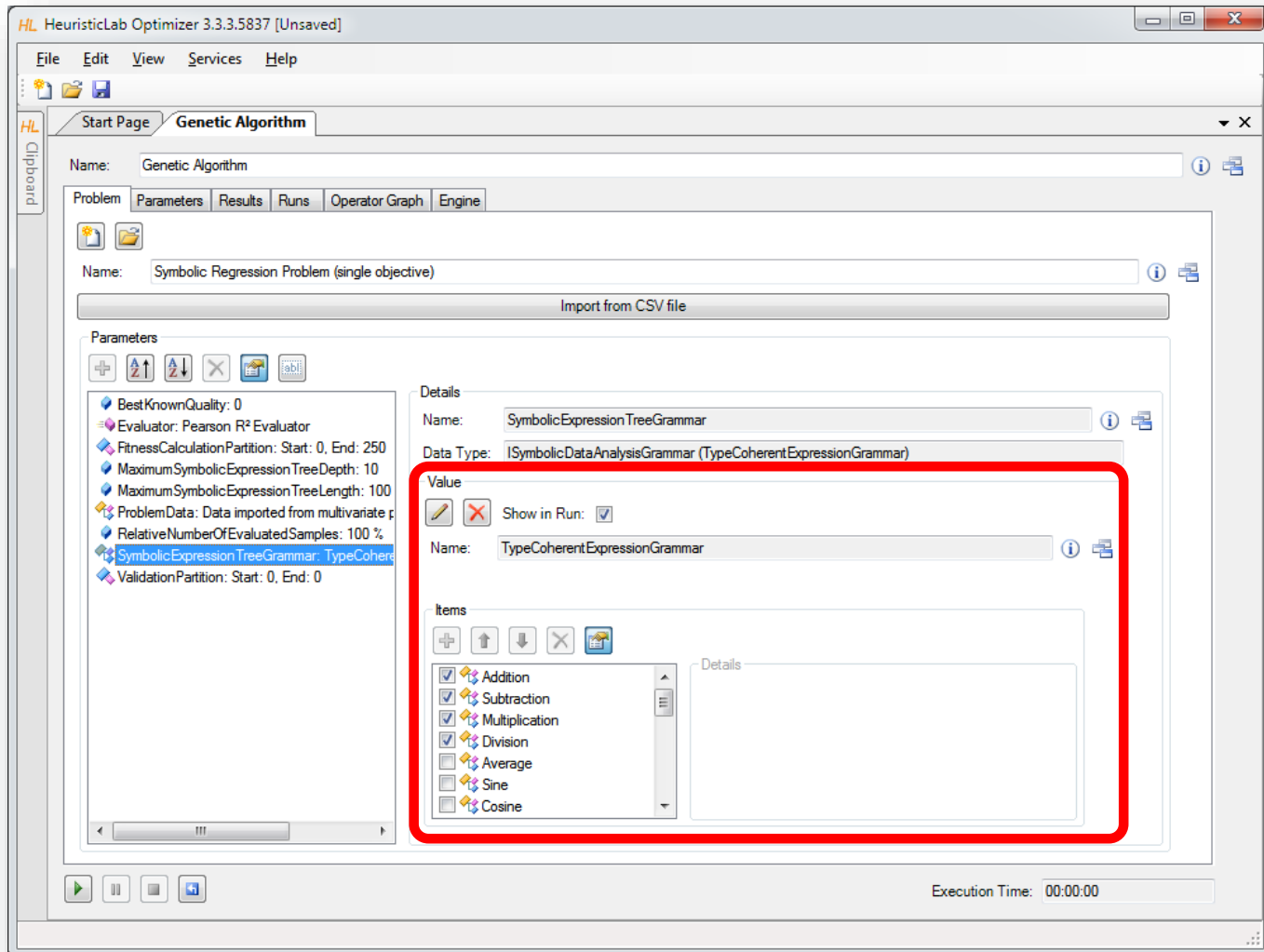
# Set Target and Input Variables



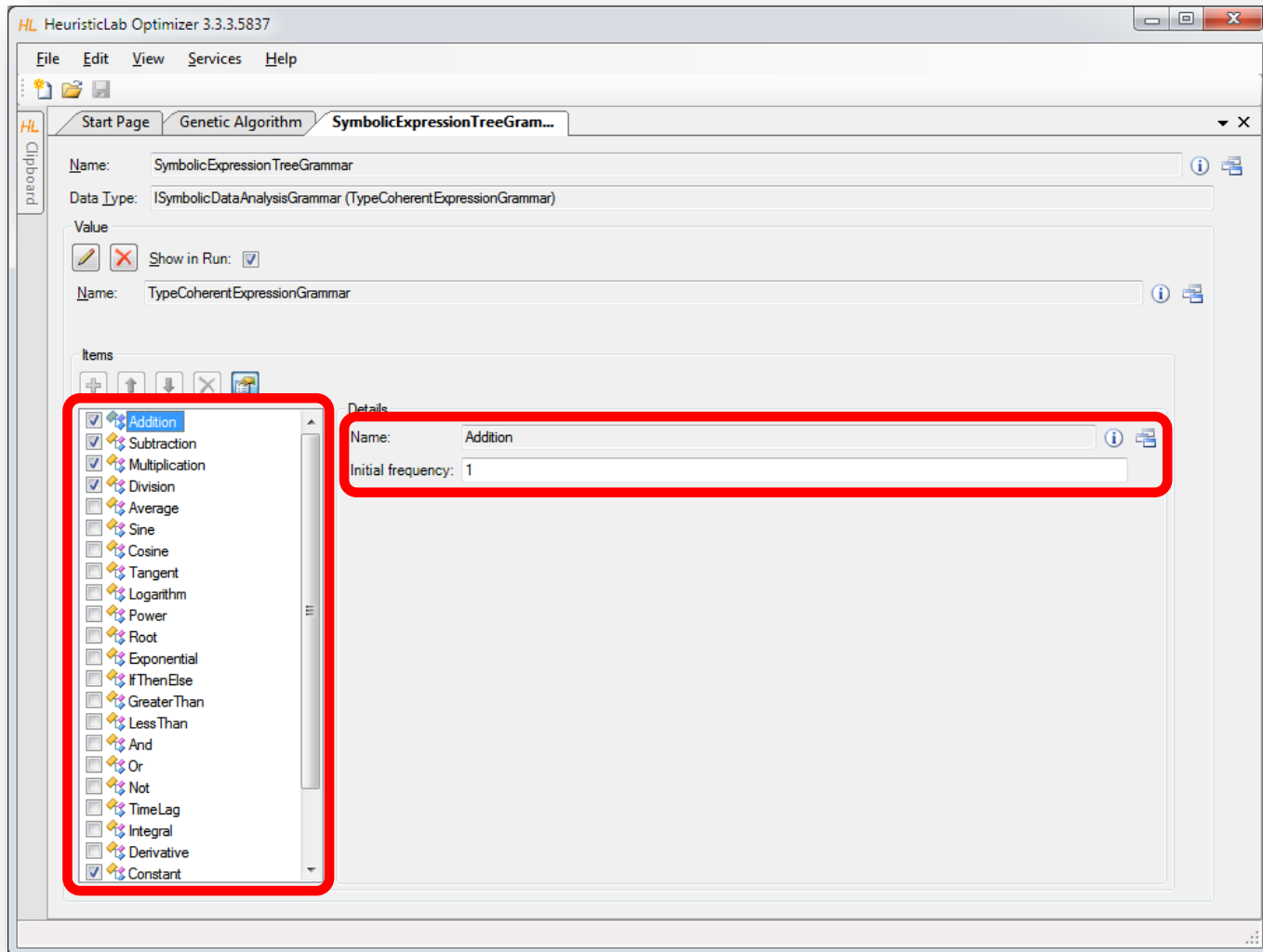
# Configure Maximal Model Depth and Length



# Configure Function Set (Grammar)

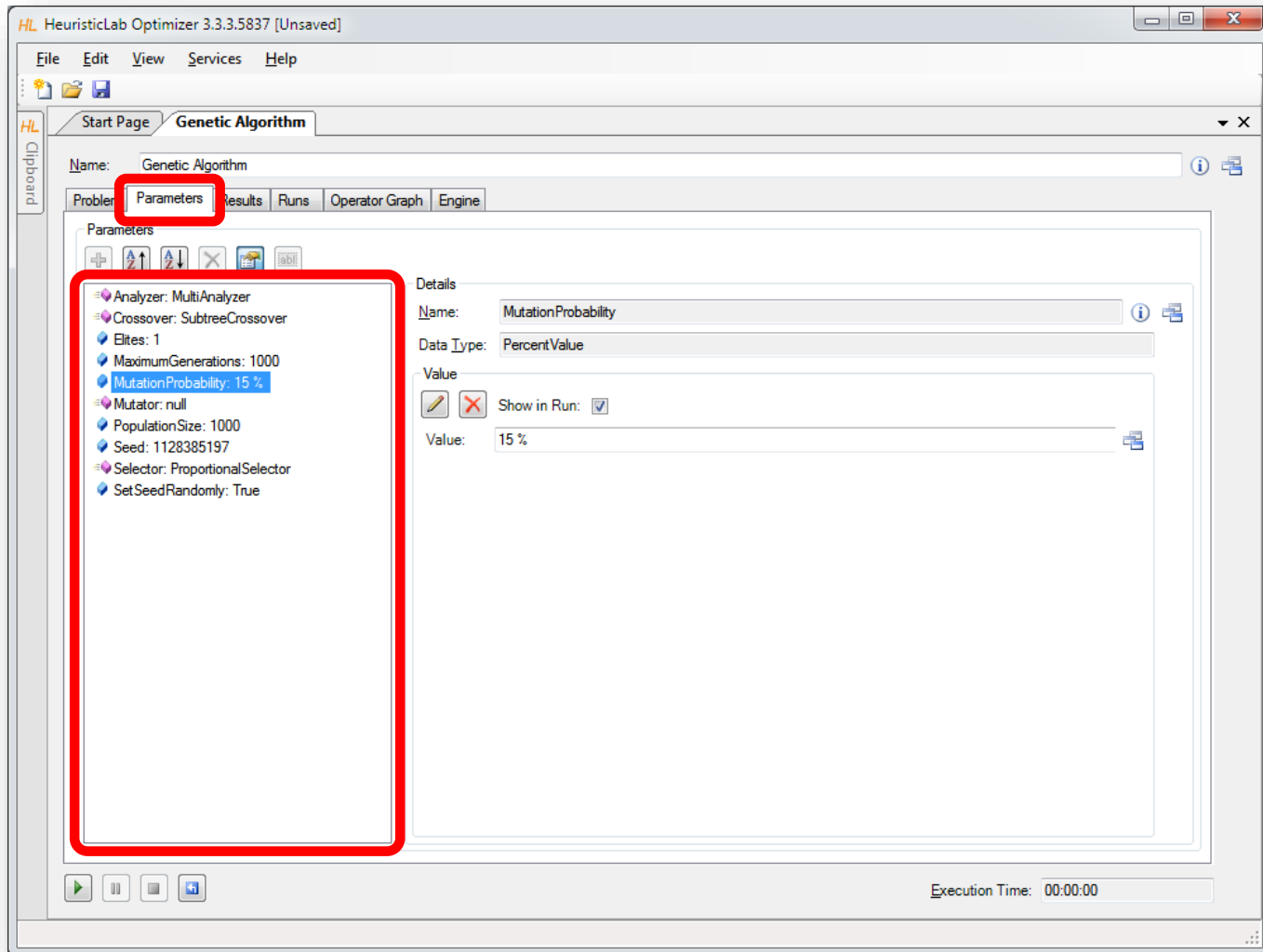


# Configure Function Set (Grammar)

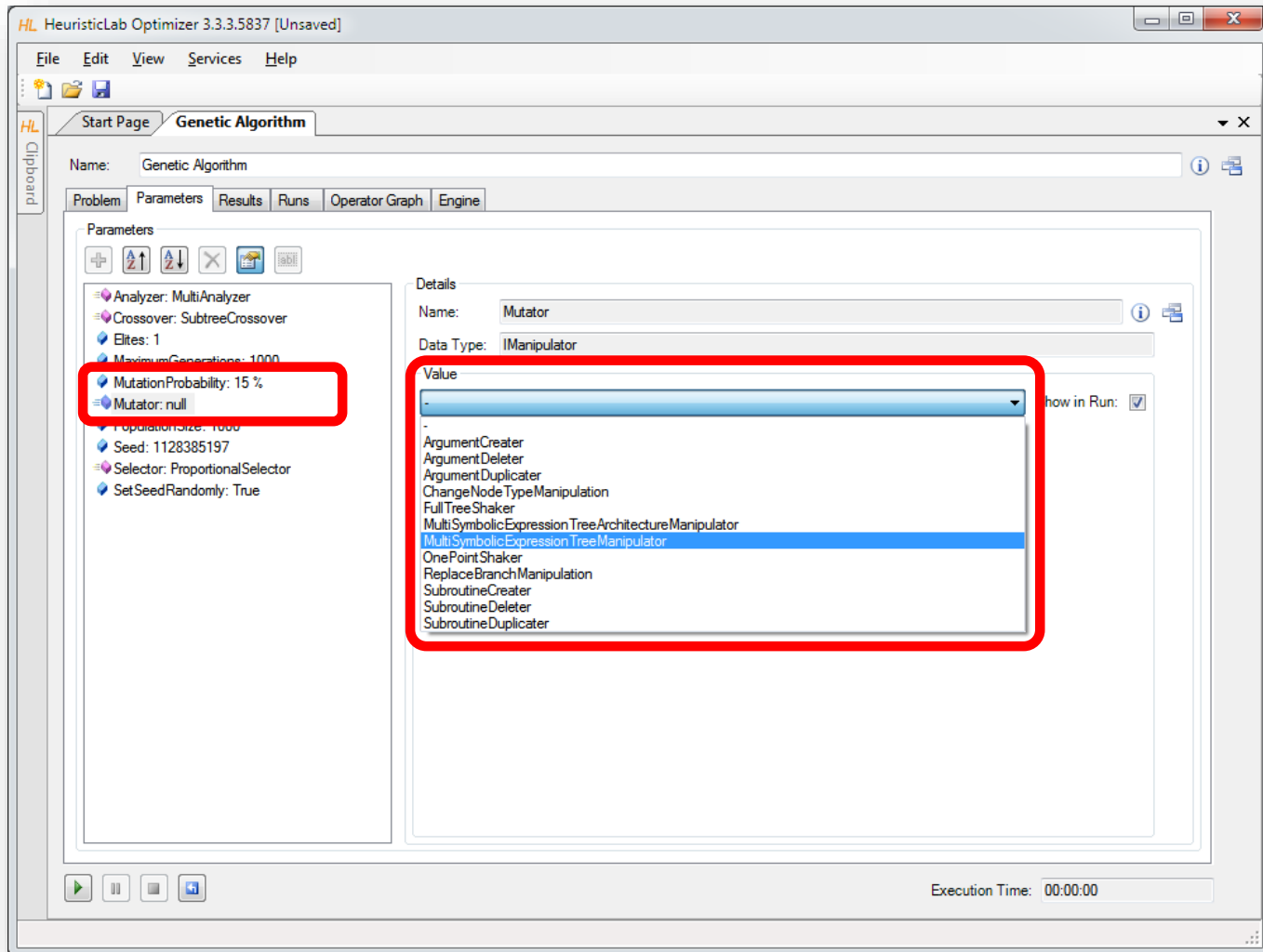




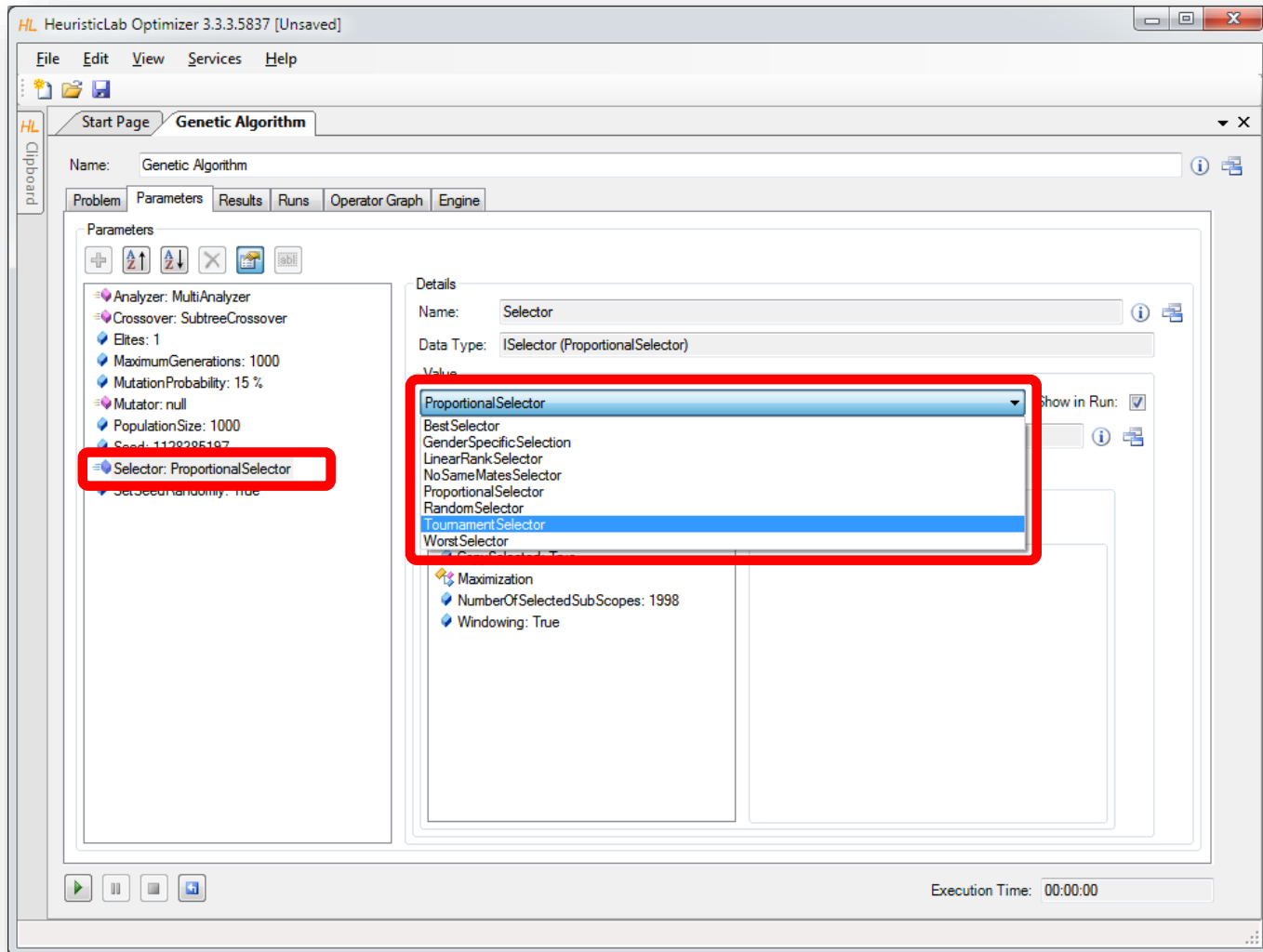
# Configure Algorithm Parameters



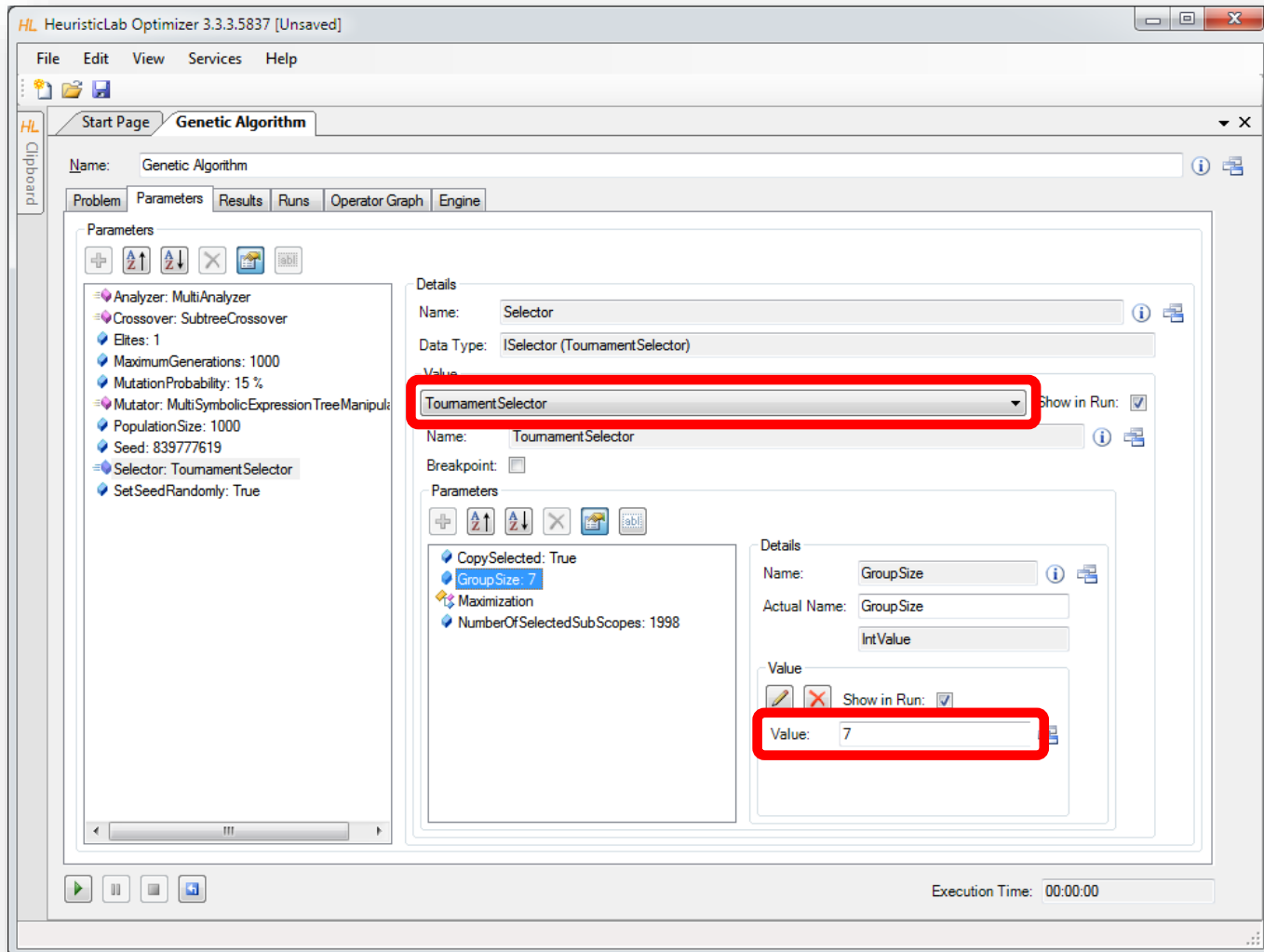
# Configure Mutation Operator



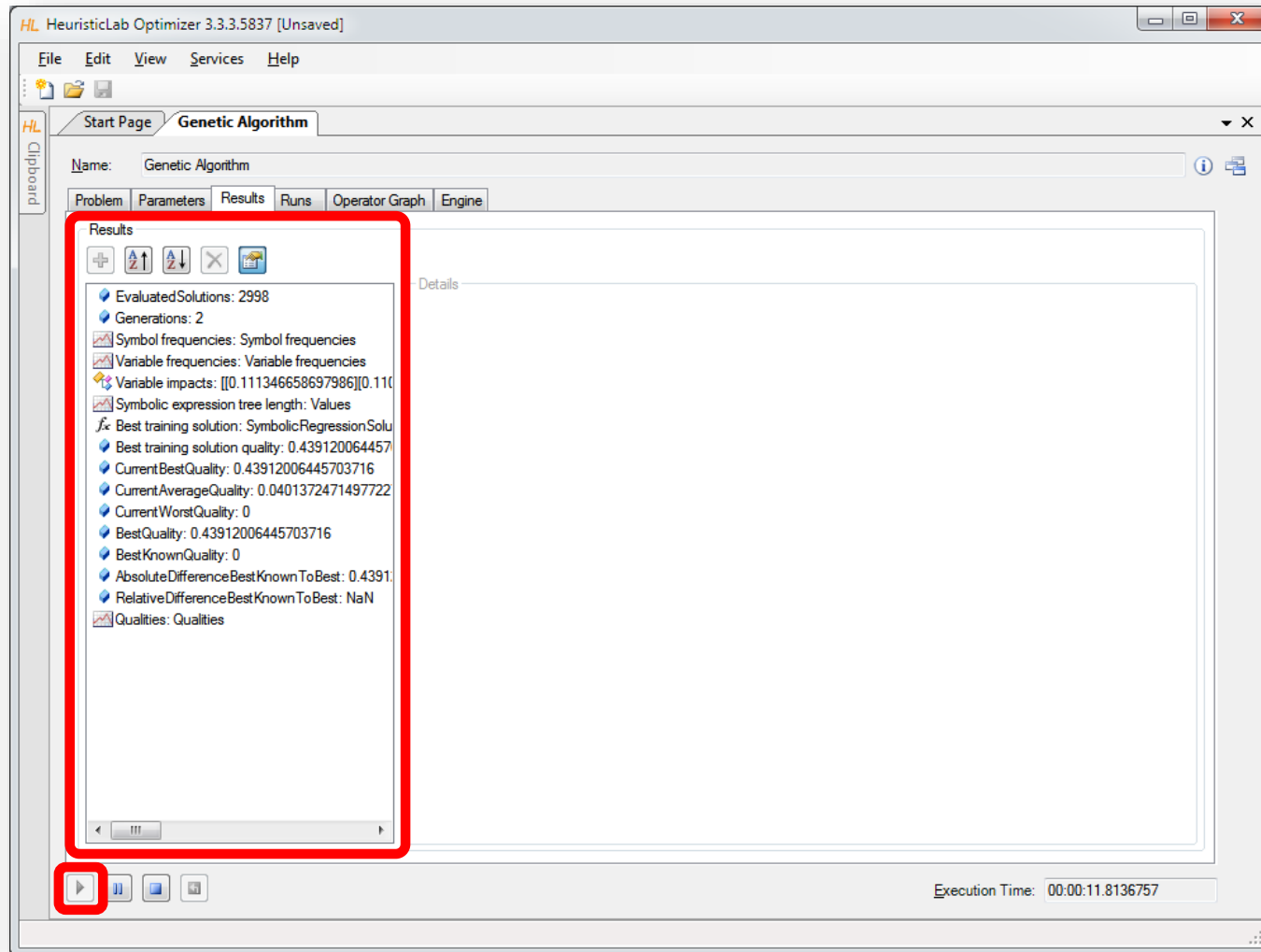
# Configure Selection Operator



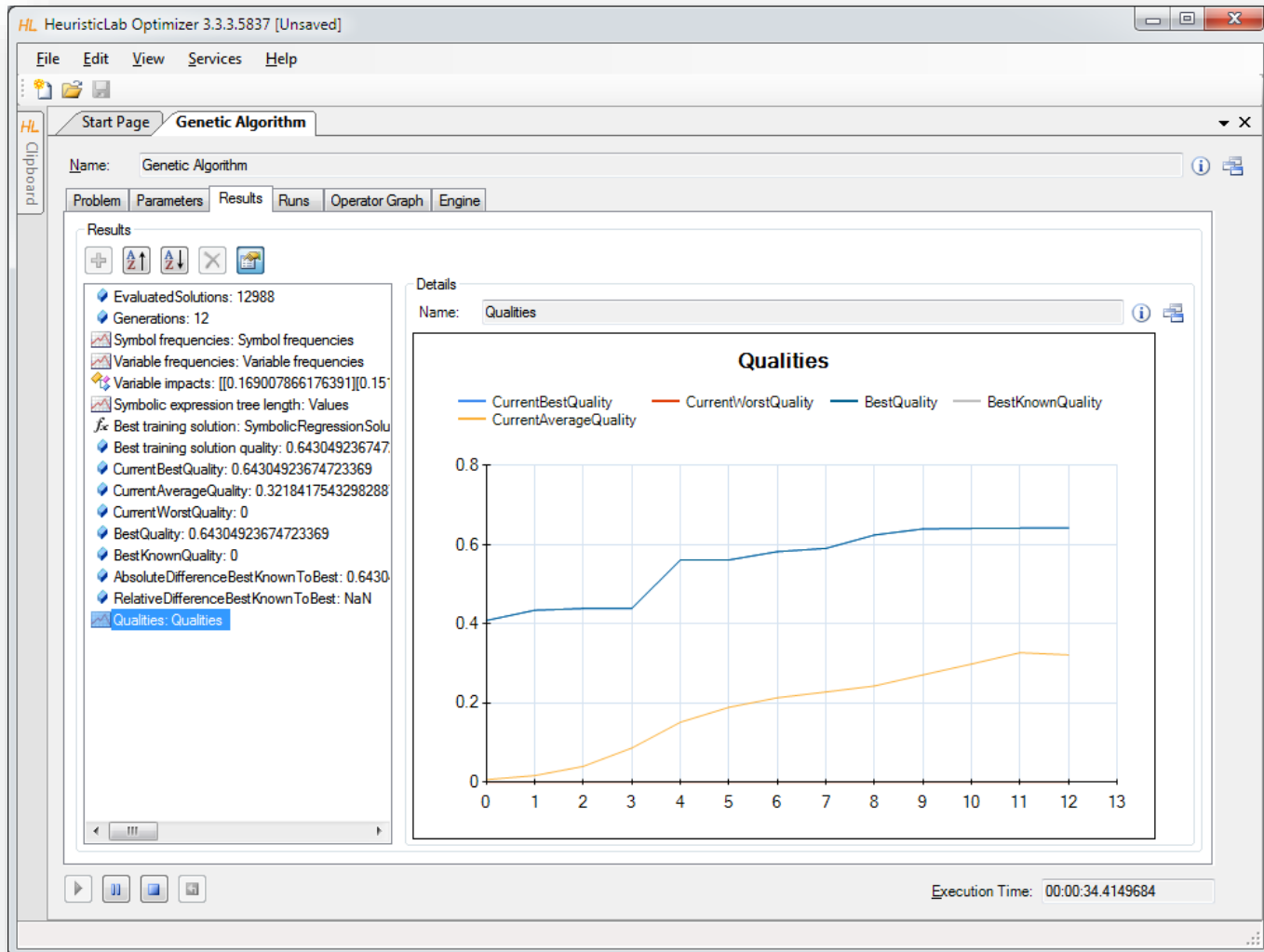
# Configure Tournament Group Size



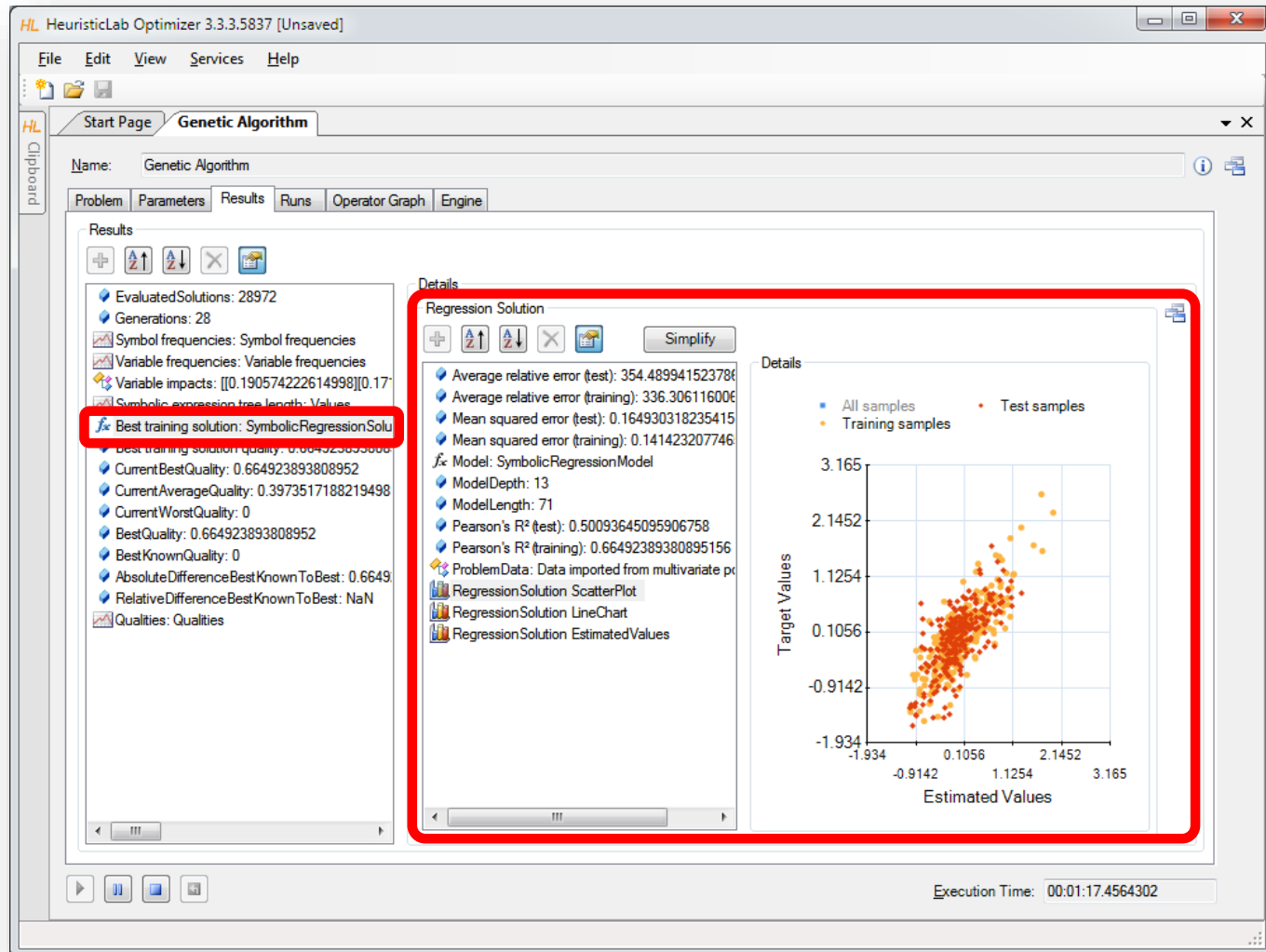
# Start Algorithm and Inspect Results



# Inspect Quality Chart



# Inspect Best Model on Training Partition



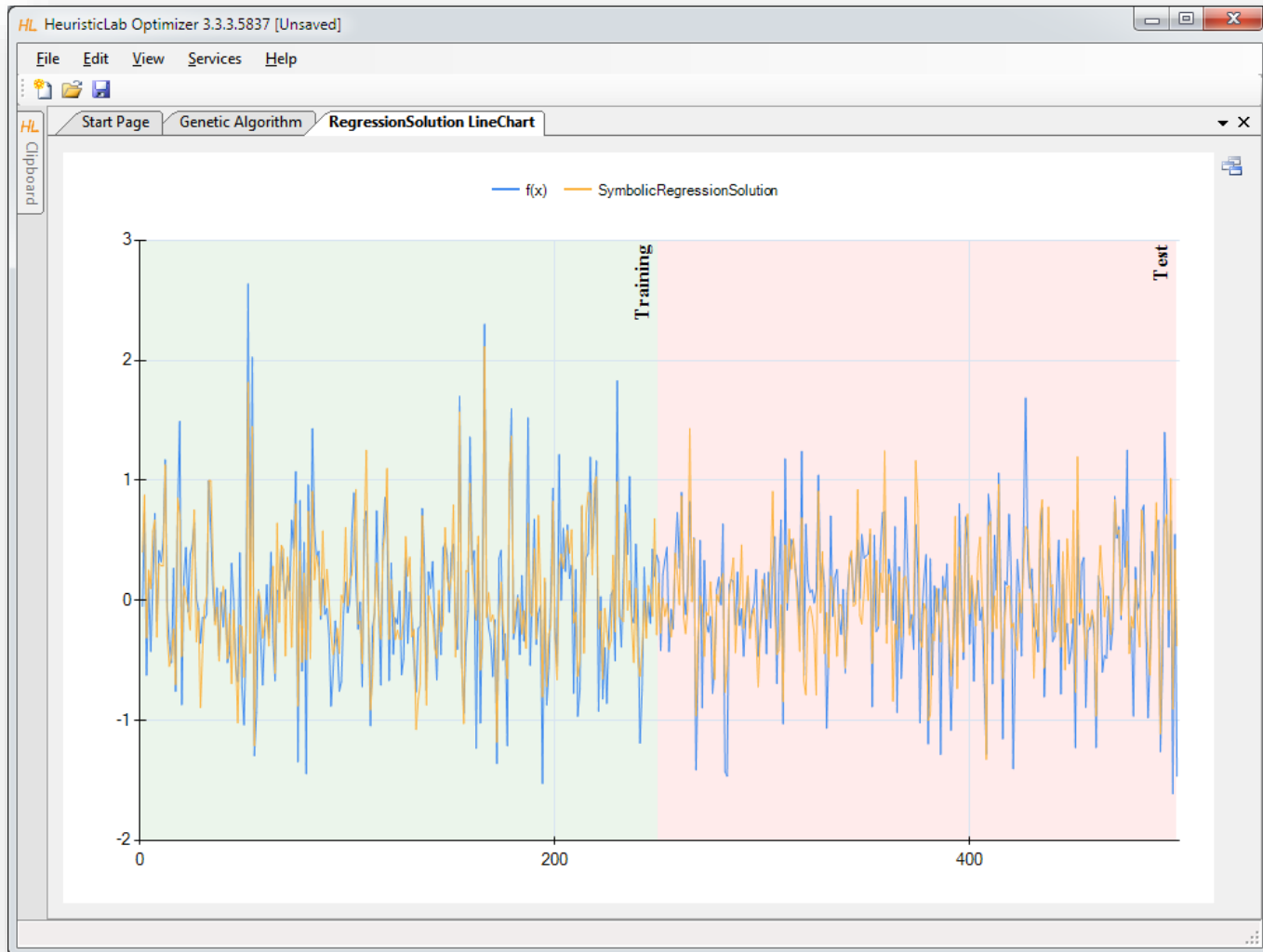
The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Genetic Algorithm" is selected in the "Start Page" tab. The "Results" tab is active, showing a list of results. The "Best training solution: SymbolicRegressionSolu" is highlighted with a red box. The "Details" panel for the "Regression Solution" is also highlighted with a red box, showing the following information:

- 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
- ModelDepth: 13
- ModelLength: 71
- Pearson's R<sup>2</sup> (test): 0.50093645095906758
- Pearson's R<sup>2</sup> (training): 0.66492389380895156
- ProblemData: Data imported from multivariate p...
- RegressionSolution ScatterPlot
- RegressionSolution LineChart
- RegressionSolution EstimatedValues

To the right of the details panel is a scatter plot titled "Regression Solution ScatterPlot". The y-axis is labeled "Target Values" and the x-axis is labeled "Estimated Values". The plot shows a positive correlation between the estimated and target values. The legend indicates that blue dots represent "All samples" and orange dots represent "Training samples". The axes range from -1.934 to 3.165.

Execution Time: 00:01:17.4564302

# Inspect Linechart of Best Model on Training Partition

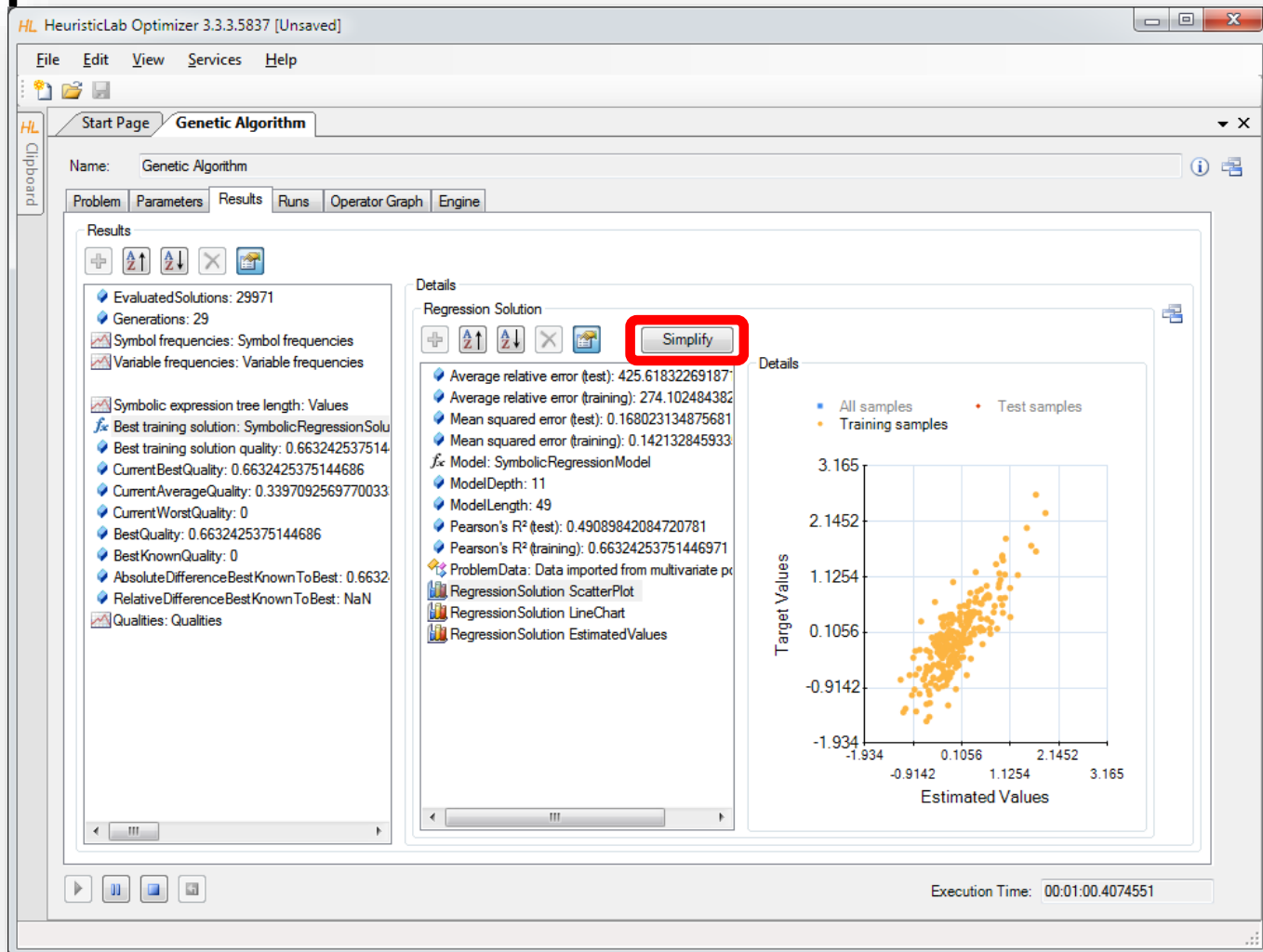




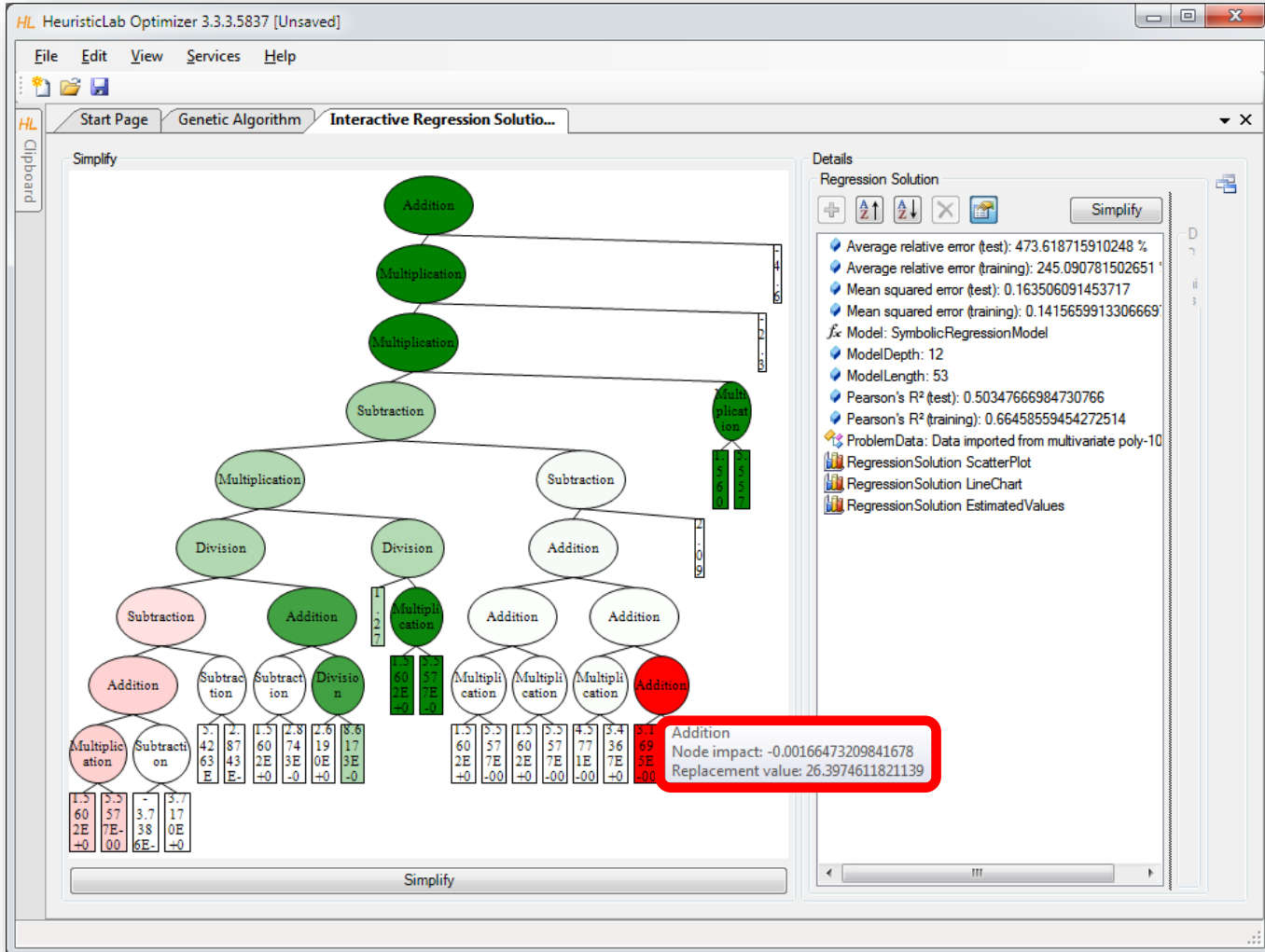




# Detailed Model Analysis and Simplification



# Symbolic Simplification and Node Impacts



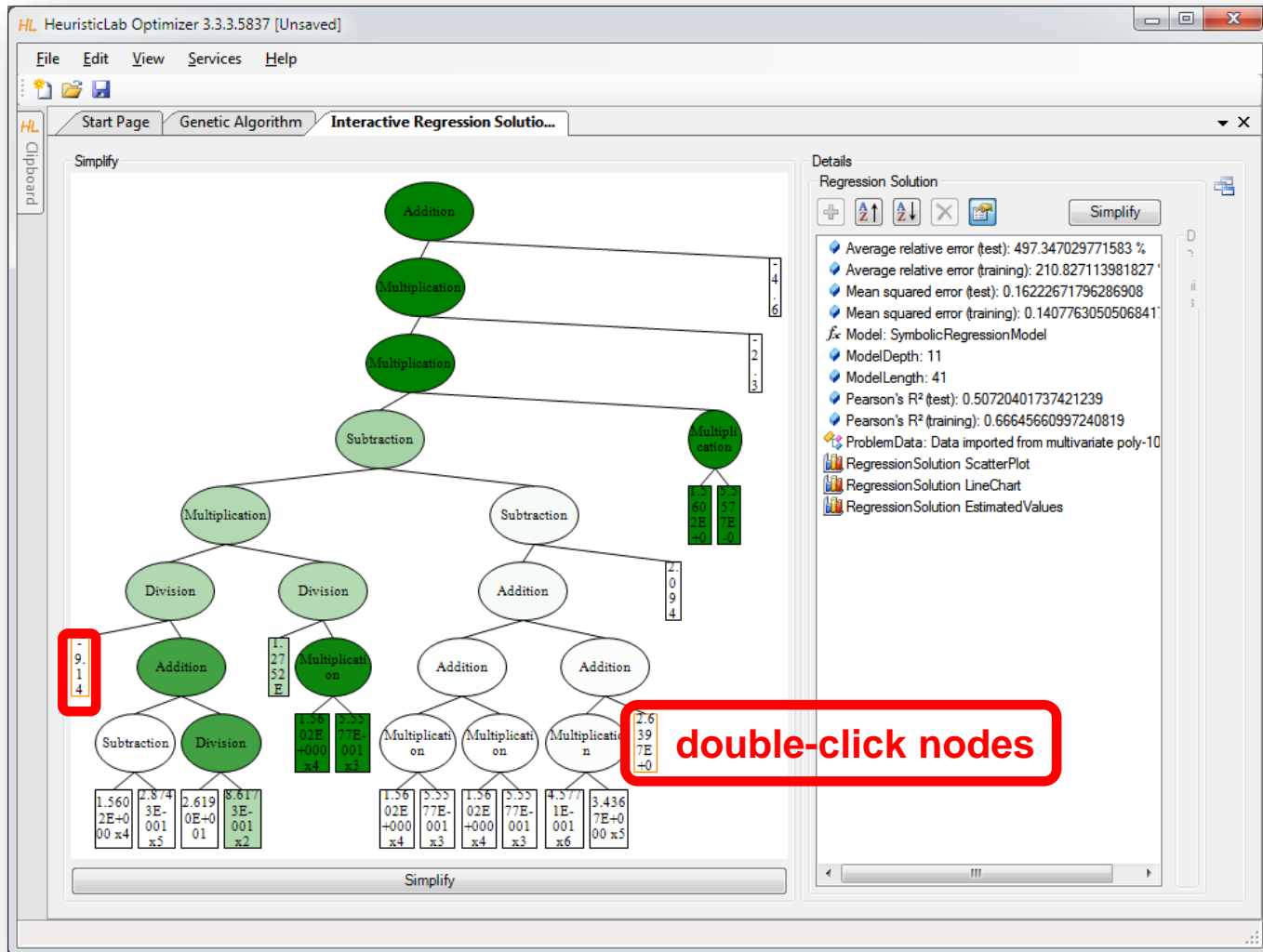
The screenshot displays the HeuristicLab Optimizer interface. The main window shows a symbolic regression tree with nodes labeled with mathematical operations: Addition, Multiplication, Subtraction, and Division. The tree is rooted at an Addition node. A red box highlights a specific node in the tree, which is an Addition node. A tooltip for this node shows the following information:

**Addition**  
Node impact: -0.00166473209841678  
Replacement value: 26.3974611821139

The right-hand side of the window shows the 'Details' panel for the 'Regression Solution'. It includes a 'Simplify' button and the following statistics:

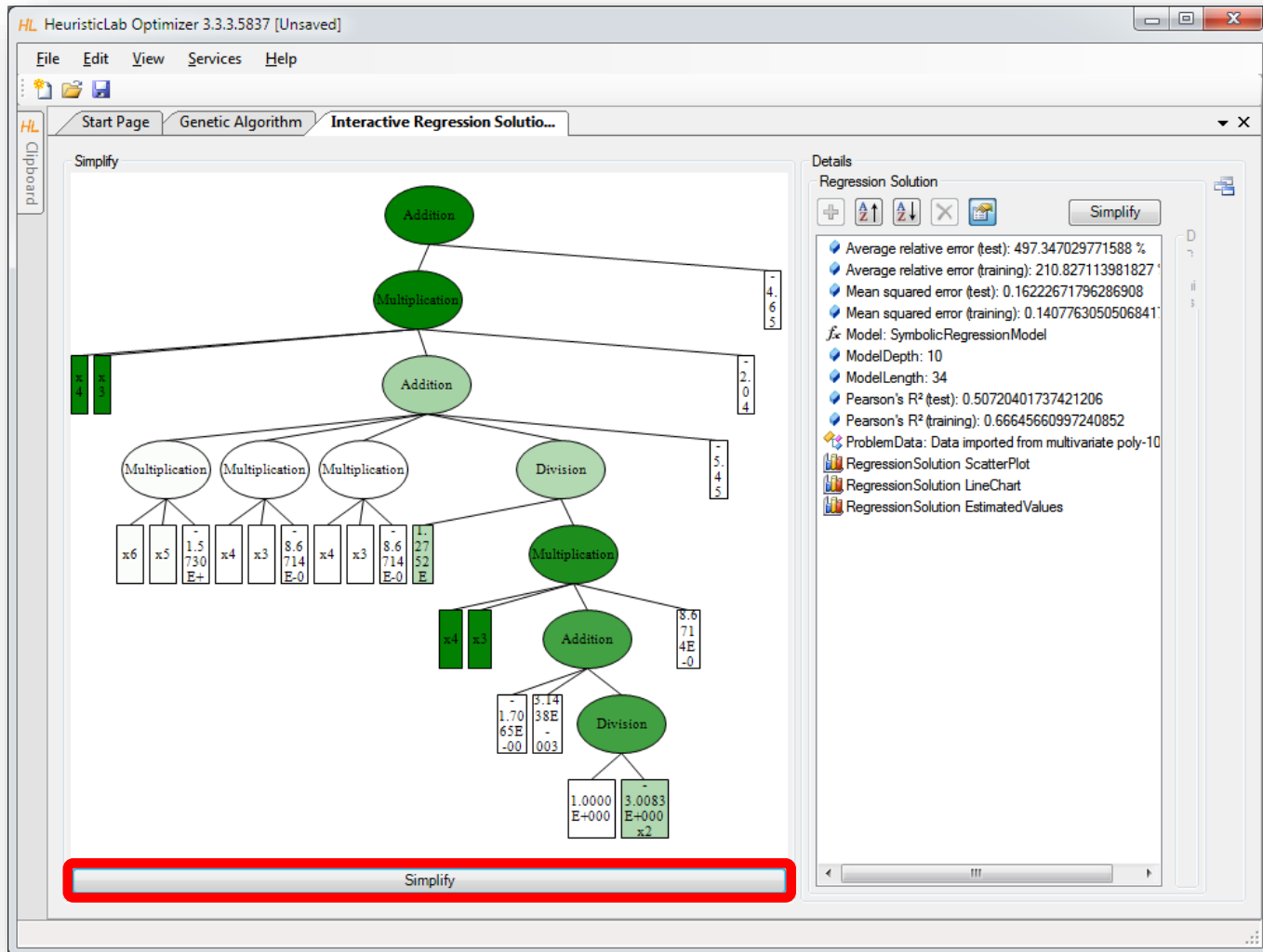
- Average relative error (test): 473.618715910248 %
- Average relative error (training): 245.090781502651 %
- Mean squared error (test): 0.163506091453717
- Mean squared error (training): 0.1415659913306669
- Model: SymbolicRegressionModel
- ModelDepth: 12
- ModelLength: 53
- Pearson's R<sup>2</sup> (test): 0.50347666984730766
- Pearson's R<sup>2</sup> (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 variables (e.g.,  $1.5602E+000x4$ ,  $2.6190E+001x2$ ,  $1.5602E+000x4$ ,  $3.5577E-001x3$ ,  $1.5602E+000x4$ ,  $3.5577E-001x3$ ,  $4.5771E-001x6$ ,  $3.4360E+000x5$ ). A red box highlights a node with the value  $-9.14$ . A red callout box with the text 'double-click nodes' points to a node with the value  $2.6397E+0$ . 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 R<sup>2</sup> (test): 0.50720401737421239, and Pearson's R<sup>2</sup> (training): 0.66645660997240819. The 'ProblemData' is noted as 'Data imported from multivariate poly-10'. The 'RegressionSolution' panel includes buttons for 'ScatterPlot', 'LineChart', and 'EstimatedValues'.

# 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 for the Multiplications:  $x^6$ ,  $x^5$ ,  $1.5730E+$ ,  $x^4$ ,  $x^3$ ,  $8.6714E-0$ ,  $x^4$ ,  $x^3$ ,  $8.6714E-0$ ,  $1.2752E$ .
- Level 4: Leaf nodes for the Division:  $x^4$ ,  $x^3$ .
- Level 5: Multiplication (Value:  $8.6714E-0$ )
- Level 6: Addition (Value:  $1.7065E-00$ ,  $5.1488E-003$ )
- Level 7: Division (Value:  $1.0000E+000$ ,  $3.0083E+000$ ,  $x^2$ )

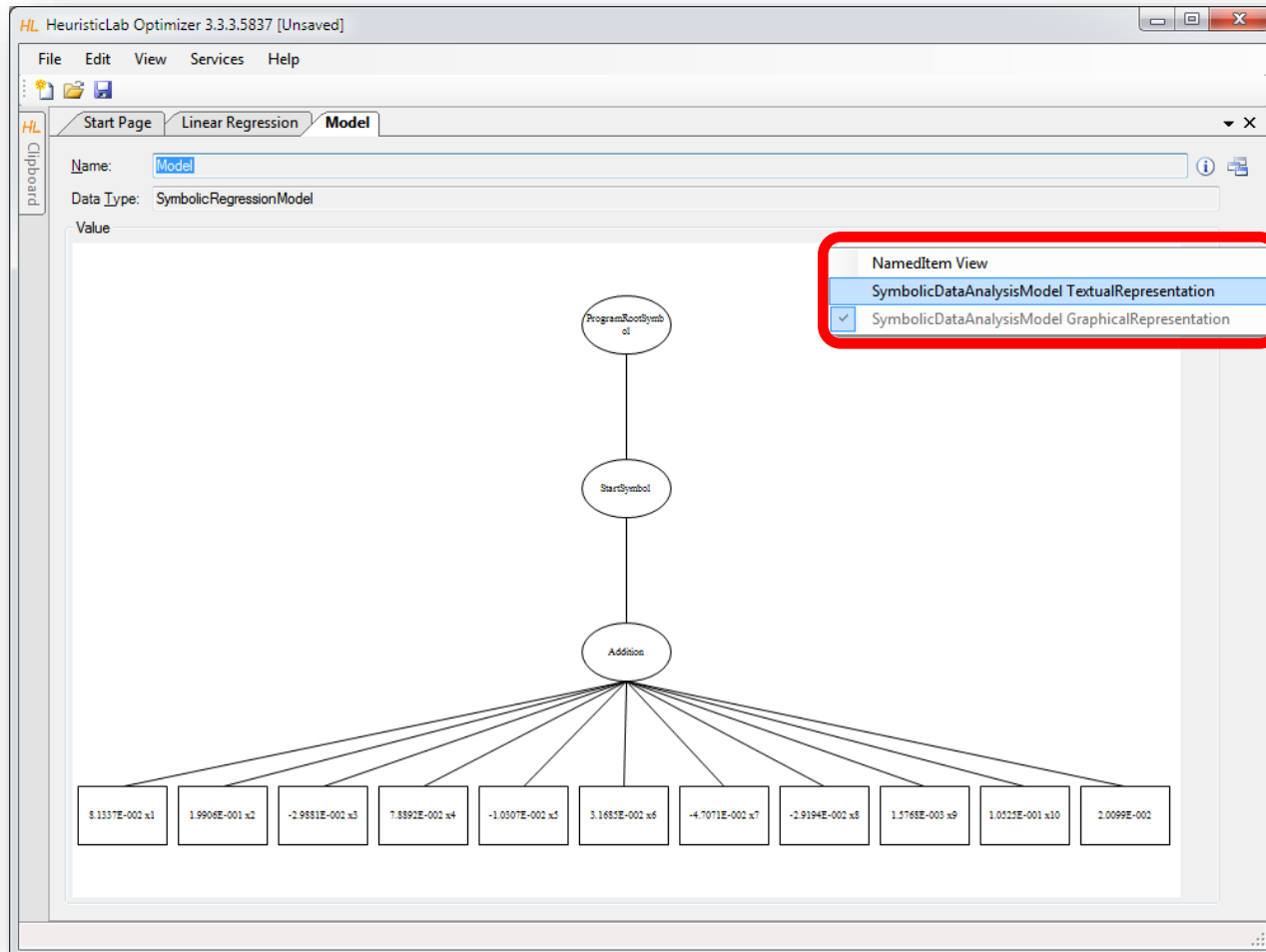
The "Details" panel on the right provides the following information:

- 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 R<sup>2</sup> (test): 0.50720401737421206
- Pearson's R<sup>2</sup> (training): 0.66645660997240852
- ProblemData: Data imported from multivariate poly-10
- RegressionSolution ScatterPlot
- RegressionSolution LineChart
- RegressionSolution EstimatedValues

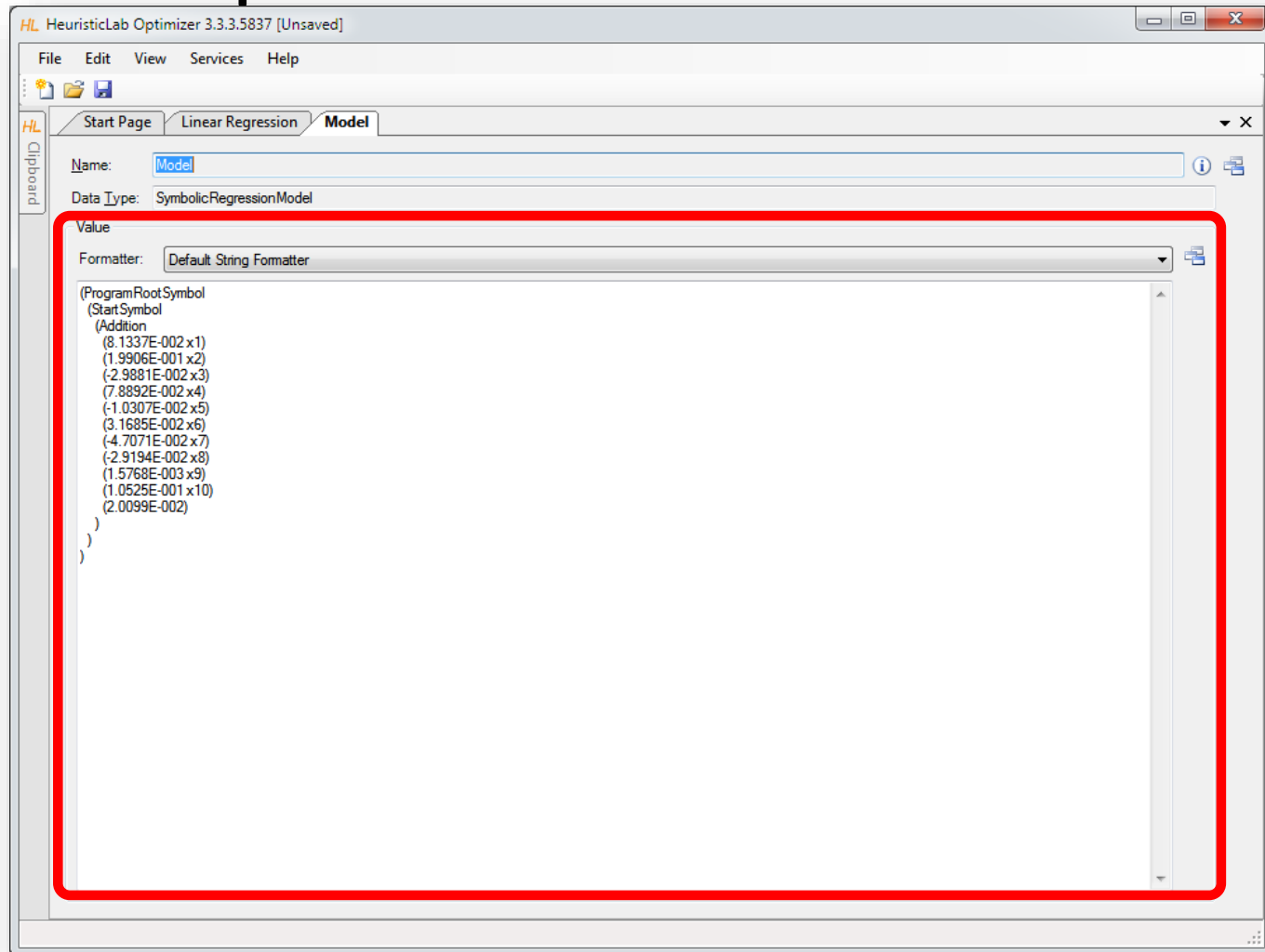
A red box highlights the "Simplify" button at the bottom of the main window.

# Textual Representations Are Also Available

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

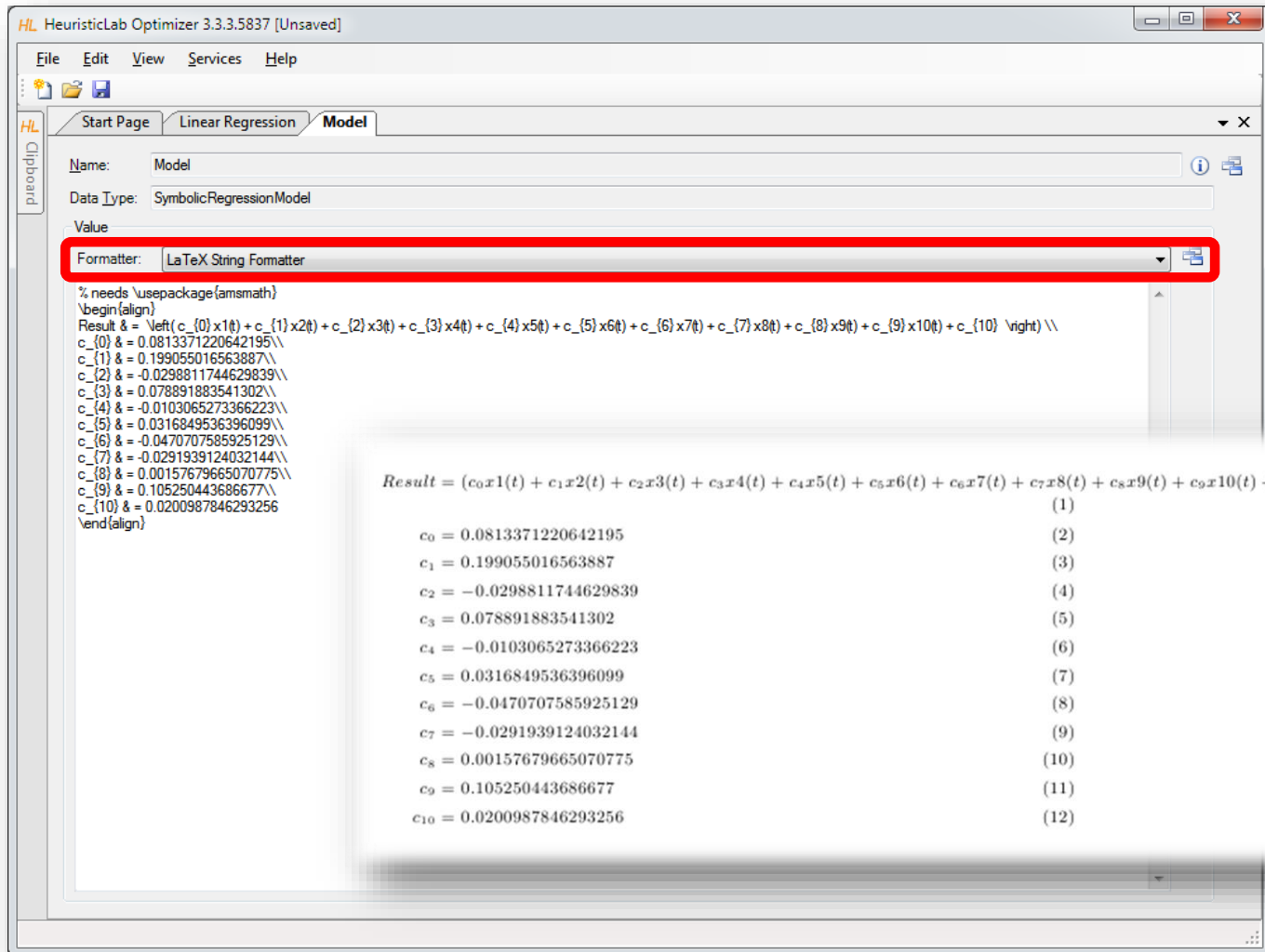


# Default Textual Representation for Model Export

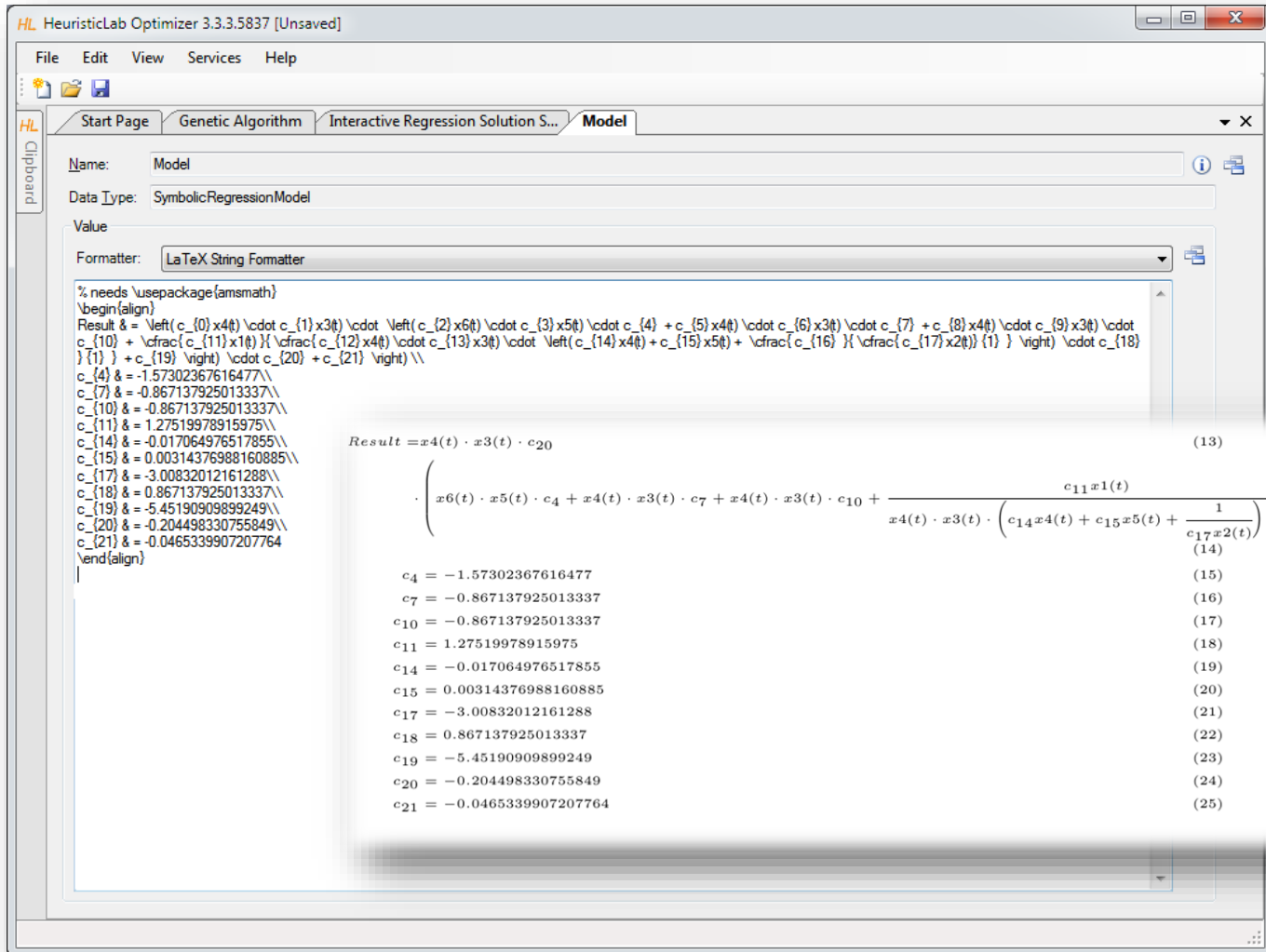




# Textual Representation for Export to LaTeX



# 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 main window displays the LaTeX code for the model's result and its coefficients.

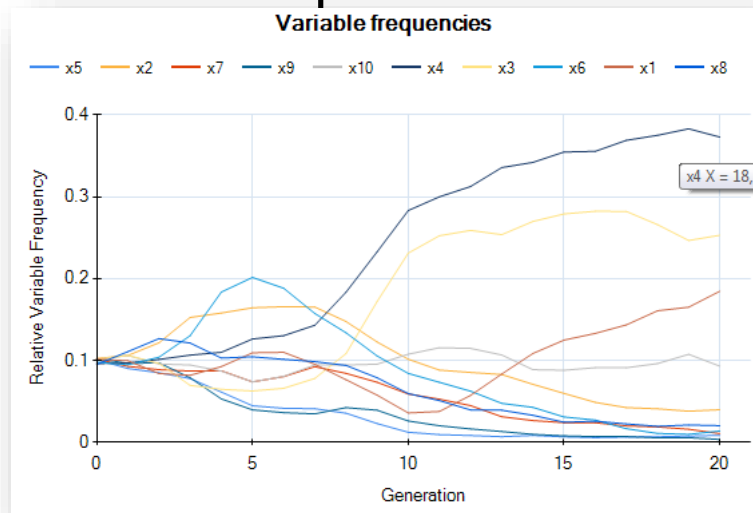
```

% needs \usepackage{amsmath}
\begin{align}
Result &= \left( c_4 x_4(t) + c_7 x_7(t) + c_{10} x_{10}(t) + c_{13} x_{13}(t) + c_{16} x_{16}(t) + c_{19} x_{19}(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}

```

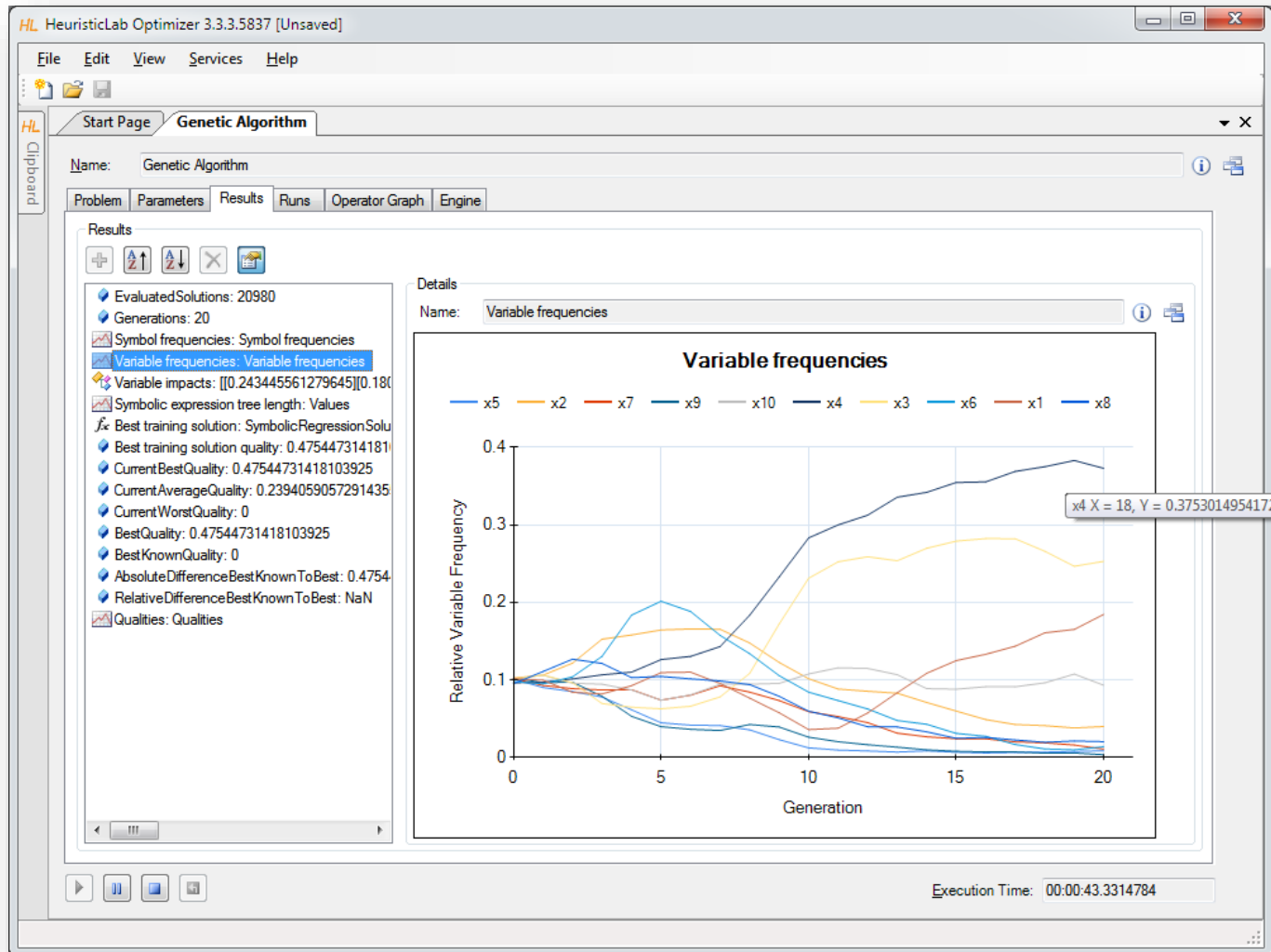
# Variable Relevance Analysis

- Which variables are important for correct predictions?
- Demonstration
  - Variable frequency analyzer
  - symbol frequency analyzer
  - variable impacts

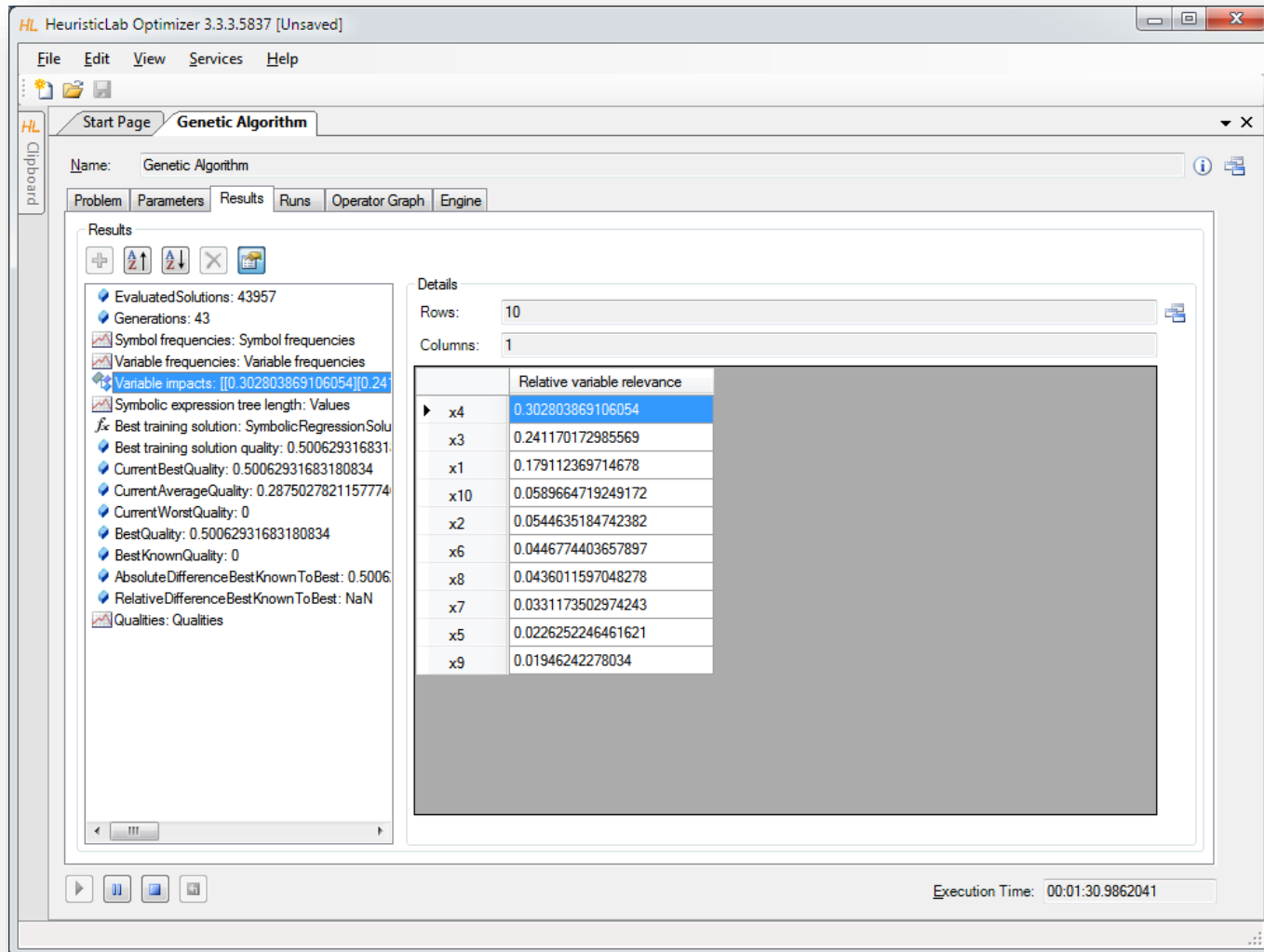


|     | Relative variable relevance |
|-----|-----------------------------|
| x4  | 0.302803869106054           |
| x3  | 0.241170172985569           |
| x1  | 0.179112369714678           |
| x10 | 0.0589664719249172          |
| x2  | 0.0544635184742382          |
| x6  | 0.0446774403657897        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

# Inspect Variable Frequency Chart



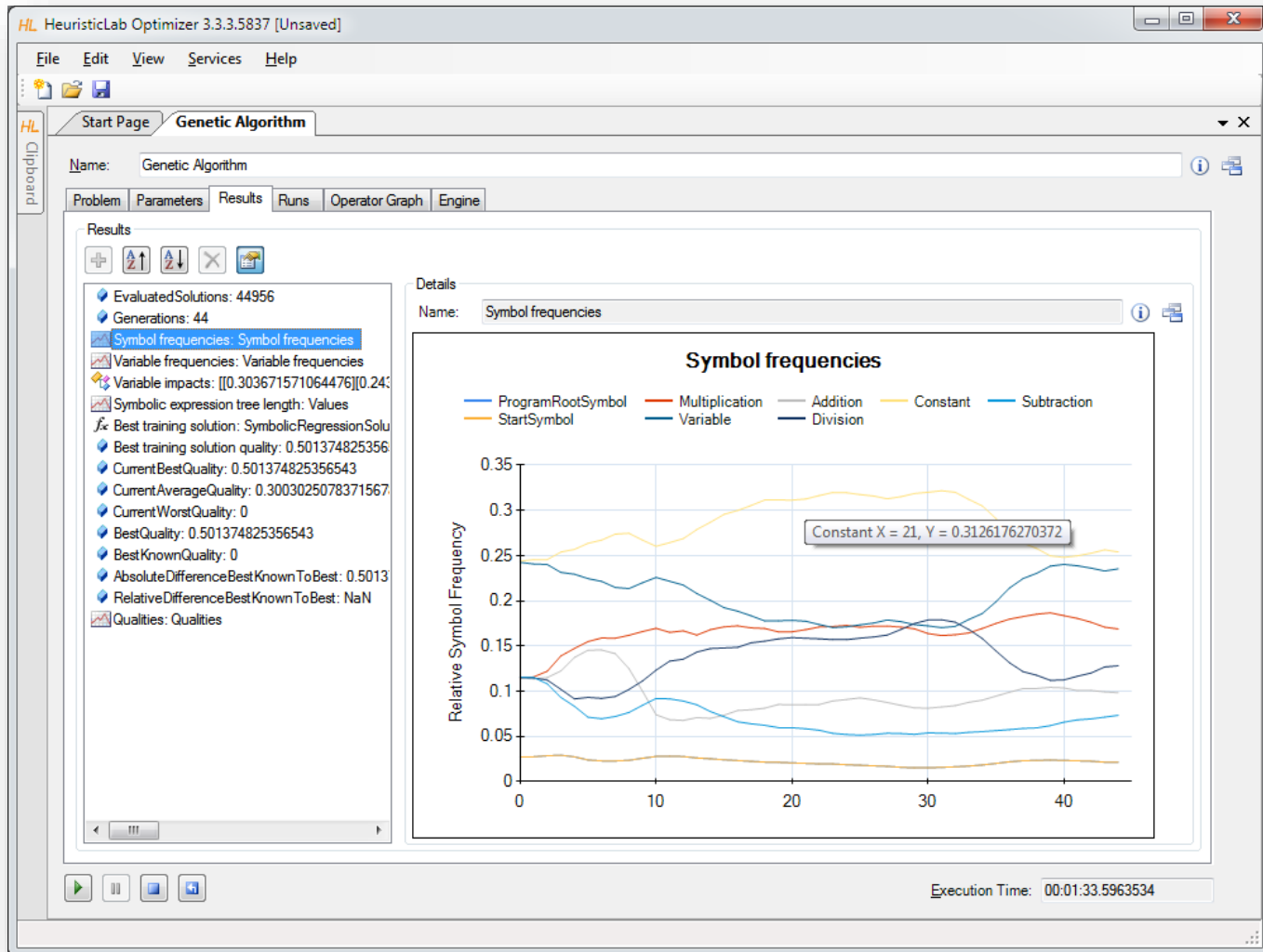
# Inspect Variable Impacts



The screenshot shows the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Results" tab is active, displaying a list of results on the left and a table of variable impacts on the right. The table is titled "Relative variable relevance" and lists variables x1 through x10 with their corresponding relevance values. The variable x4 has the highest relevance, highlighted in blue.

| Variable | 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            |

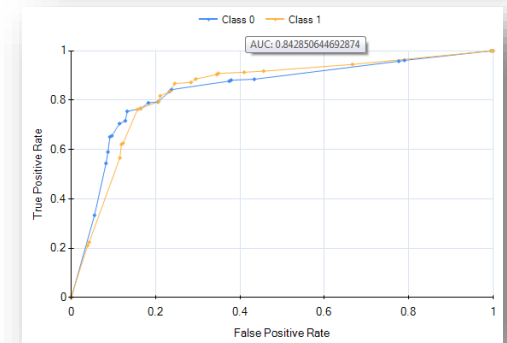
# Inspect Symbol Frequencies



# 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



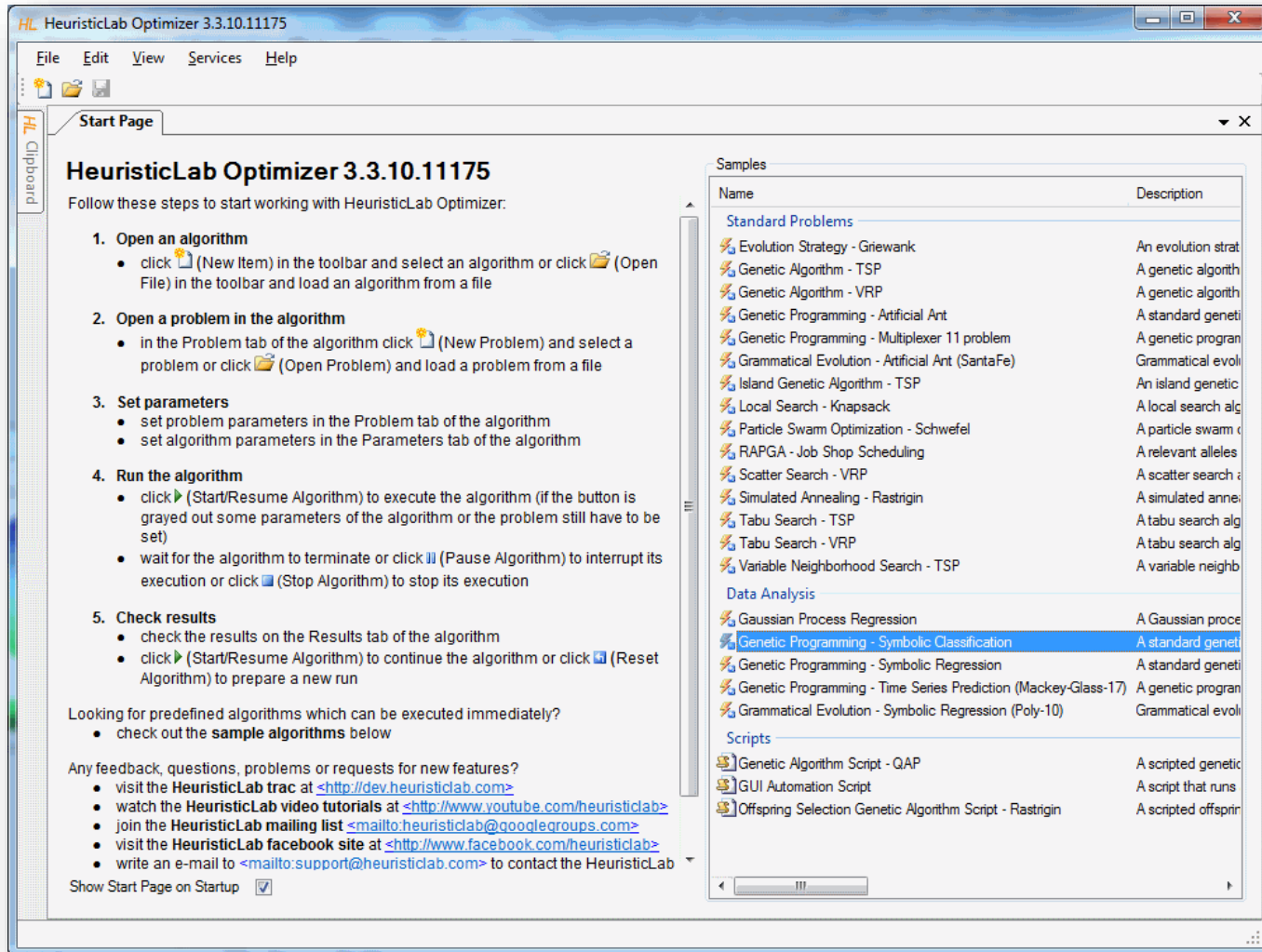
|                   | Actual Class 0 | Actual Class 1 |
|-------------------|----------------|----------------|
| Predicted Class 0 | 197            | 29             |
| Predicted Class 1 | 64             | 190            |

# 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)
  - available as a benchmark problem instance in HeuristicLab












# Open Sample



The screenshot shows the HeuristicLab Optimizer 3.3.10.11175 application window. The main area displays a 'Start Page' with a list of instructions for starting a new optimization run. On the right, a 'Samples' panel lists various optimization problems and scripts, with 'Genetic Programming - Symbolic Classification' selected.

**HeuristicLab Optimizer 3.3.10.11175**  
Follow these steps to start working with HeuristicLab Optimizer:

- 1. Open an algorithm**
  - click  (New Item) in the toolbar and select an algorithm or click  (Open File) in the toolbar and load an algorithm from a file
- 2. Open a problem in the algorithm**
  - in the Problem tab of the algorithm click  (New Problem) and select a problem or click  (Open Problem) and load a problem from a file
- 3. Set parameters**
  - set problem parameters in the Problem tab of the algorithm
  - set algorithm parameters in the Parameters tab of the algorithm
- 4. Run the algorithm**
  - click  (Start/Resume Algorithm) to execute the algorithm (if the button is grayed out some parameters of the algorithm or the problem still have to be set)
  - wait for the algorithm to terminate or click  (Pause Algorithm) to interrupt its execution or click  (Stop Algorithm) to stop its execution
- 5. Check results**
  - check the results on the Results tab of the algorithm
  - click  (Start/Resume Algorithm) to continue the algorithm or click  (Reset Algorithm) to prepare a new run
























Looking for predefined algorithms which can be executed immediately?

- check out the **sample algorithms** below

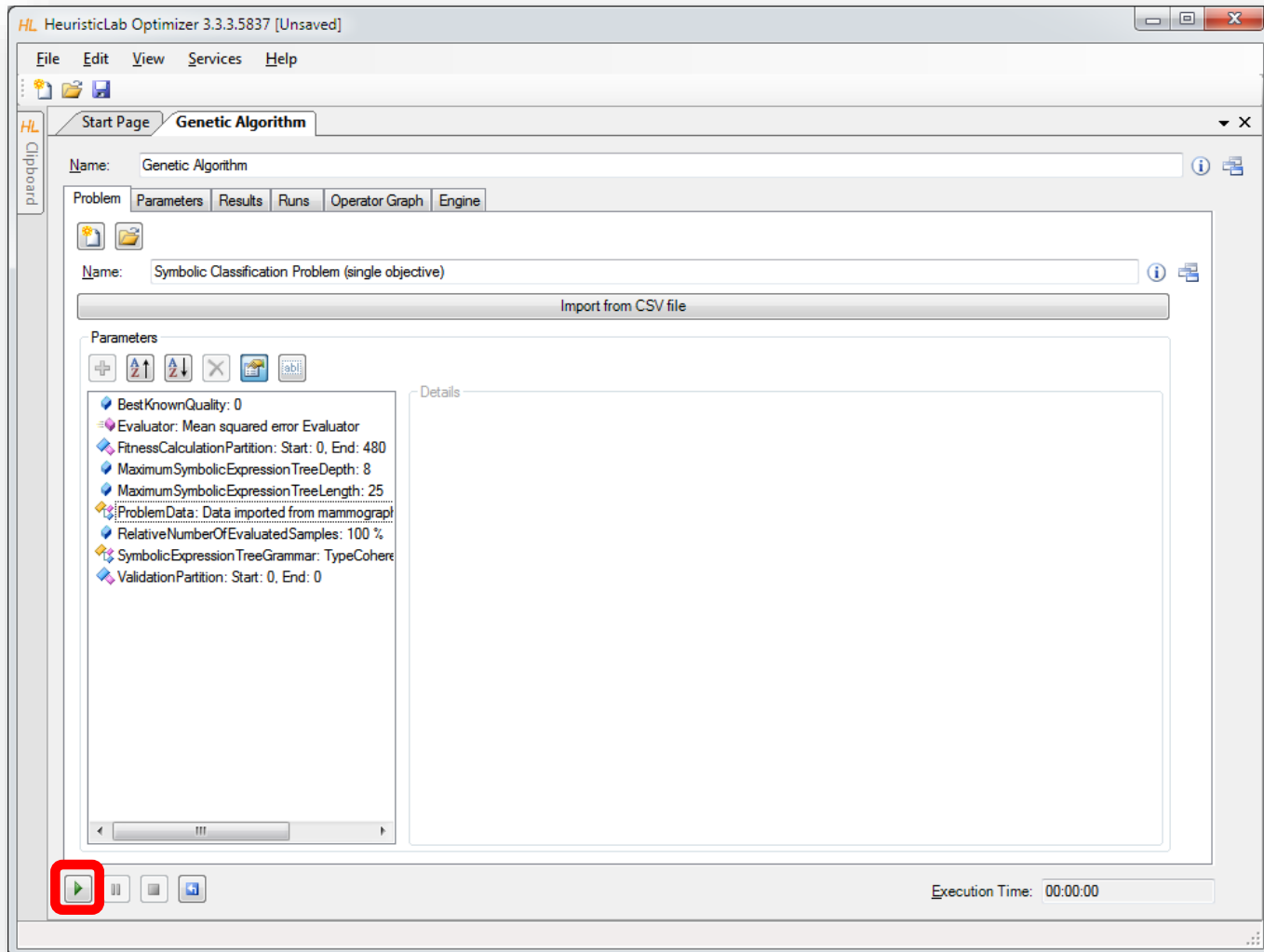
Any feedback, questions, problems or requests for new features?

- visit the **HeuristicLab trac** at <http://dev.heuristiclab.com>
- watch the **HeuristicLab video tutorials** at <http://www.youtube.com/heuristiclab>
- join the **HeuristicLab mailing list** <mailto:heuristiclab@googlegroups.com>
- visit the **HeuristicLab facebook site** at <http://www.facebook.com/heuristiclab>
- write an e-mail to <mailto:support@heuristiclab.com> to contact the HeuristicLab

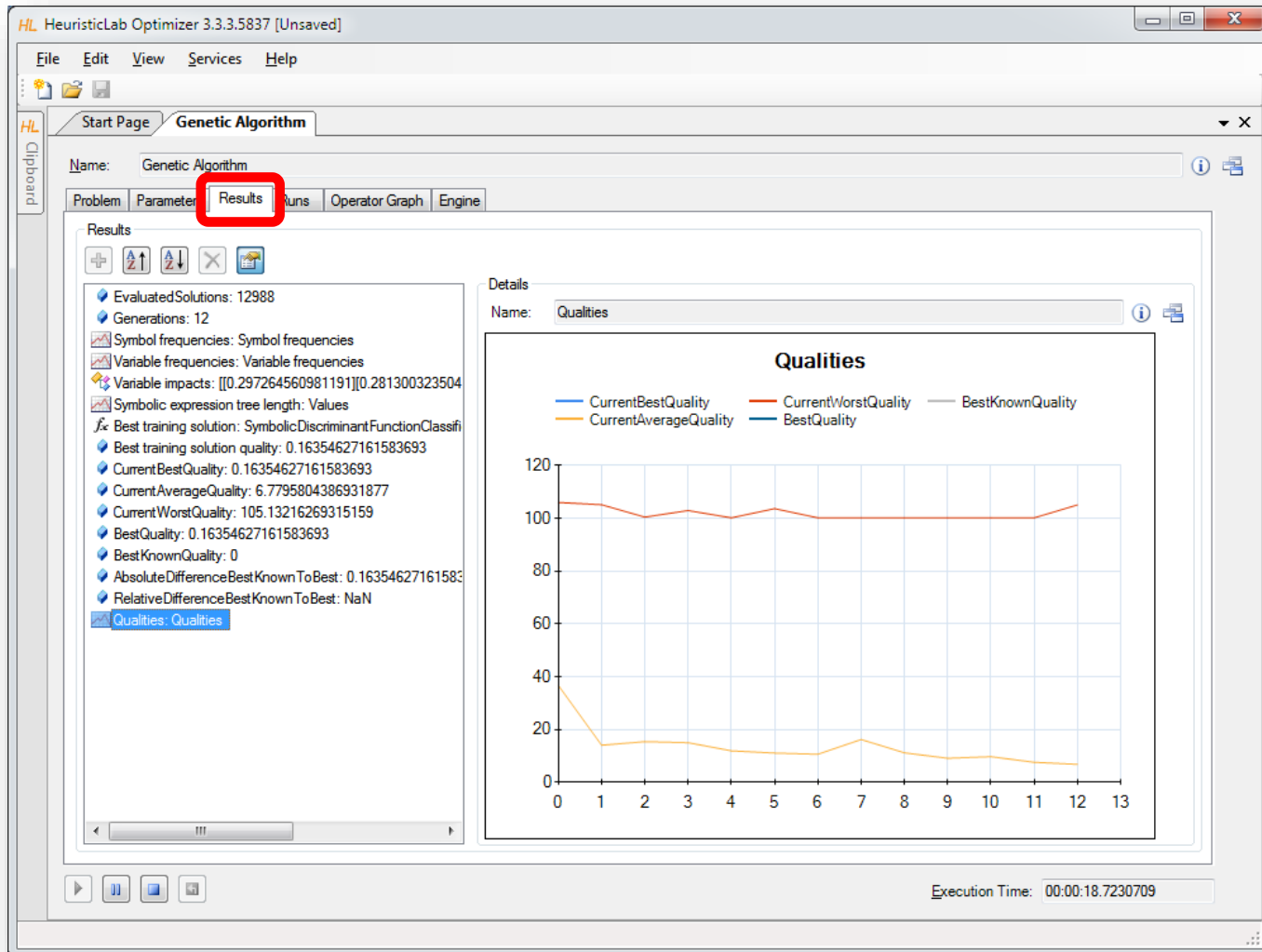
Show Start Page on Startup

| Name   | Description         |
|--|---------------------|
| <b>Standard Problems</b>   |                     |
|  Evolution Strategy - Griewank                                    | An evolution strat  |
|  Genetic Algorithm - TSP  | A genetic algorithm |
|  Genetic Algorithm - VRP  | A genetic algorithm |
|  Genetic Programming - Artificial Ant                             | A standard geneti   |
|  Genetic Programming - Multiplexer 11 problem                     | A genetic program   |
|  Grammatical Evolution - Artificial Ant (SantaFe)                 | Grammatical evoli   |
|  Island Genetic Algorithm - TSP                                   | An island genetic   |
|  Local Search - Knapsack  | A local search alg  |
|  Particle Swarm Optimization - Schwefel                           | A particle swam e   |
|  RAPGA - Job Shop Scheduling                                      | A relevant alleles  |
|  Scatter Search - VRP   | A scatter search a  |
|  Simulated Annealing - Rastrigin                                  | A simulated anne    |
|  Tabu Search - TSP  | A tabu search alg   |
|  Tabu Search - VRP  | A tabu search alg   |
|  Variable Neighborhood Search - TSP                               | A variable neighb   |
| <b>Data Analysis</b>   |                     |
|  Gaussian Process Regression                                      | A Gaussian proce    |
|  Genetic Programming - Symbolic Classification                    | A standard geneti   |
|  Genetic Programming - Symbolic Regression                       | A standard geneti   |
|  Genetic Programming - Time Series Prediction (Mackey-Glass-17) | A genetic program   |
|  Grammatical Evolution - Symbolic Regression (Poly-10)          | Grammatical evoli   |
| <b>Scripts</b>   |                     |
|  Genetic Algorithm Script - QAP                                 | A scripted genetic  |
|  GUI Automation Script  | A script that runs  |
|  Offspring Selection Genetic Algorithm Script - Rastrigin       | A scripted offsprin |

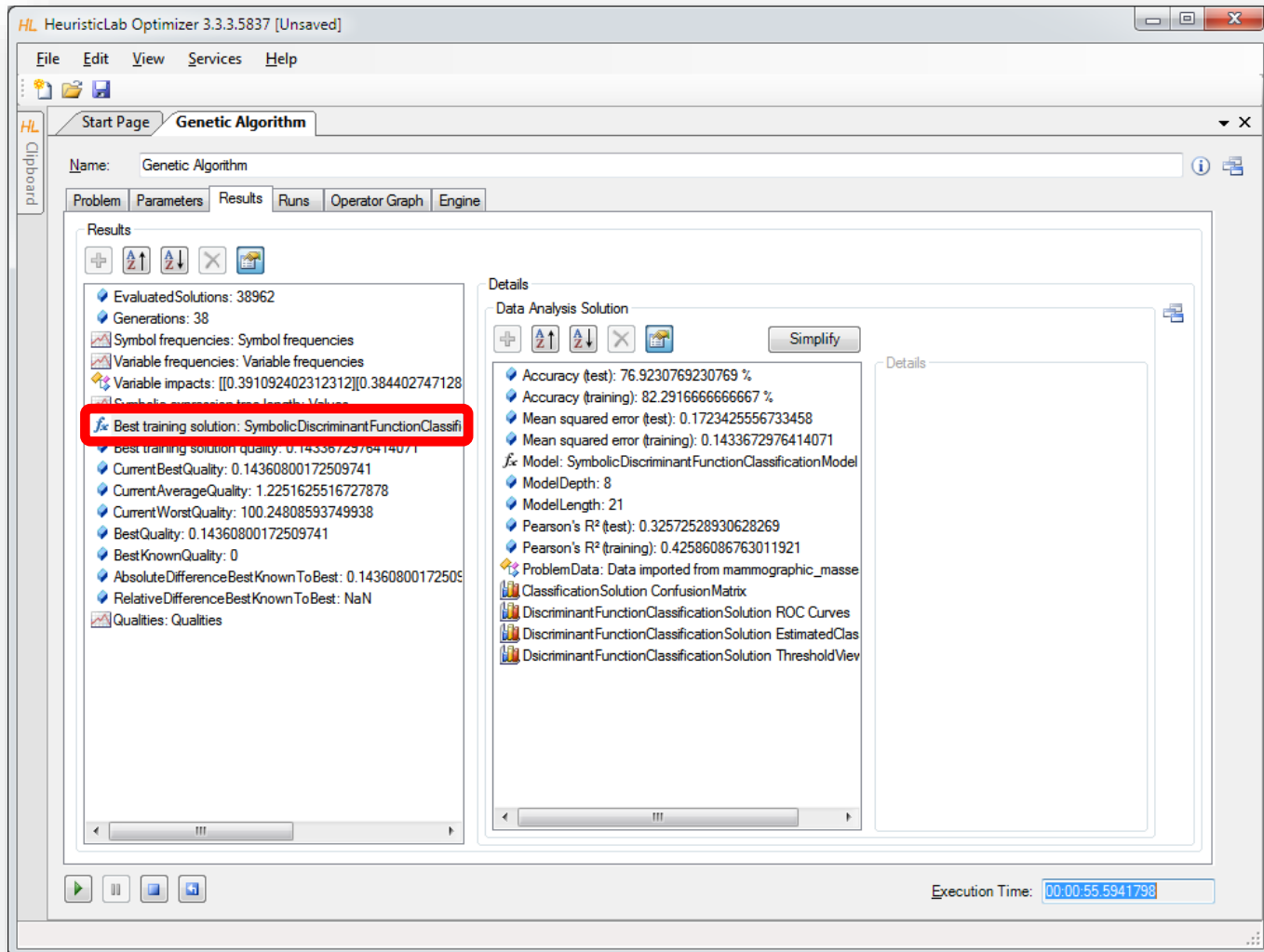
# Configure and Run Algorithm



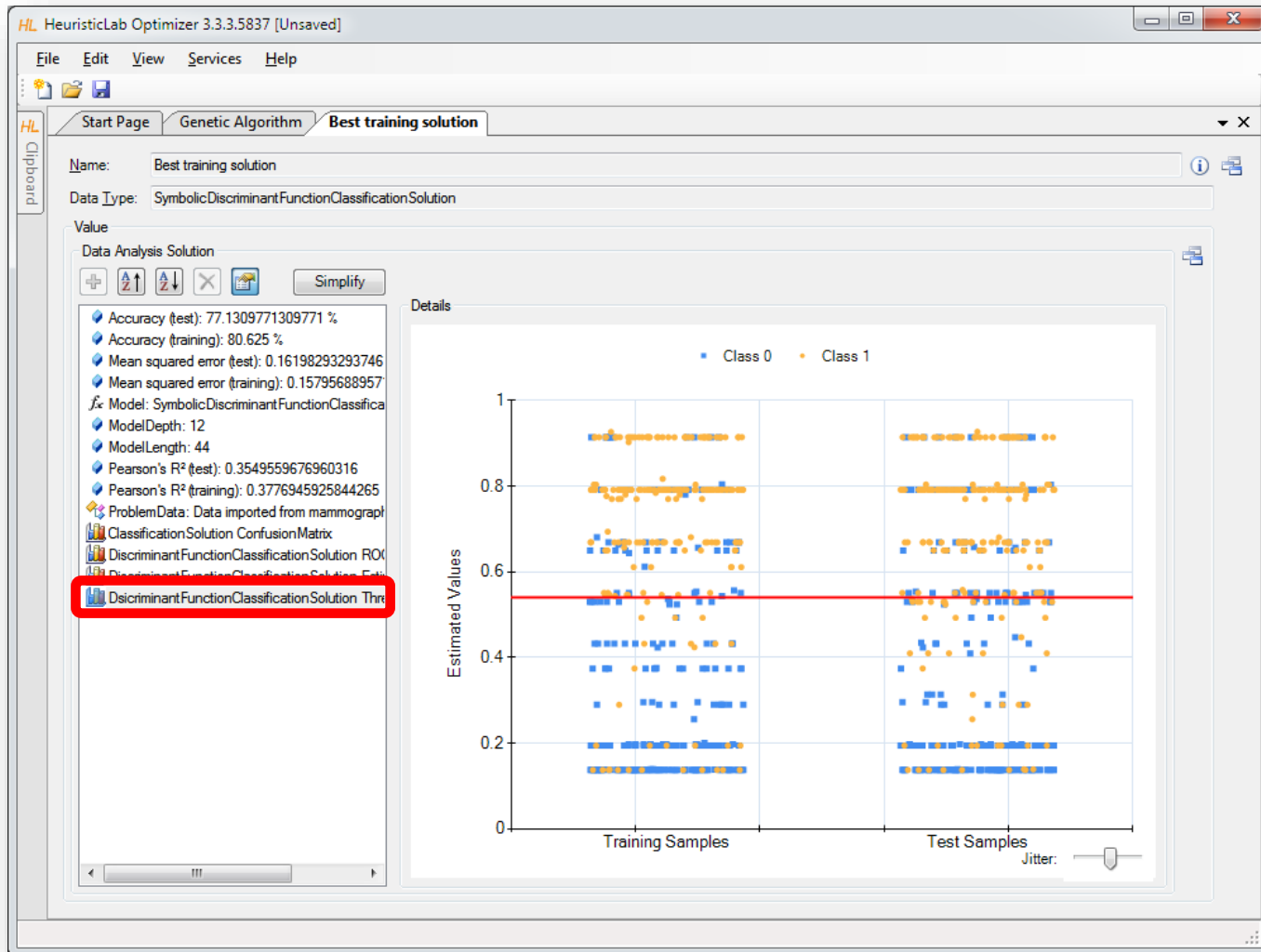
# Inspect Quality Linechart



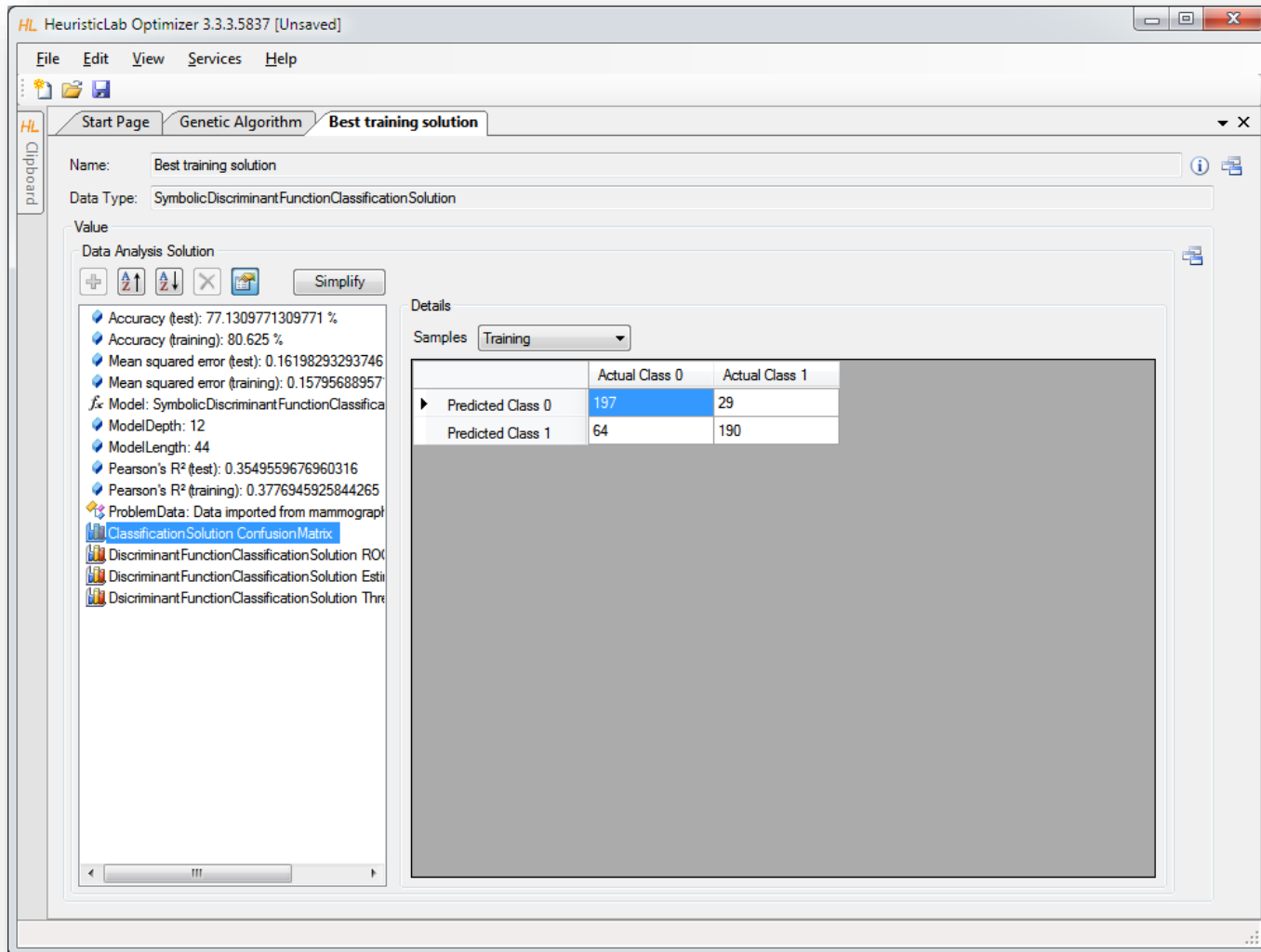
# Inspect Best Training Solution



# Inspect Model Output and Thresholds



# Inspect Confusion Matrix



HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

Start Page Genetic Algorithm **Best training solution**

Name: Best training solution

Data Type: SymbolicDiscriminantFunctionClassificationSolution

Value

Data Analysis Solution

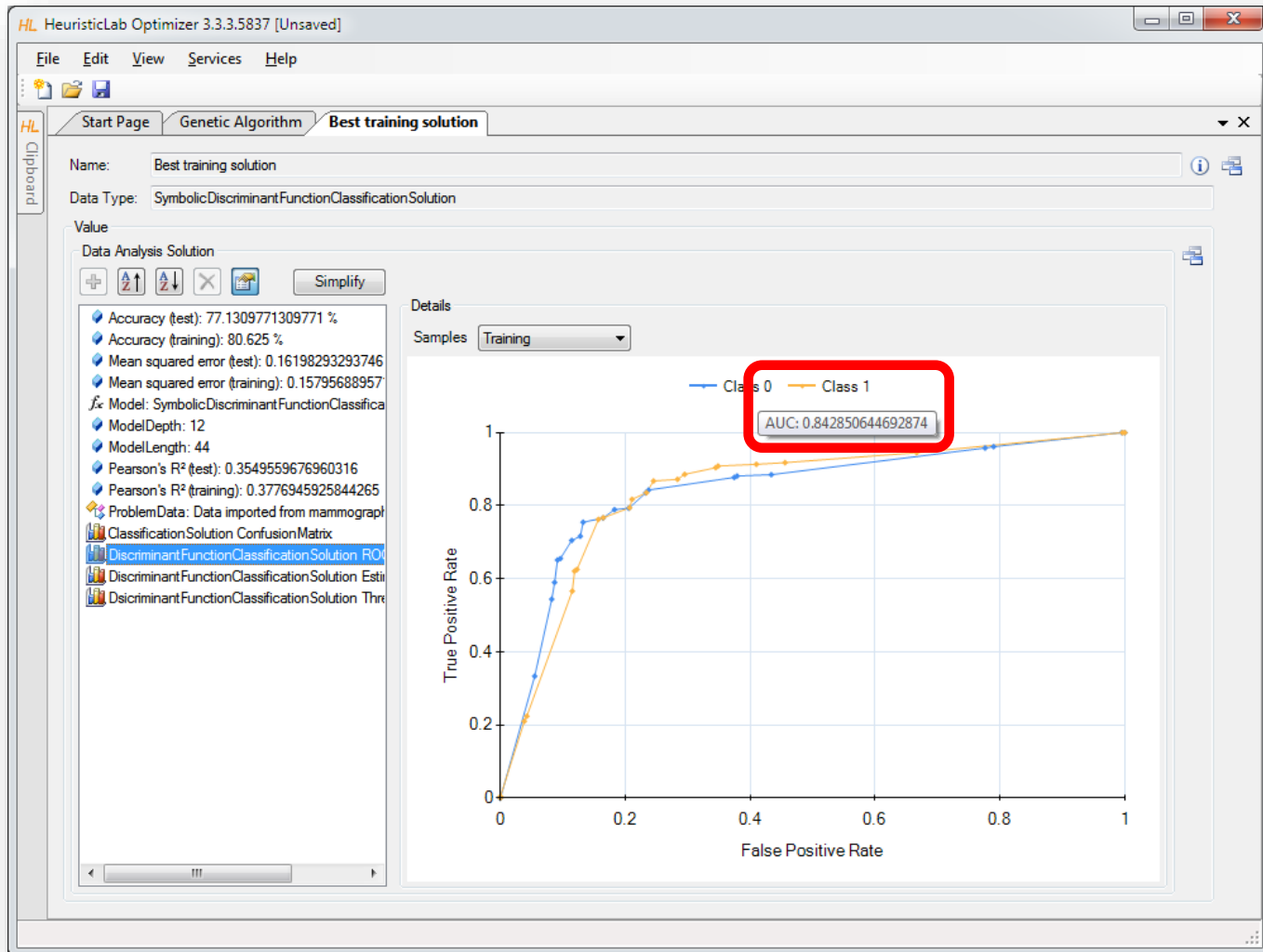
Accuracy (test): 77.1309771309771 %  
Accuracy (training): 80.625 %  
Mean squared error (test): 0.16198293293746  
Mean squared error (training): 0.15795688957  
Model: SymbolicDiscriminantFunctionClassifica  
ModelDepth: 12  
ModelLength: 44  
Pearson's R<sup>2</sup> (test): 0.3549559676960316  
Pearson's R<sup>2</sup> (training): 0.3776945925844265  
ProblemData: Data imported from mammograph  
ClassificationSolution ConfusionMatrix  
DiscriminantFunctionClassificationSolution RO  
DiscriminantFunctionClassificationSolution Esti  
DiscriminantFunctionClassificationSolution Thre

Details

Samples Training

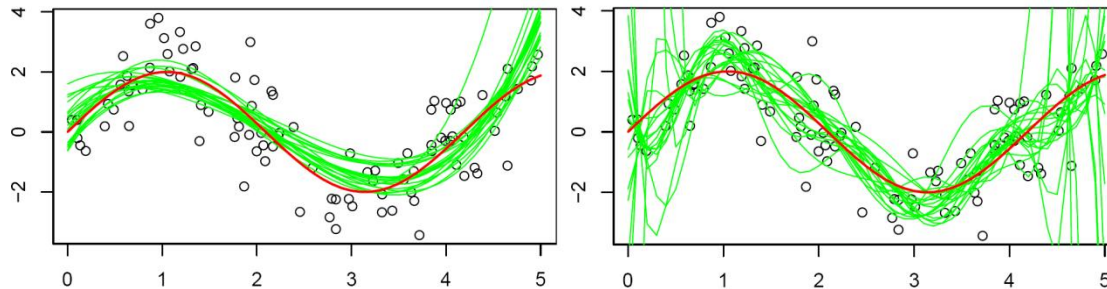
|                   | Actual Class 0 | Actual Class 1 |
|-------------------|----------------|----------------|
| Predicted Class 0 | 197            | 29             |
| Predicted Class 1 | 64             | 190            |

# Inspect ROC Curve



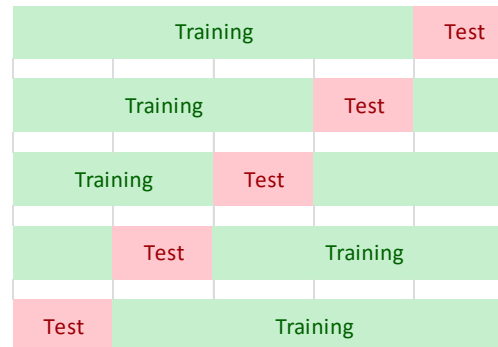
# Validation of Results

- Overfitting = memorizing data



- Strategies to reduce overfitting

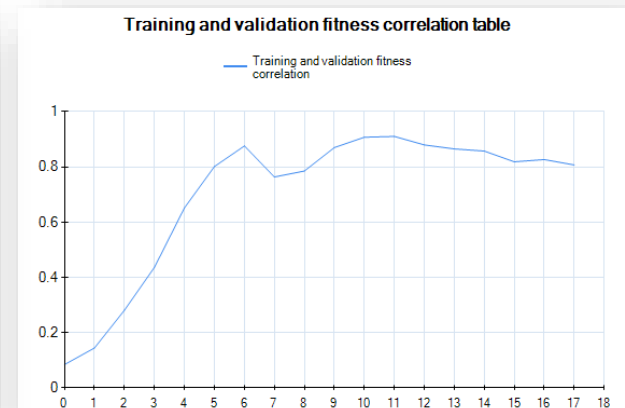
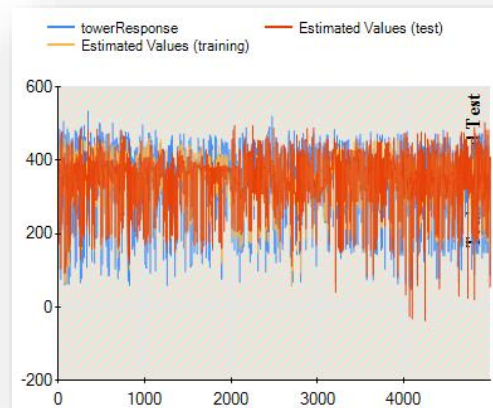
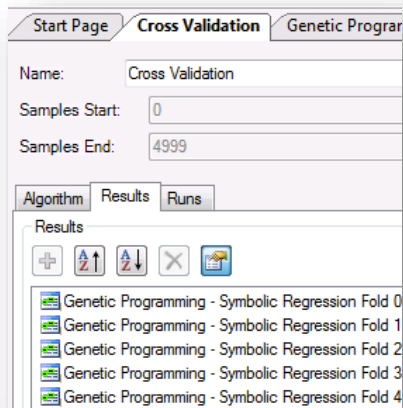
- validation partition
- cross-validation



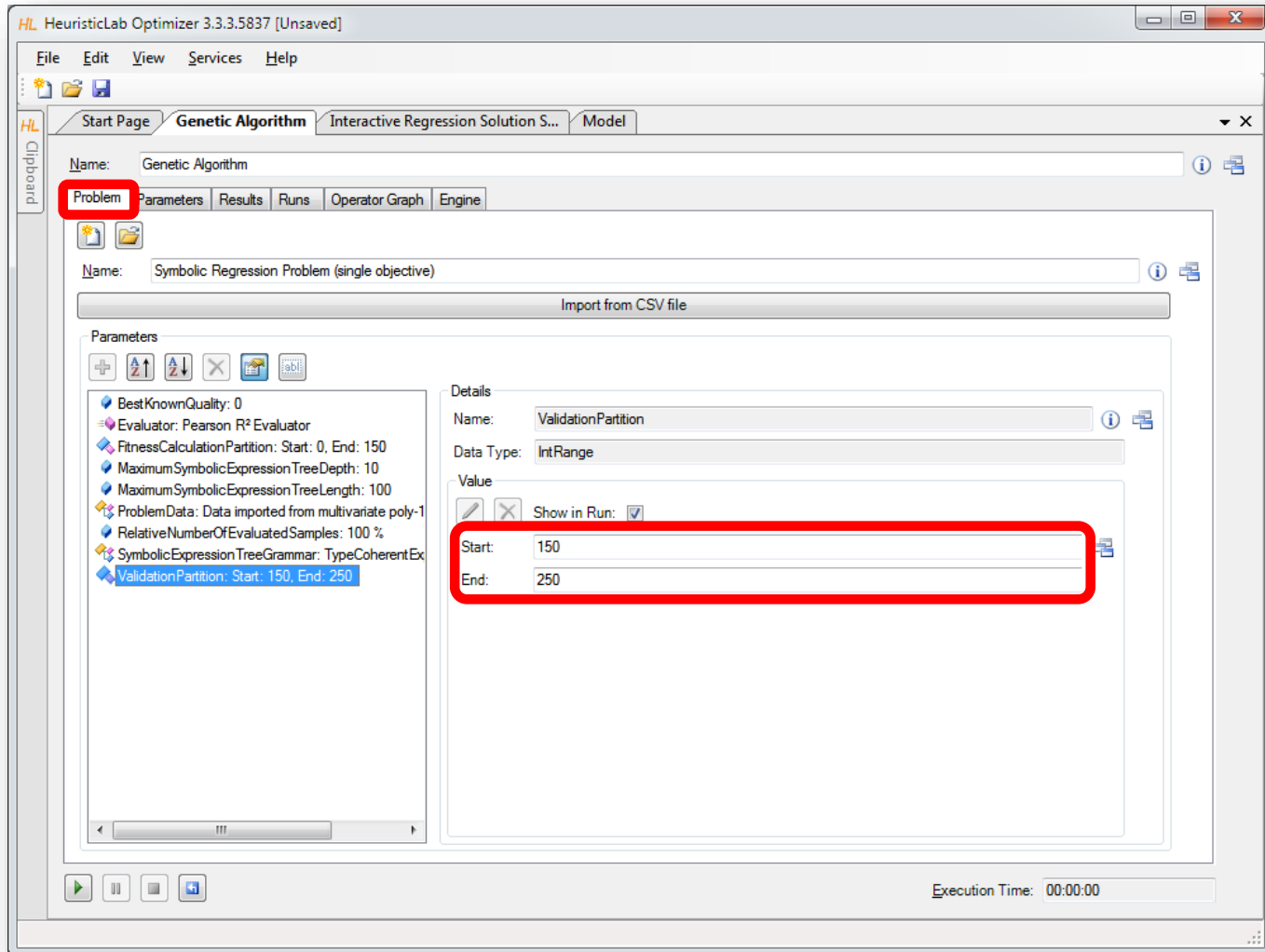


# Validation of Results

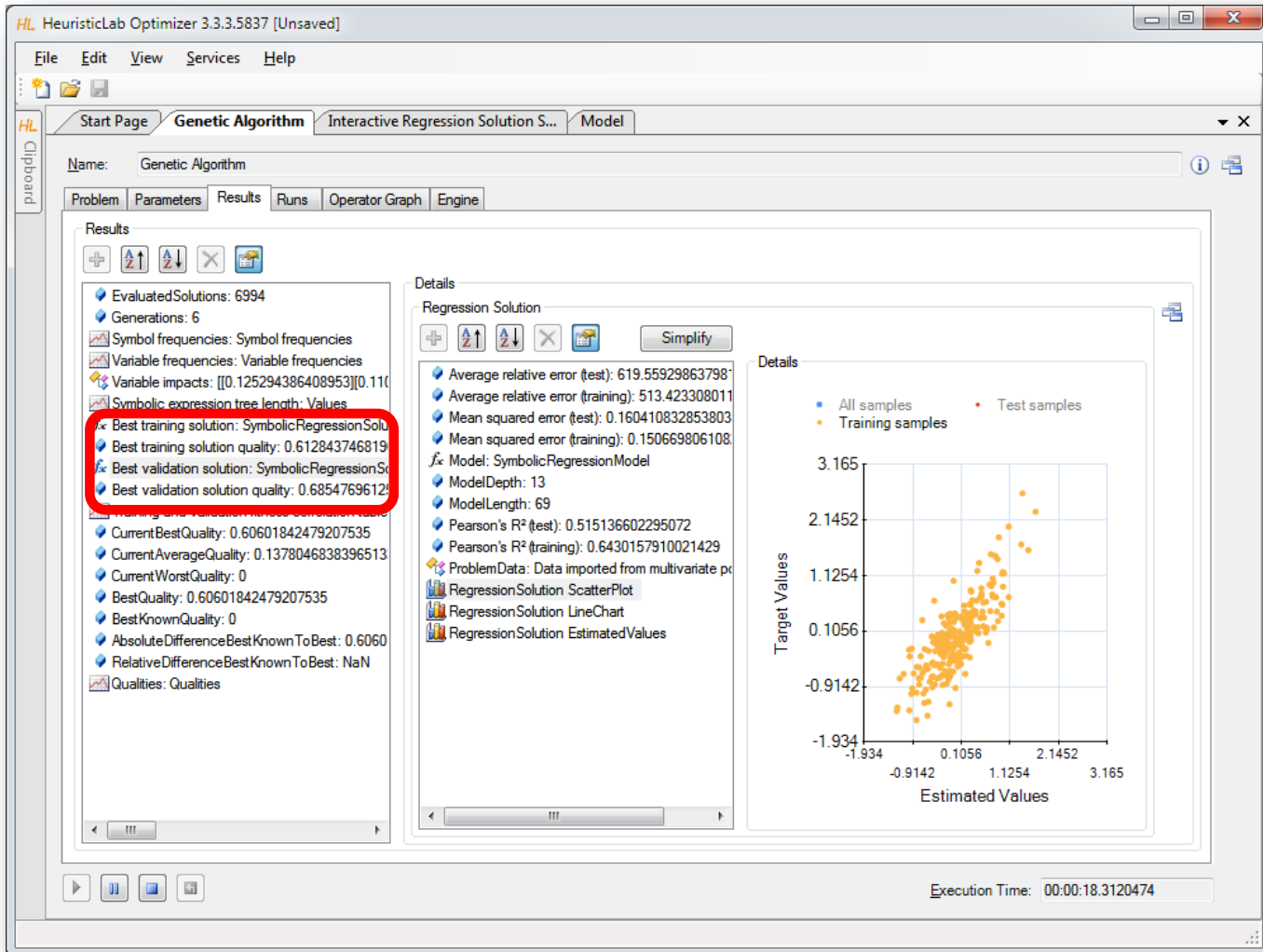
- Demonstration
  - Configuration of a validation set
  - Inspection of best solution on validation set
  - Analysis of training- and validation fitness correlation
  - Cross-validation
    - Configuration
    - Analysis of results



# Configuration of Validation Partition



# Inspect Best Model on Validation Partition



The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Results" tab is active, showing the following information:

- Evaluated Solutions: 6994
- Generations: 6
- Symbol frequencies: Symbol frequencies
- Variable frequencies: Variable frequencies
- Variable impacts:  $[[0.125294386408953][0.110...]]$
- Symbolic expression tree length: Values
- Best training solution: SymbolicRegressionSol...
- Best training solution quality: 0.612843746819
- Best validation solution: SymbolicRegressionS...
- Best validation solution quality: 0.6854769612...

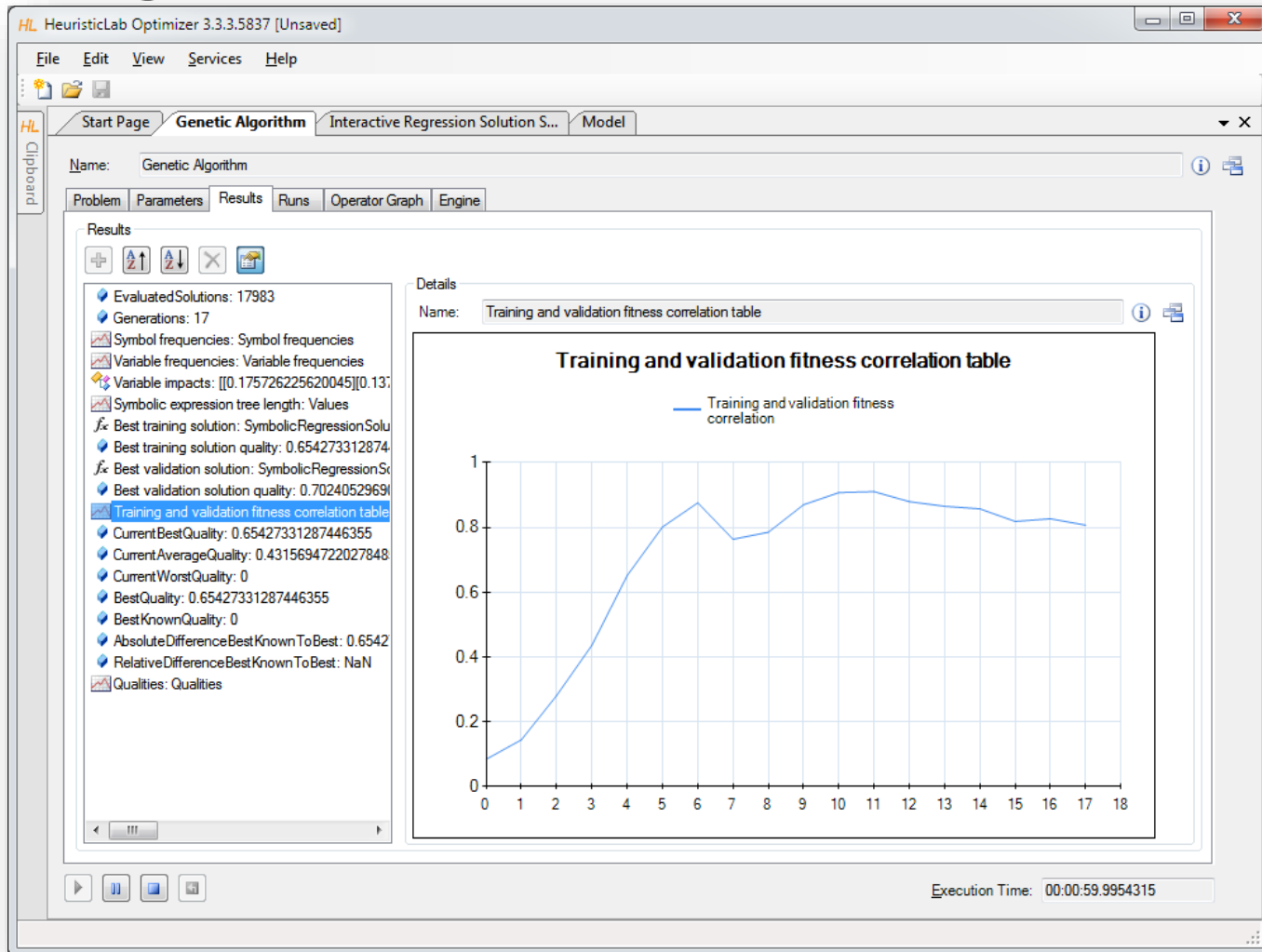
The "Details" tab is also active, showing the following information:

- Regression Solution
- Average relative error (test): 619.55929863798
- Average relative error (training): 513.423308011
- Mean squared error (test): 0.160410832853803
- Mean squared error (training): 0.150669806108
- Model: SymbolicRegressionModel
- ModelDepth: 13
- ModelLength: 69
- Pearson's R<sup>2</sup> (test): 0.515136602295072
- Pearson's R<sup>2</sup> (training): 0.6430157910021429
- ProblemData: Data imported from multivariate p...
- RegressionSolution ScatterPlot
- RegressionSolution LineChart
- RegressionSolution EstimatedValues

A scatter plot titled "Target Values" vs "Estimated Values" is shown on the right. The plot displays two data series: "All samples" (blue dots) and "Training samples" (orange dots). The x-axis is labeled "Estimated Values" and ranges from -1.934 to 3.165. The y-axis is labeled "Target Values" and ranges from -1.934 to 3.165. The training samples are clustered around the diagonal line, indicating a good fit to the training data.

Execution Time: 00:00:18.3120474

# Inspect Linechart of Correlation of Training and Validation Fitness



# Agenda



- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
  
- **Demonstration Part I: Working with HeuristicLab**
- **Demonstration Part II: Data-based Modeling**
  
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

# Some Additional Features

- HeuristicLab Hive
  - parallel and distributed execution of algorithms and experiments on many computers in a network
- Optimization Knowledge Base (OKB)
  - database to store algorithms, problems, parameters and results
  - open to the public
  - open for other frameworks
  - analyze and store characteristics of problem instances and problem classes
- External solution evaluation and simulation-based optimization
  - interface to couple HeuristicLab with other applications (MATLAB, Simulink, SciLab, AnyLogic, ...)
  - supports different protocols (command line parameters, TCP, ...)
- Parameter grid tests and meta-optimization
  - automatically create experiments to test large ranges of parameters
  - apply heuristic optimization algorithms to find optimal parameter settings for heuristic optimization algorithms



# Planned Features

- Algorithms & Problems
  - steady-state genetic algorithm
  - unified tabu search for vehicle routing
  - estimation of distribution algorithms
  - evolution of arbitrary code (Robocode, controller, etc.)
  - ...
- Cloud Computing
  - port HeuristicLab Hive to Windows Azure
- Have a look at the HeuristicLab roadmap
  - <http://dev.heuristiclab.com/trac.fcgi/roadmap>
- Any other ideas, requests or recommendations?
  - join our HeuristicLab Google group [heuristiclab@googlegroups.com](mailto:heuristiclab@googlegroups.com)
  - write an e-mail to [support@heuristiclab.com](mailto:support@heuristiclab.com)

# HeuristicLab Team



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

Softwarepark 11  
A-4232 Hagenberg  
AUSTRIA

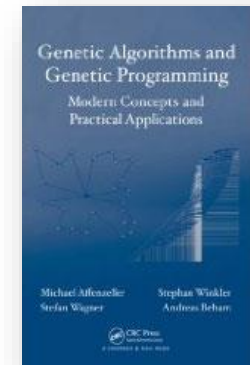
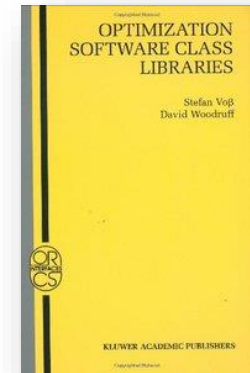
WWW: <http://heal.heuristiclab.com>





# Suggested Readings

- S. Voß, D. Woodruff (Edts.)  
**Optimization Software Class Libraries**  
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
- S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, M. Affenzeller  
**Architecture and Design of the HeuristicLab Optimization Environment**  
Advanced Methods and Applications in Computational Intelligence, vol. 6, pp. 197-261, Springer, 2014
- Detailed list of all publications of the HEAL research group: <http://research.fh-ooe.at/de/orgunit/356#showpublications>

# Questions & Answers



<http://dev.heuristiclab.com>

[heuristiclab@googlegroups.com](mailto:heuristiclab@googlegroups.com)

<http://www.youtube.com/heuristiclab>

<http://www.facebook.com/heuristiclab>