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

fh OBERÖSTERREICH

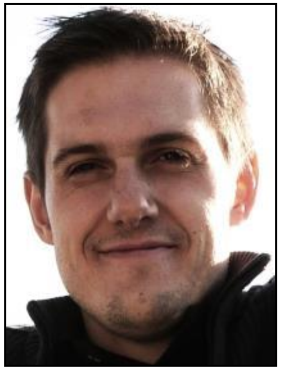
Heuristic Optimization in Production and Logistics

Instructor Biographies

HeuristicLab

- 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

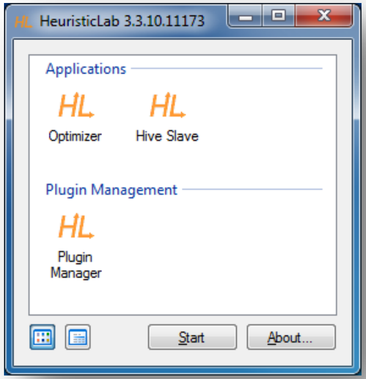
- Facts
  - development of HeuristicLab started in 2002
  - based on Microsoft .NET and C#
  - used in research and education
  - second place at the *Microsoft Innovation Award 2009*
  - open source (GNU General Public License)
  - version 3.3.0 released on May 18th, 2010
  - latest version 3.3.10 "Vancouver" released on July 10th, 2014

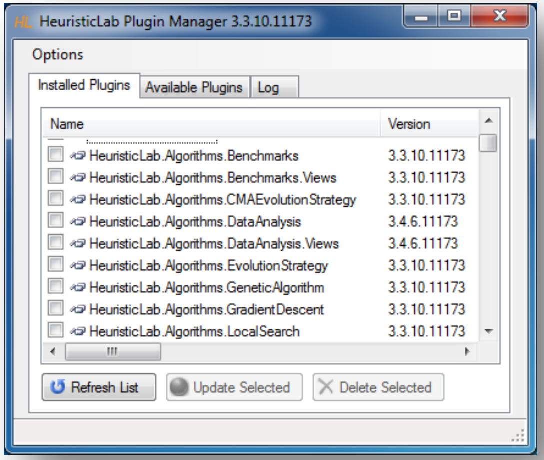
Where to get HeuristicLab?

- Download binaries
  - deployed as ZIP archives
  - latest stable version 3.3.10 "Vancouver"
    - released on July 10th, 2014
  - daily trunk builds
  - http://dev.heuristiclab.com/download
- Check out sources
  - SVN repository
  - HeuristicLab 3.3.10 tag
    - http://svn.heuristiclab.com/svn/core/tags/3.3.10
  - Stable development version
    - http://svn.heuristiclab.com/svn/core/stable
- License
  - GNU General Public License (Version 3)
- System requirements
  - Microsoft .NET Framework 4.0 Full Version
  - enough RAM and CPU power ;-)

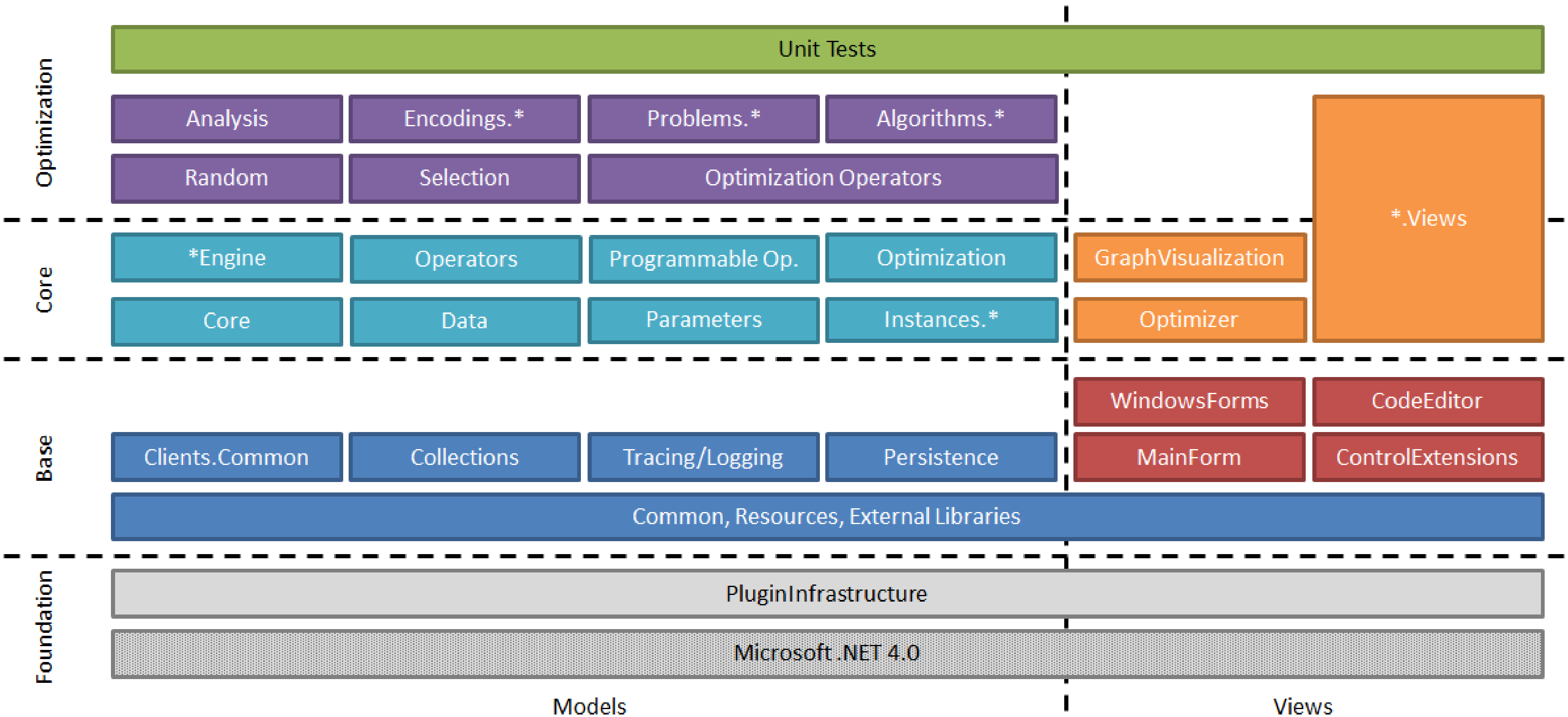
Plugin Infrastructure

- HeuristicLab consists of many assemblies
  - 142 plugins in HeuristicLab 3.3.10
  - 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

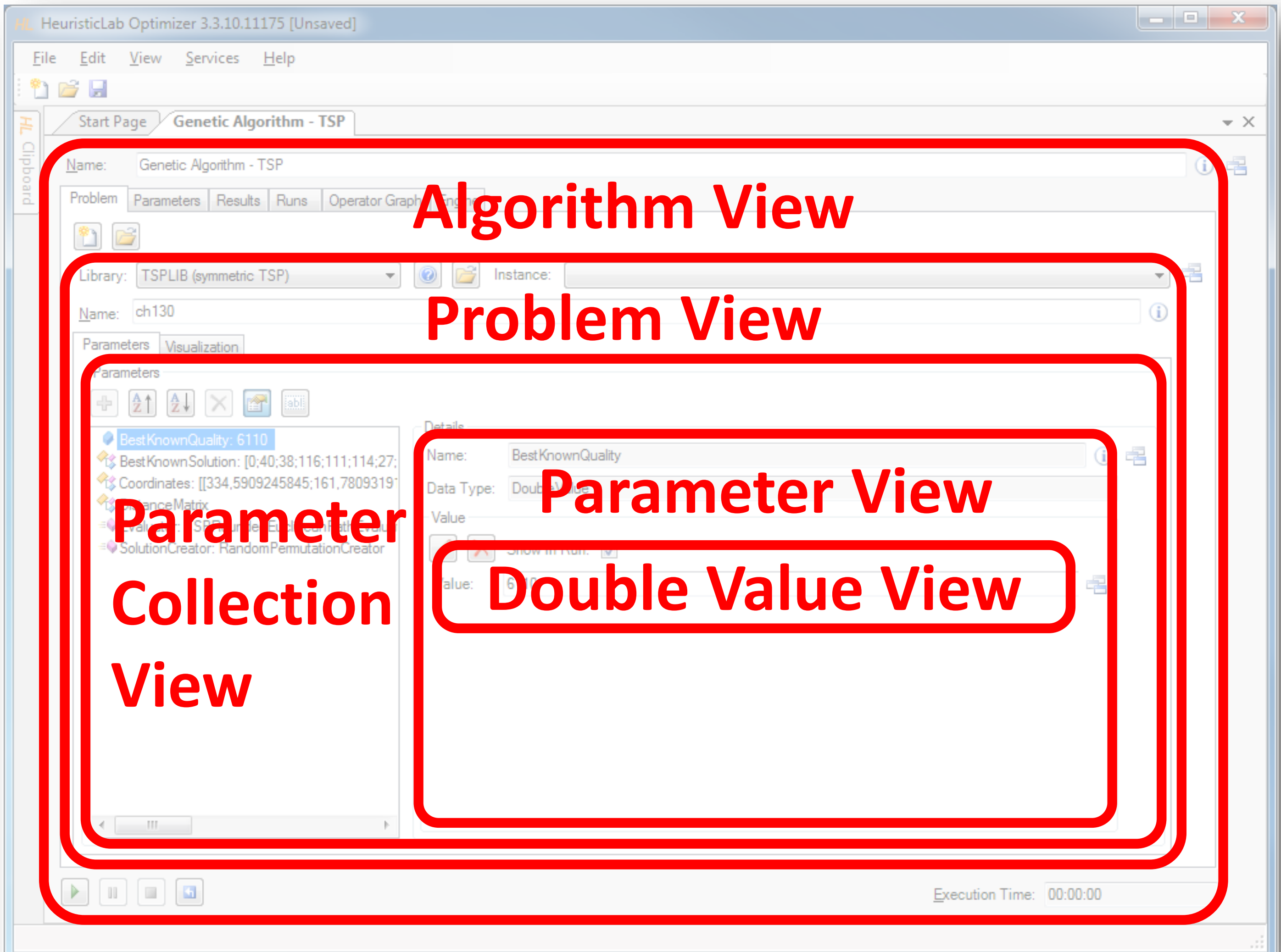


Graphical User Interface

- 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

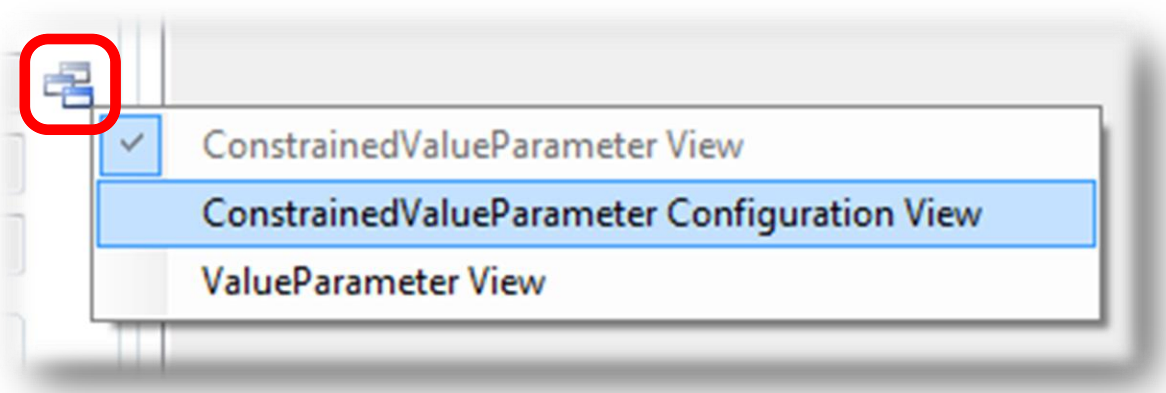
- Drag & Drop
  - views support drag & drop operations
  - content objects can be copied or moved (shift key)
  - enabled for collection items and content objects

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
- 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
- Job Shop Scheduling
- Linear Assignment
- Quadratic Assignment
- OneMax

Genetic Programming Problems

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

Additional Problems

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

Agenda

HeuristicLab

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

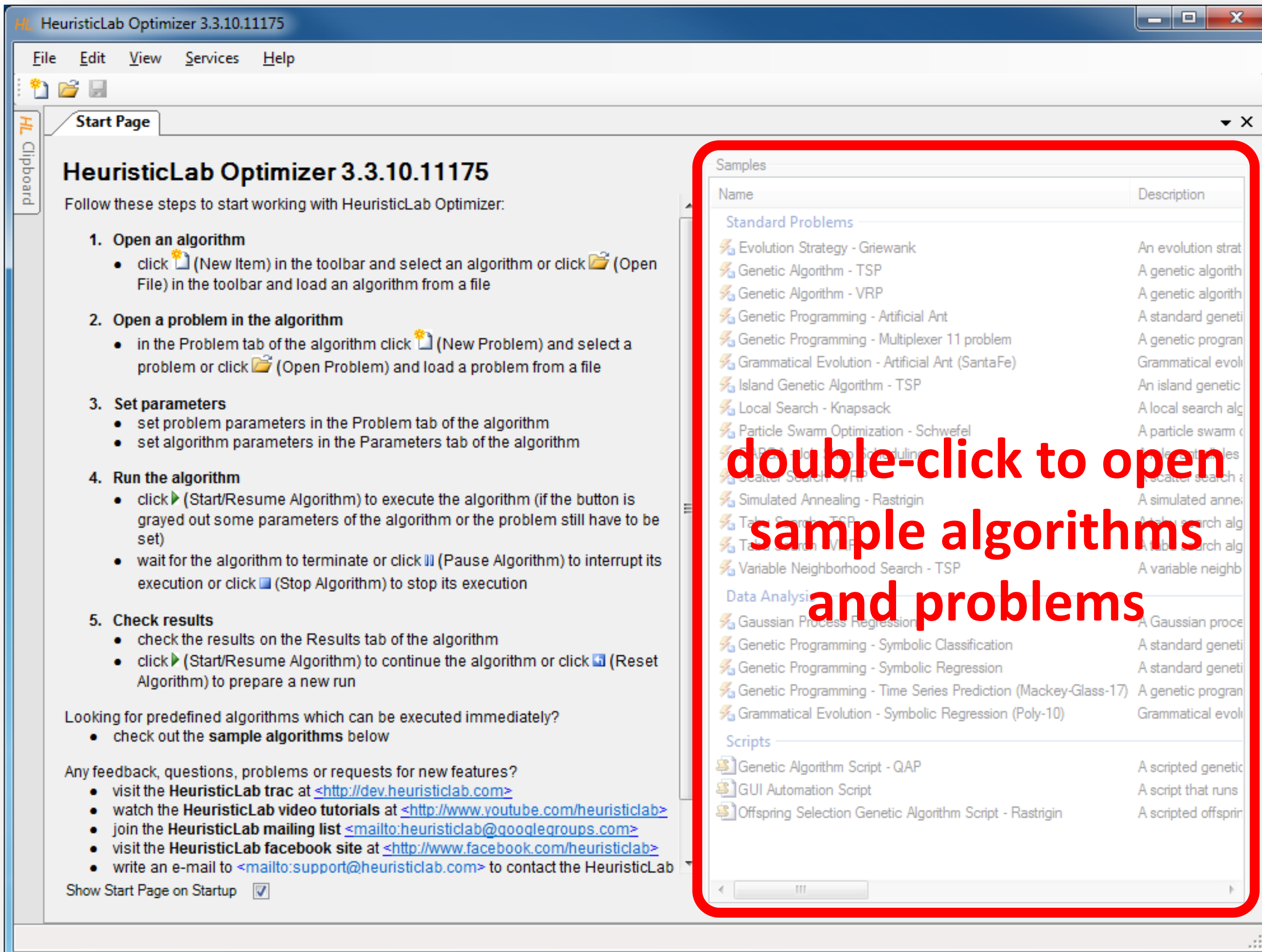
- **Demonstration Part I: Working with HeuristicLab**
- **Demonstration Part II: Data-based Modeling**

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

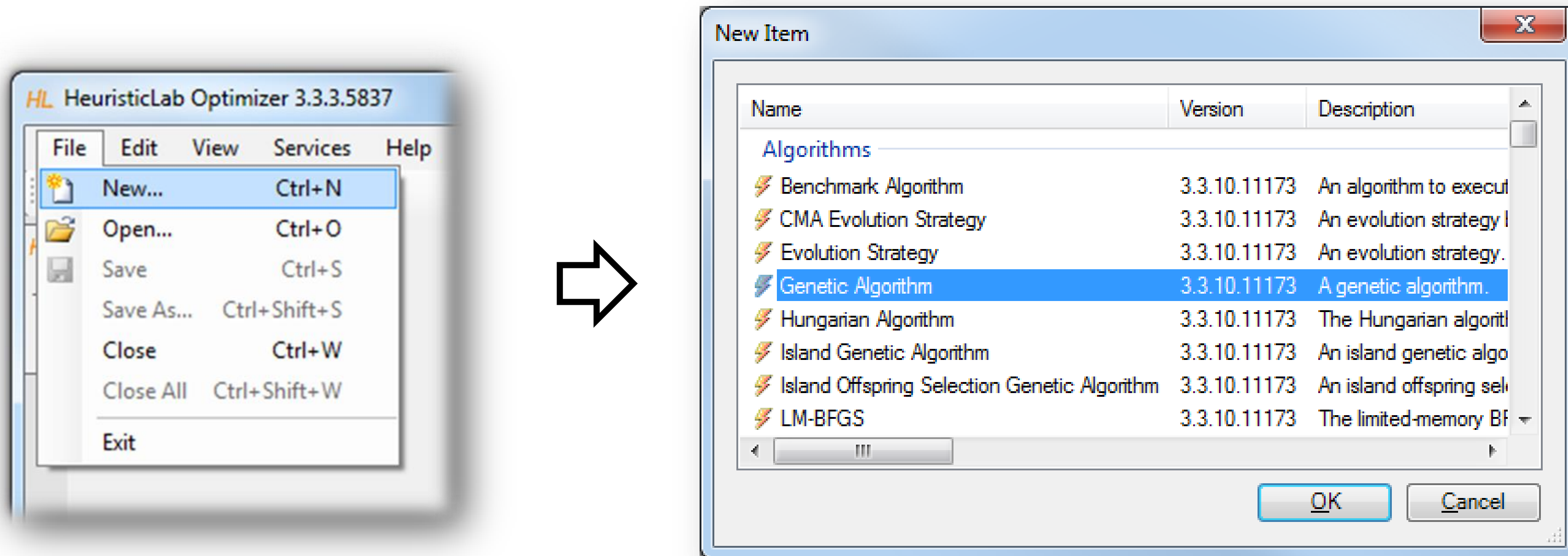
Demonstration Part I: Working with HeuristicLab

- 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

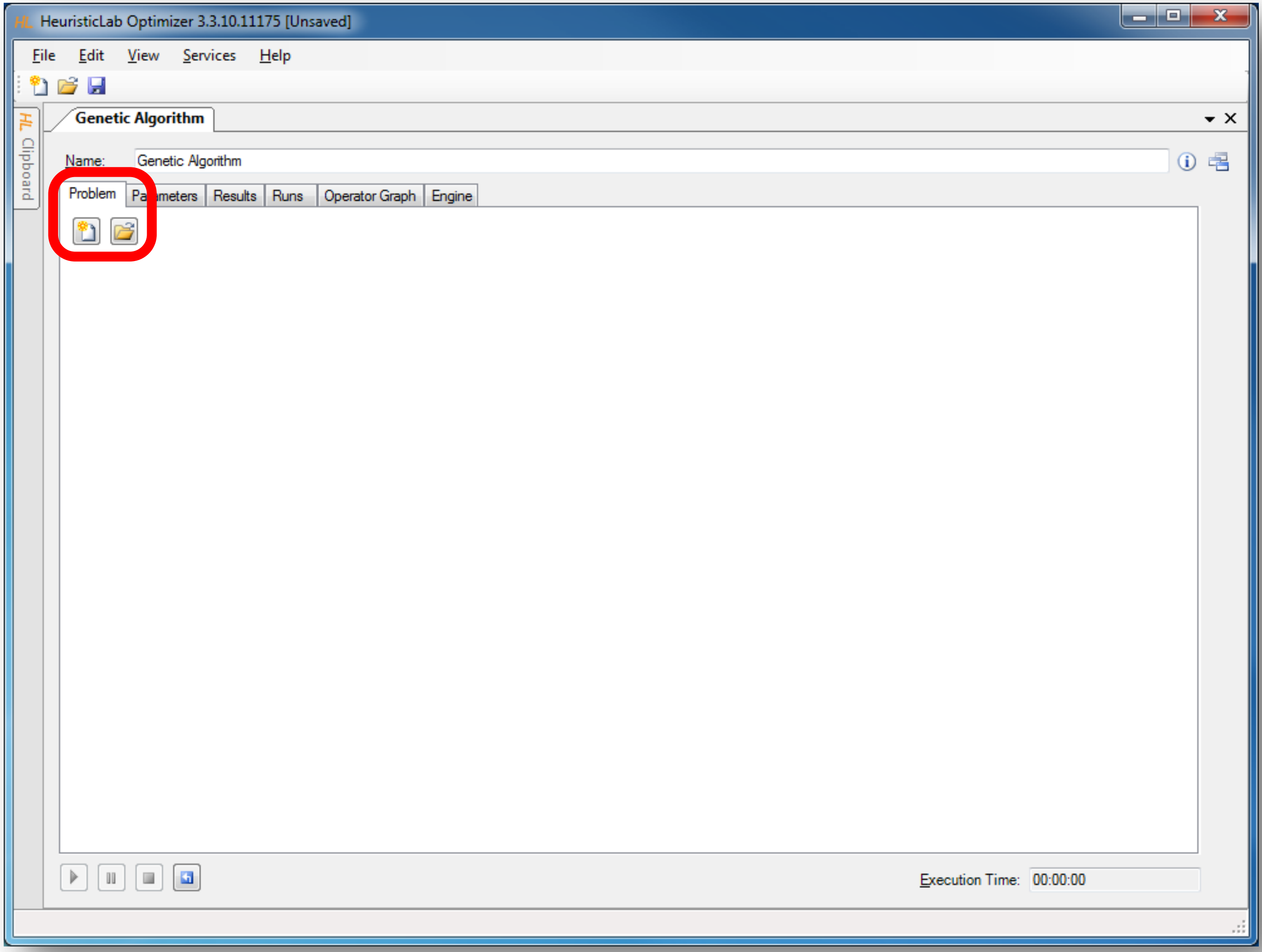
HeuristicLab Optimizer



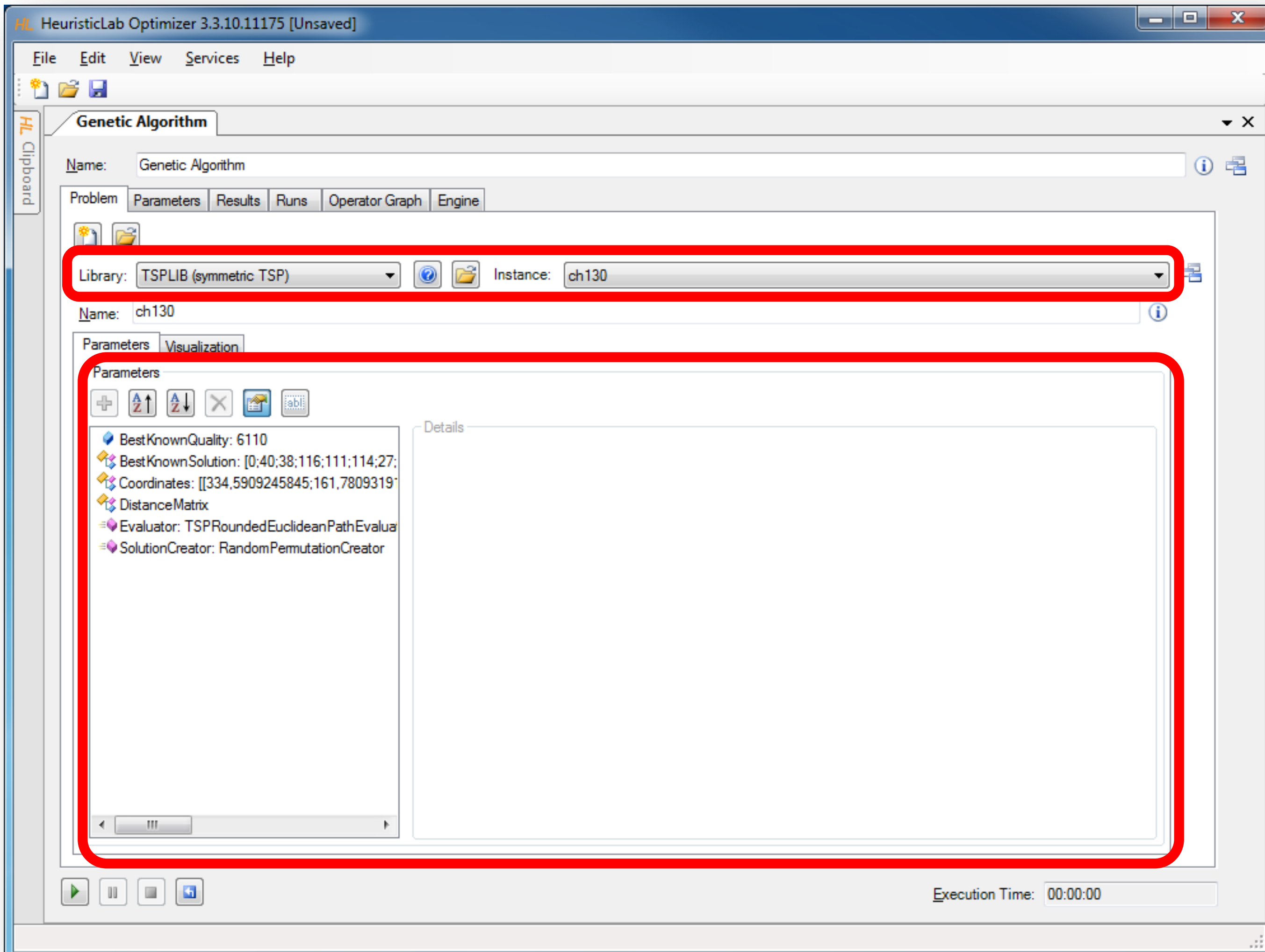
Create Algorithm



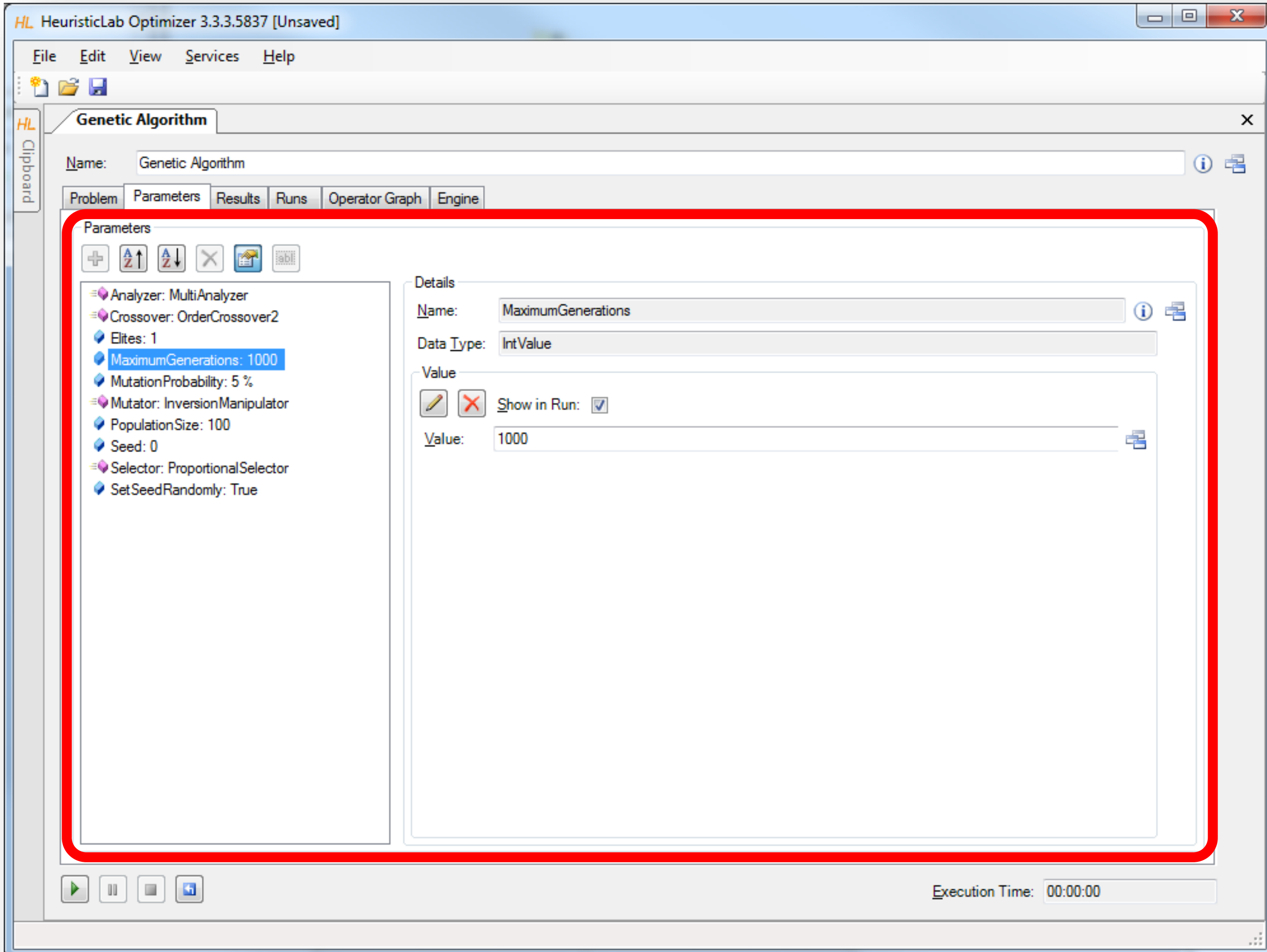
Create or Load Problem



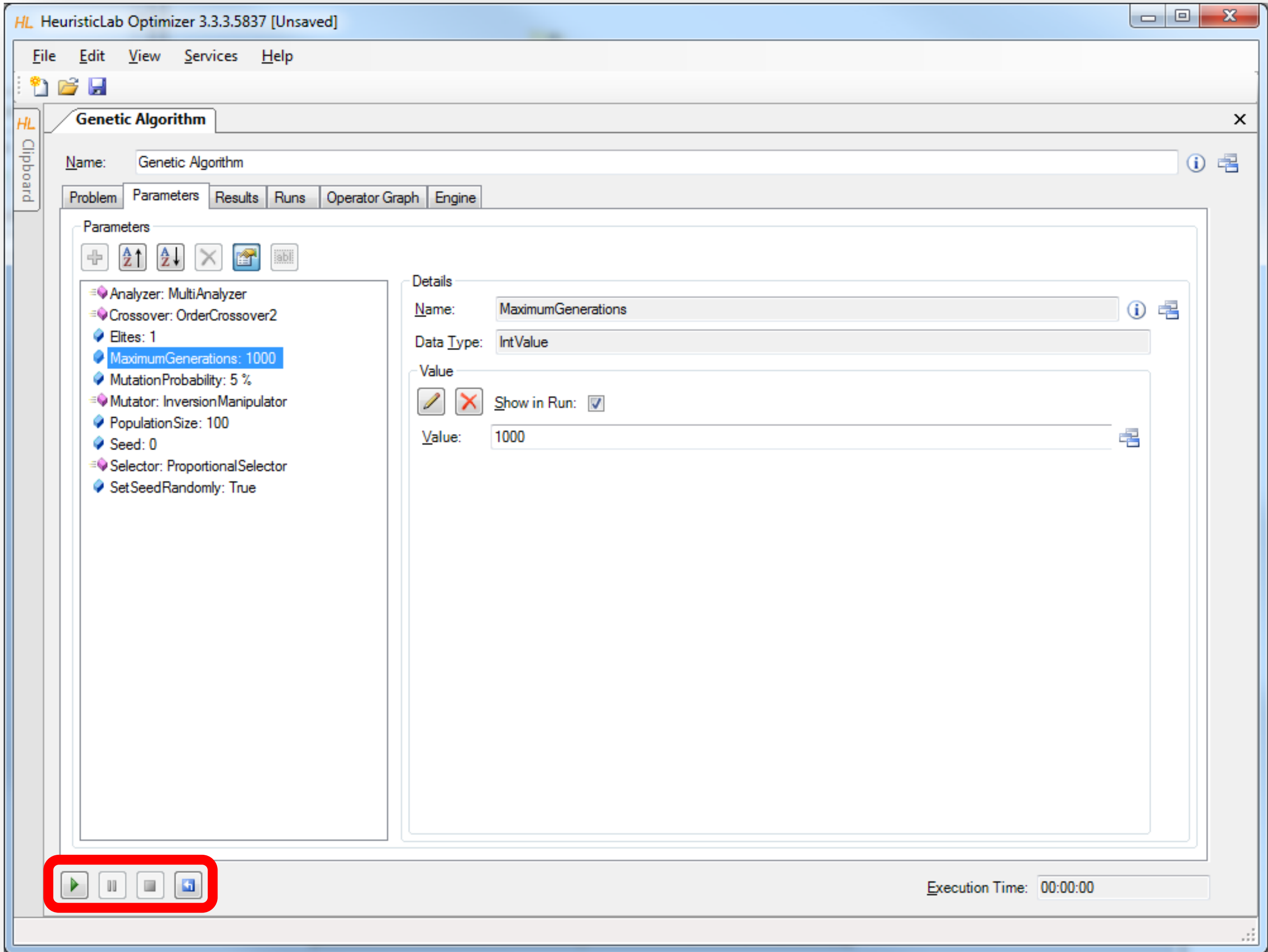
Import or Parameterize Problem Data



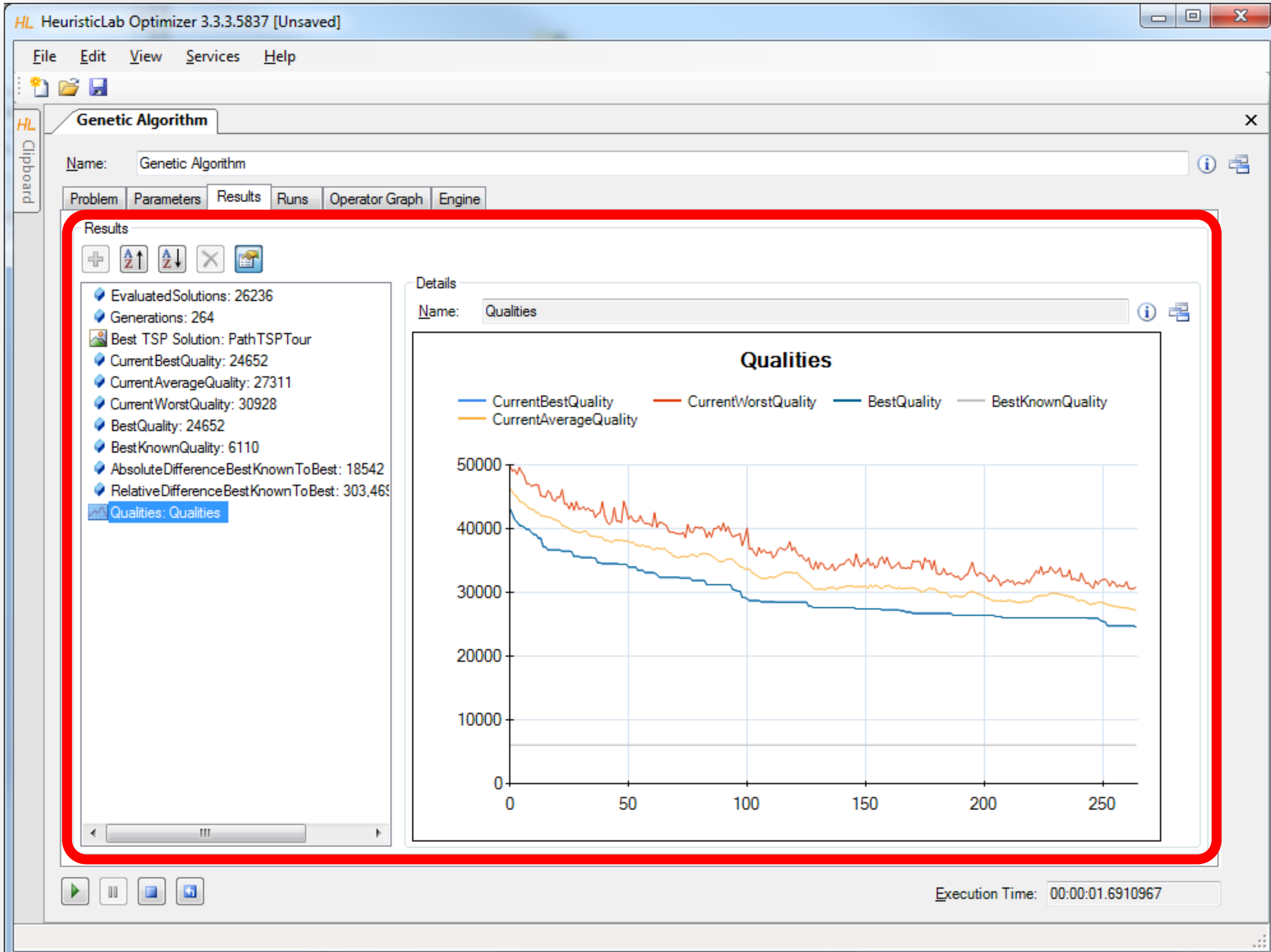
Parameterize Algorithm



Start, Pause, Resume, Stop and Reset

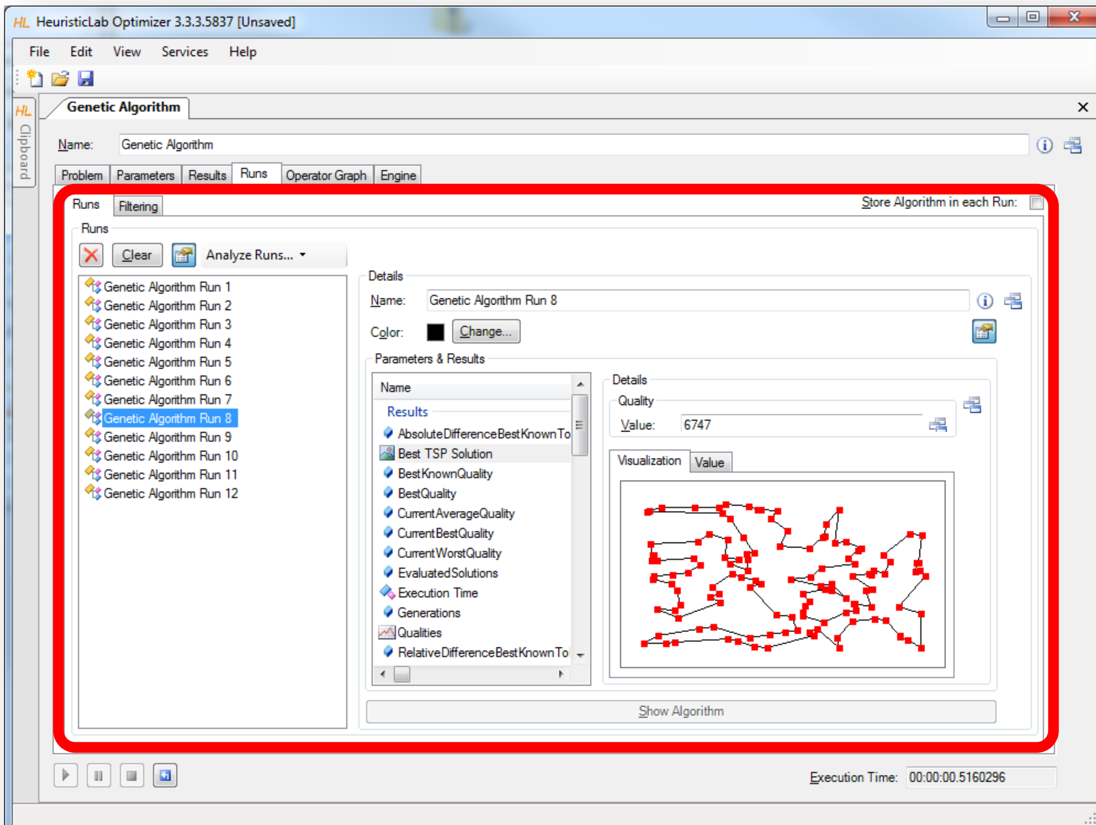


Inspect Results



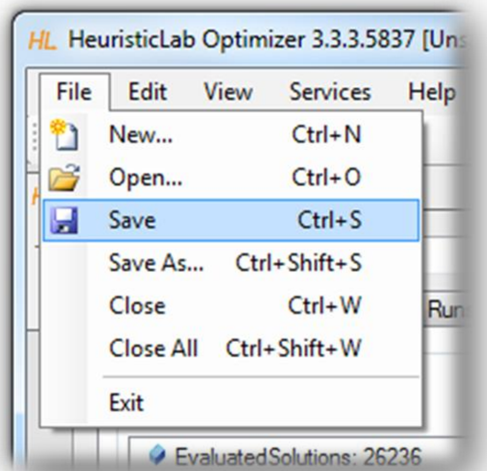
Compare Runs

- 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



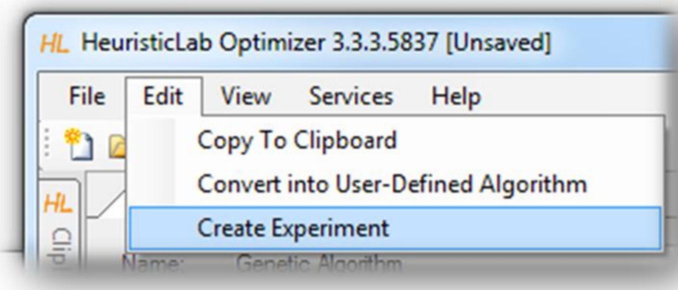
Save and Load

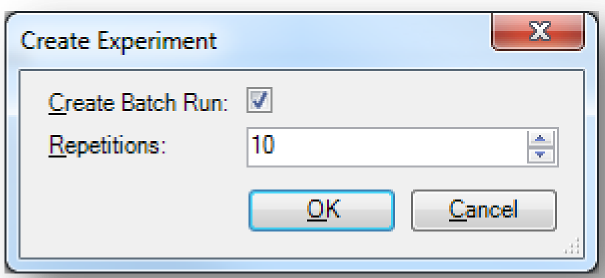
- Save to and load from disk
  - HeuristicLab items (i.e., algorithms, problems, experiments, …) can be saved to and loaded from a file
  - algorithms can be paused, saved, loaded and resumed
  - data format is custom compressed XML
  - saving and loading files might take several minutes
  - saving and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



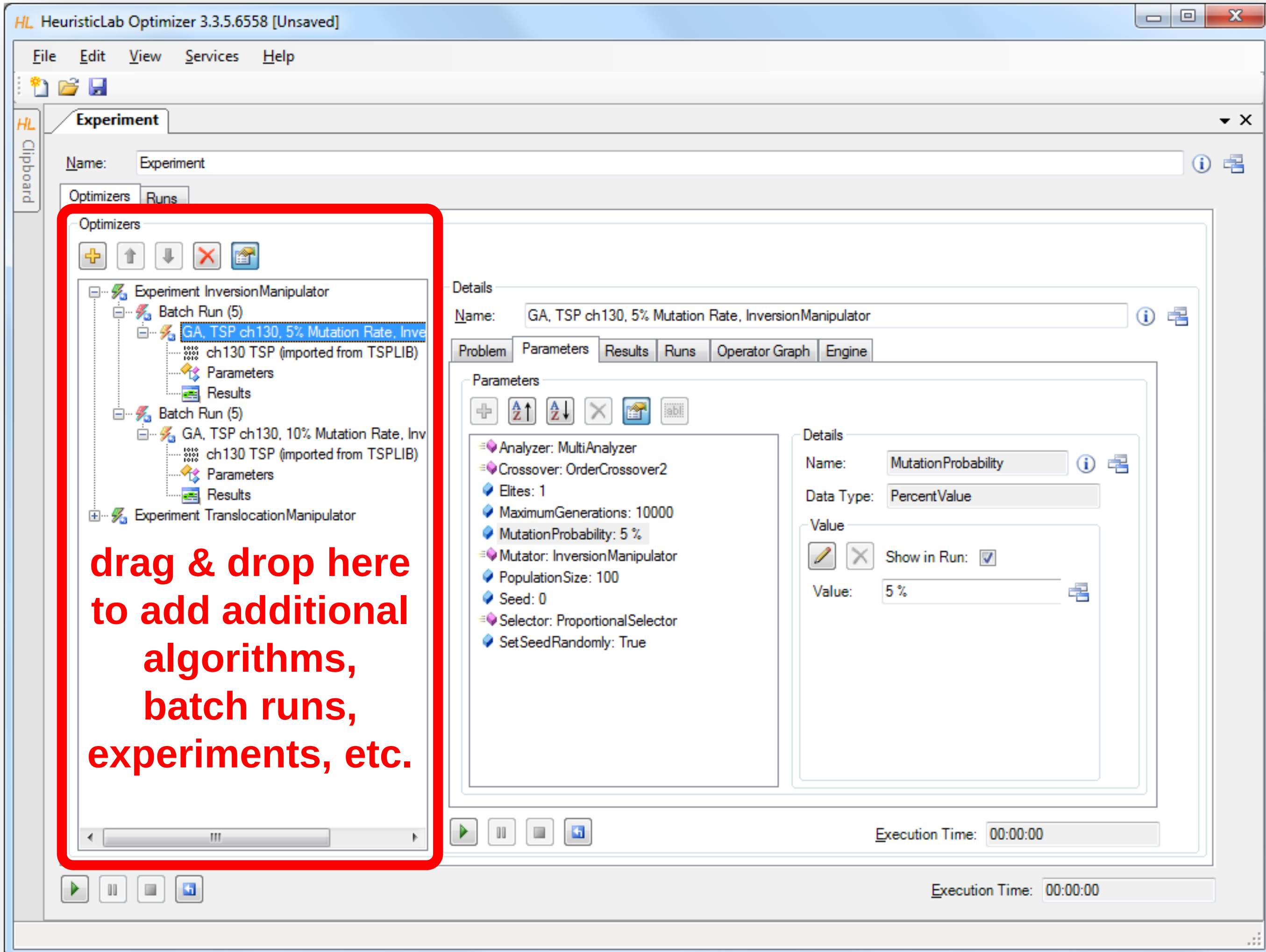
Create Batch Runs and Experiments

- 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

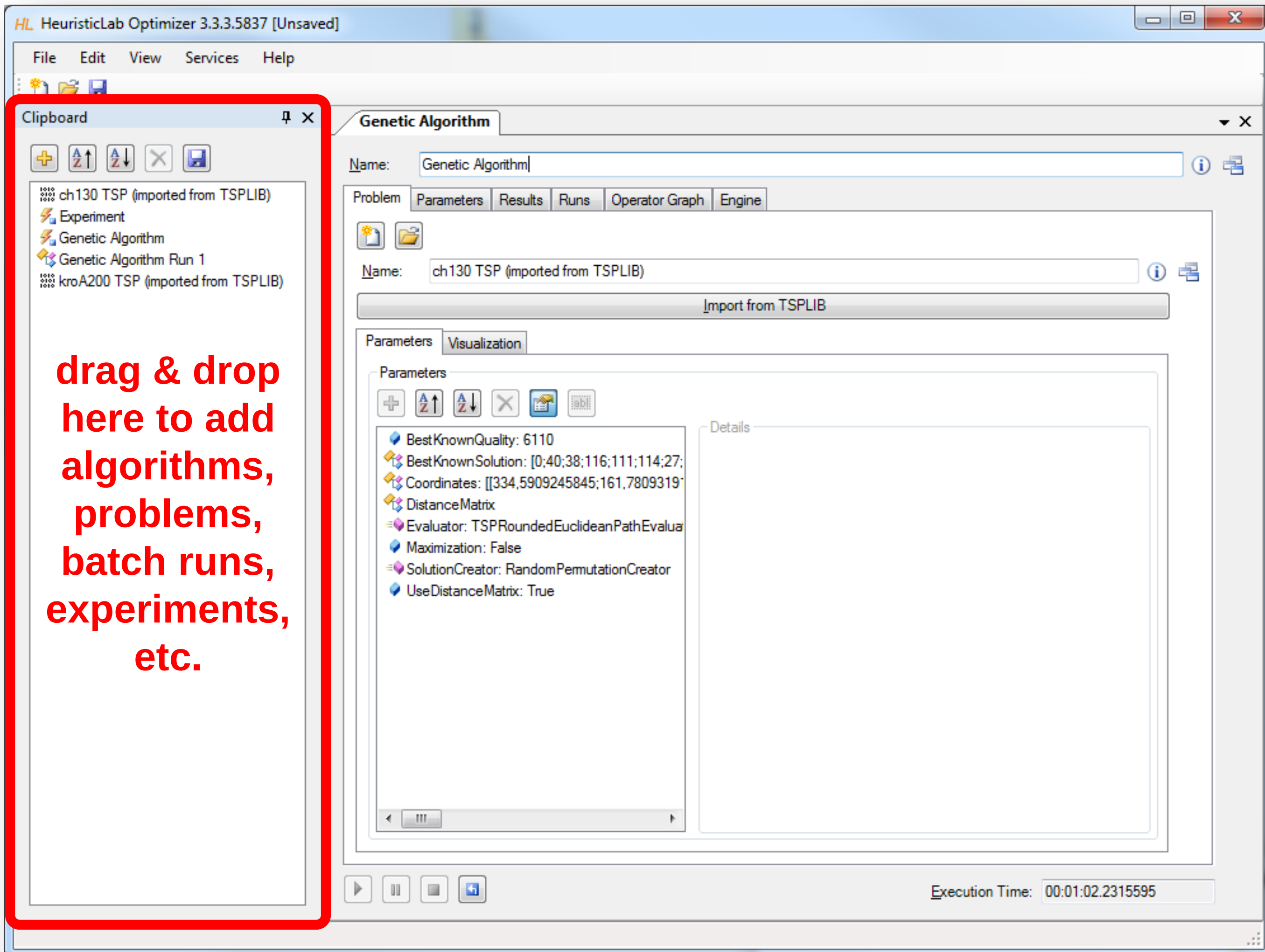




Create Batch Runs and Experiments

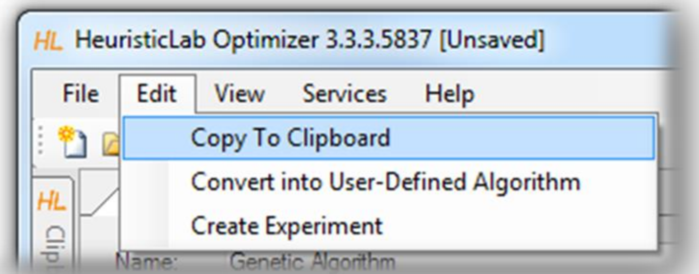


Clipboard


HeuristicLab Optimizer 3.3.3.5837 [Unsaved]
File Edit View Services Help
Clipboard
ch130 TSP (imported from TSPLIB)
Experiment
Genetic Algorithm
Genetic Algorithm Run 1
kroA200 TSP (imported from TSPLIB)
drag & drop here to add algorithms, problems, batch runs, experiments, etc.
Genetic Algorithm
Name: Genetic Algorithm
Problem Parameters Results Runs Operator Graph Engine
Name: ch130 TSP (imported from TSPLIB)
Import from TSPLIB
Parameters Visualization
Parameters
Details
BestKnownQuality: 6110
BestKnownSolution: [0;40;38;116;111;114;27;
Coordinates: [[334,5909245845;161,7809319
DistanceMatrix
Evaluator: TSPRoundedEuclideanPathEvalua
Maximization: False
SolutionCreator: RandomPermutationCreator
UseDistanceMatrix: True
Execution Time: 00:01:02.2315595

Clipboard

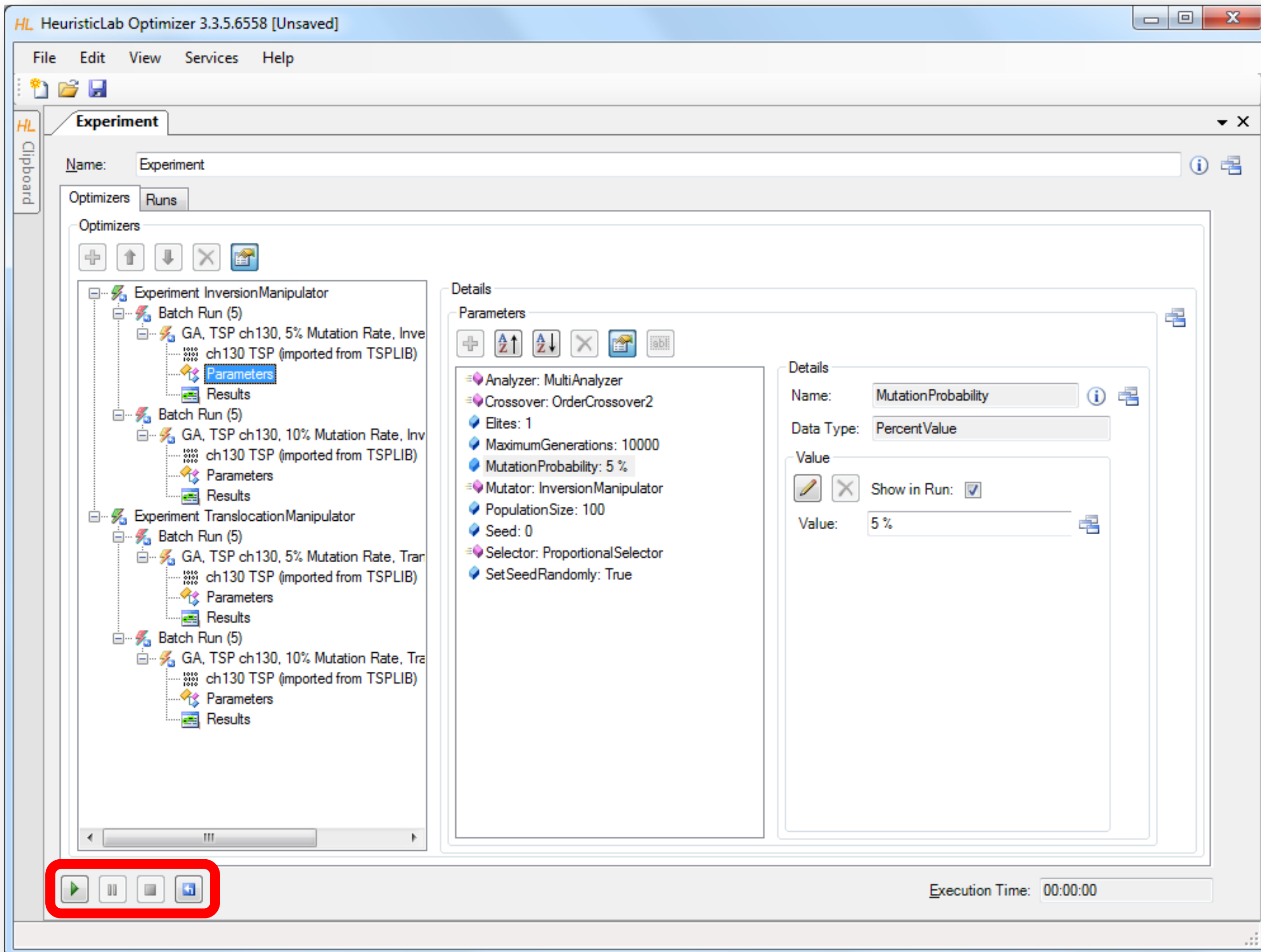
- Store items
  - click on the buttons to add or remove items
  - drag & drop items on the clipboard
  - use the menu to add a copy of a shown item to the clipboard



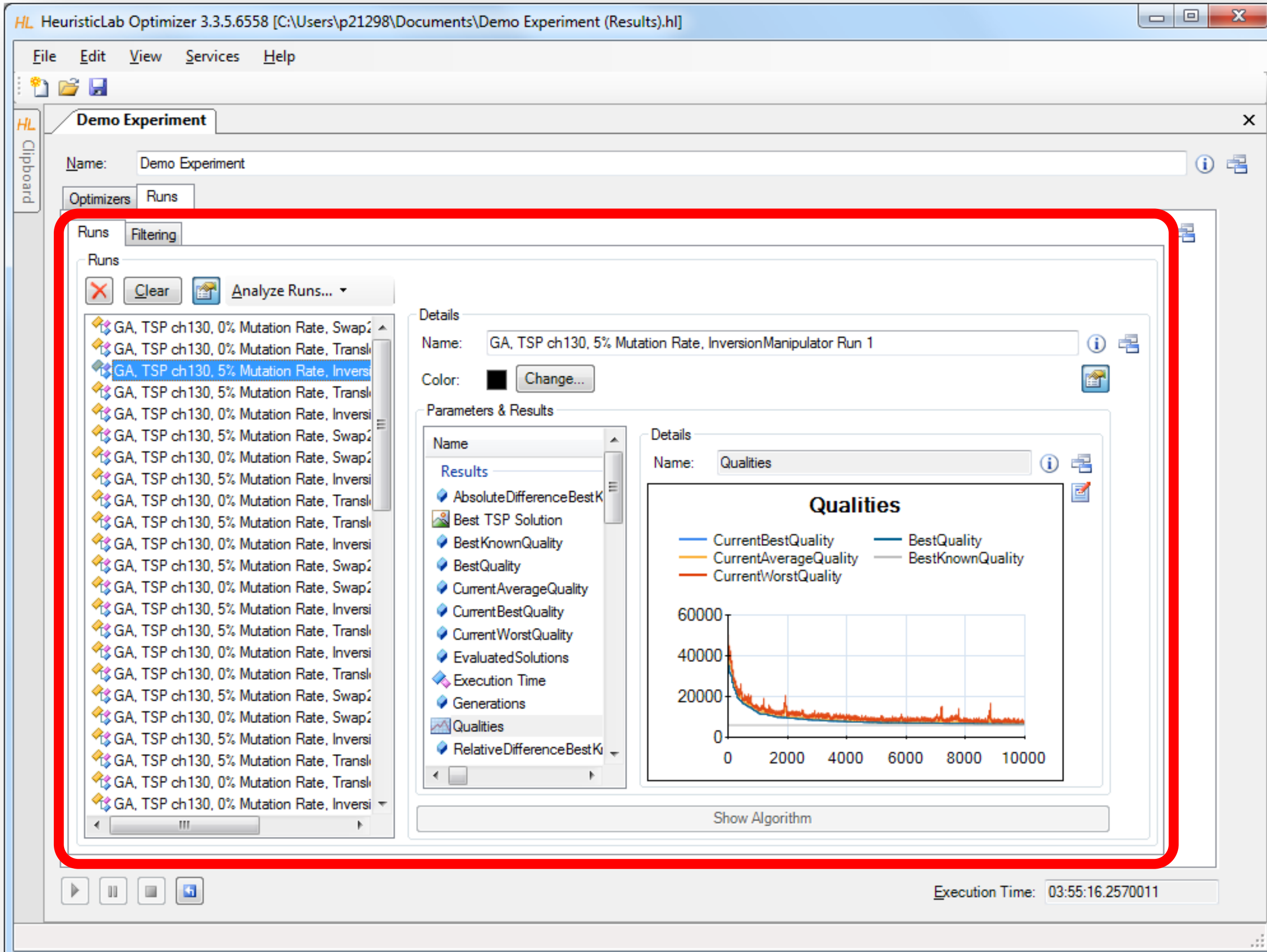
- Show items
  - double-click on an item in the clipboard to show its view

- Save and restore clipboard content
  - click on the save button to write the clipboard content to disk
  - clipboard is automatically restored when HeuristicLab is started the next time

Start, Pause, Resume, Stop, Reset



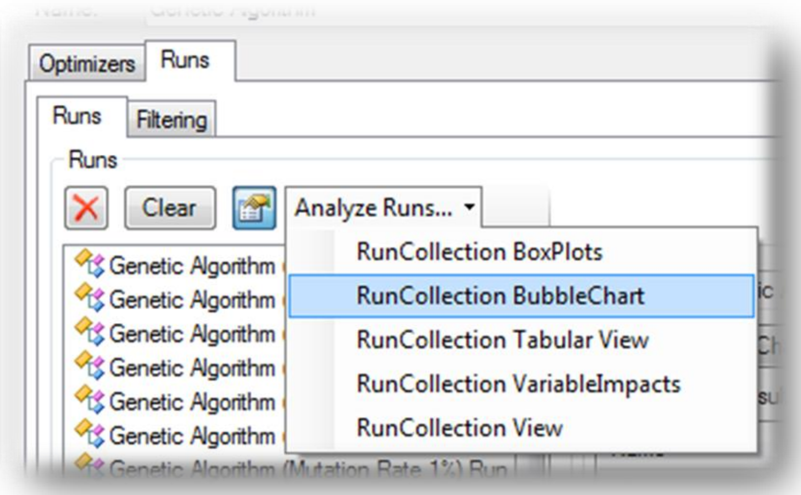
Compare Runs



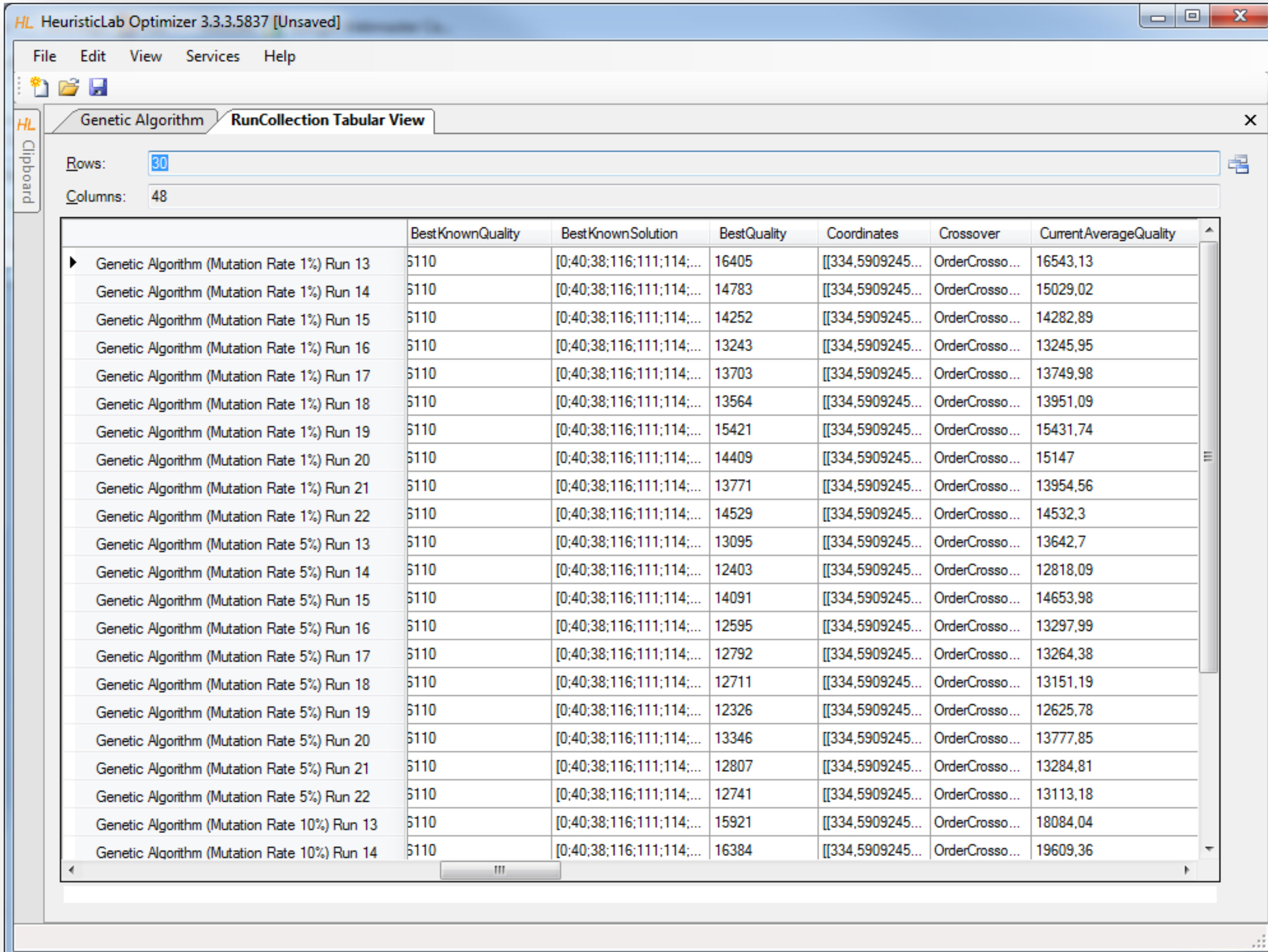
Analyze Runs

- HeuristicLab provides interactive views to analyze and compare all runs of a run collection
  - textual analysis
    - RunCollection Tabular View
  - graphical analysis
    - RunCollection BubbleChart
    - RunCollection BoxPlots

- Filtering is automatically applied to all open run collection views



Runs – Tabular View

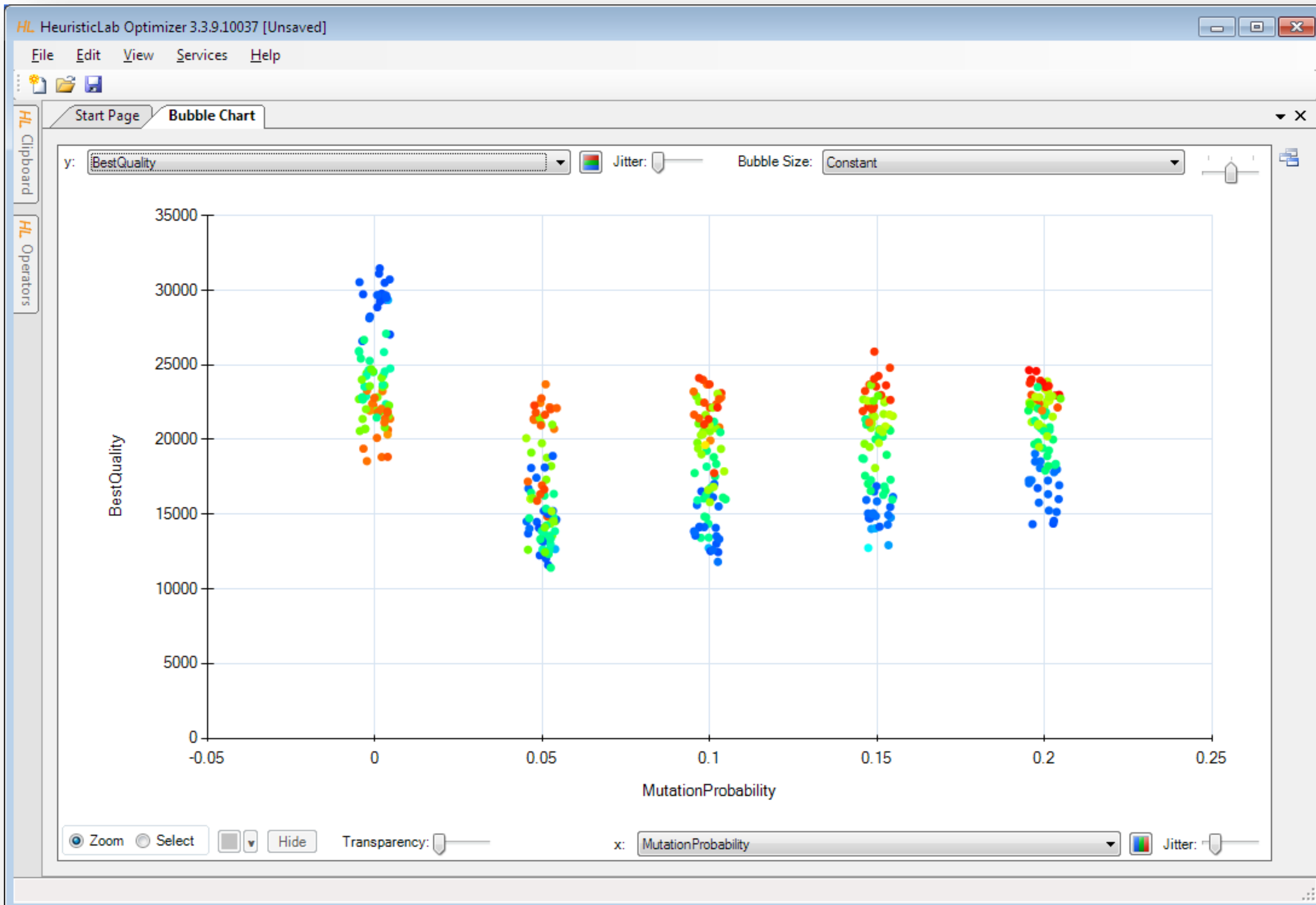


	BestKnownQuality	BestKnownSolution	BestQuality	Coordinates	Crossover	CurrentAverageQuality
Genetic Algorithm (Mutation Rate 1%) Run 13	6110	[0;40;38;116;111;114;...	16405	[[334,5909245...	OrderCrosso...	16543,13
Genetic Algorithm (Mutation Rate 1%) Run 14	6110	[0;40;38;116;111;114;...	14783	[[334,5909245...	OrderCrosso...	15029,02
Genetic Algorithm (Mutation Rate 1%) Run 15	6110	[0;40;38;116;111;114;...	14252	[[334,5909245...	OrderCrosso...	14282,89
Genetic Algorithm (Mutation Rate 1%) Run 16	6110	[0;40;38;116;111;114;...	13243	[[334,5909245...	OrderCrosso...	13245,95
Genetic Algorithm (Mutation Rate 1%) Run 17	6110	[0;40;38;116;111;114;...	13703	[[334,5909245...	OrderCrosso...	13749,98
Genetic Algorithm (Mutation Rate 1%) Run 18	6110	[0;40;38;116;111;114;...	13564	[[334,5909245...	OrderCrosso...	13951,09
Genetic Algorithm (Mutation Rate 1%) Run 19	6110	[0;40;38;116;111;114;...	15421	[[334,5909245...	OrderCrosso...	15431,74
Genetic Algorithm (Mutation Rate 1%) Run 20	6110	[0;40;38;116;111;114;...	14409	[[334,5909245...	OrderCrosso...	15147
Genetic Algorithm (Mutation Rate 1%) Run 21	6110	[0;40;38;116;111;114;...	13771	[[334,5909245...	OrderCrosso...	13954,56
Genetic Algorithm (Mutation Rate 1%) Run 22	6110	[0;40;38;116;111;114;...	14529	[[334,5909245...	OrderCrosso...	14532,3
Genetic Algorithm (Mutation Rate 5%) Run 13	6110	[0;40;38;116;111;114;...	13095	[[334,5909245...	OrderCrosso...	13642,7
Genetic Algorithm (Mutation Rate 5%) Run 14	6110	[0;40;38;116;111;114;...	12403	[[334,5909245...	OrderCrosso...	12818,09
Genetic Algorithm (Mutation Rate 5%) Run 15	6110	[0;40;38;116;111;114;...	14091	[[334,5909245...	OrderCrosso...	14653,98
Genetic Algorithm (Mutation Rate 5%) Run 16	6110	[0;40;38;116;111;114;...	12595	[[334,5909245...	OrderCrosso...	13297,99
Genetic Algorithm (Mutation Rate 5%) Run 17	6110	[0;40;38;116;111;114;...	12792	[[334,5909245...	OrderCrosso...	13264,38
Genetic Algorithm (Mutation Rate 5%) Run 18	6110	[0;40;38;116;111;114;...	12711	[[334,5909245...	OrderCrosso...	13151,19
Genetic Algorithm (Mutation Rate 5%) Run 19	6110	[0;40;38;116;111;114;...	12326	[[334,5909245...	OrderCrosso...	12625,78
Genetic Algorithm (Mutation Rate 5%) Run 20	6110	[0;40;38;116;111;114;...	13346	[[334,5909245...	OrderCrosso...	13777,85
Genetic Algorithm (Mutation Rate 5%) Run 21	6110	[0;40;38;116;111;114;...	12807	[[334,5909245...	OrderCrosso...	13284,81
Genetic Algorithm (Mutation Rate 5%) Run 22	6110	[0;40;38;116;111;114;...	12741	[[334,5909245...	OrderCrosso...	13113,18
Genetic Algorithm (Mutation Rate 10%) Run 13	6110	[0;40;38;116;111;114;...	15921	[[334,5909245...	OrderCrosso...	18084,04
Genetic Algorithm (Mutation Rate 10%) Run 14	6110	[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

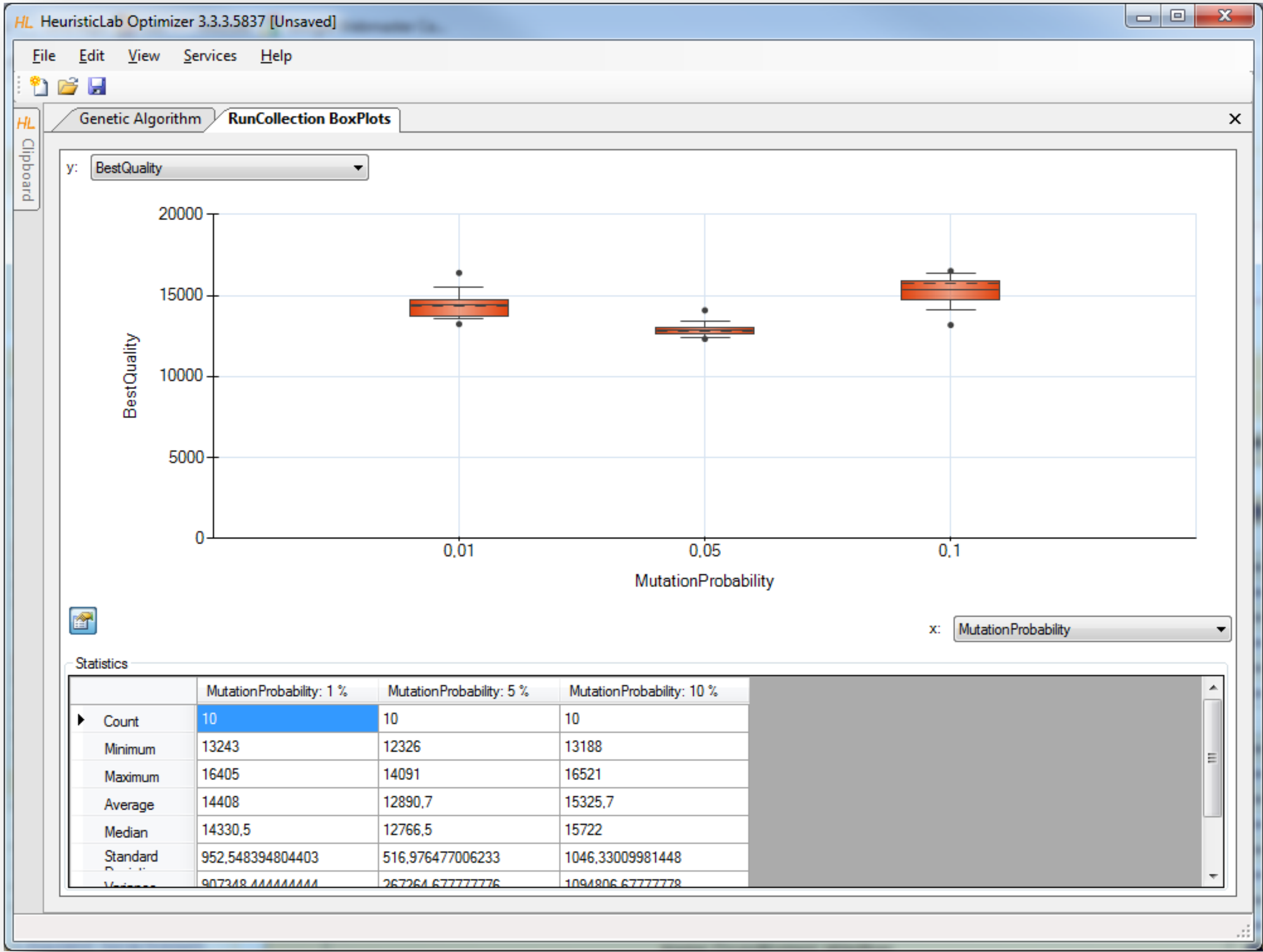
Runs – BubbleChart



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

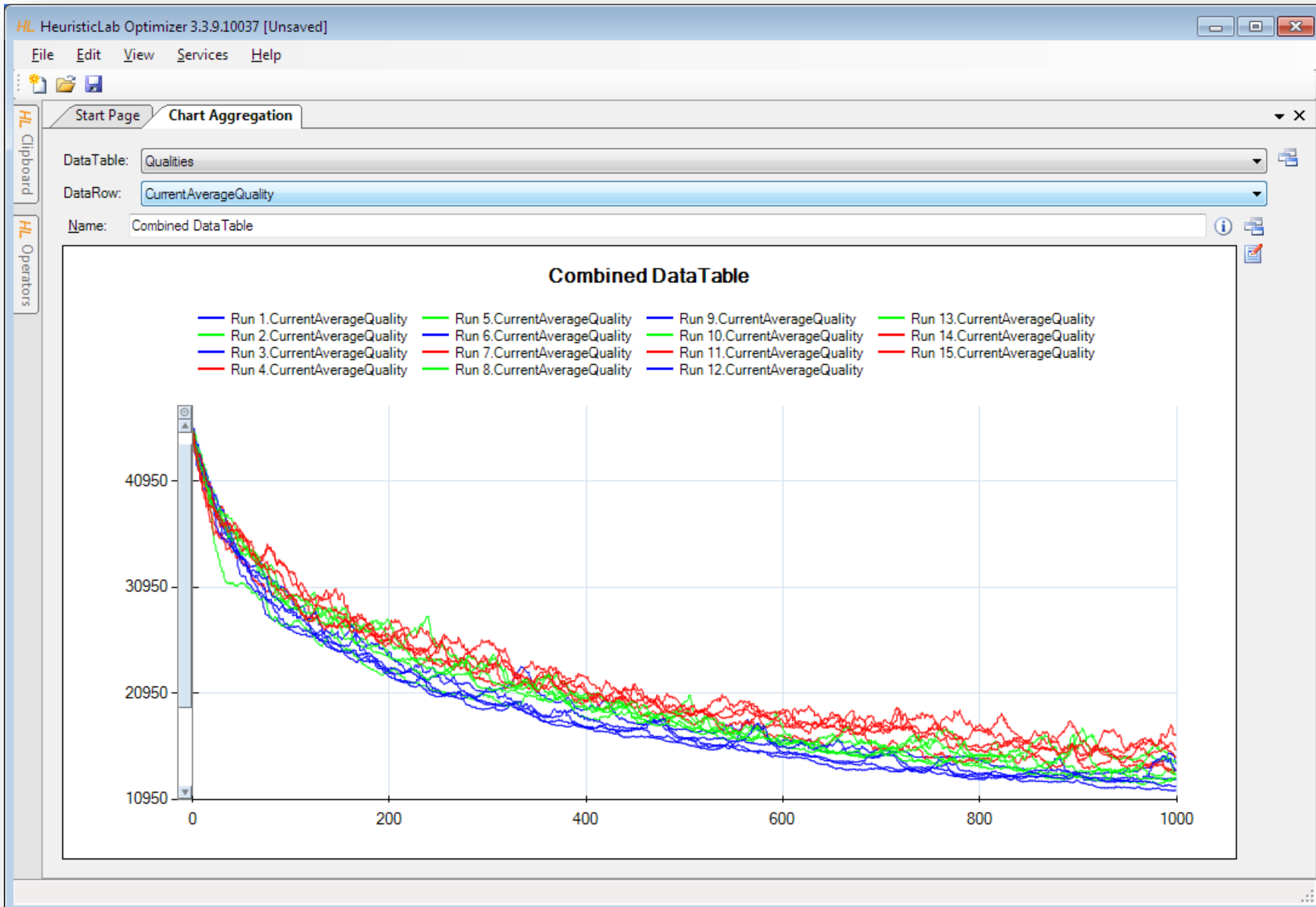
Runs – BoxPlots



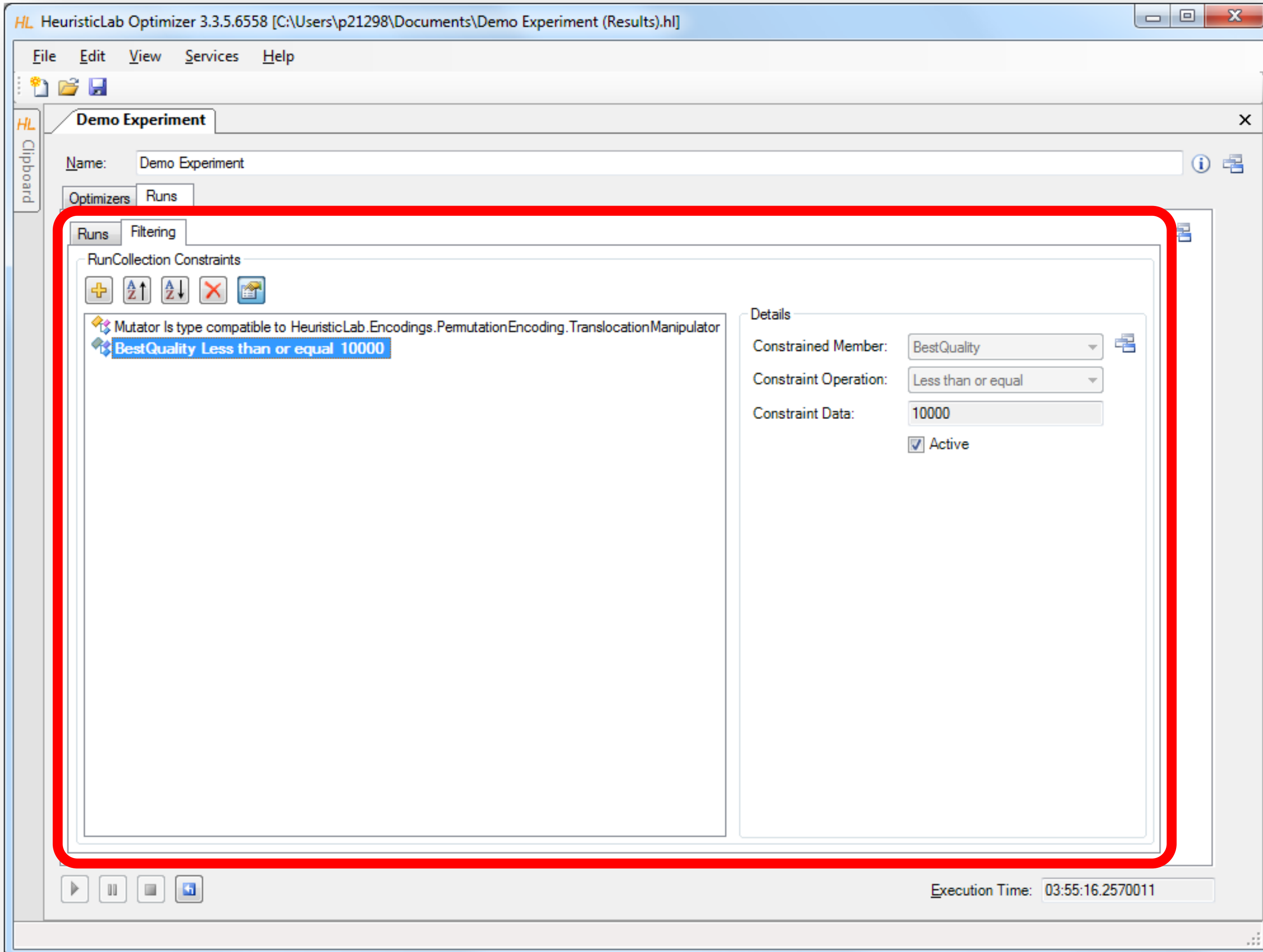
Runs – BoxPlots

- 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



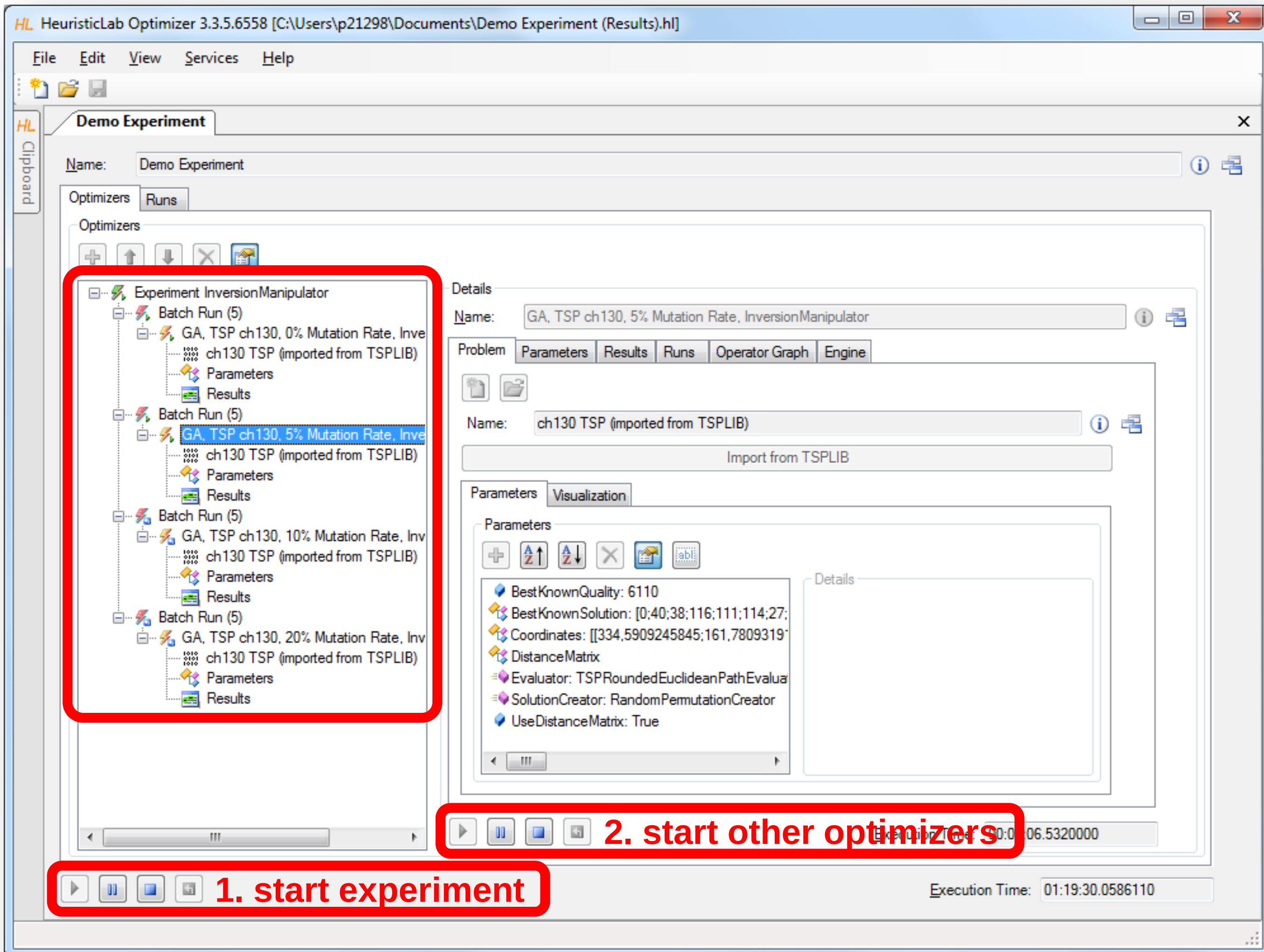
Filter Runs


HeuristicLab Optimizer 3.3.5.6558 [C:\Users\p21298\Documents\Demo Experiment (Results).hl]
File Edit View Services Help
Clipboard
Demo Experiment
Name: Demo Experiment
Optimizers Runs
Runs Filtering
RunCollection Constraints
Mutator Is type compatible to HeuristicLab.Encodings.PermutationEncoding.TranslocationManipulator
BestQuality Less than or equal 10000
Details
Constrained Member: BestQuality
Constraint Operation: Less than or equal
Constraint Data: 10000
Active
Execution Time: 03:55:16.2570011

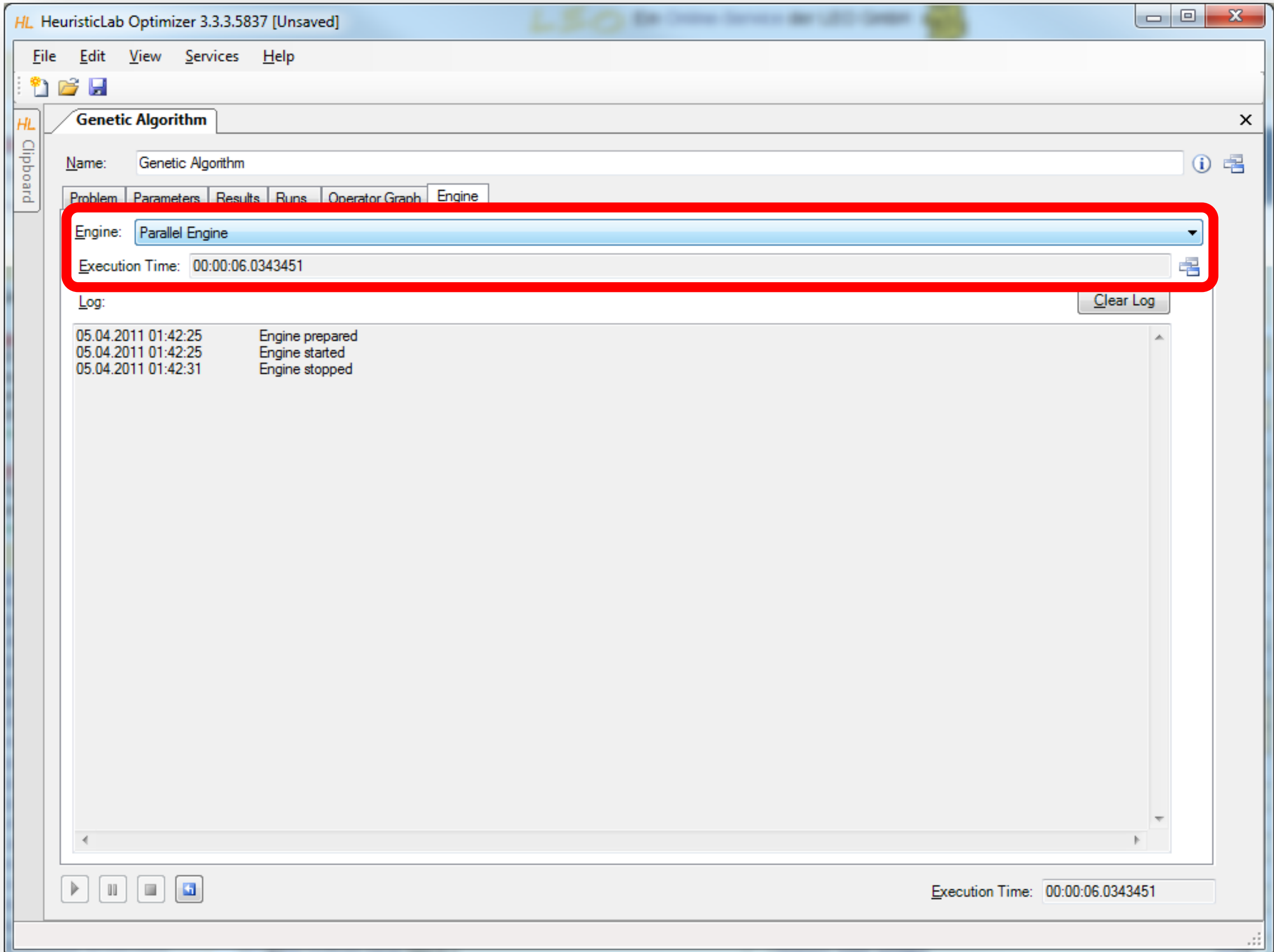
Multi-core CPUs and Parallelization

- 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

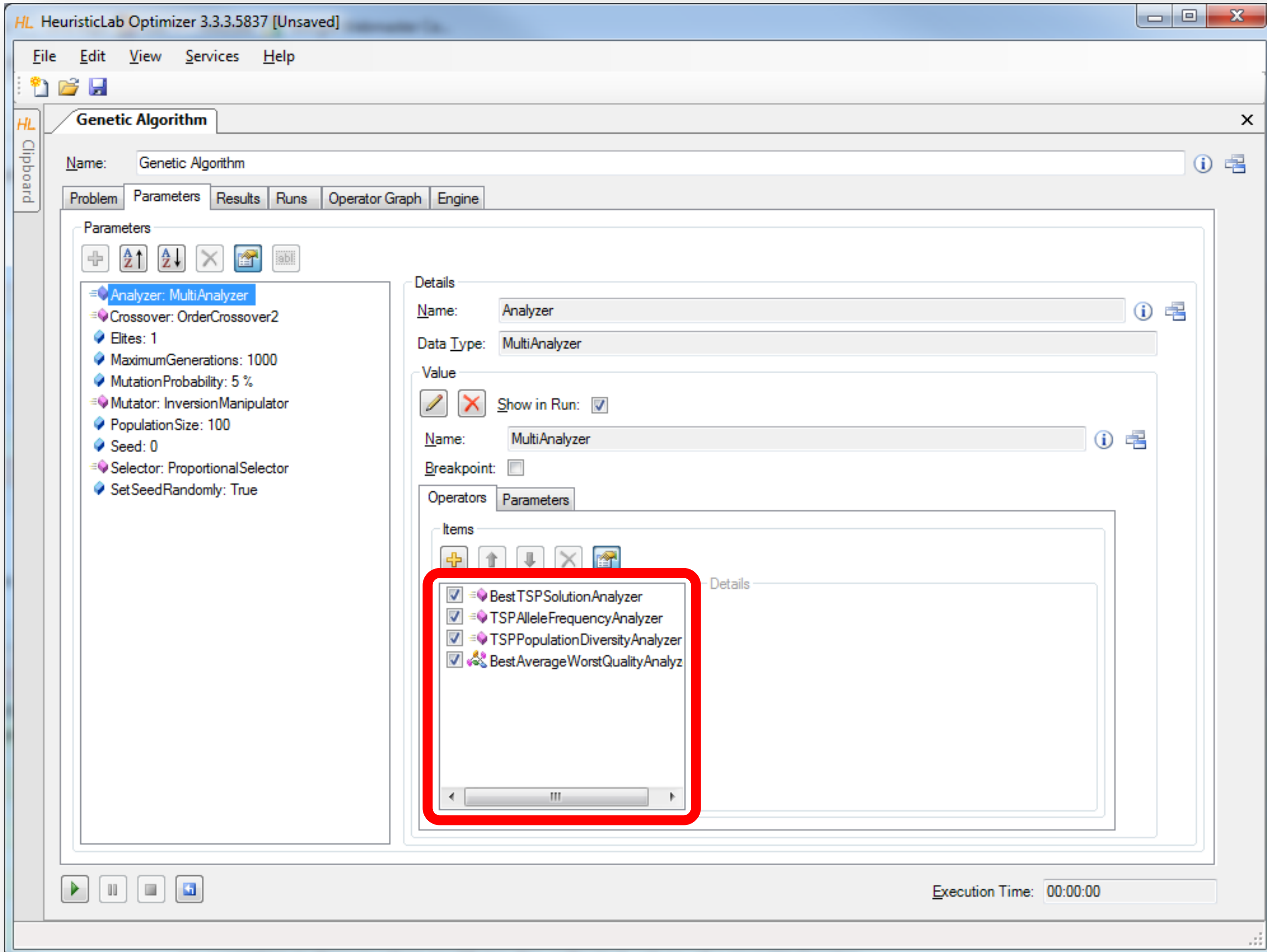


Analyzers

- 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

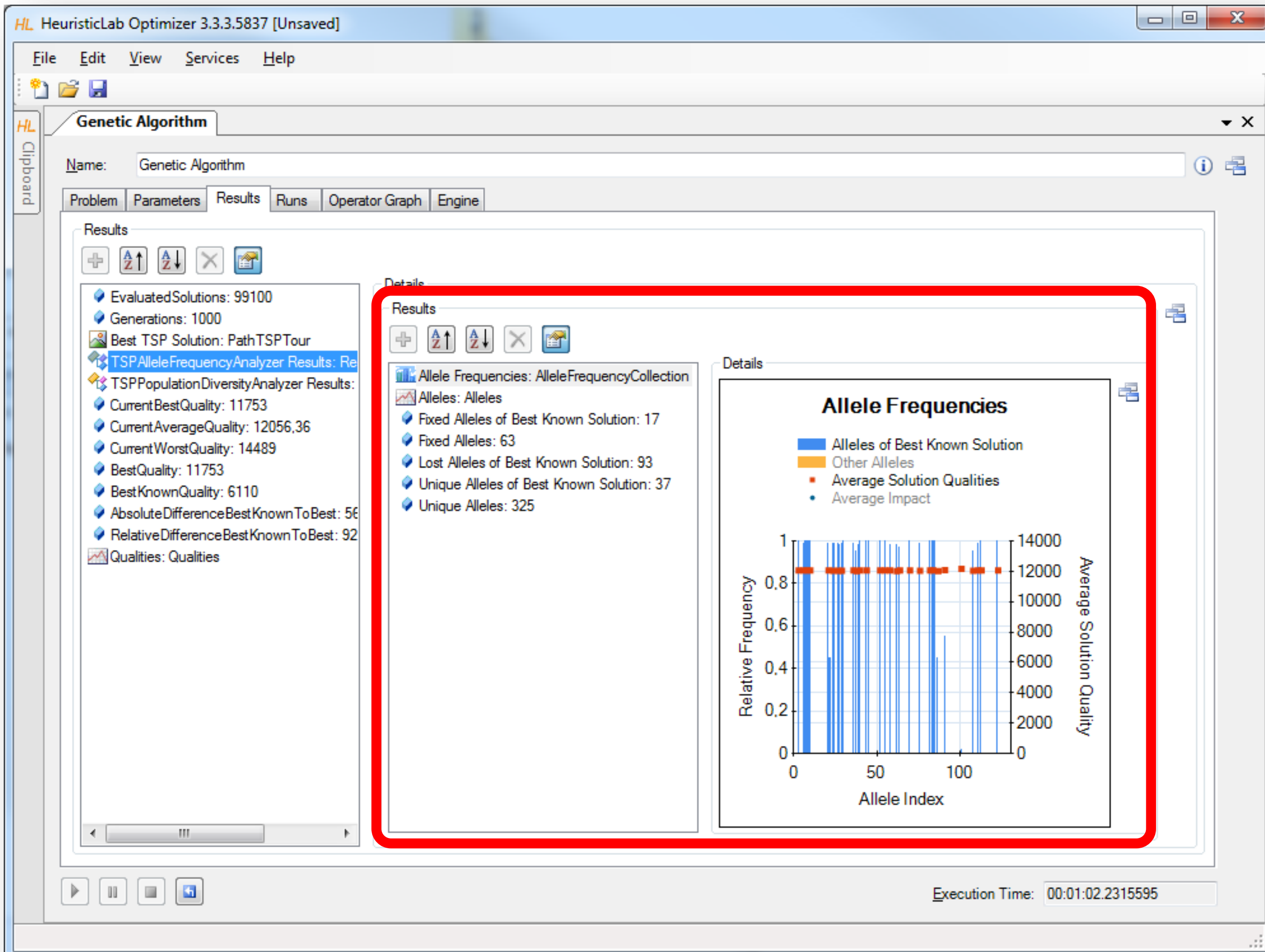
- Examples
  - TSPAlleleFrequencyAnalyzer
  - TSPPopulationDiversityAnalyzer
  - SuccessfulOffspringAnalyzer
  - SymbolicDataAnalysisVariableFrequencyAnalyzer
  - SymbolicRegressionSingleObjectiveTrainingBestSolutionAnalyzer
  - …

Analyzers

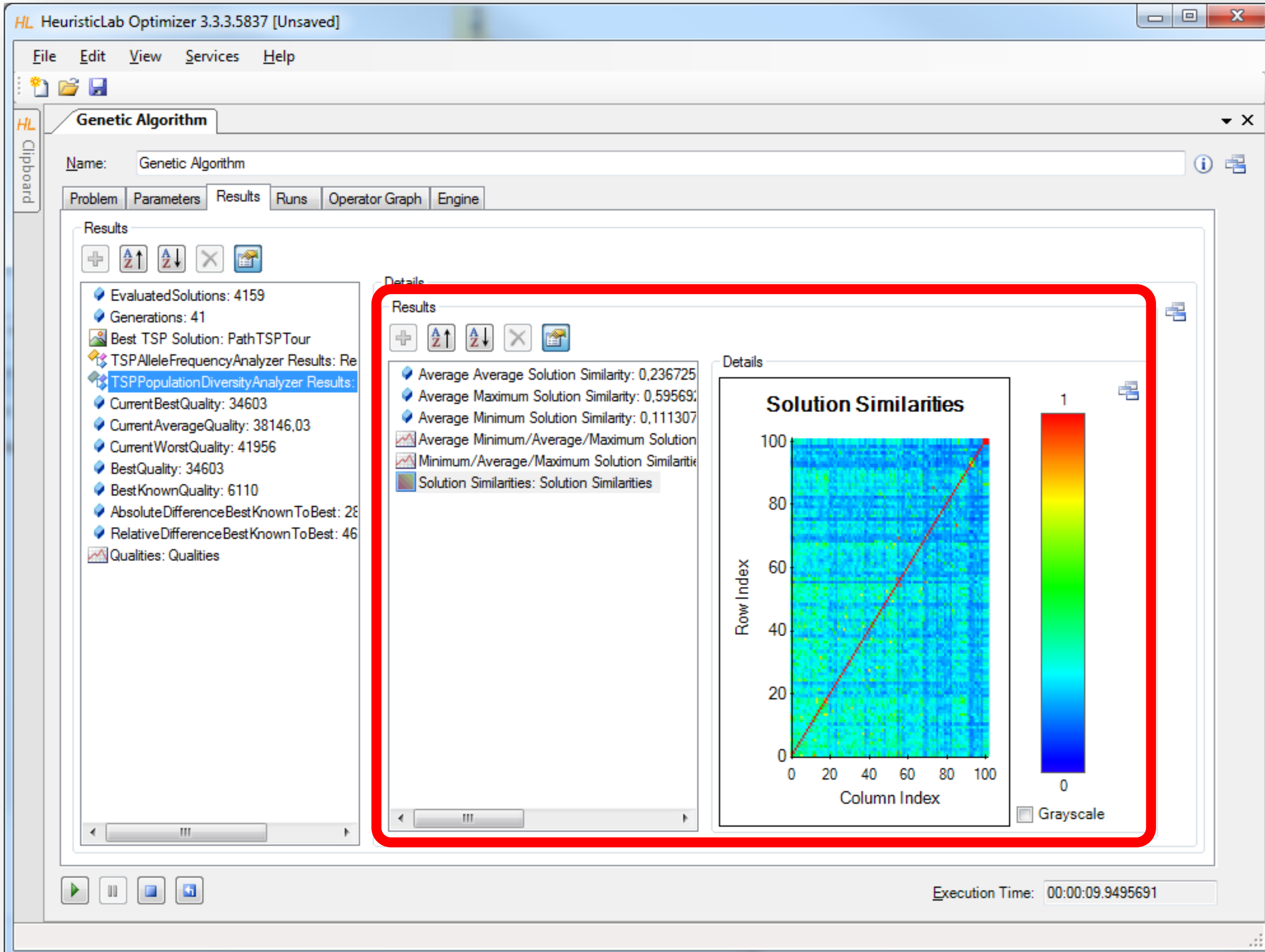


HeuristicLab Optimizer 3.3.3.5837 [Unsaved]
File Edit View Services Help
Genetic Algorithm
Name: Genetic Algorithm
Problem Parameters Results Runs Operator Graph Engine
Parameters
Analyzer: MultiAnalyzer
Crossover: OrderCrossover2
Elites: 1
MaximumGenerations: 1000
MutationProbability: 5 %
Mutator: InversionManipulator
PopulationSize: 100
Seed: 0
Selector: ProportionalSelector
SetSeedRandomly: True
Details
Name: Analyzer
Data Type: MultiAnalyzer
Value
Show in Run:
Name: MultiAnalyzer
Breakpoint:
Operators Parameters
Items
BestTSPSolutionAnalyzer
TSPAlleleFrequencyAnalyzer
TSPPopulationDiversityAnalyzer
BestAverageWorstQualityAnalyz
Details
Execution Time: 00:00:00

TSPAlleleFrequencyAnalyzer

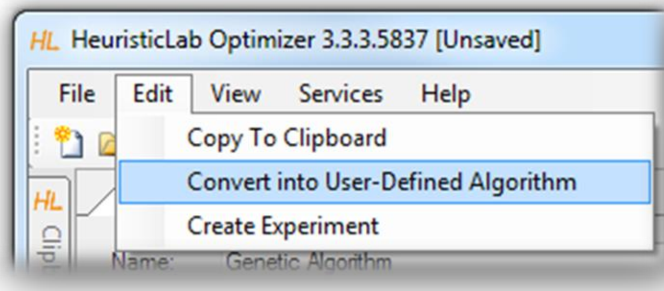


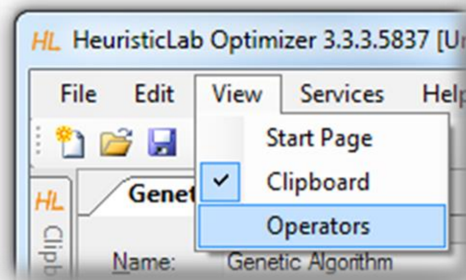
TSPPopulationDiversityAnalyzer



Building User-Defined Algorithms

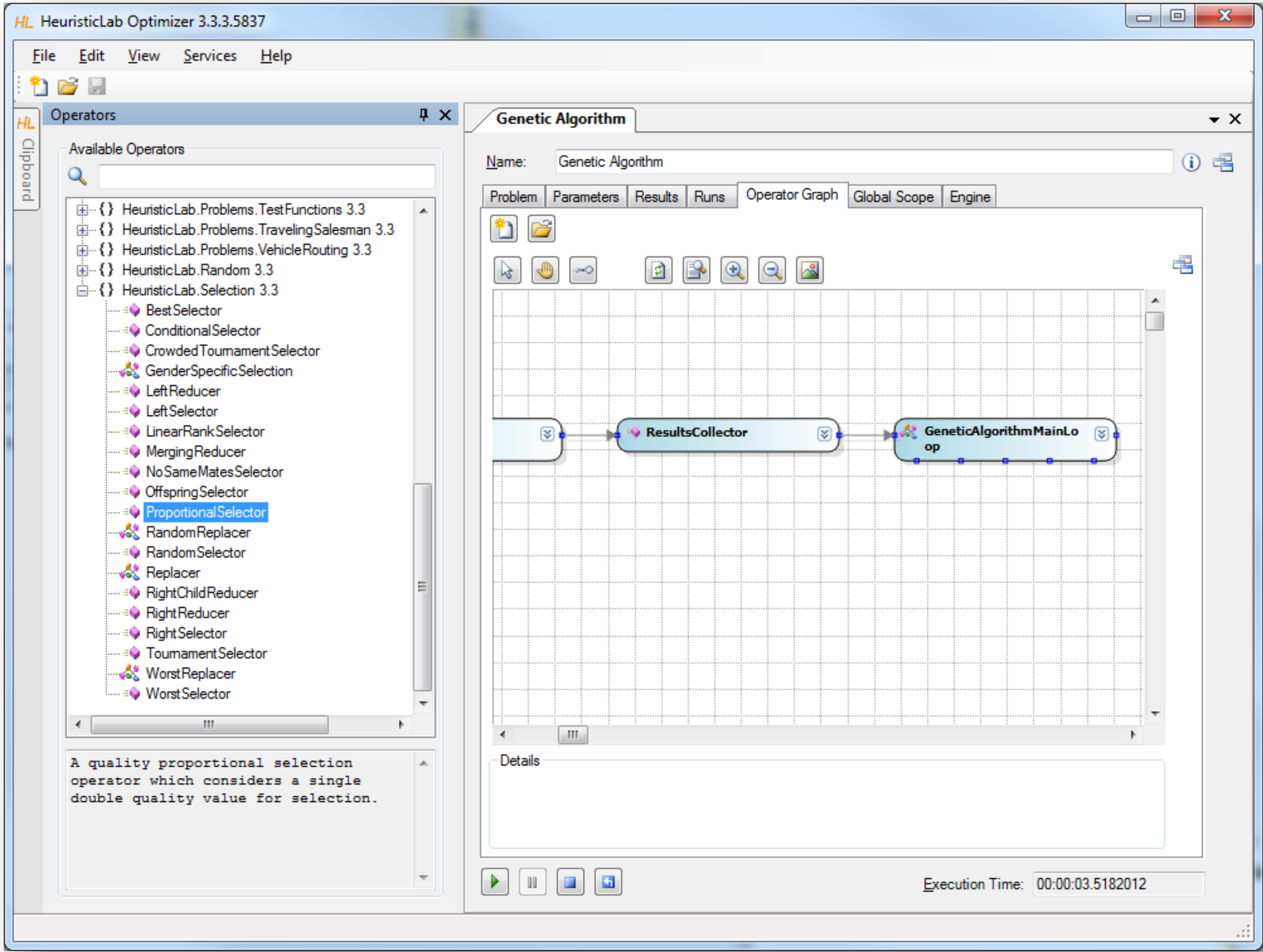
- Operator graphs
  - algorithms are represented as operator graphs
  - operator graphs of user-defined algorithms can be changed
  - algorithms can be defined in the graphical algorithm designer
  - use the menu to convert a standard algorithm into a user-defined algorithm



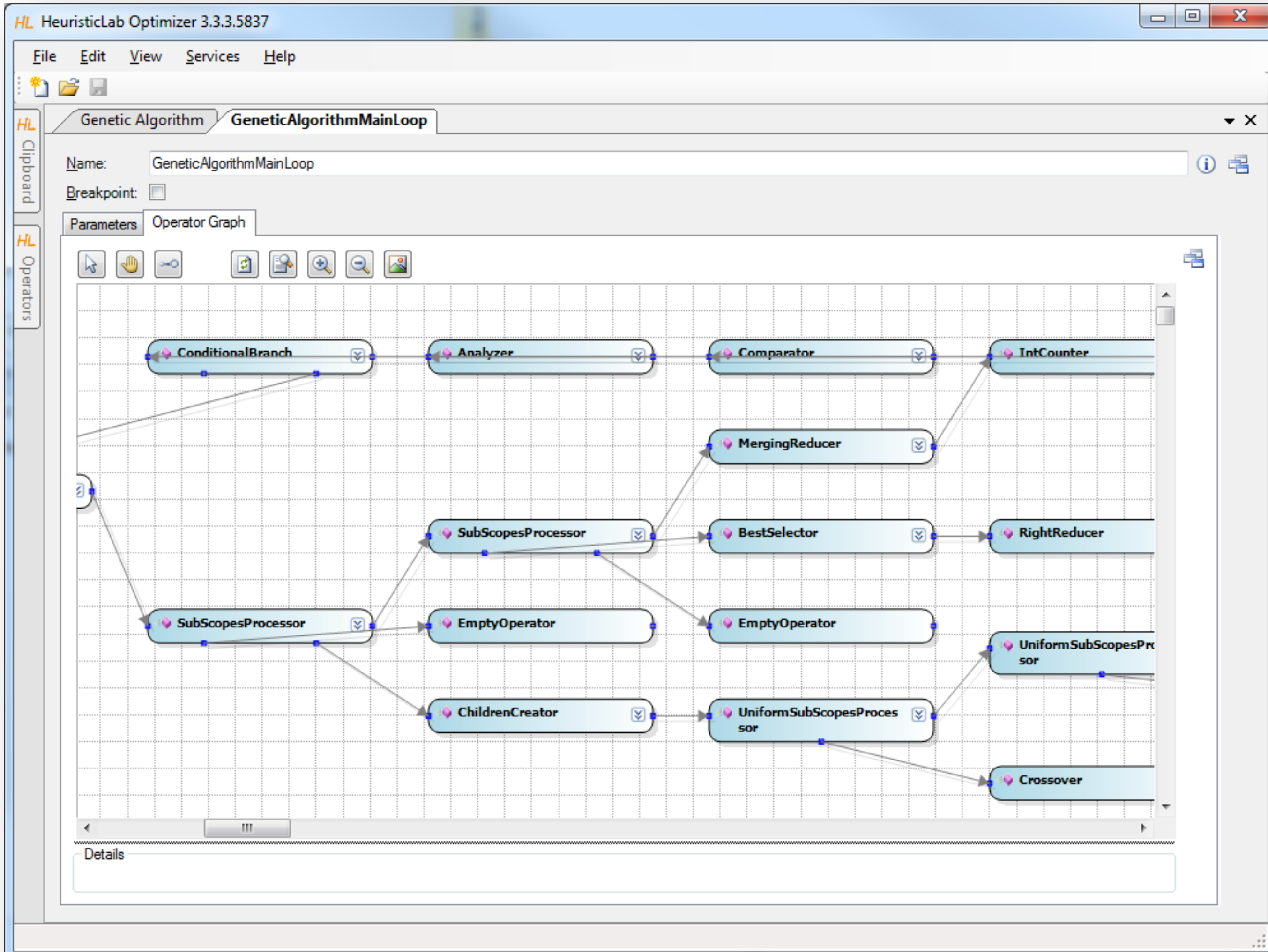


- Operators sidebar
  - drag & drop operators into an operator graph
- Programmable operators
  - add programmable operators in order to implement custom logic in an algorithm
  - no additional development environment needed
- Debug algorithms
  - use the debug engine to obtain detailed information during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

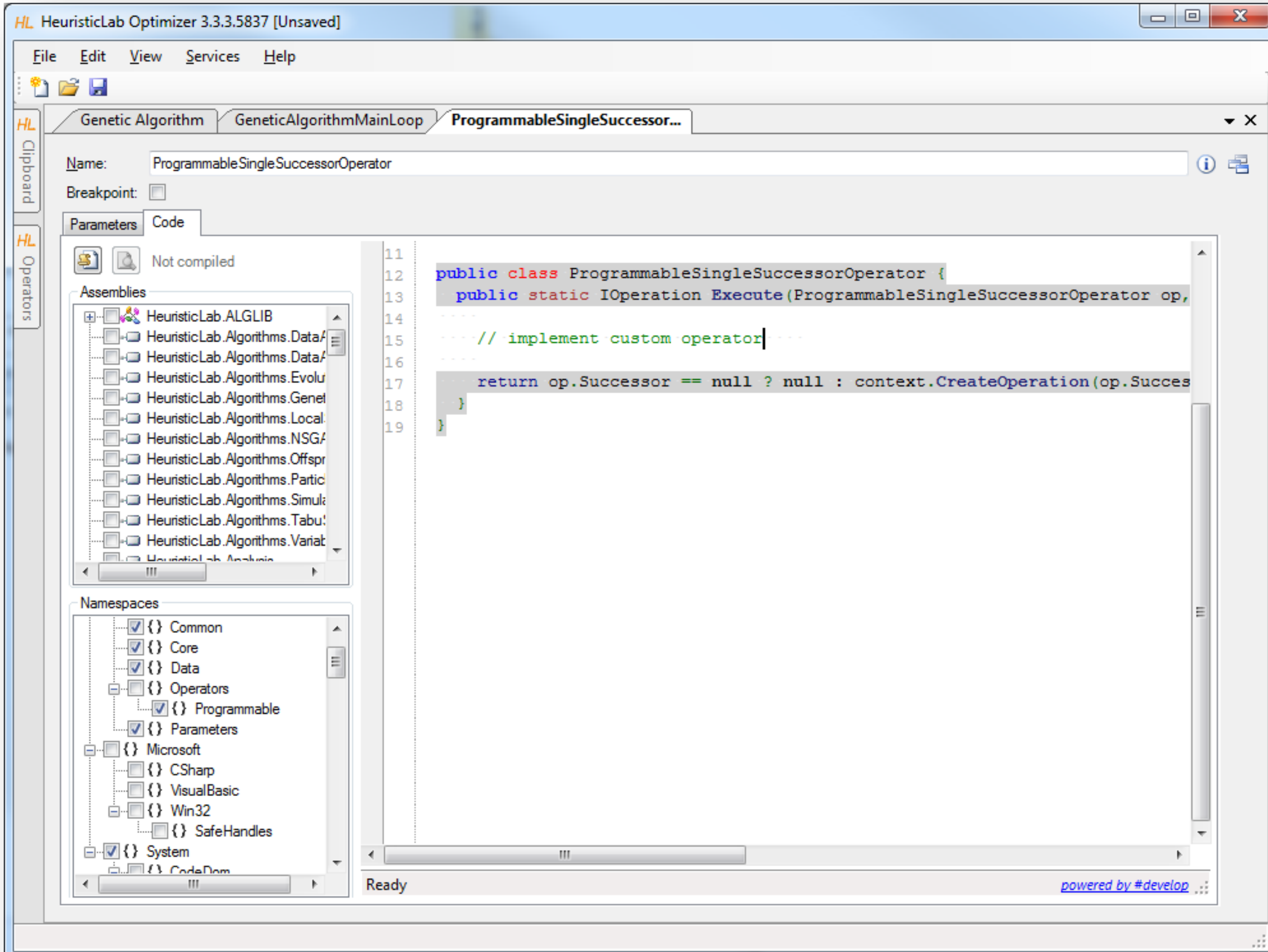
Building User-Defined Algorithms



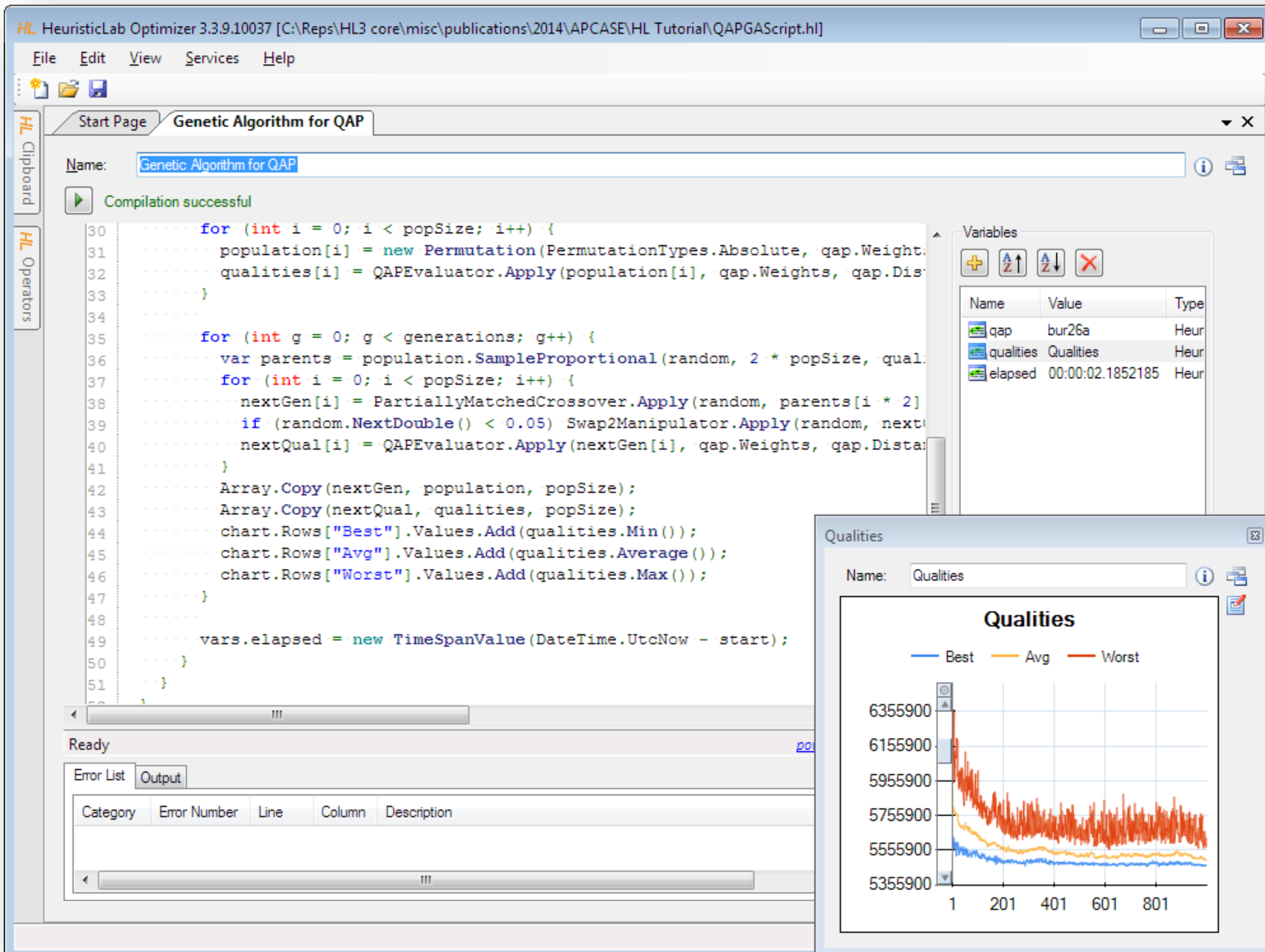
Building User-Defined Algorithms



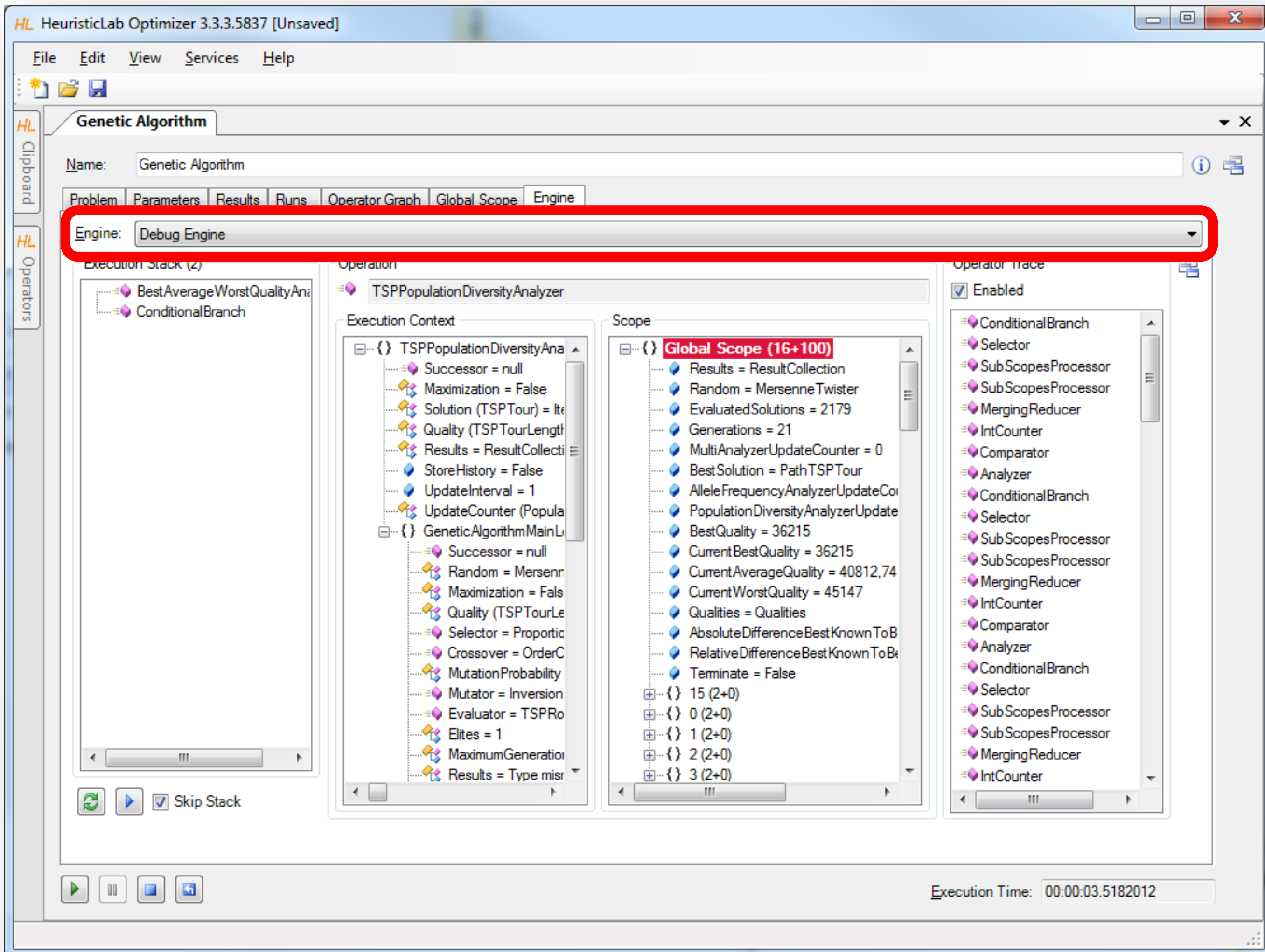
Programmable Operators



Scripting Environment



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 $(y_1 \ldots y_N)^T$

- Goal: learn association of input variable values to labels

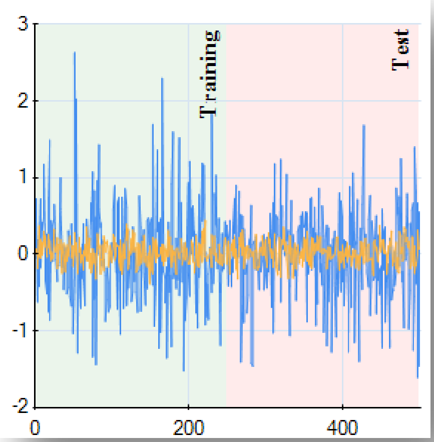
- Common tasks
  - Regression (real-valued labels)
  - Classification (discrete labels)
  - Clustering (no labels, group similar observations)

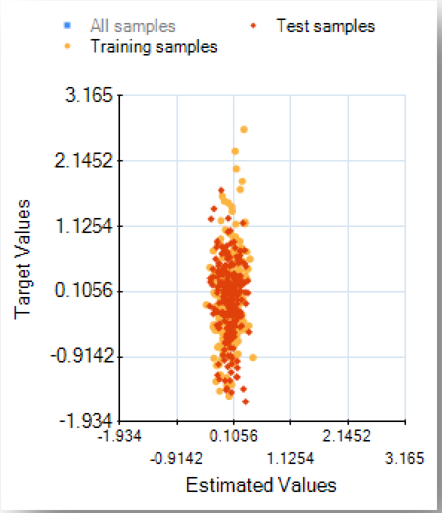
Data-based Modeling Algorithms in HeuristicLab

- Symbolic regression and classification using genetic programming
- External Libraries:
  - Linear Regression, Linear Discriminate Analysis
  - K-Means clustering
  - Support Vector Machines

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



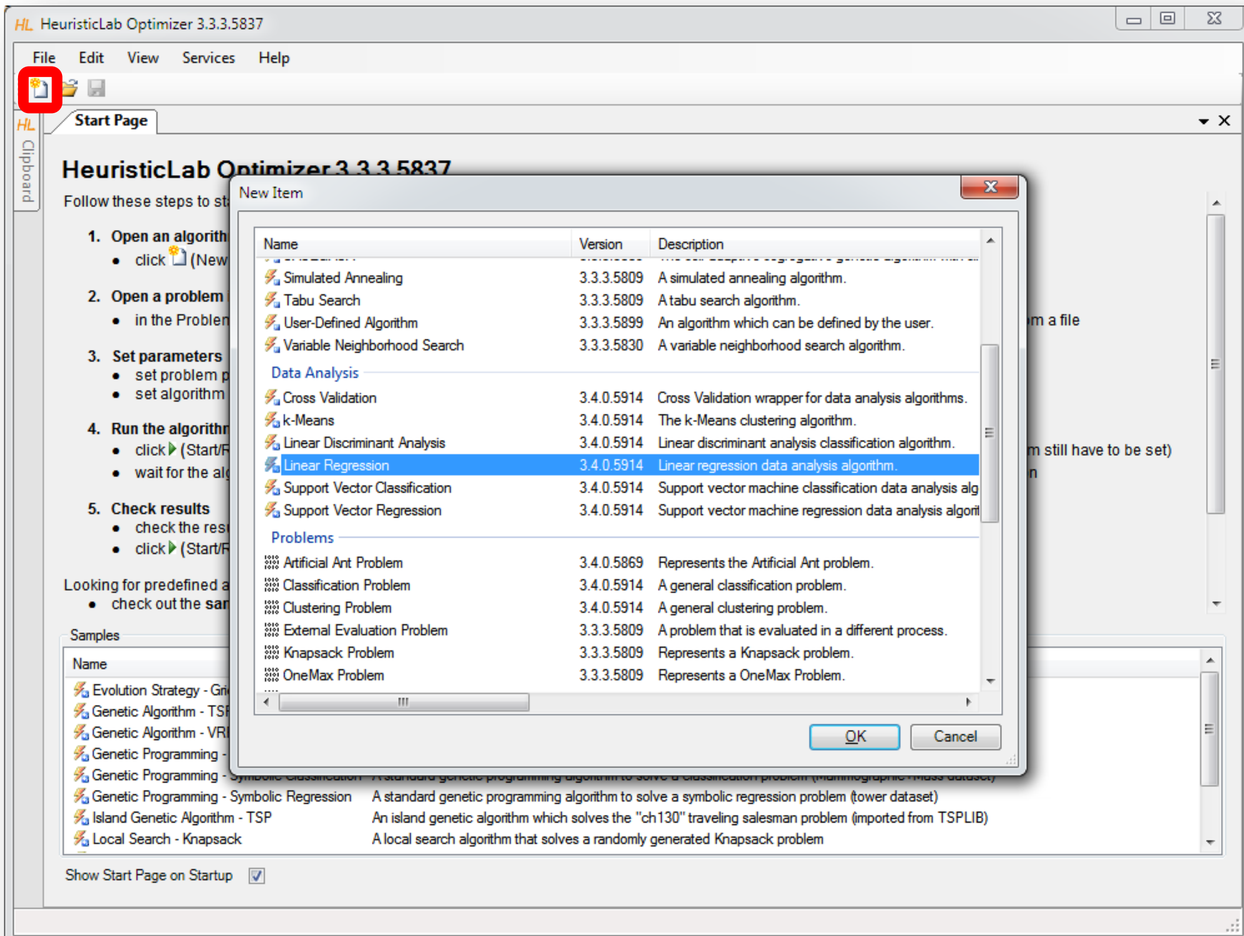


Case Study: Regression

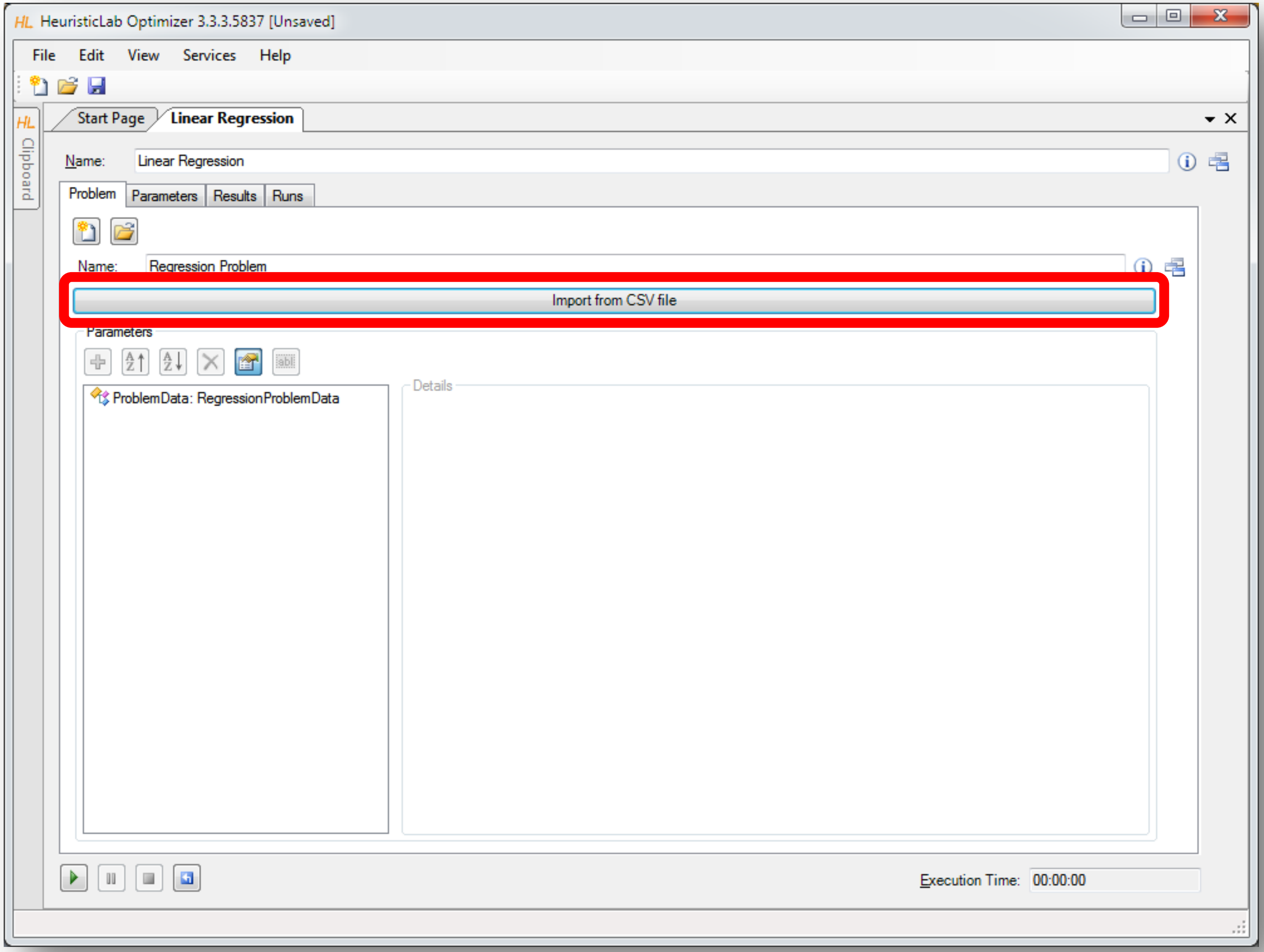
- Poly-10 benchmark problem dataset
  - 10 input variables $x_1 \dots x_{10}$
  - $y = x_1 \cdot x_2 + x_3 \cdot x_4 + x_5 \cdot x_6 + x_1 \cdot x_7 \cdot x_9 + x_3 \cdot x_6 \cdot x_{10}$
  - non-linear modeling approach necessary
  - frequently used in GP literature
  - download
    http://dev.heuristiclab.com/AdditionalMaterial#GECCO2012

Linear Regression

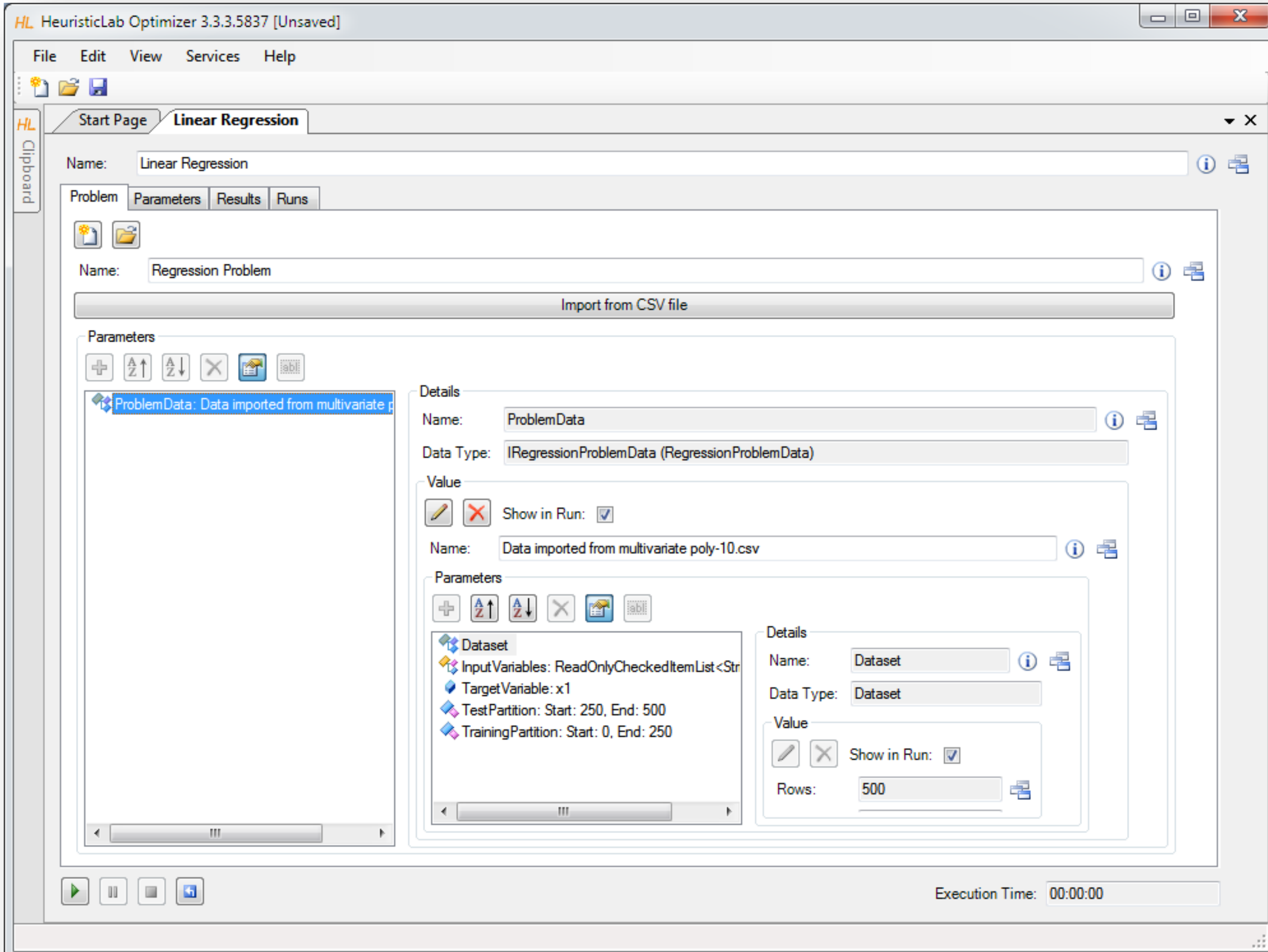
- Create new algorithm



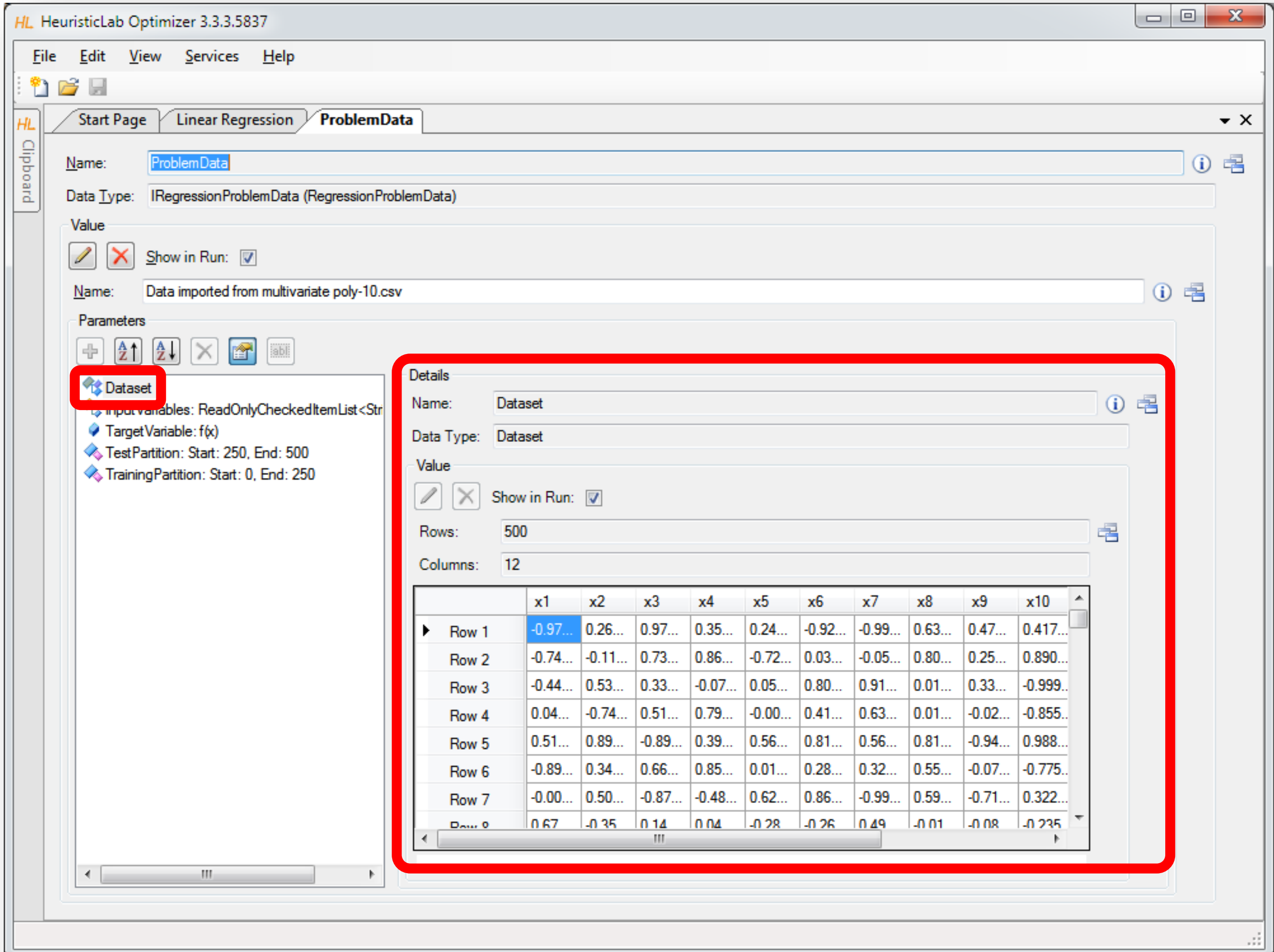
Import Data from CSV-File



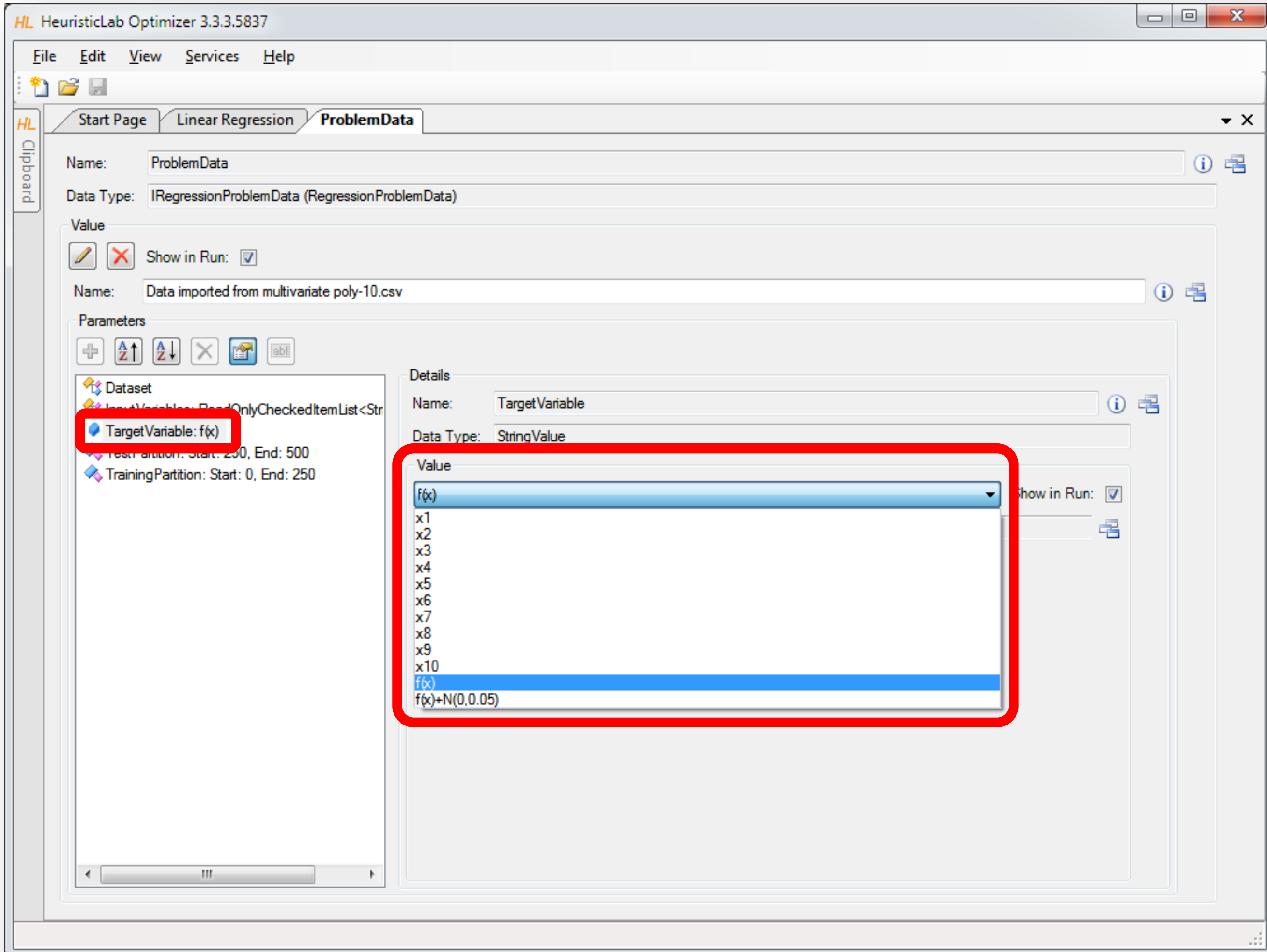
Inspect and Configure Dataset



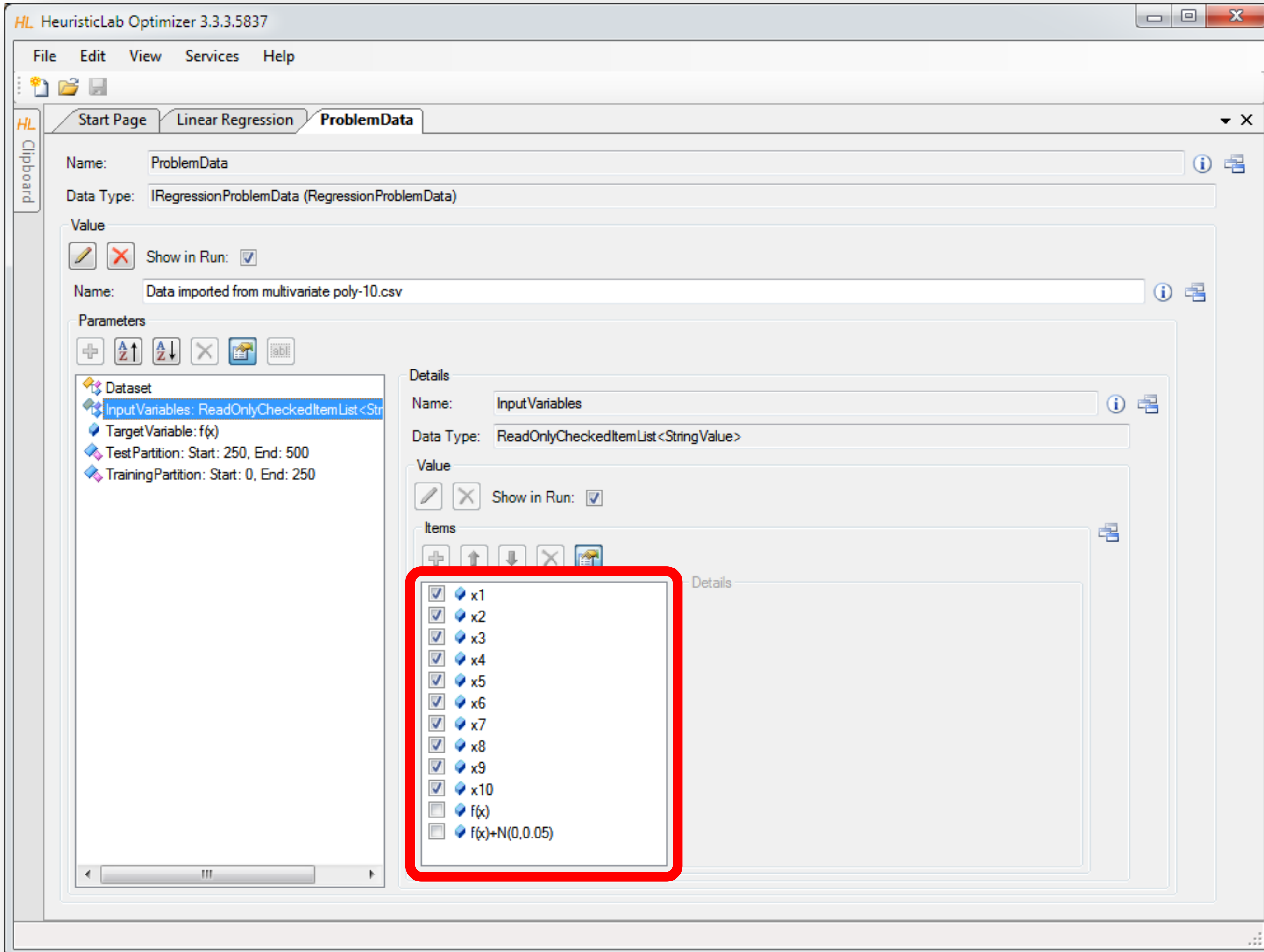
Inspect Imported Data



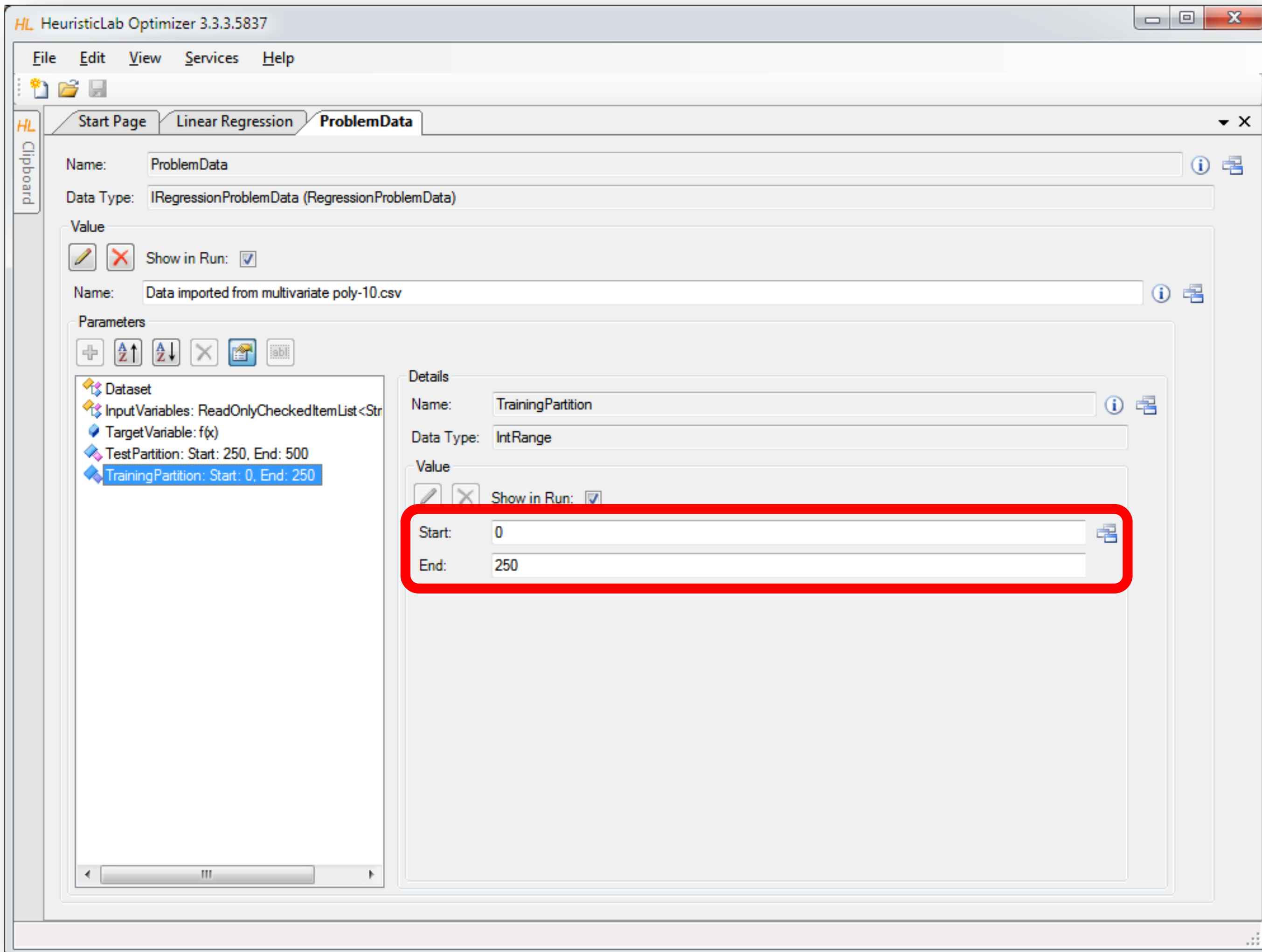
Set Target Variable



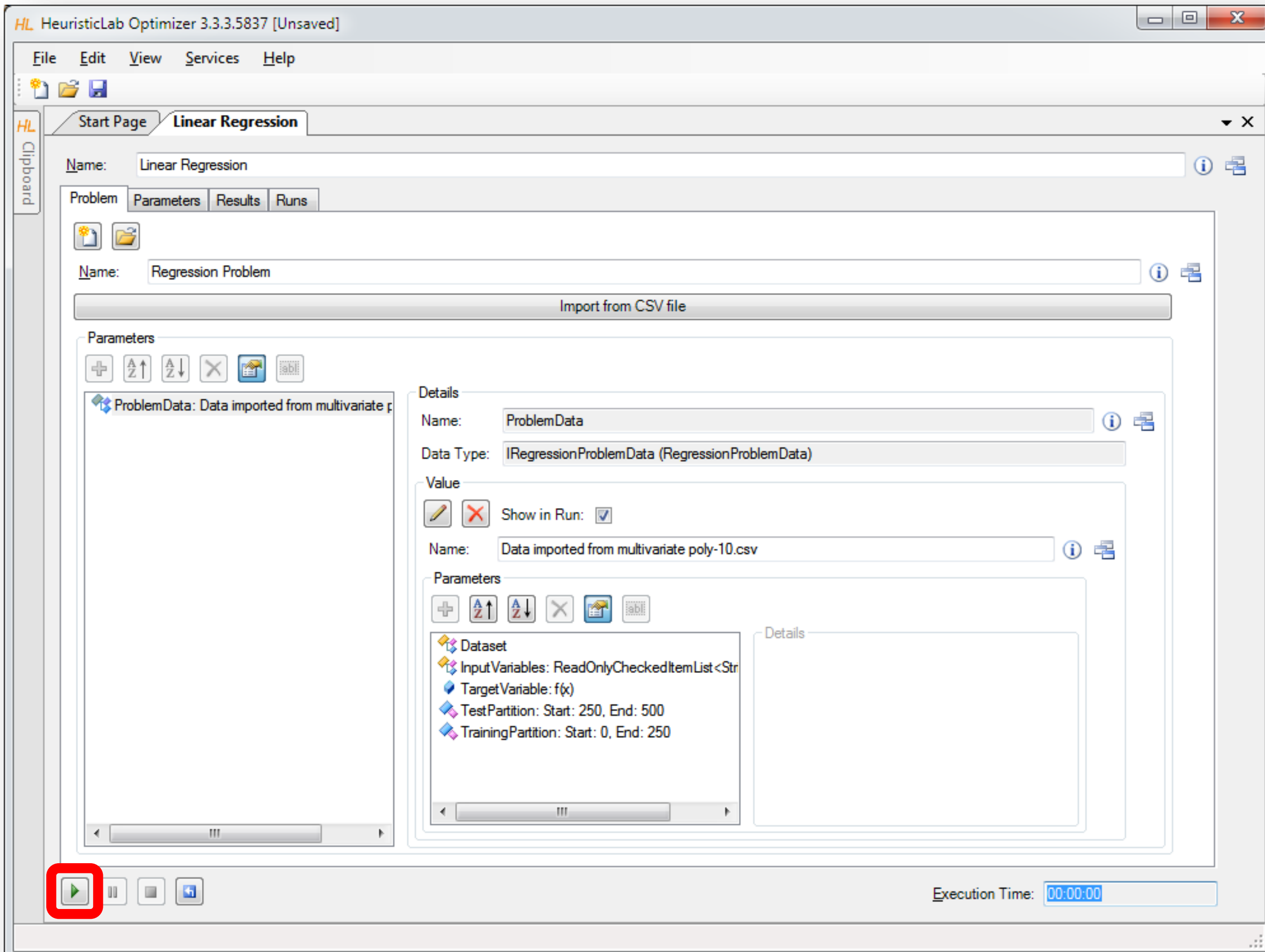
Select Input Variables



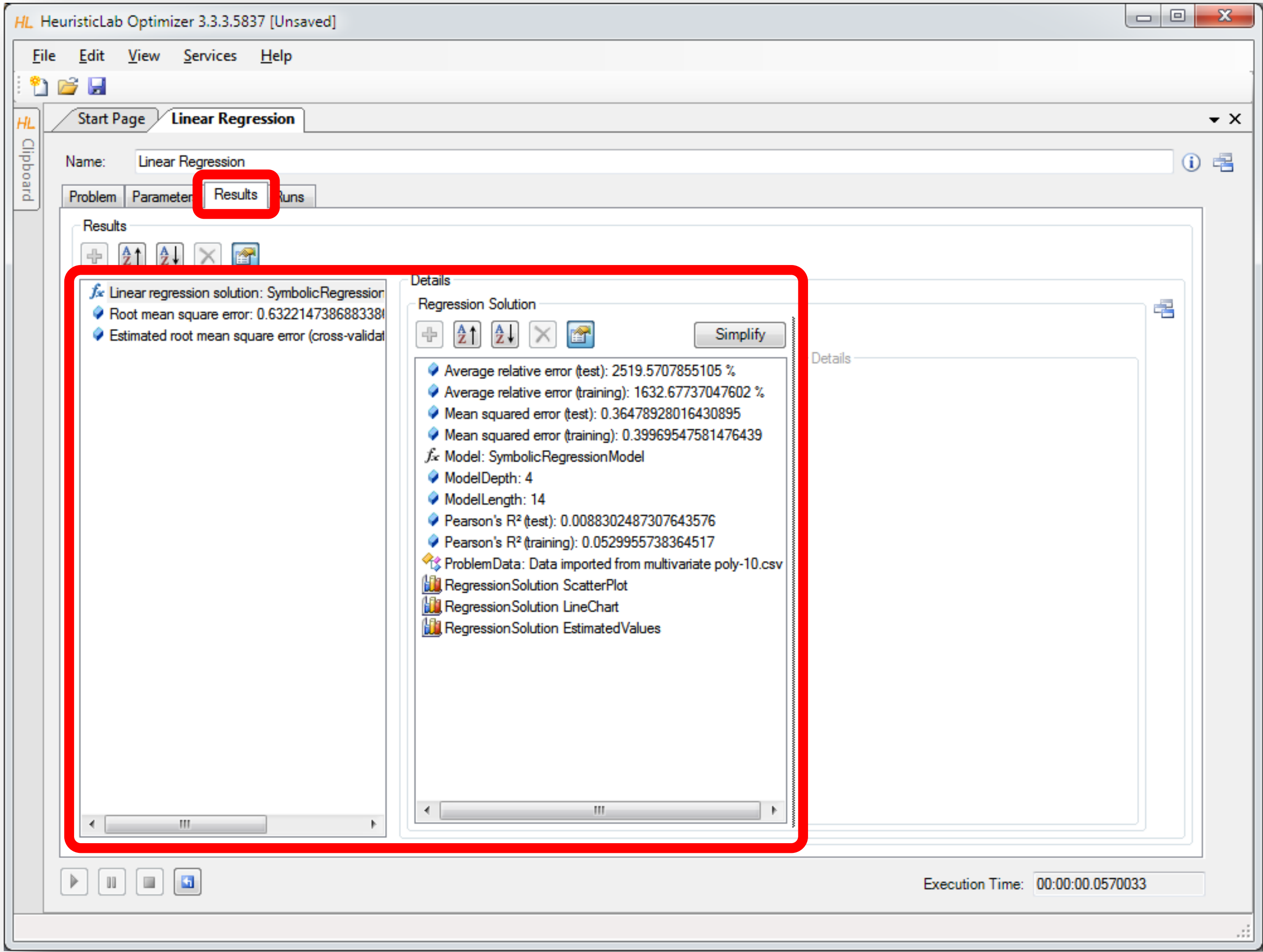
Configure Training and Test Partitions



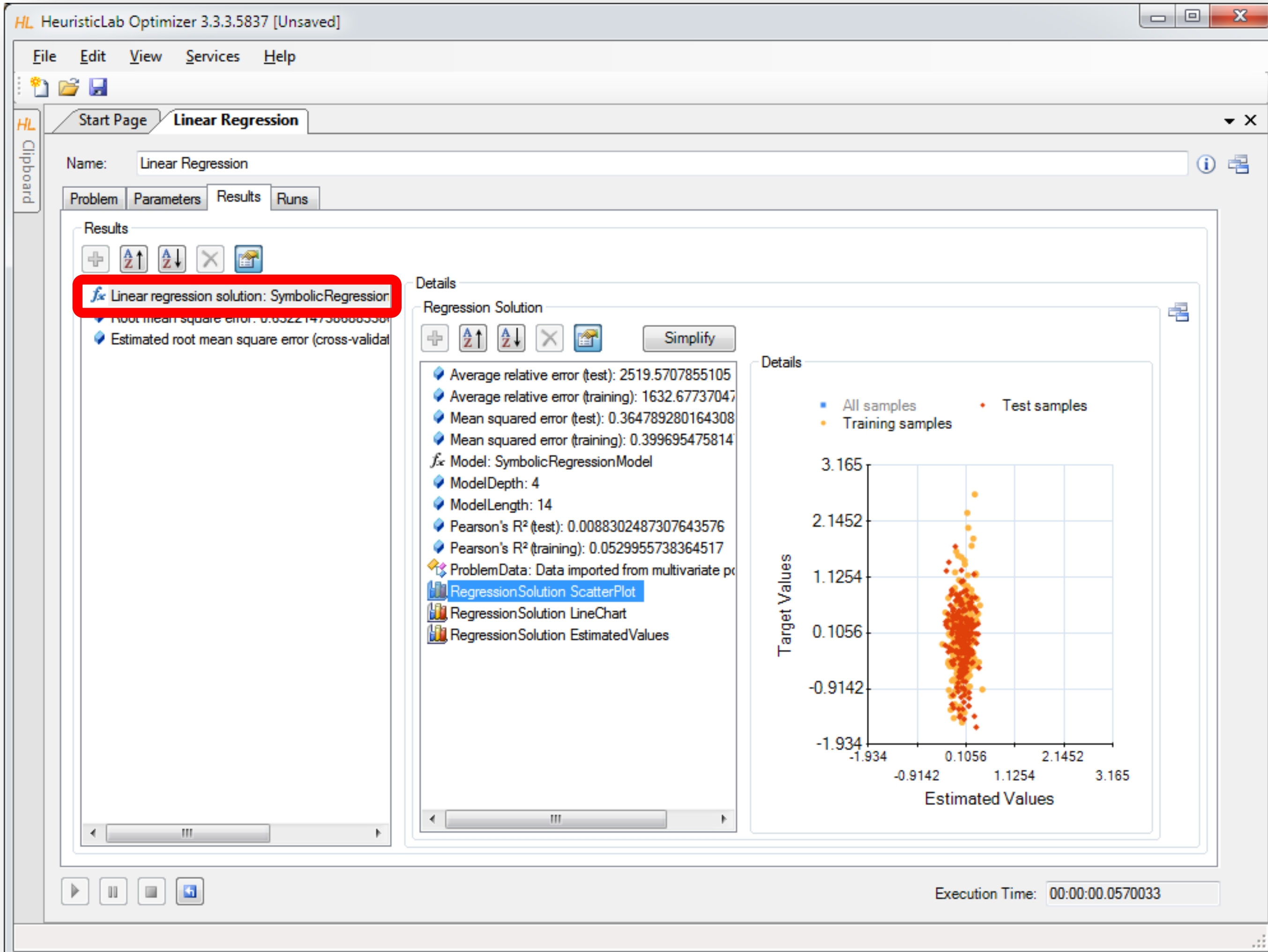
Run Linear Regression



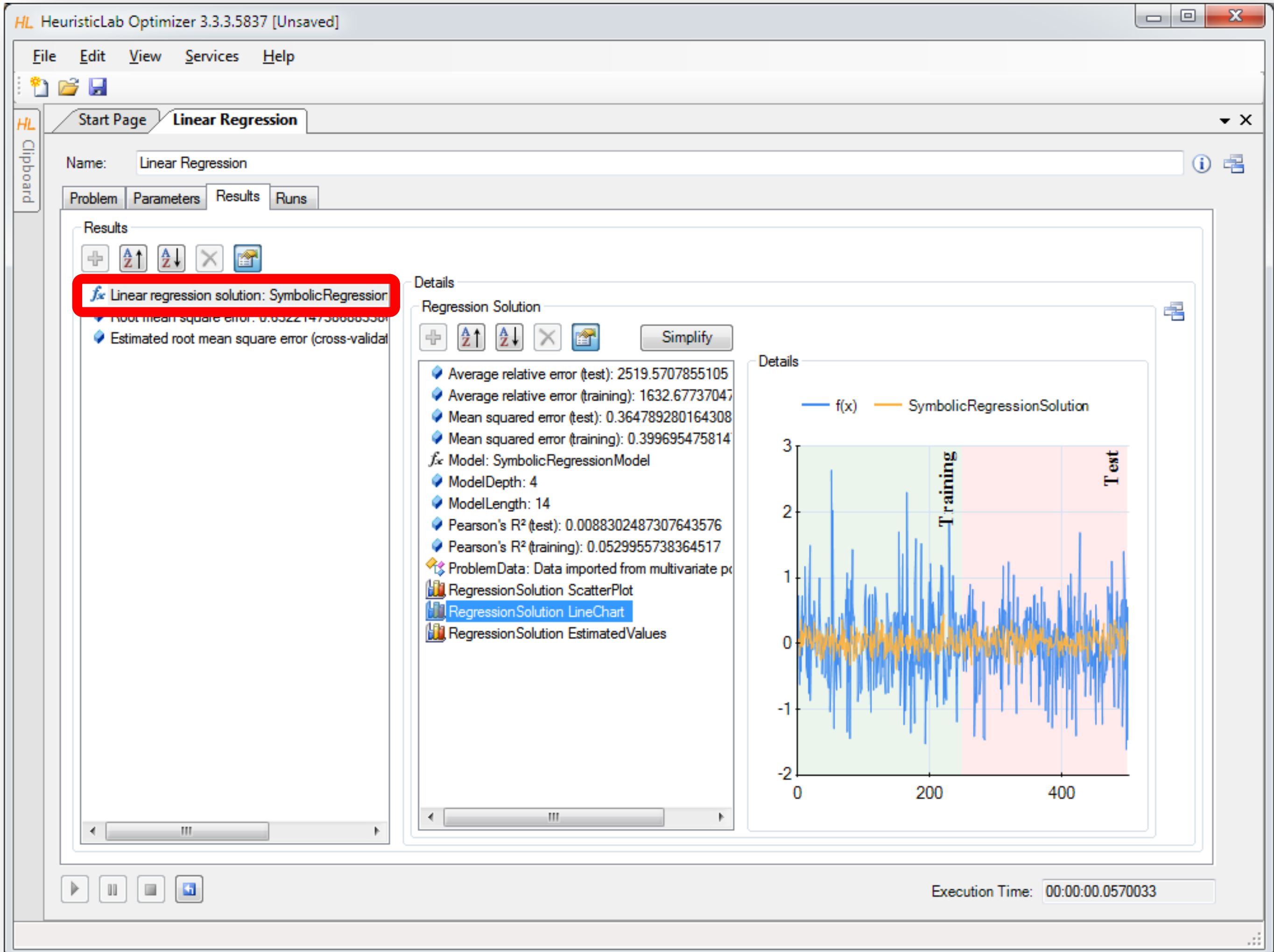
Inspect Results



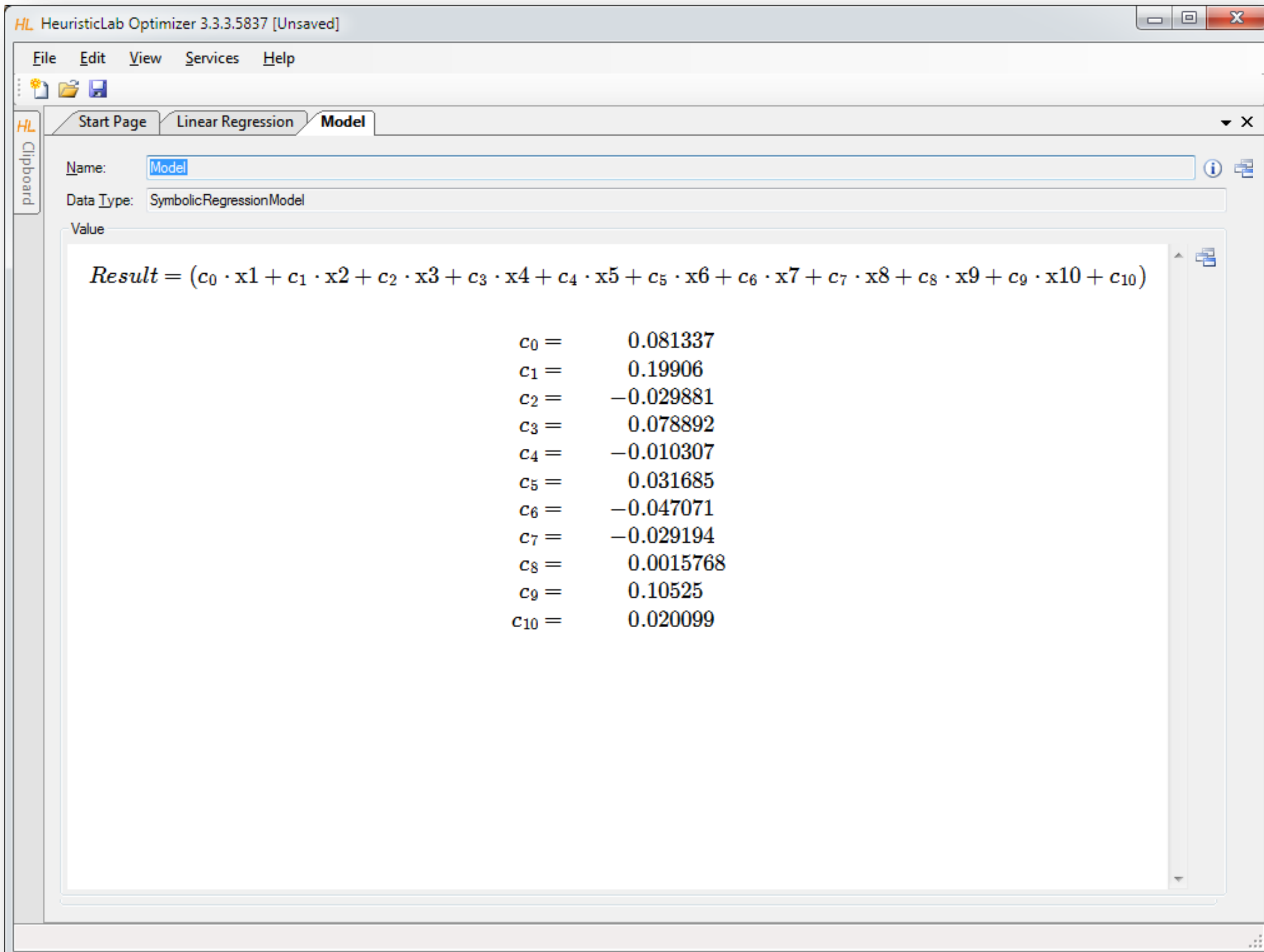
Inspect Scatterplot of Predicted and Target Values



Inspect Linechart



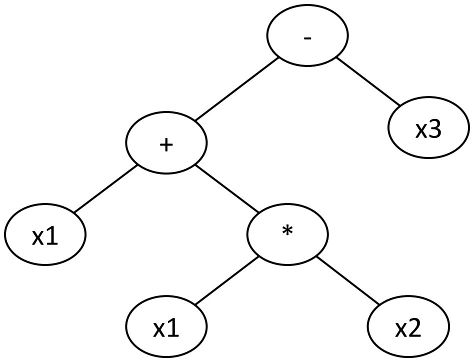
Inspect the Model



Symbolic Regression with HeuristicLab

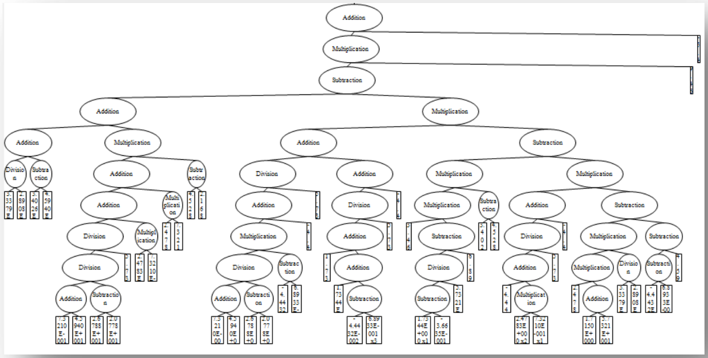
- Linear regression produced an inaccurate model.
- Next: produce a nonlinear symbolic regression model using genetic programming

- Genetic programming
  - evolve variable-length models
  - model representation: symbolic expression tree
  - structure and model parameters are evolved side-by-side
  - white-box models

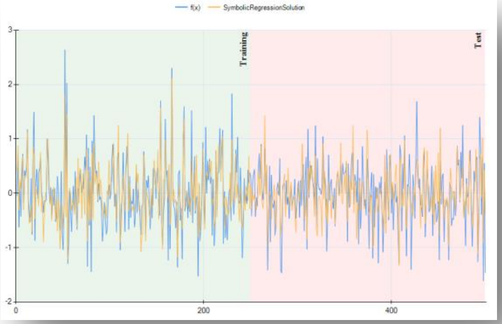


Symbolic Regression with HeuristicLab

- Demonstration
  - problem configuration
  - function set and terminal set
  - model size constraints
  - Evaluation

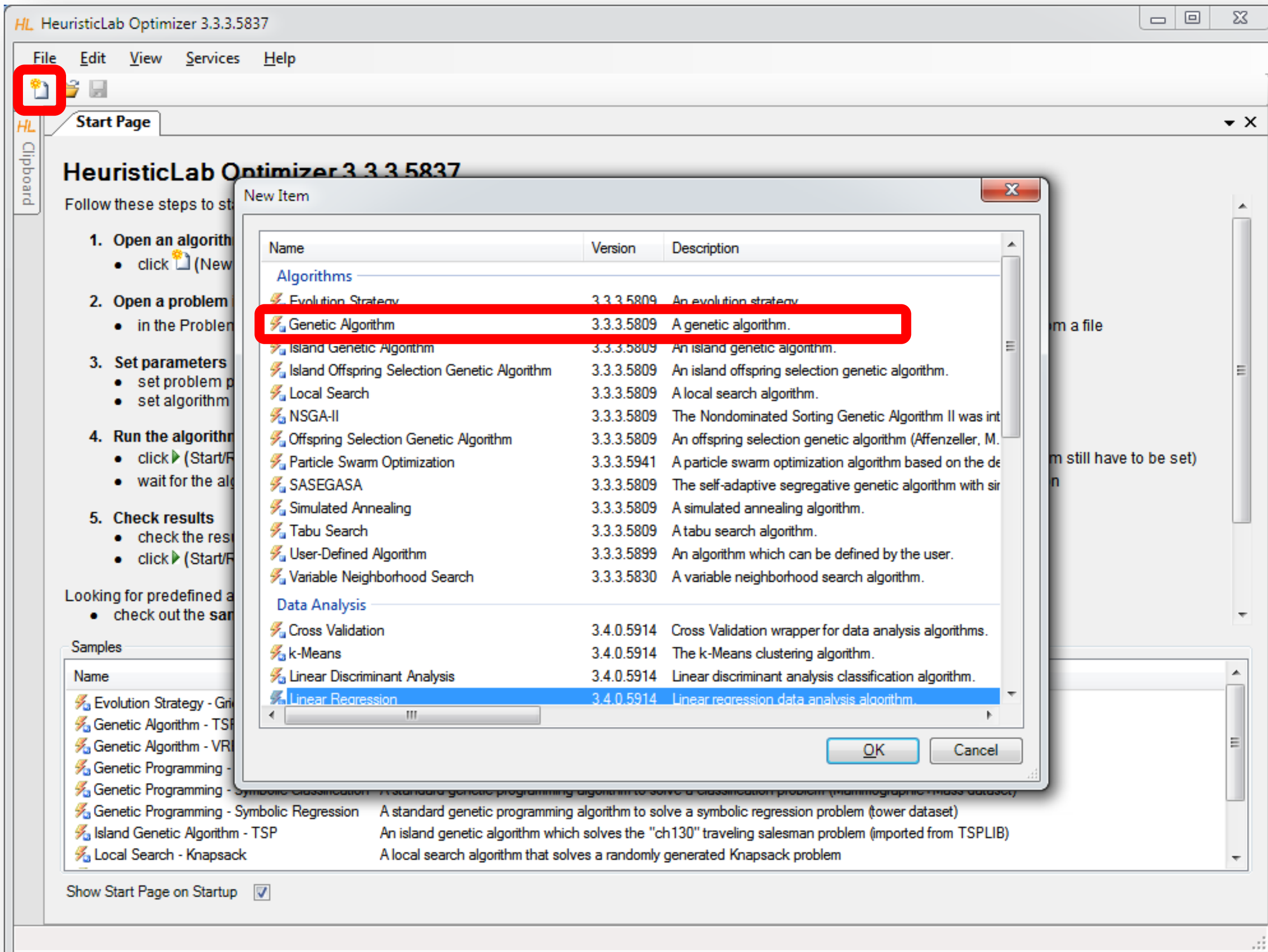


- Algorithm configuration
  - selection
  - Mutation

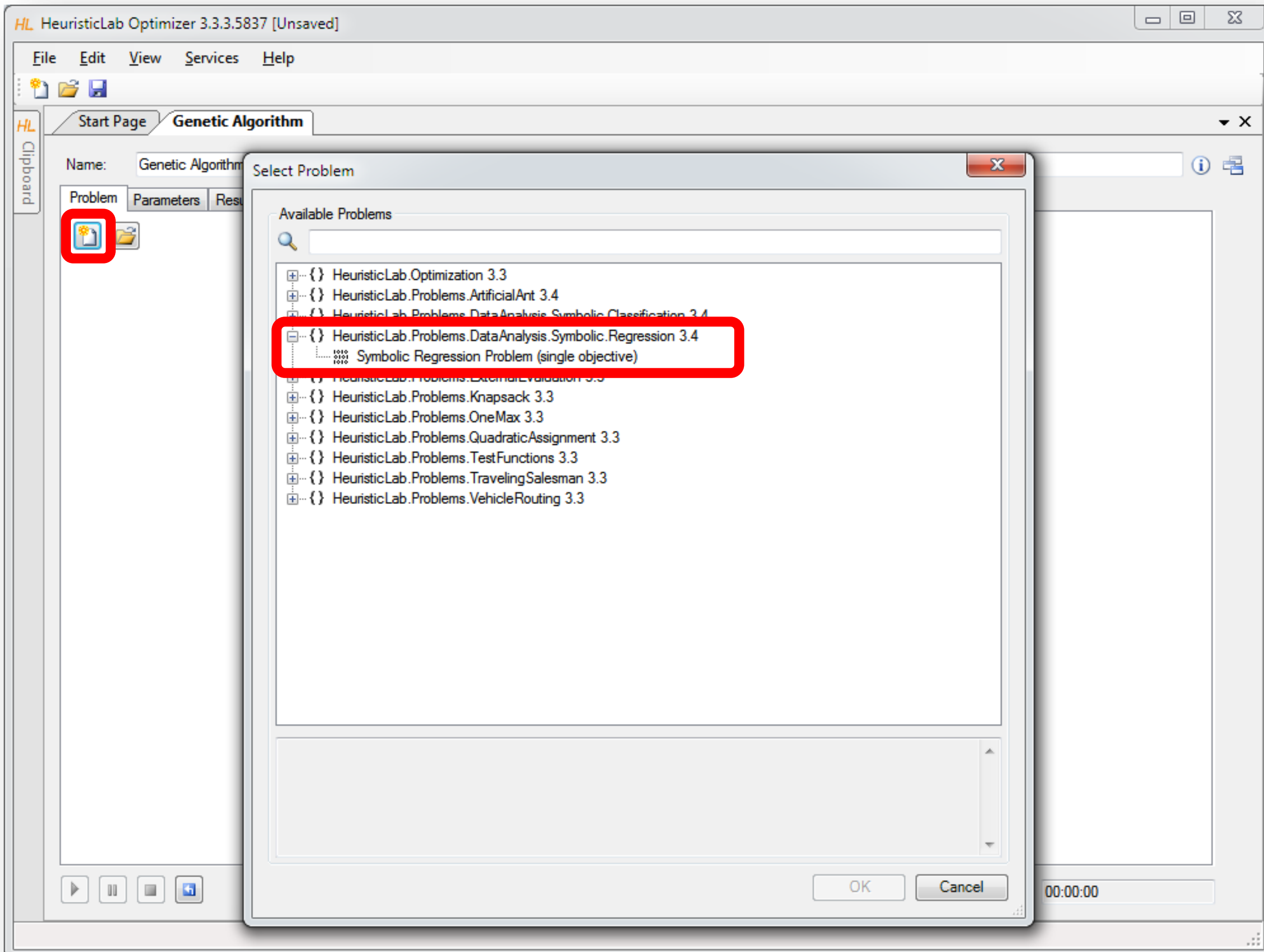


- Analysis of results
  - model accuracy
  - model structure and parameters

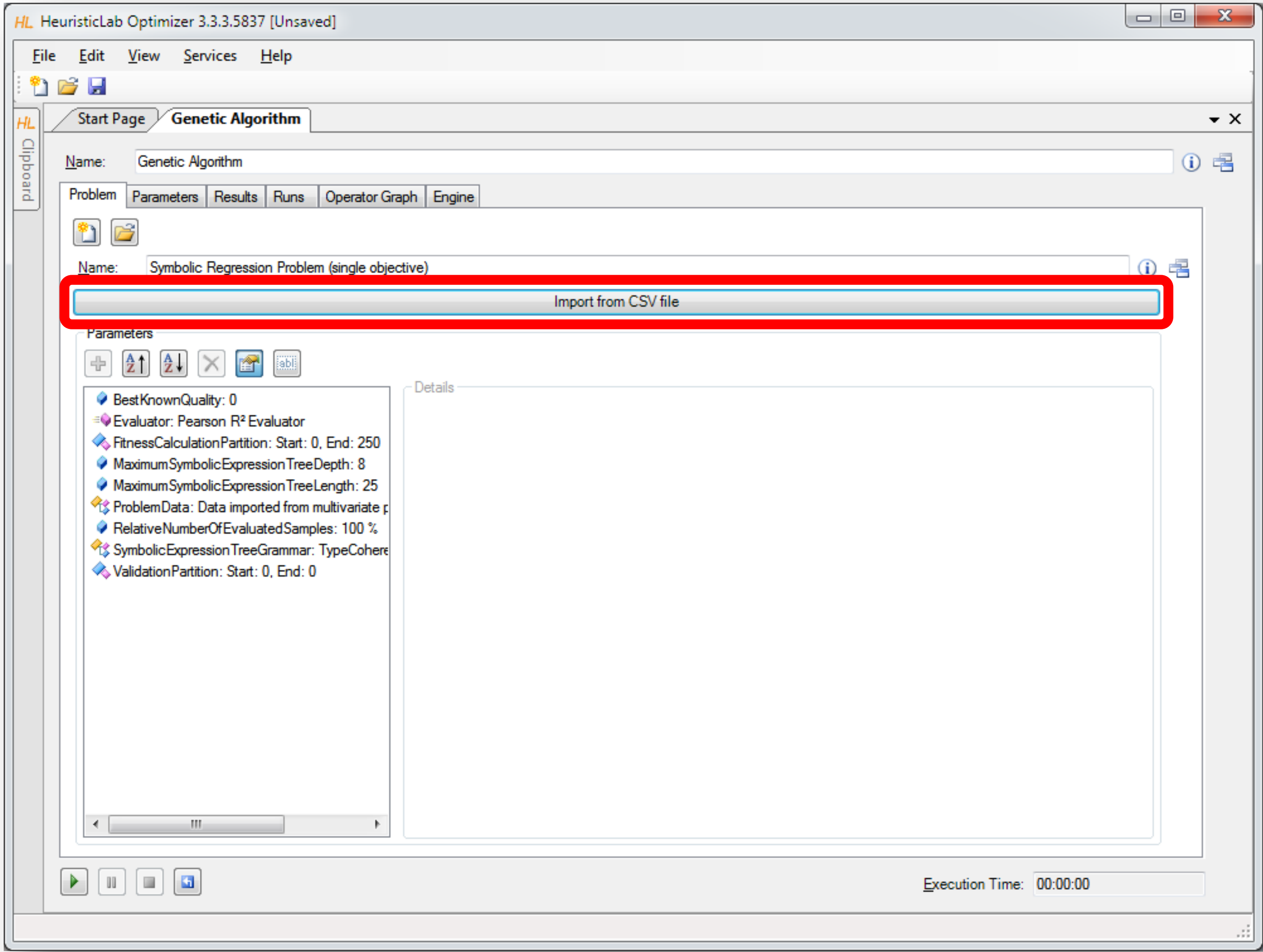
Create New Genetic Algorithm



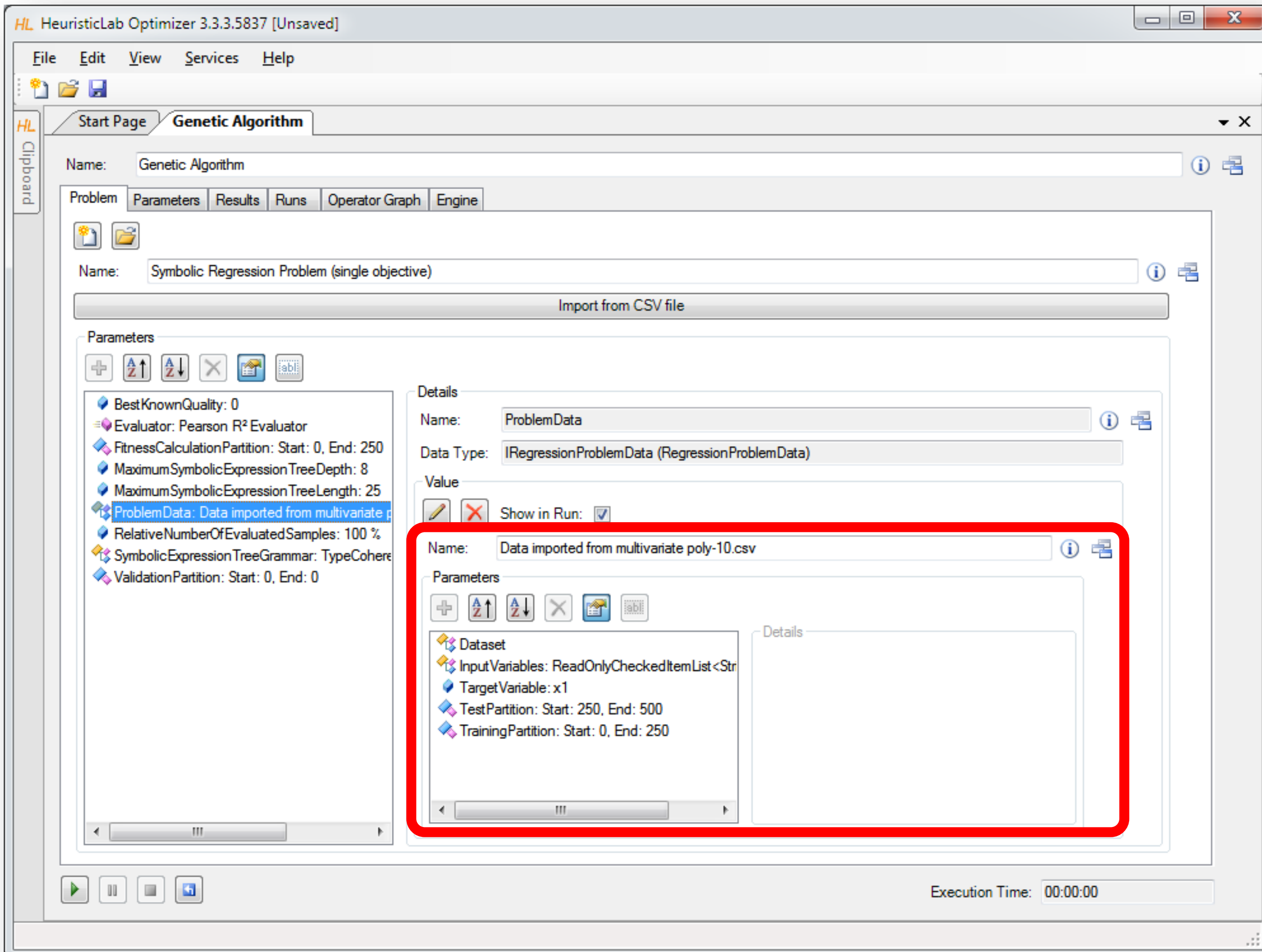
Create New Symbolic Regression Problem



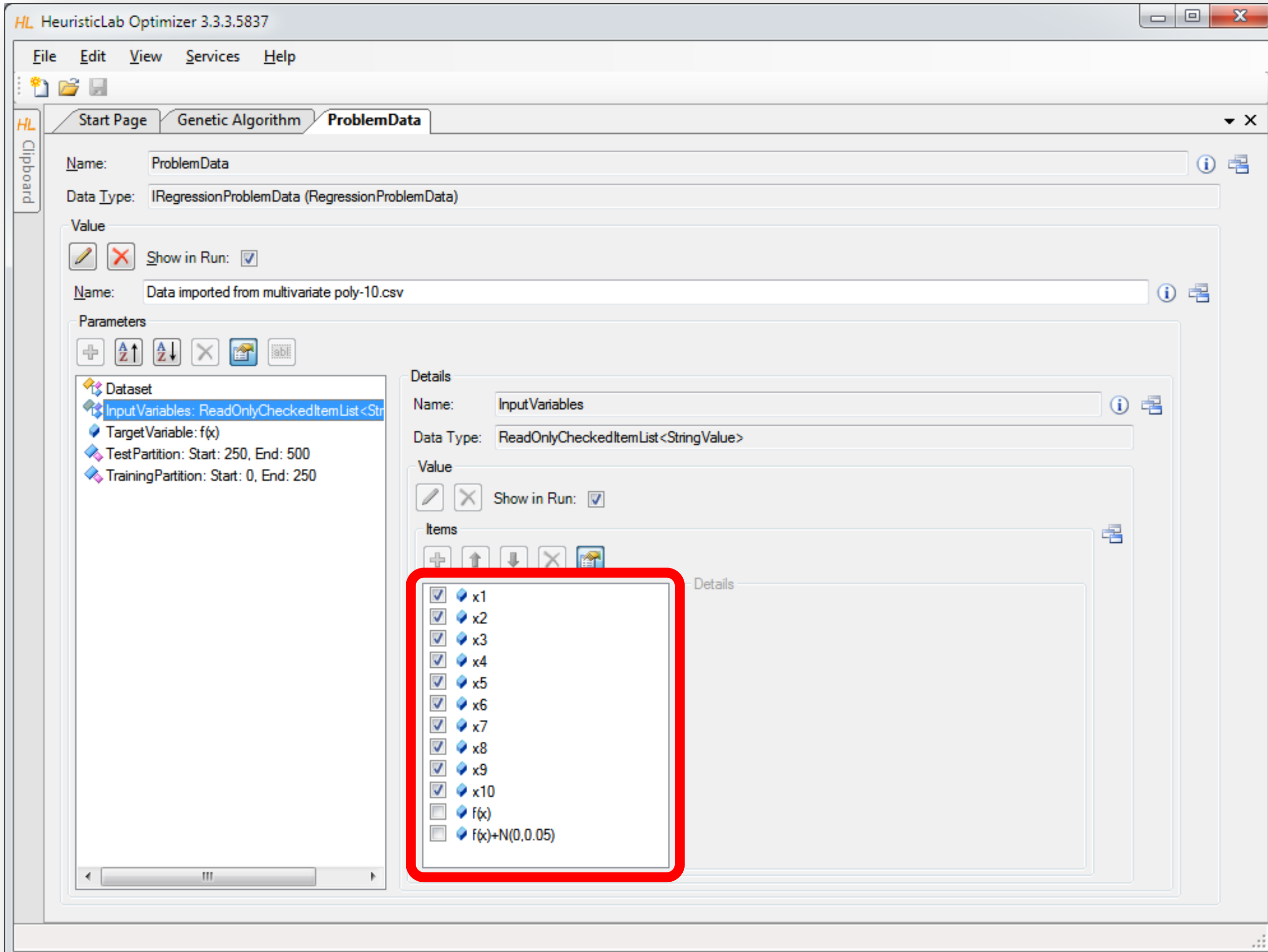
Import Data


HeuristicLab Optimizer 3.3.3.5837 [Unsaved]
File Edit View Services Help
Clipboard
Start Page
Genetic Algorithm
Name: Genetic Algorithm
Problem Parameters Results Runs Operator Graph Engine
Name: Symbolic Regression Problem (single objective)
Import from CSV file
Parameters
BestKnownQuality: 0
Evaluator: Pearson R² Evaluator
FitnessCalculationPartition: Start: 0, End: 250
MaximumSymbolicExpressionTreeDepth: 8
MaximumSymbolicExpressionTreeLength: 25
ProblemData: Data imported from multivariate p
RelativeNumberOfEvaluatedSamples: 100 %
SymbolicExpressionTreeGrammar: TypeCohere
ValidationPartition: Start: 0, End: 0
Details
Execution Time: 00:00:00

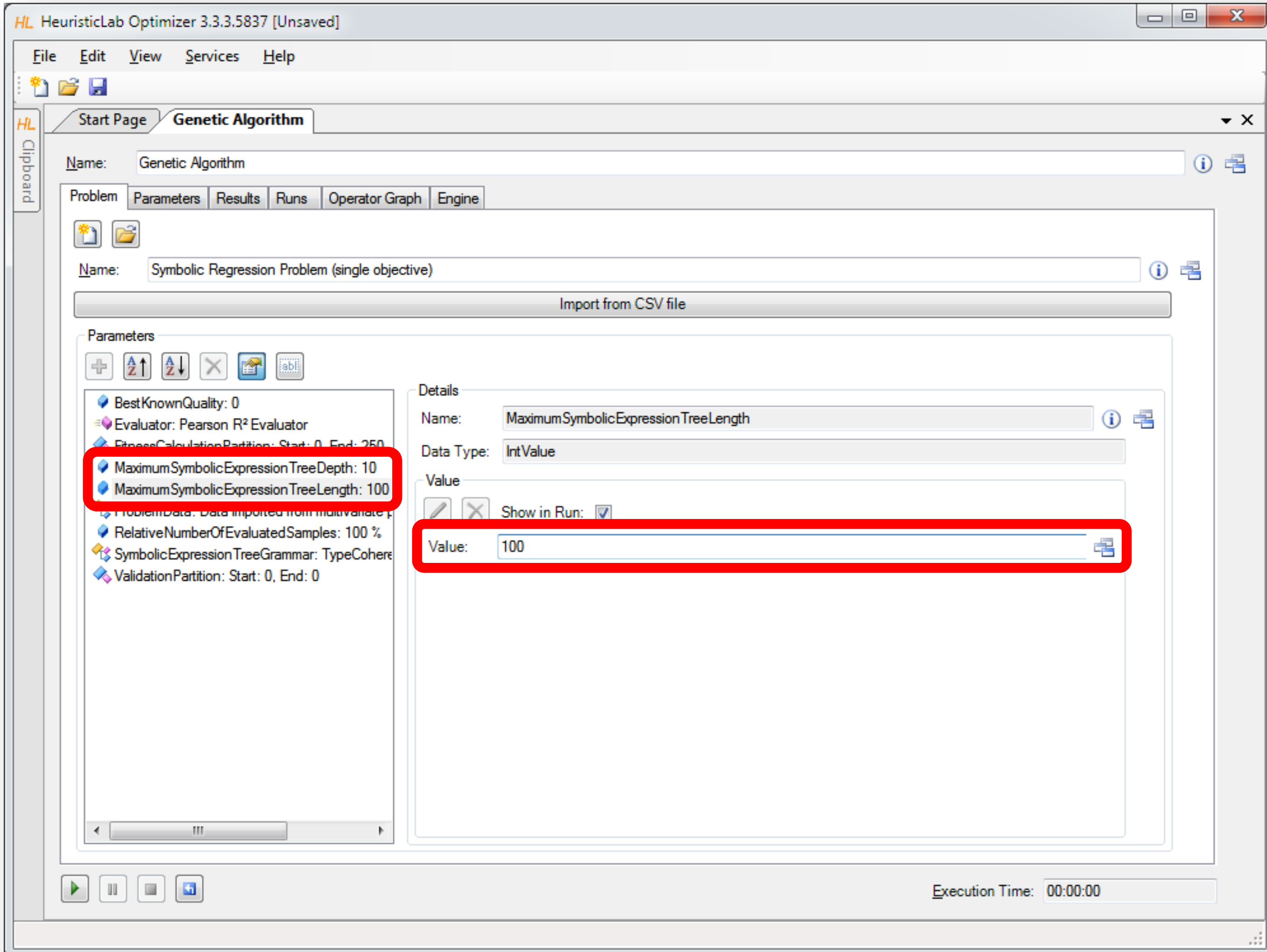
Inspect Data and Configure Dataset



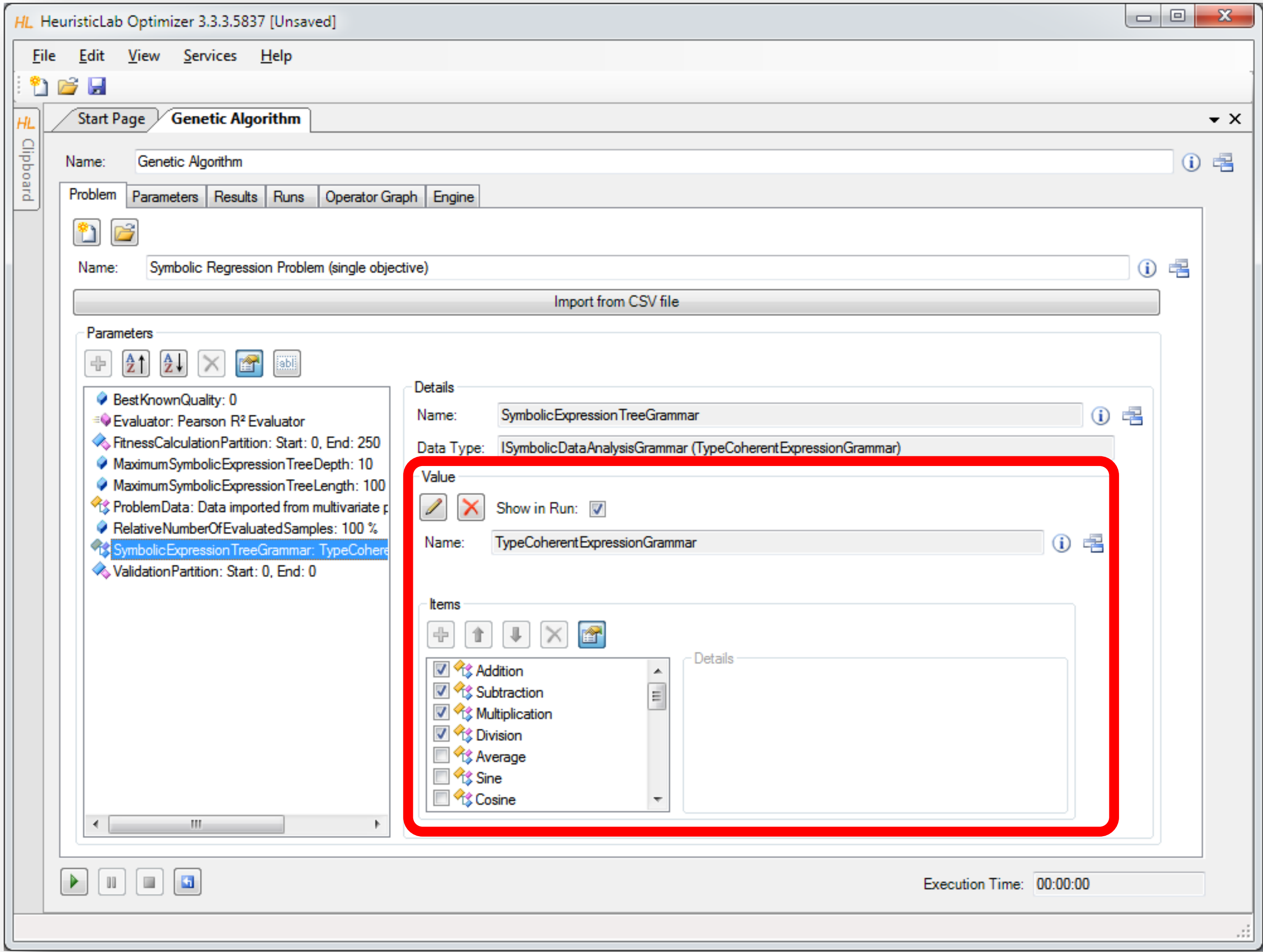
Set Target and Input Variables



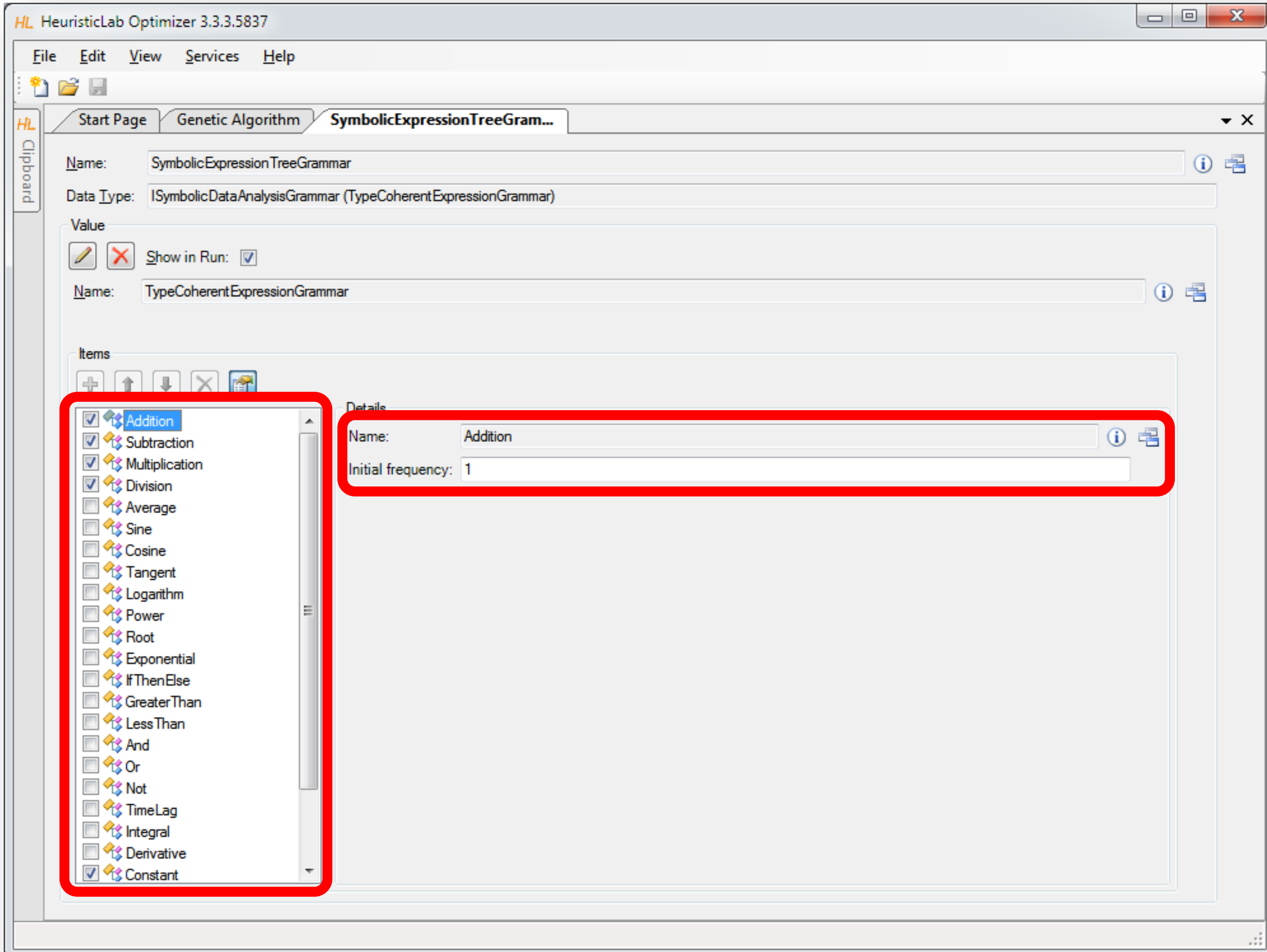
Configure Maximal Model Depth and Length



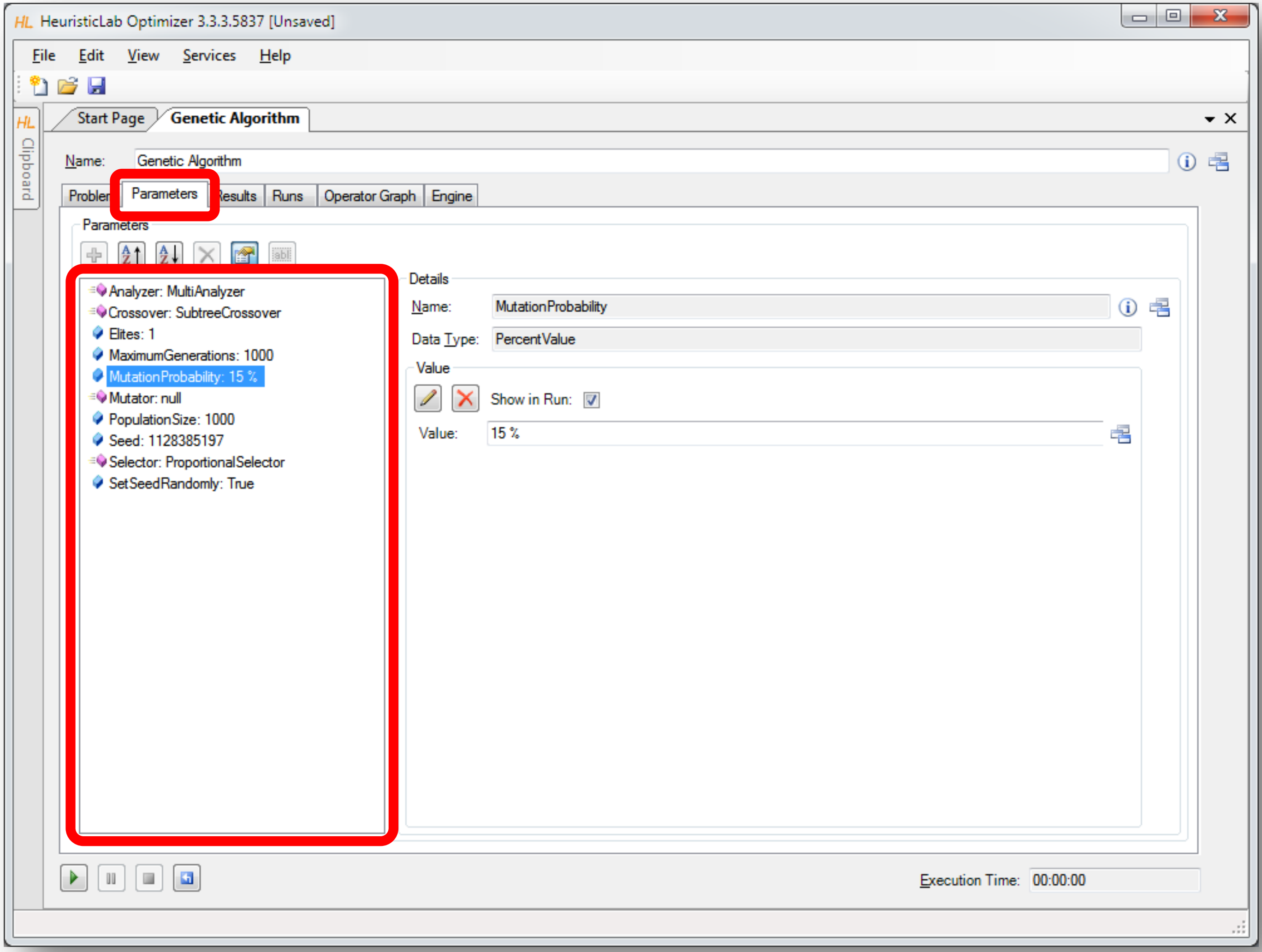
Configure Function Set (Grammar)



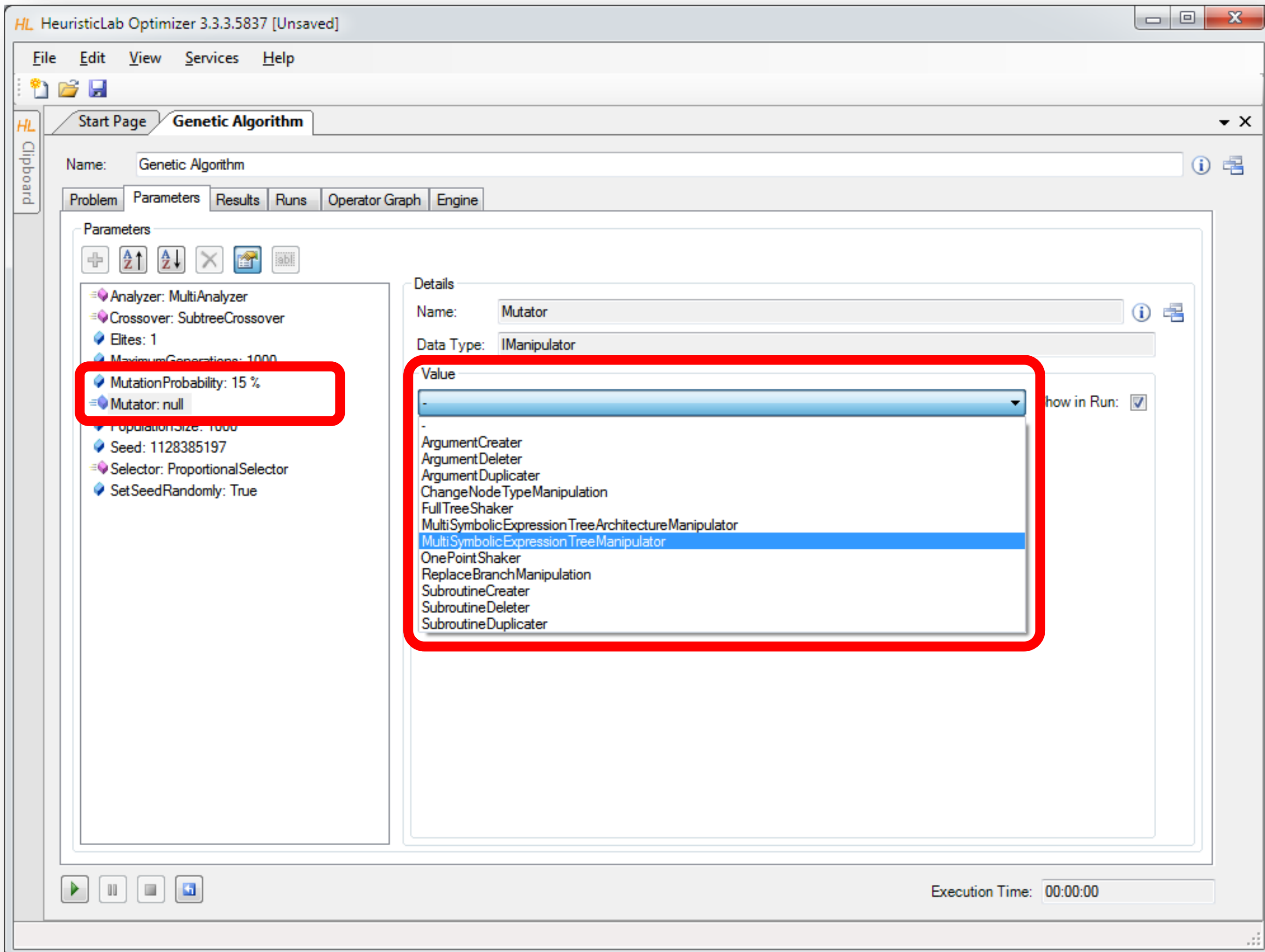
Configure Function Set (Grammar)



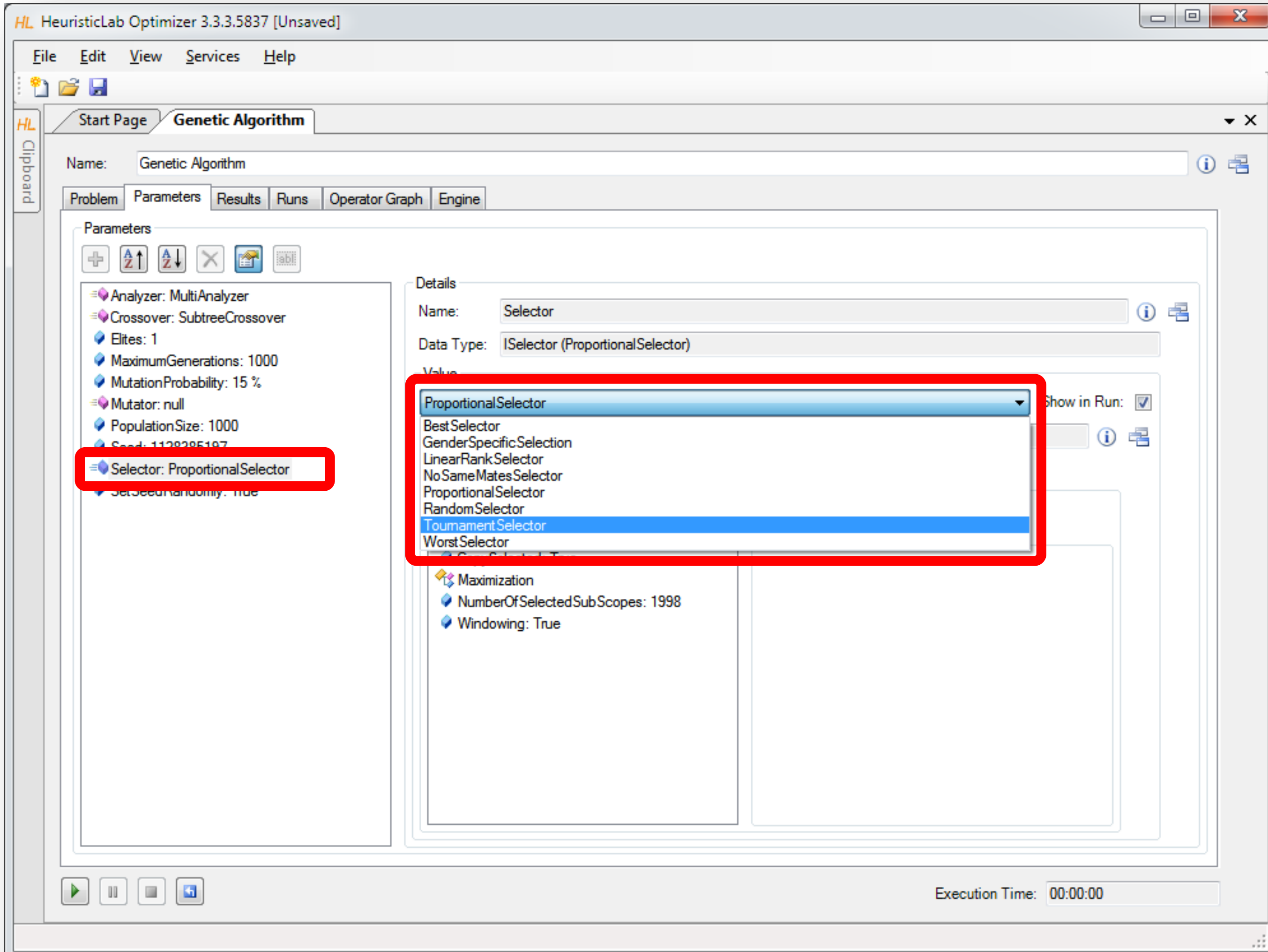
Configure Algorithm Parameters



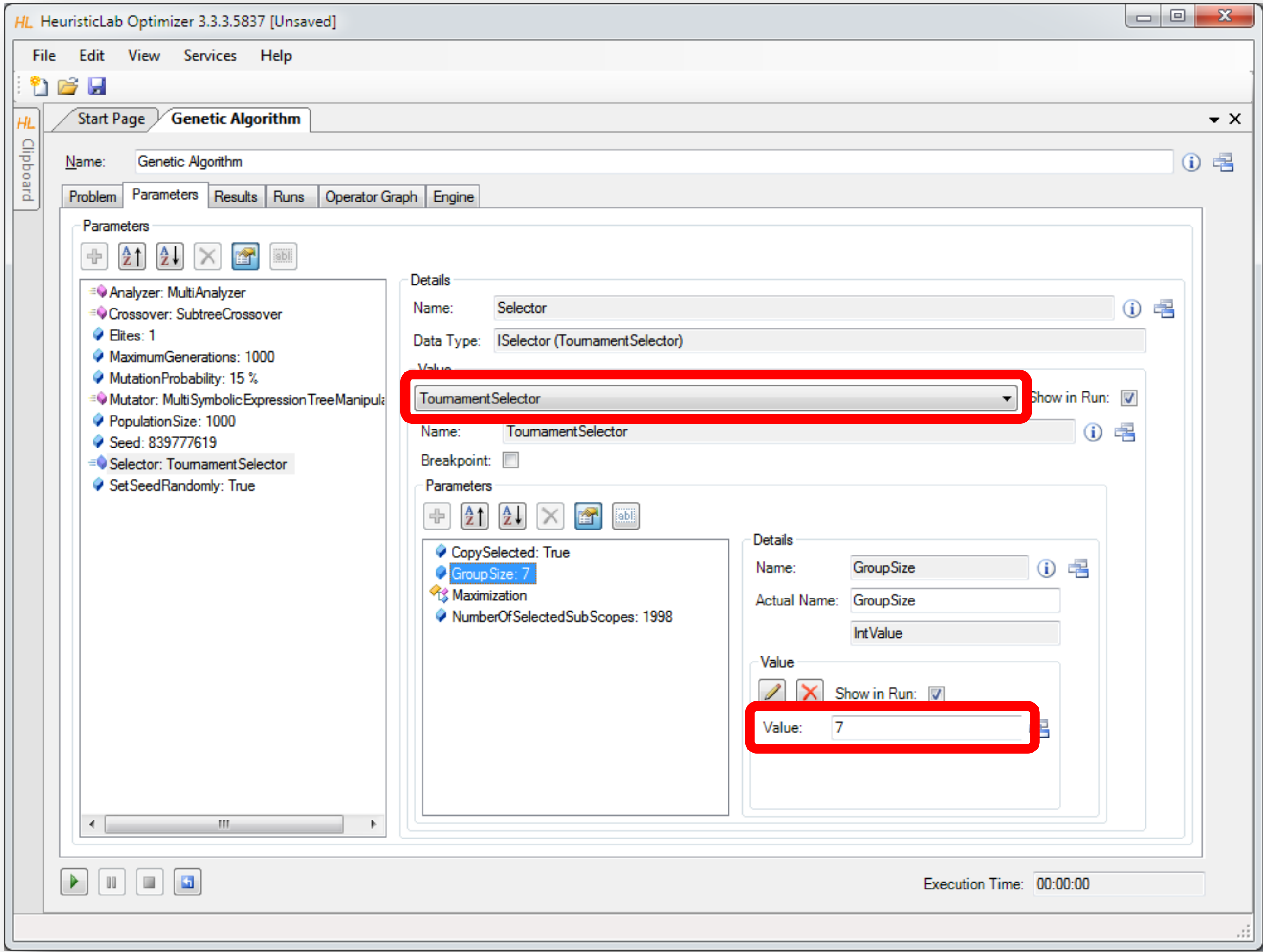
Configure Mutation Operator



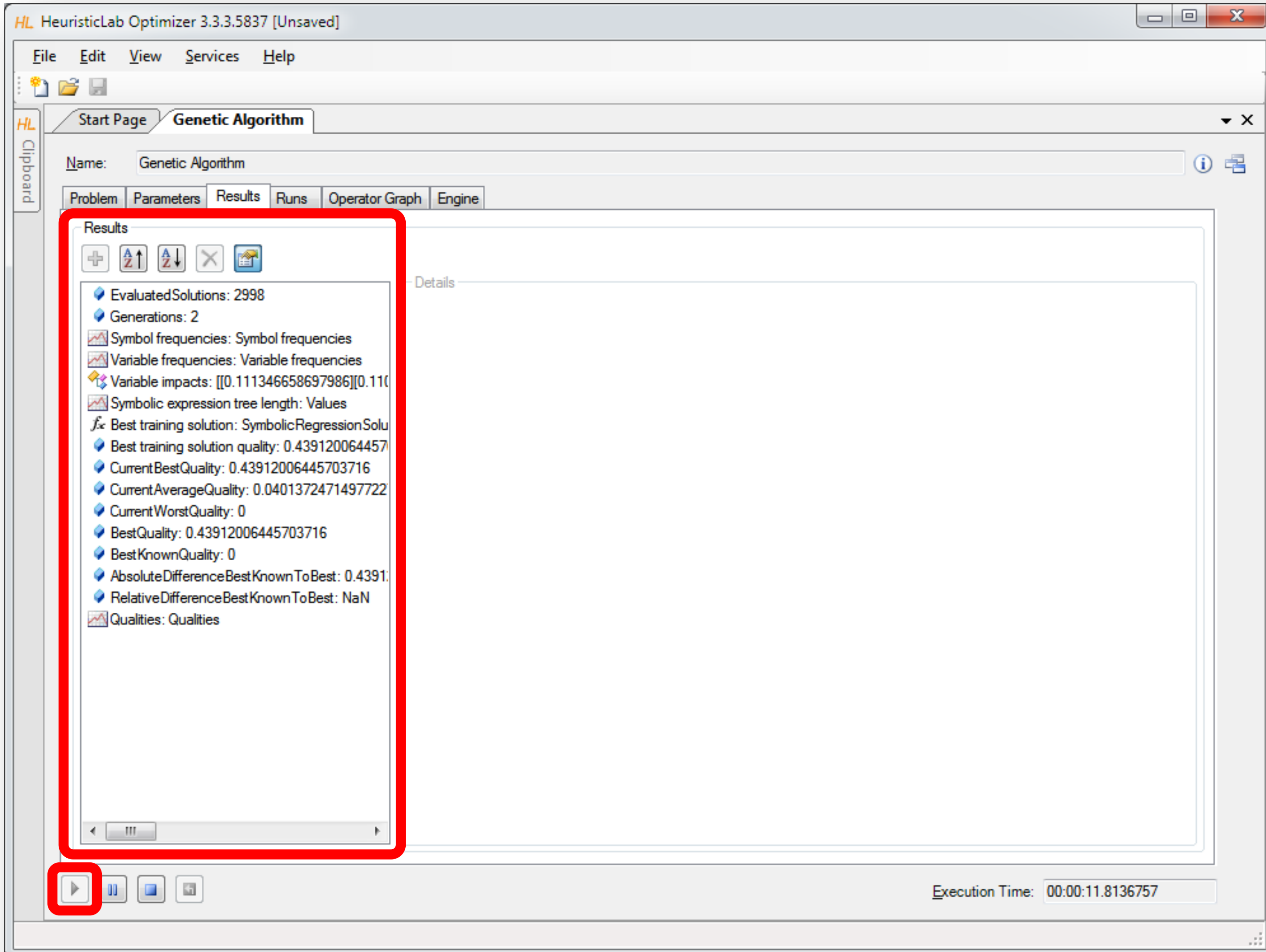
Configure Selection Operator



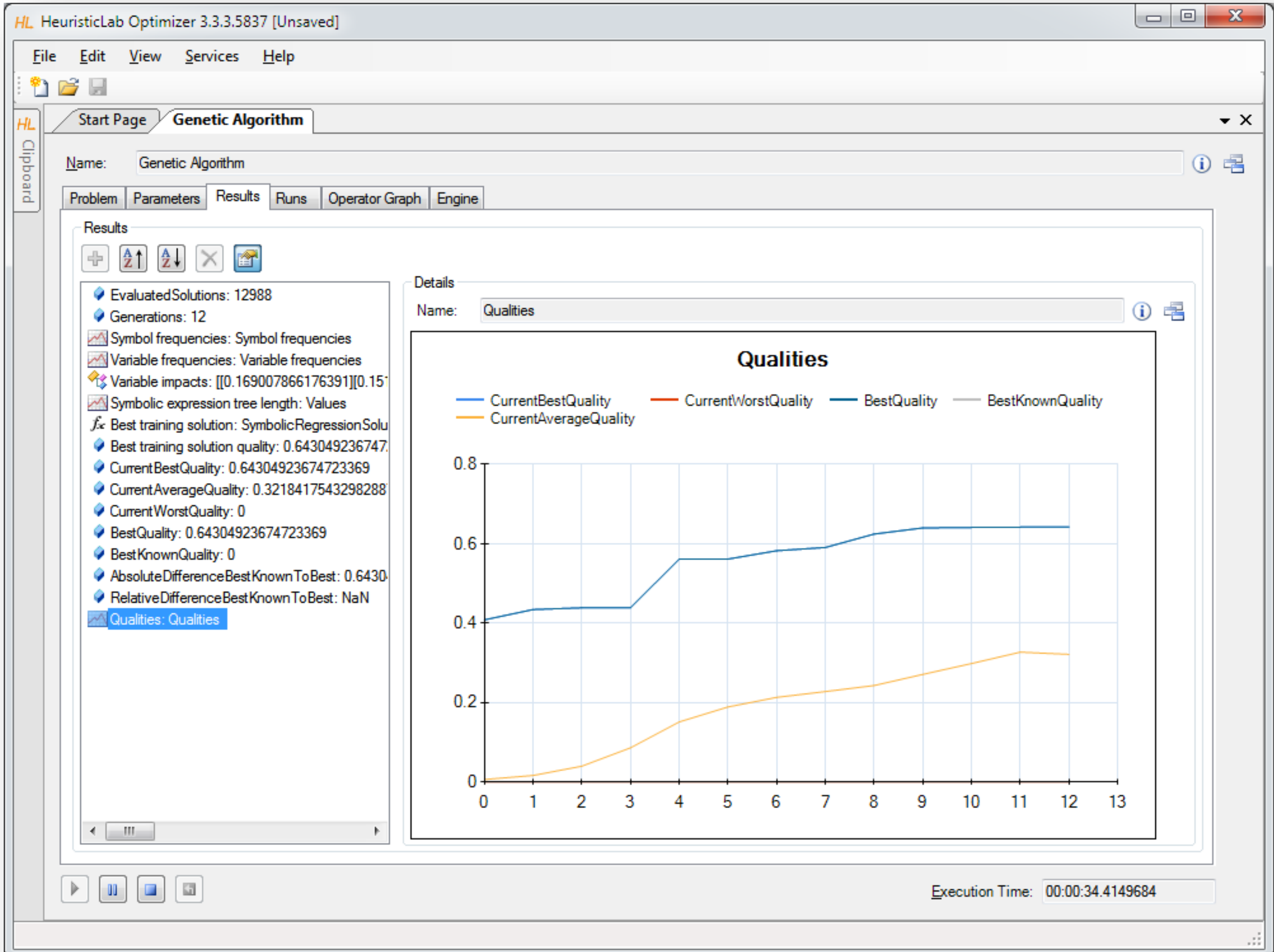
Configure Tournament Group Size



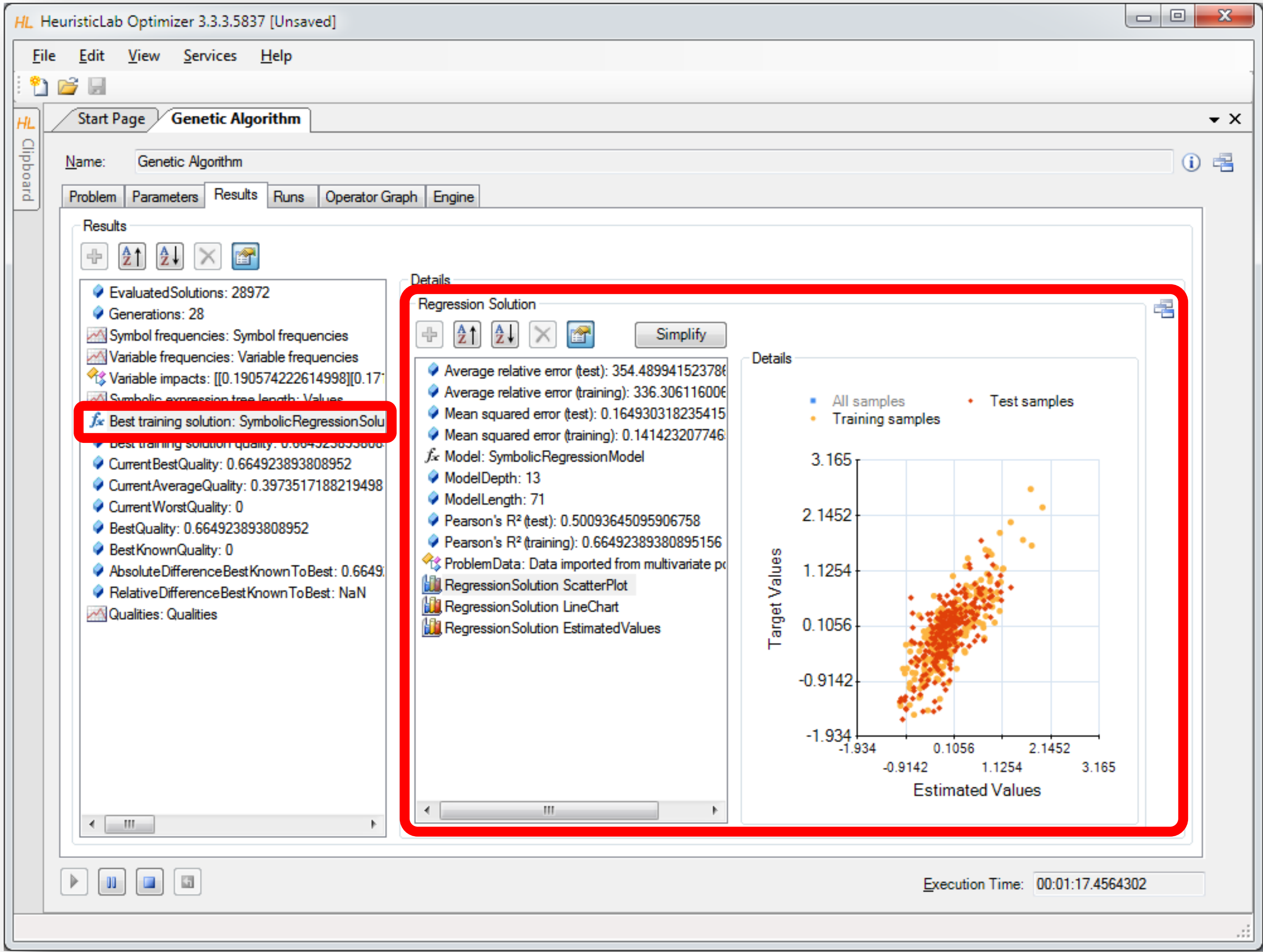
Start Algorithm and Inspect Results



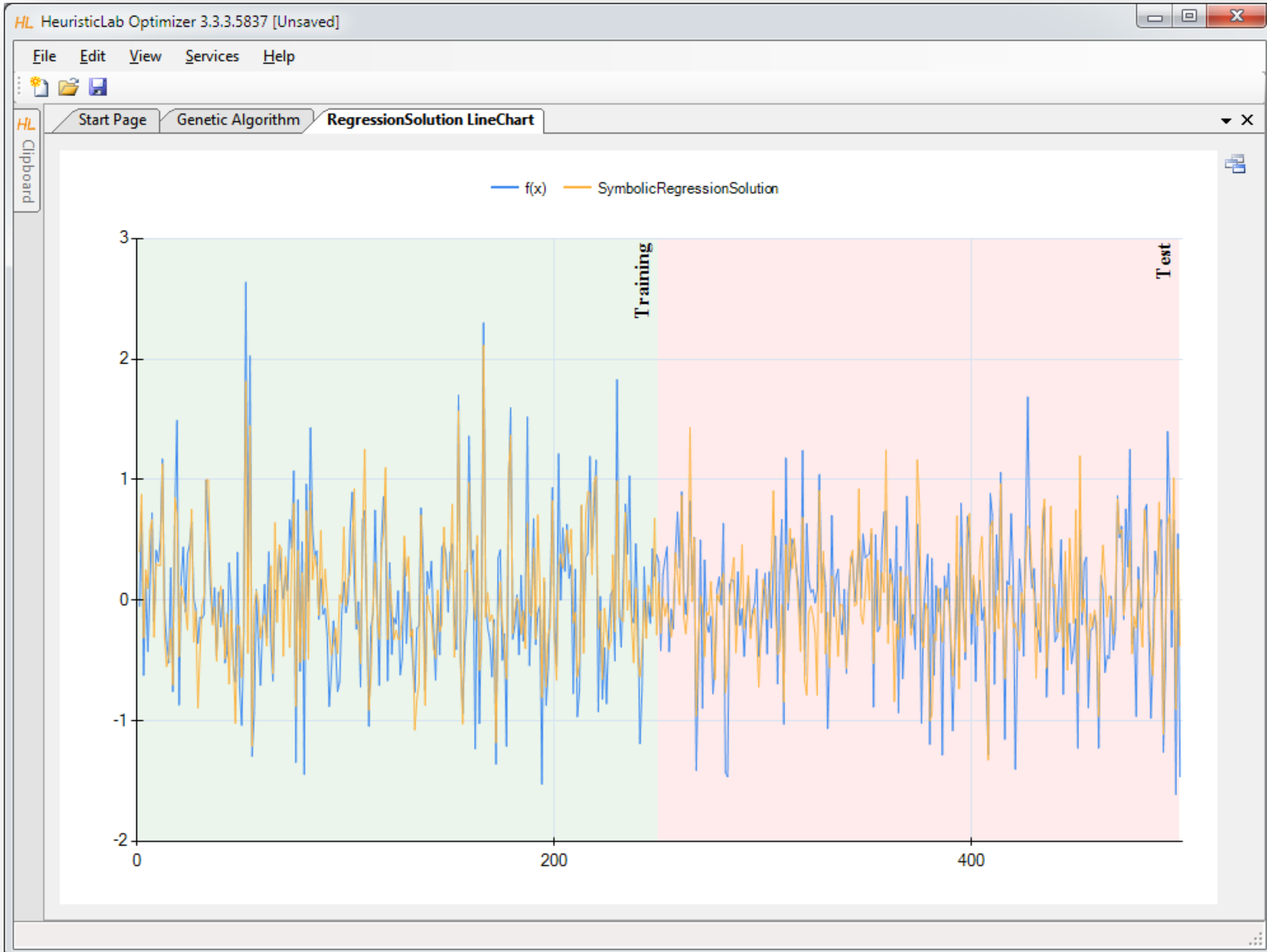
Inspect Quality Chart



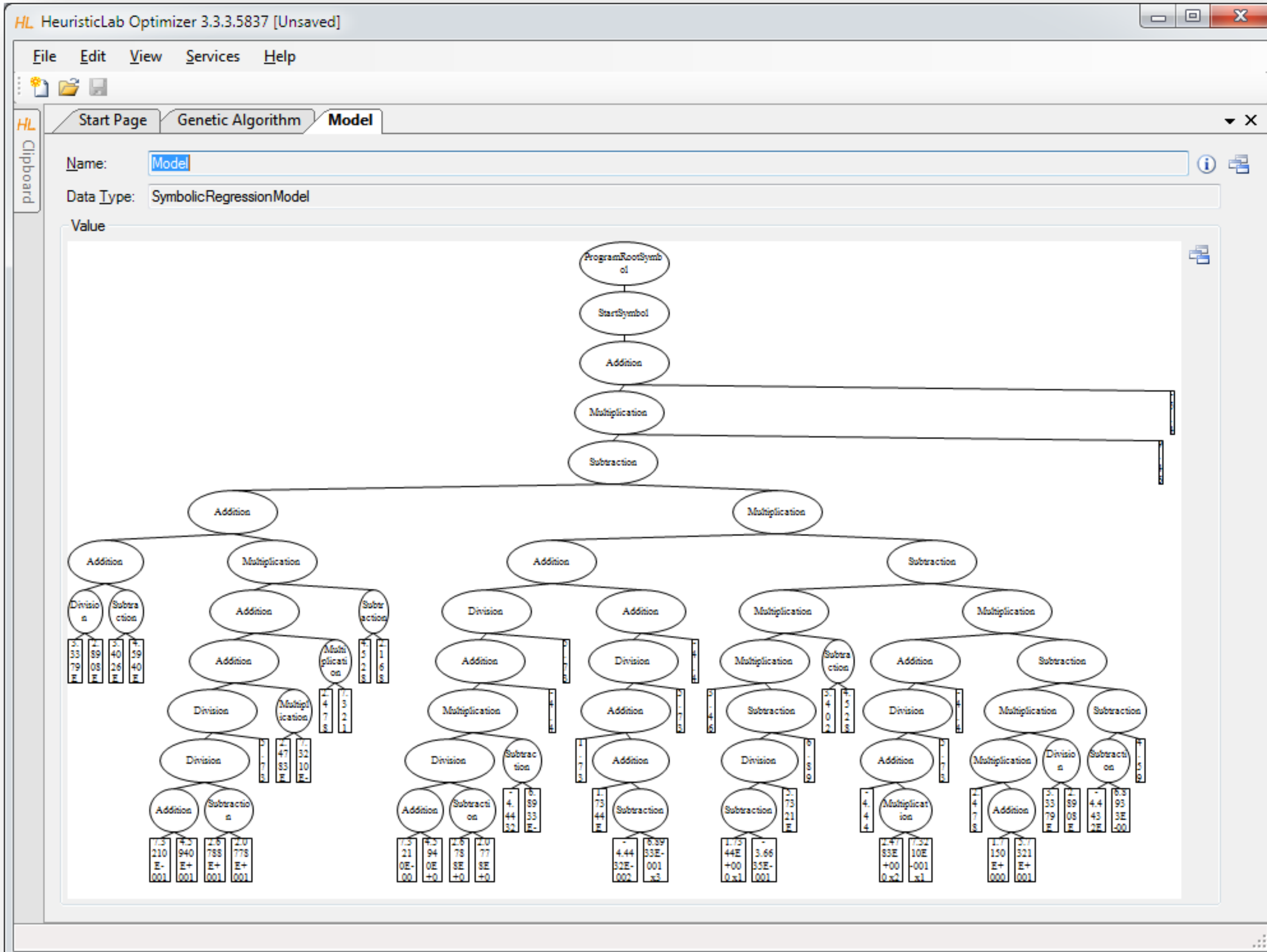
Inspect Best Model on Training Partition



Inspect Linechart of Best Model on Training Partition

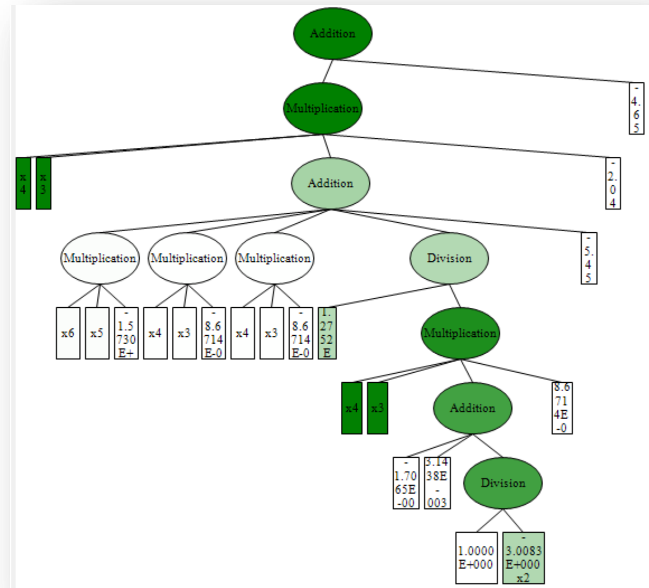


Inspect Structure of Best Model on Training Partition



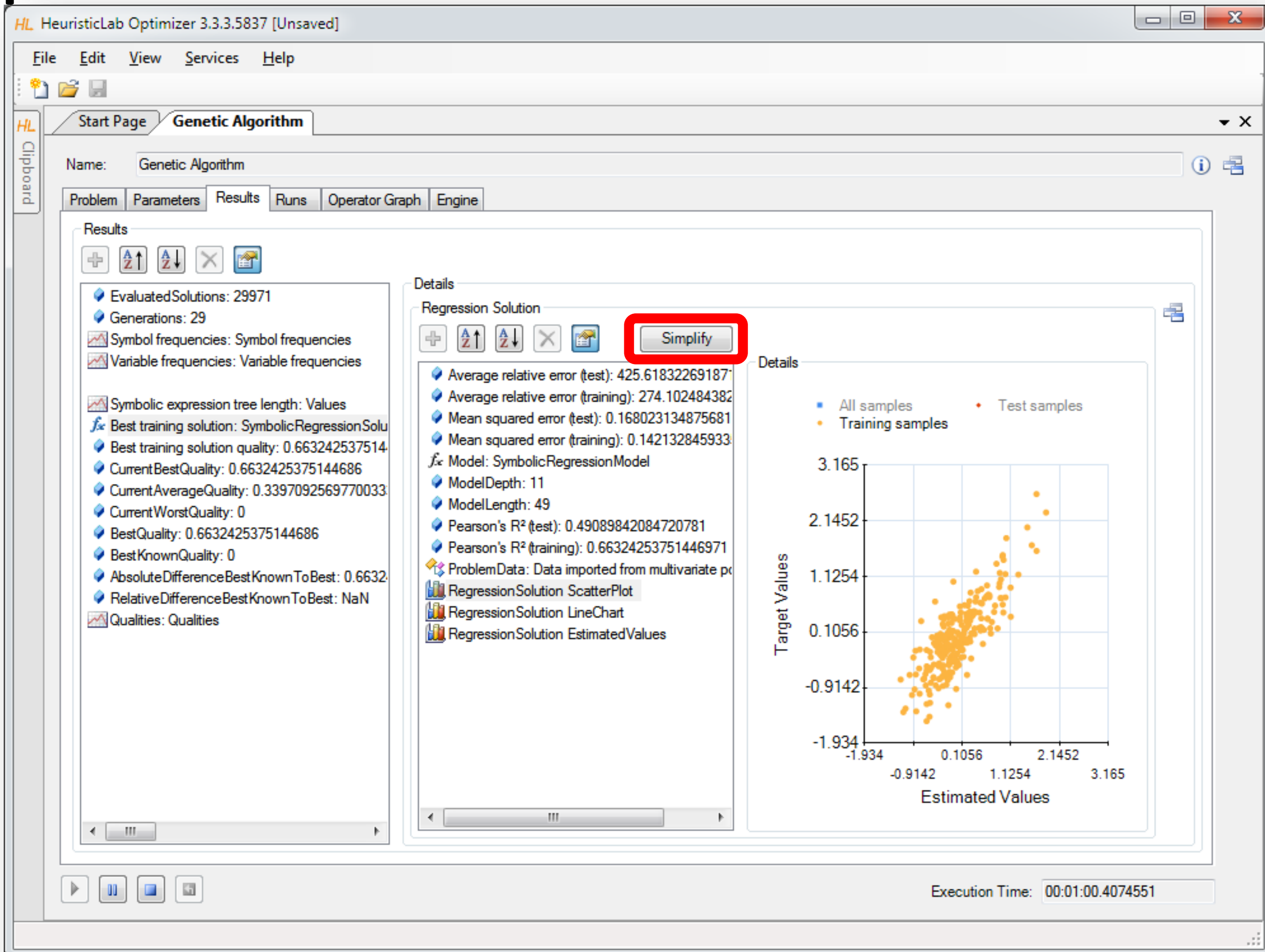
Model Simplification and Export

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

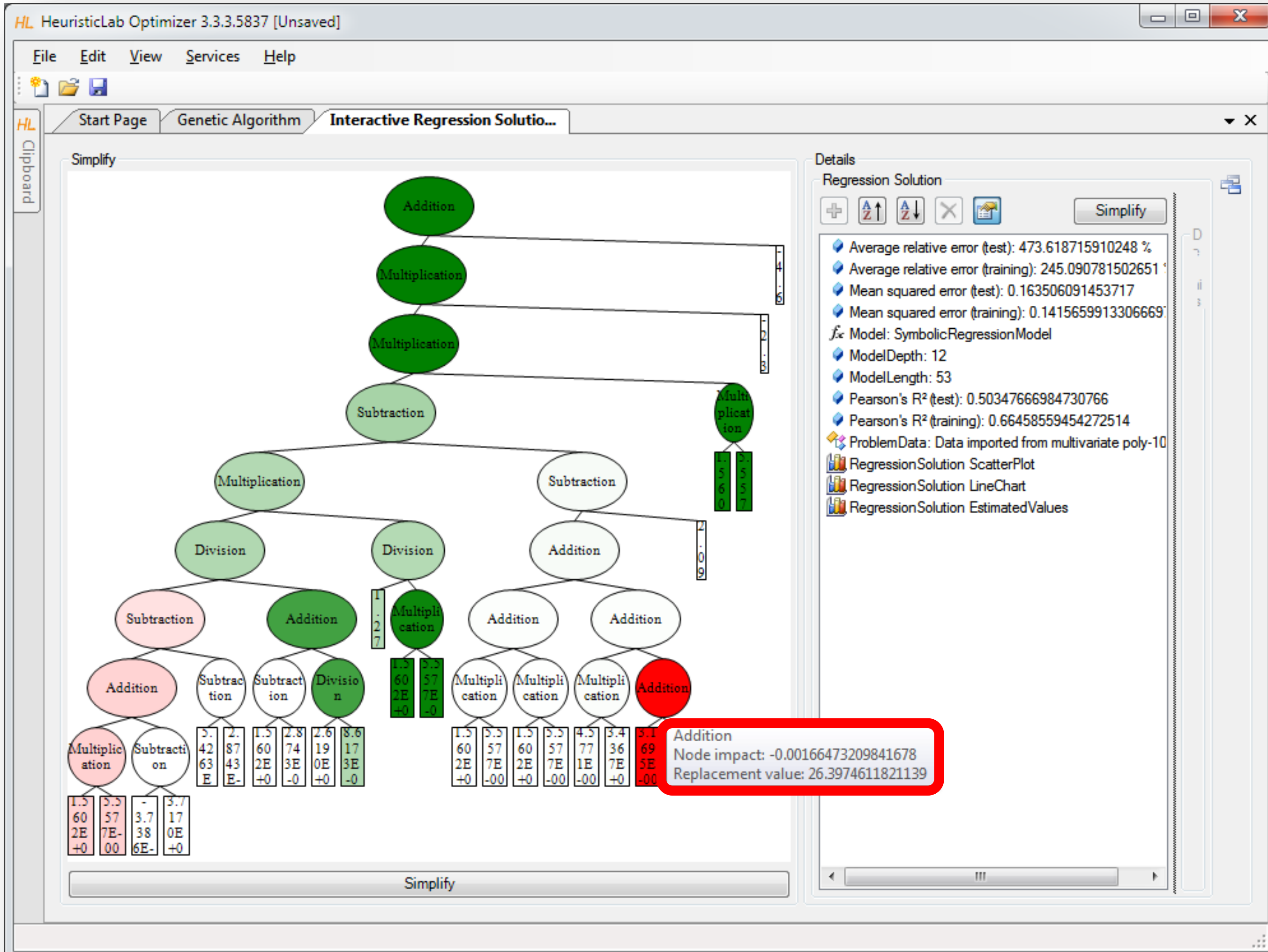


$$
\begin{aligned}
Result =\ & x4(t)\cdot x3(t)\cdot c_{20} && (13)\\
& \cdot\left(x6(t)\cdot x5(t)\cdot c_4 + x4(t)\cdot x3(t)\cdot c_7 + x4(t)\cdot x3(t)\cdot c_{10} + \frac{c_{11}x1(t)}{x4(t)\cdot x3(t)\cdot\left(c_{14}x4(t)+c_{15}x5(t)+\frac{1}{c_{17}x2(t)}\right)\cdot c_{18}} + c_{19}\right) + c_{21} && (14)
\end{aligned}
$$

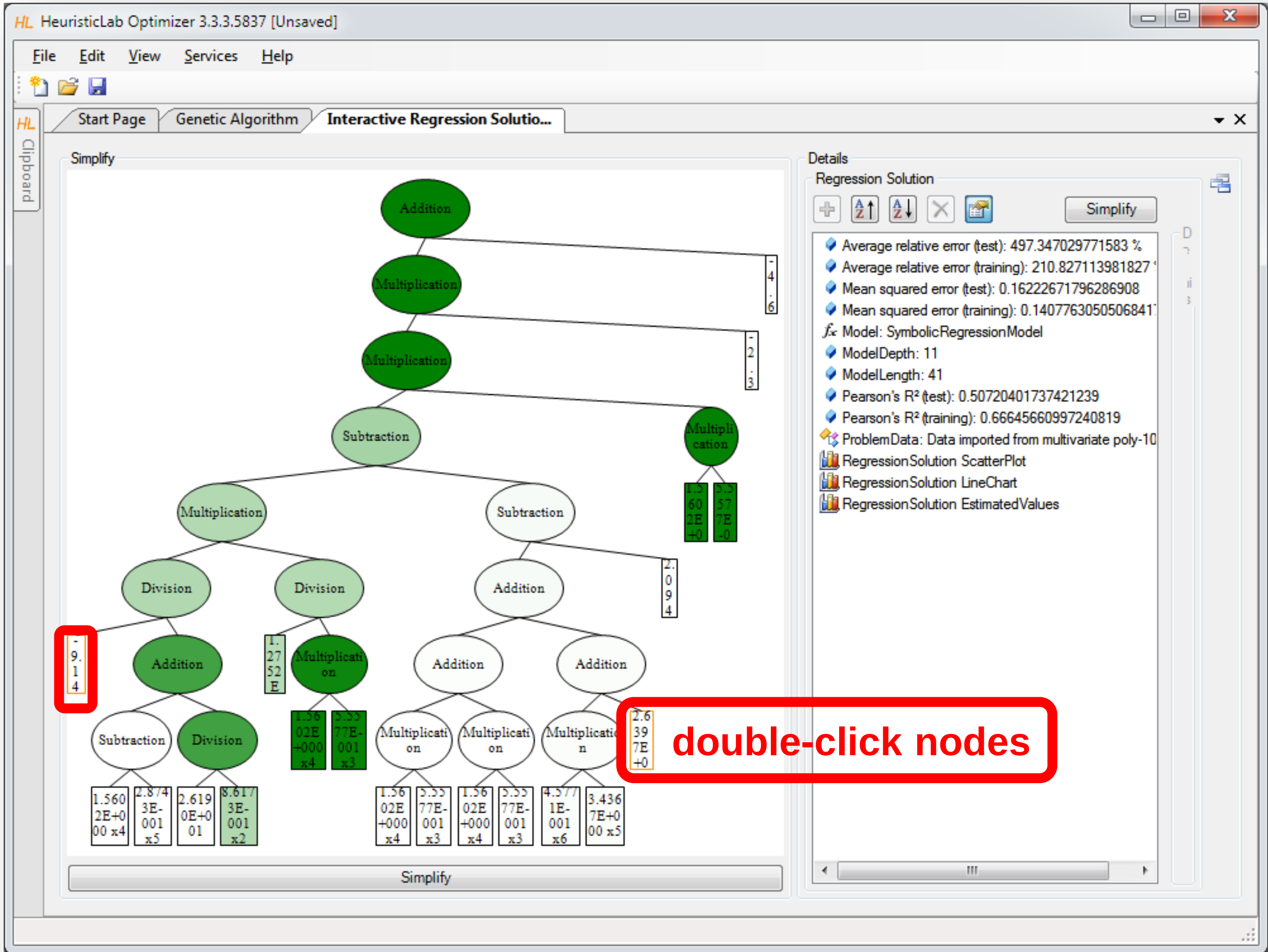
Detailed Model Analysis and Simplification



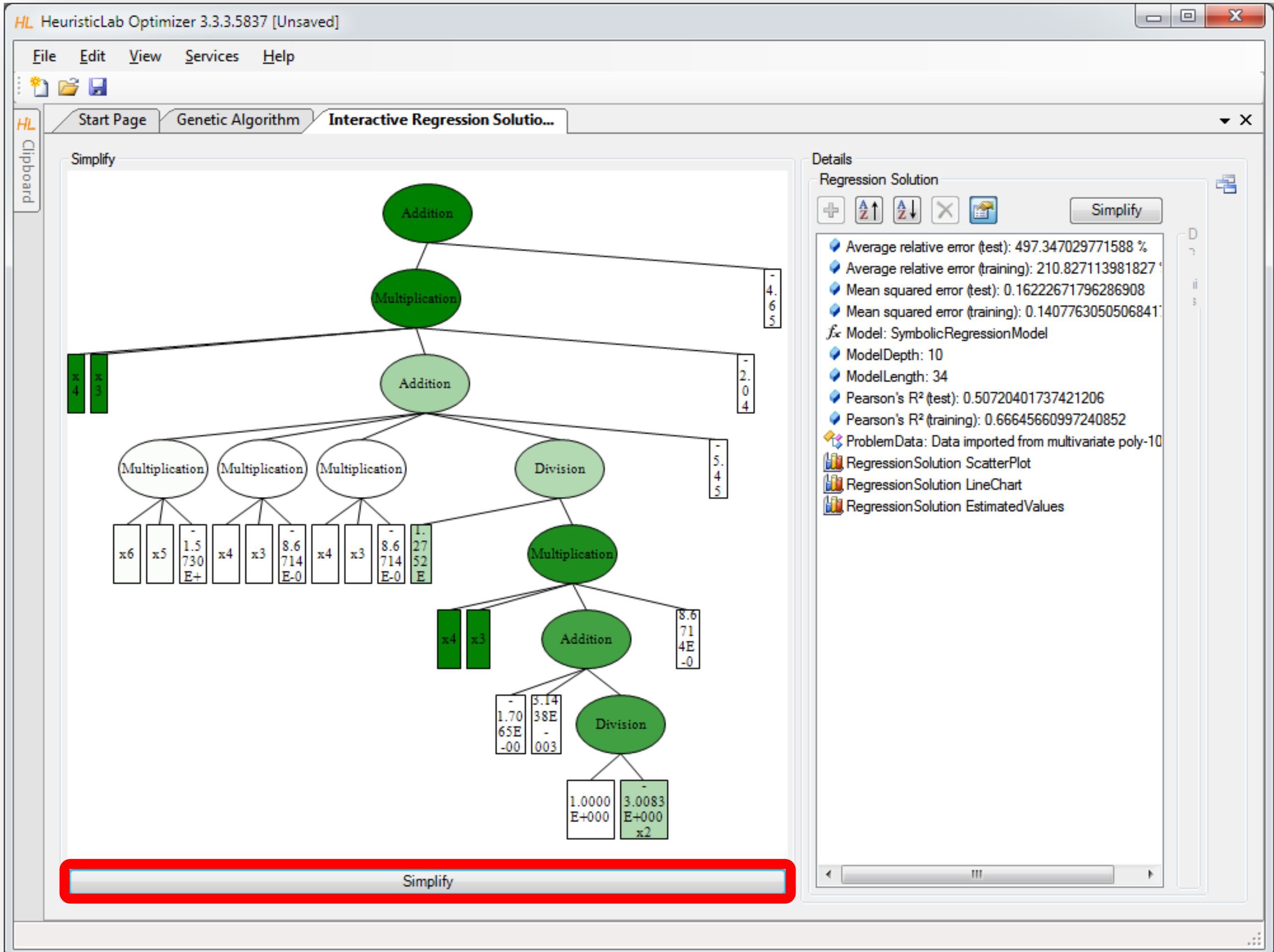
Symbolic Simplification and Node Impacts



Manual Simplification

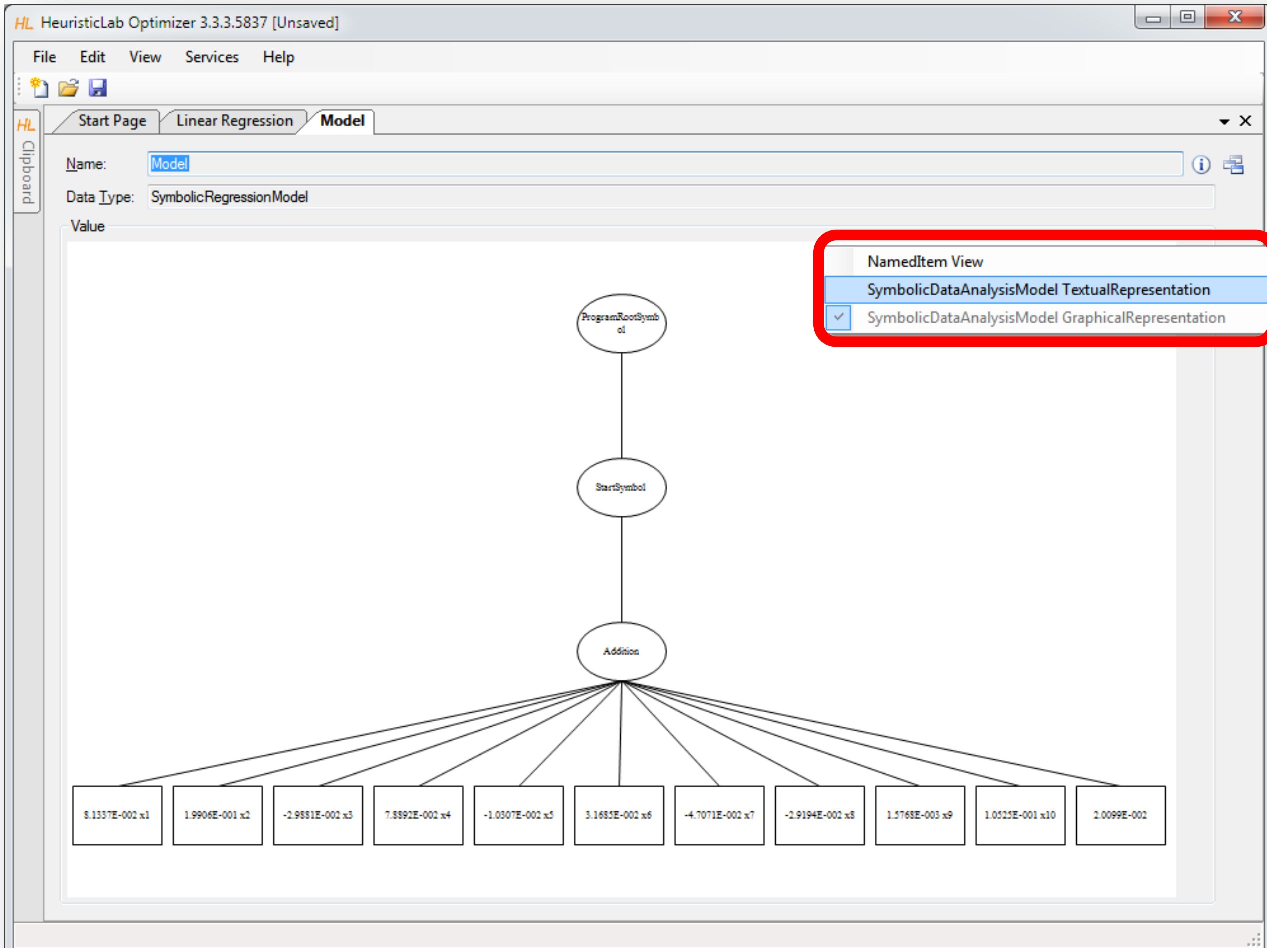


Automatic Symbolic Simplification

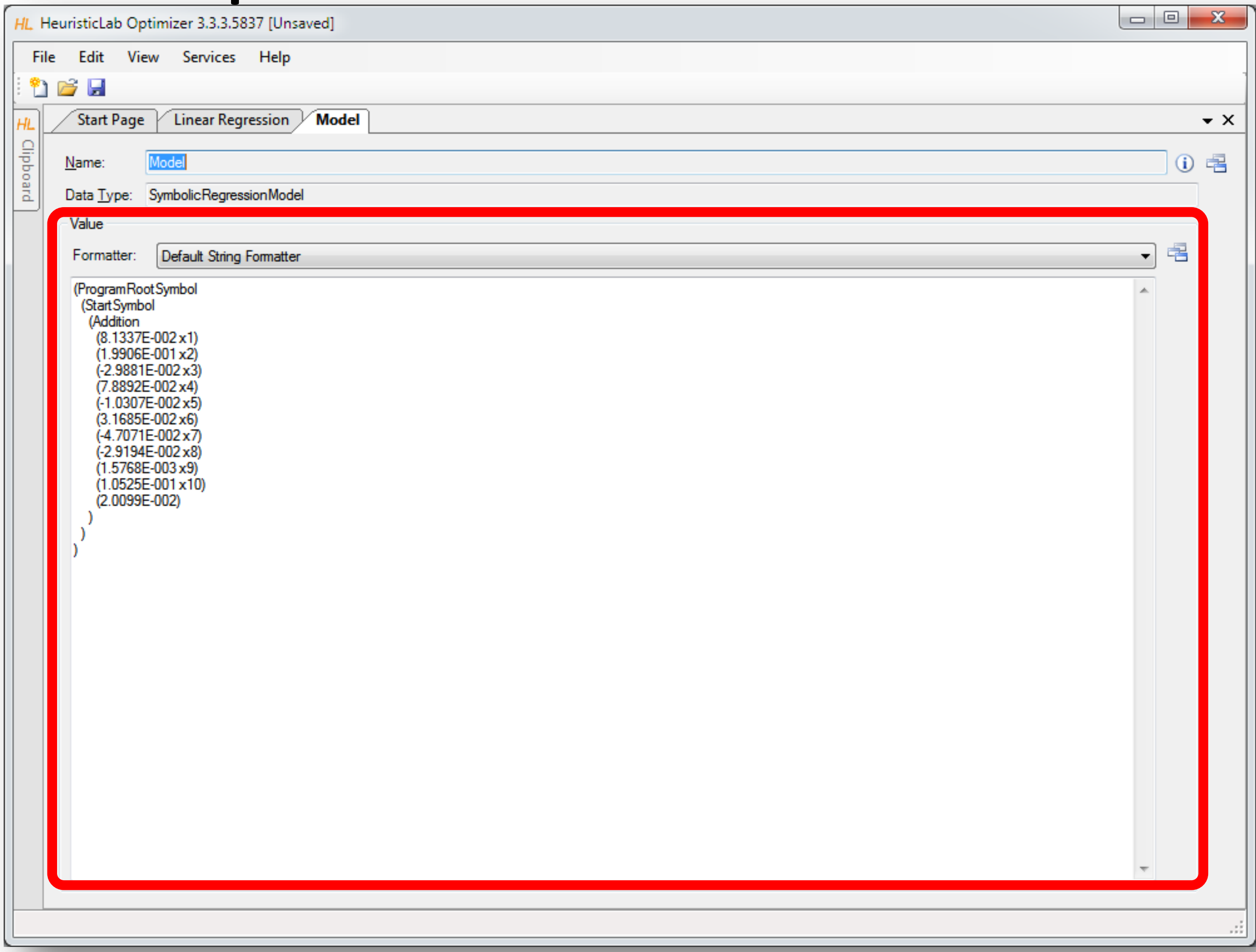


Textual Representations Are Also Available

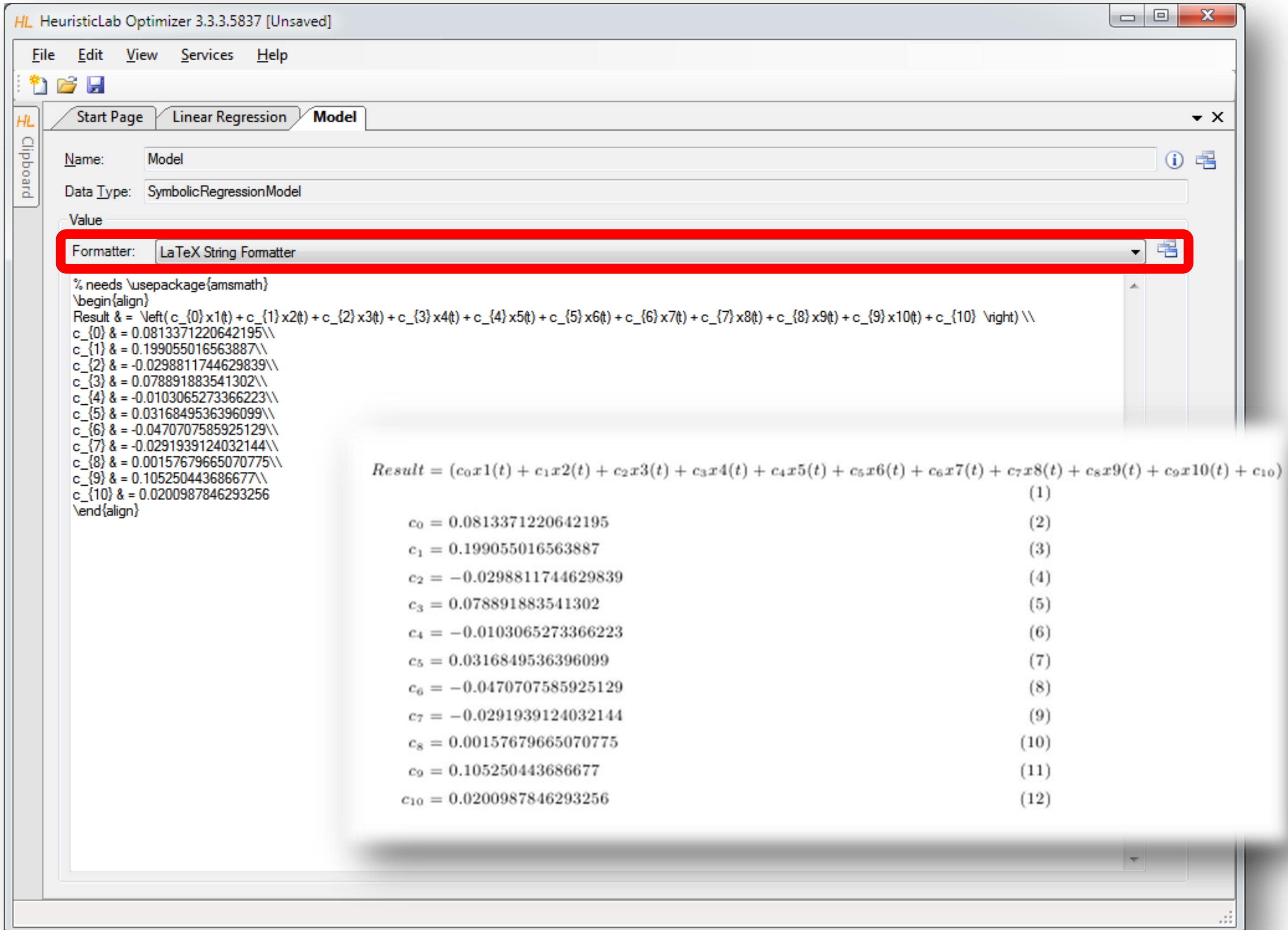
- Use *ViewHost* to switch to textual representation view.



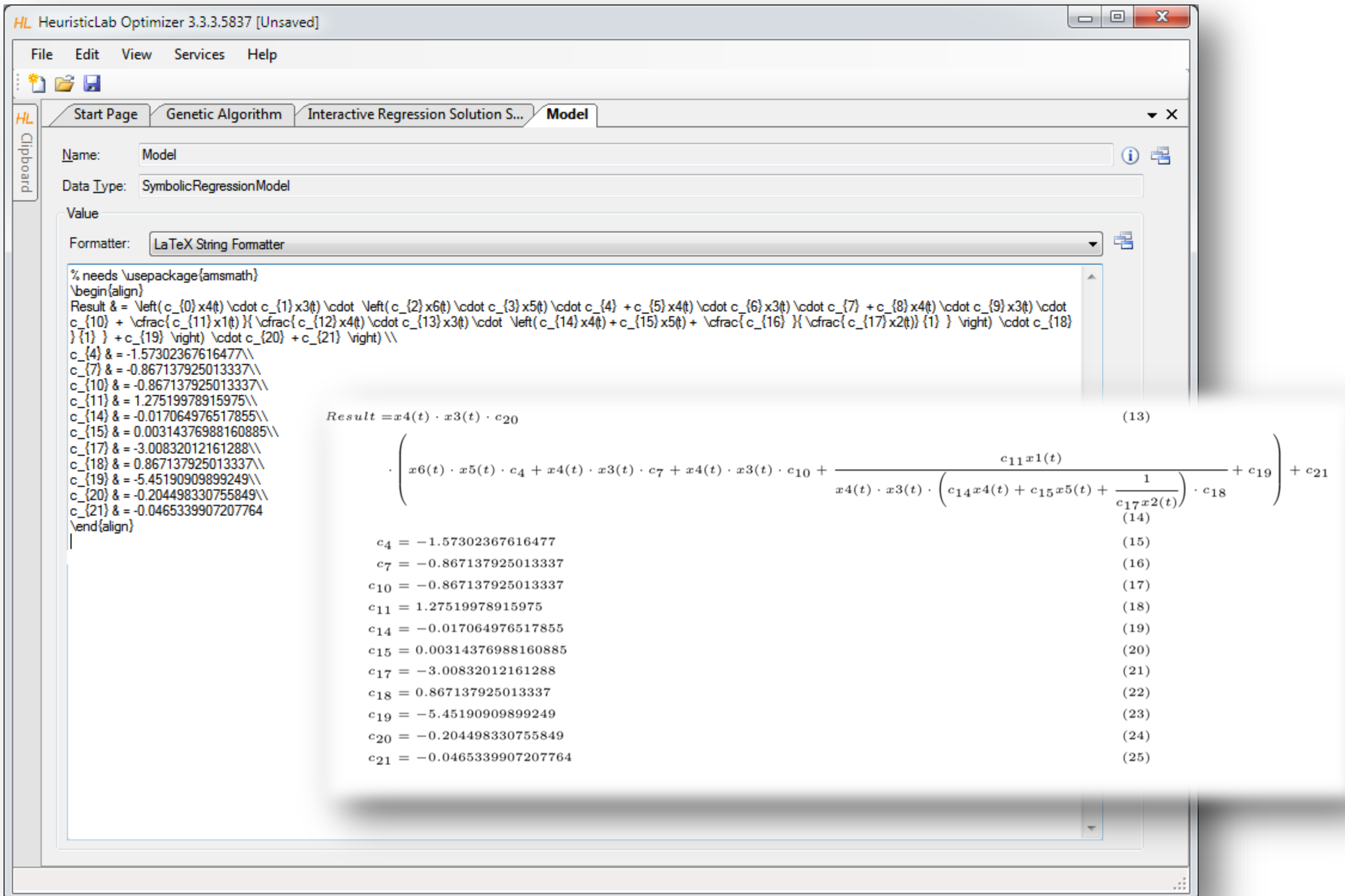
Default Textual Representation for Model Export



Textual Representation for Export to LaTeX

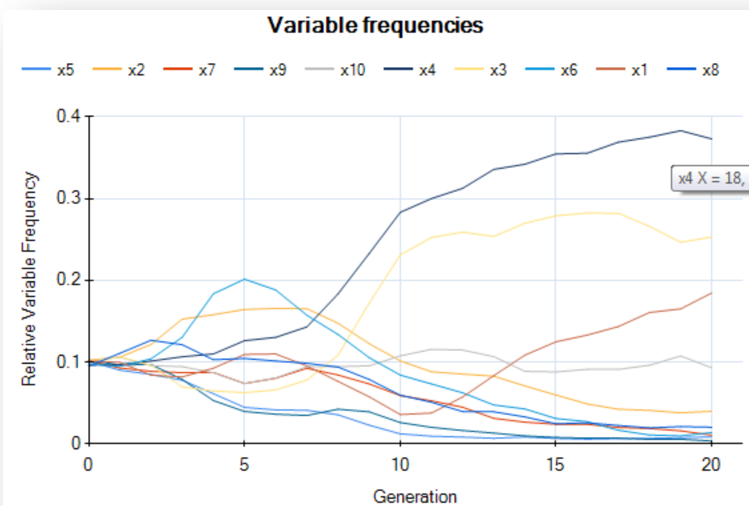


LaTeX Export



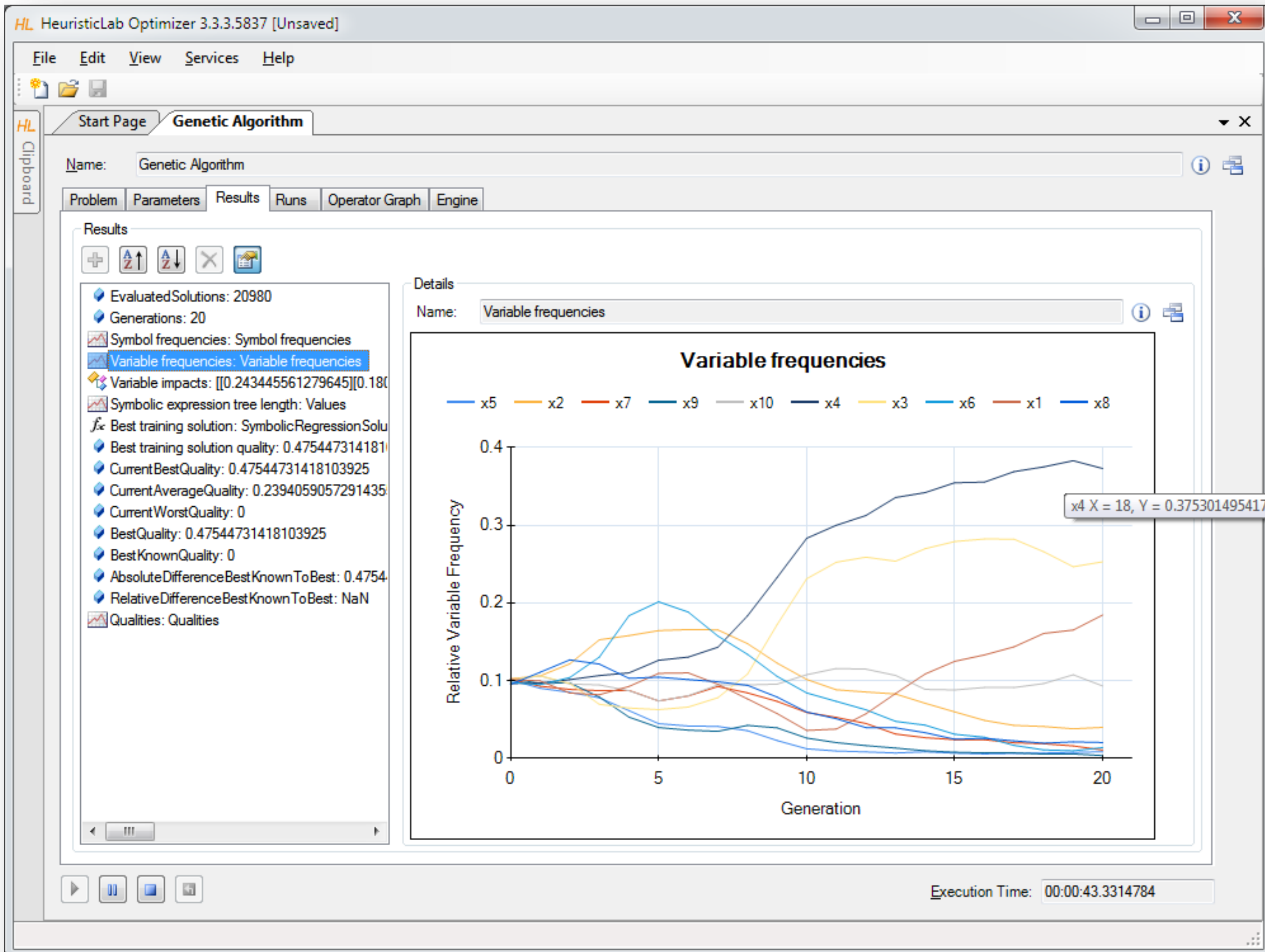
Variable Relevance Analysis

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

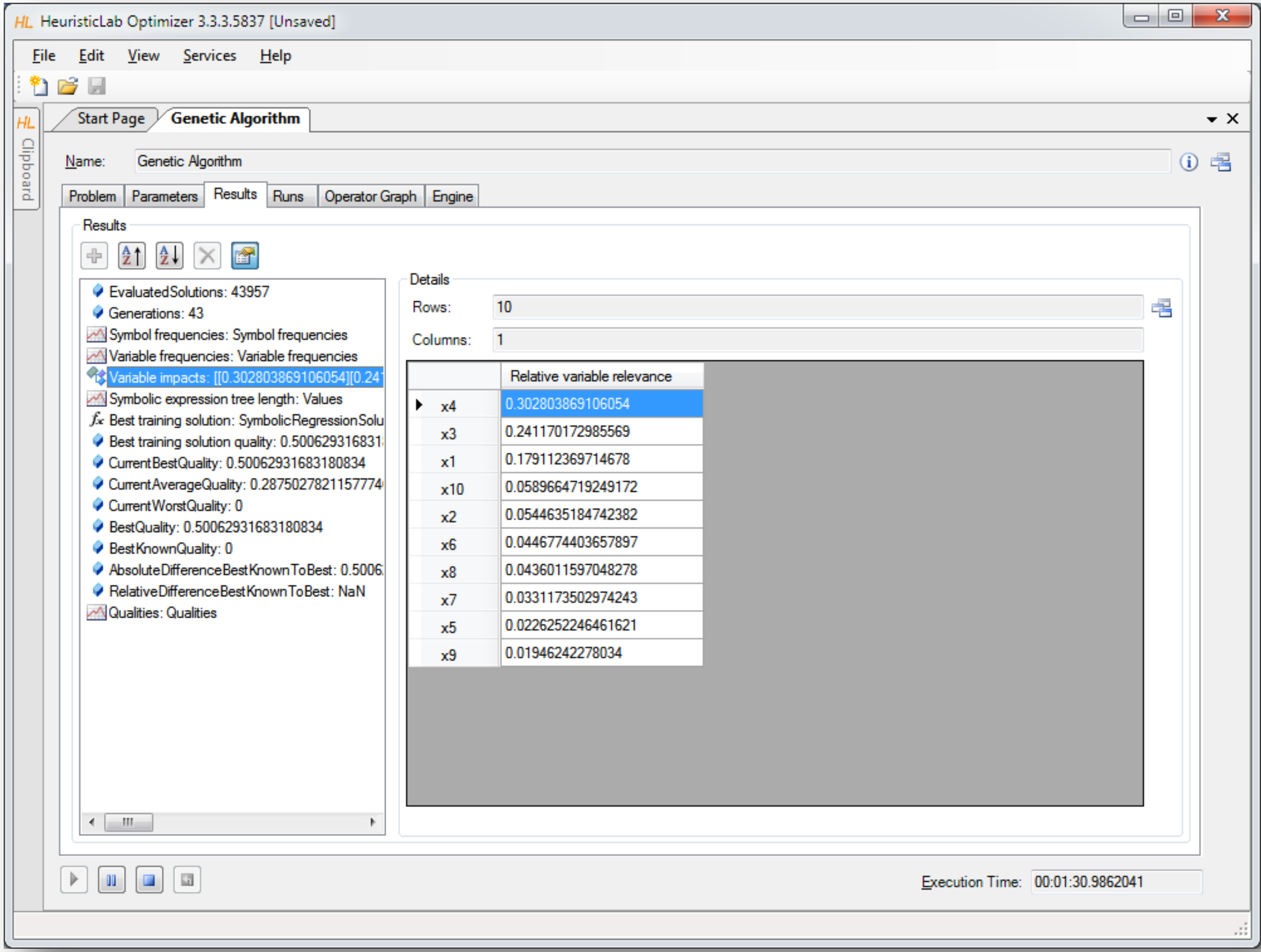


	Relative variable relevance
x4	0.302803869106054
x3	0.241170172985569
x1	0.179112369714678
x10	0.0589664719249172
x2	0.0544635184742382
x6	0.0446774403657897
x8	0.0436011597048278
x7	0.0331173502974243
x5	0.0226252246461621
x9	0.01946242278034

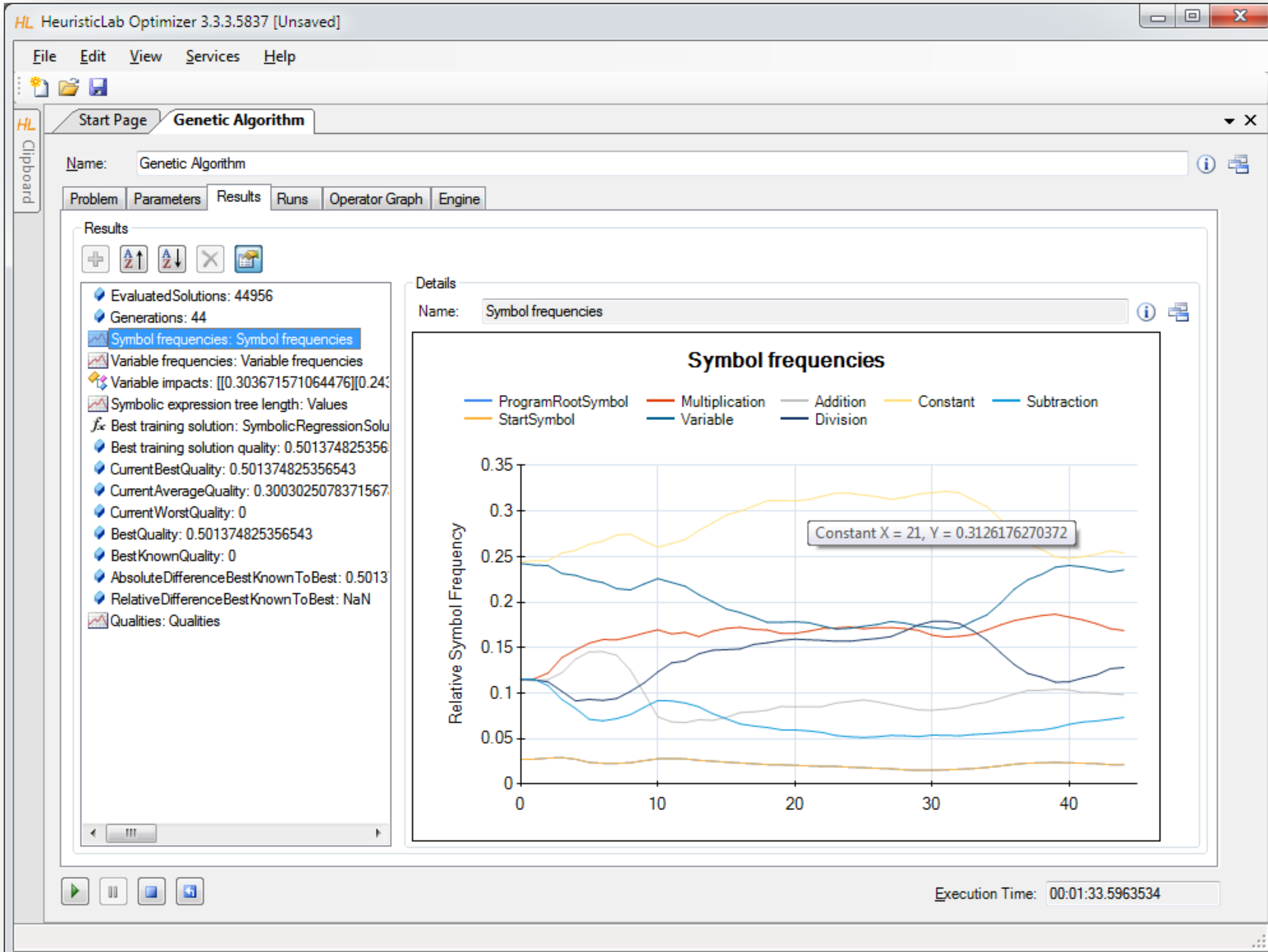
Inspect Variable Frequency Chart



Inspect Variable Impacts

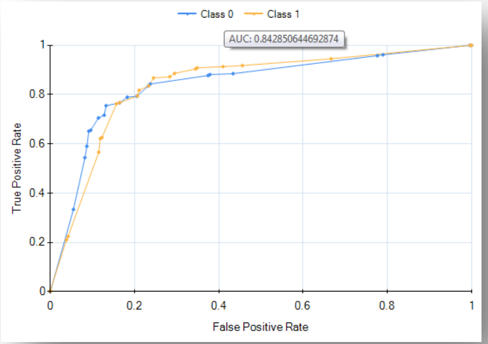


Inspect Symbol Frequencies



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

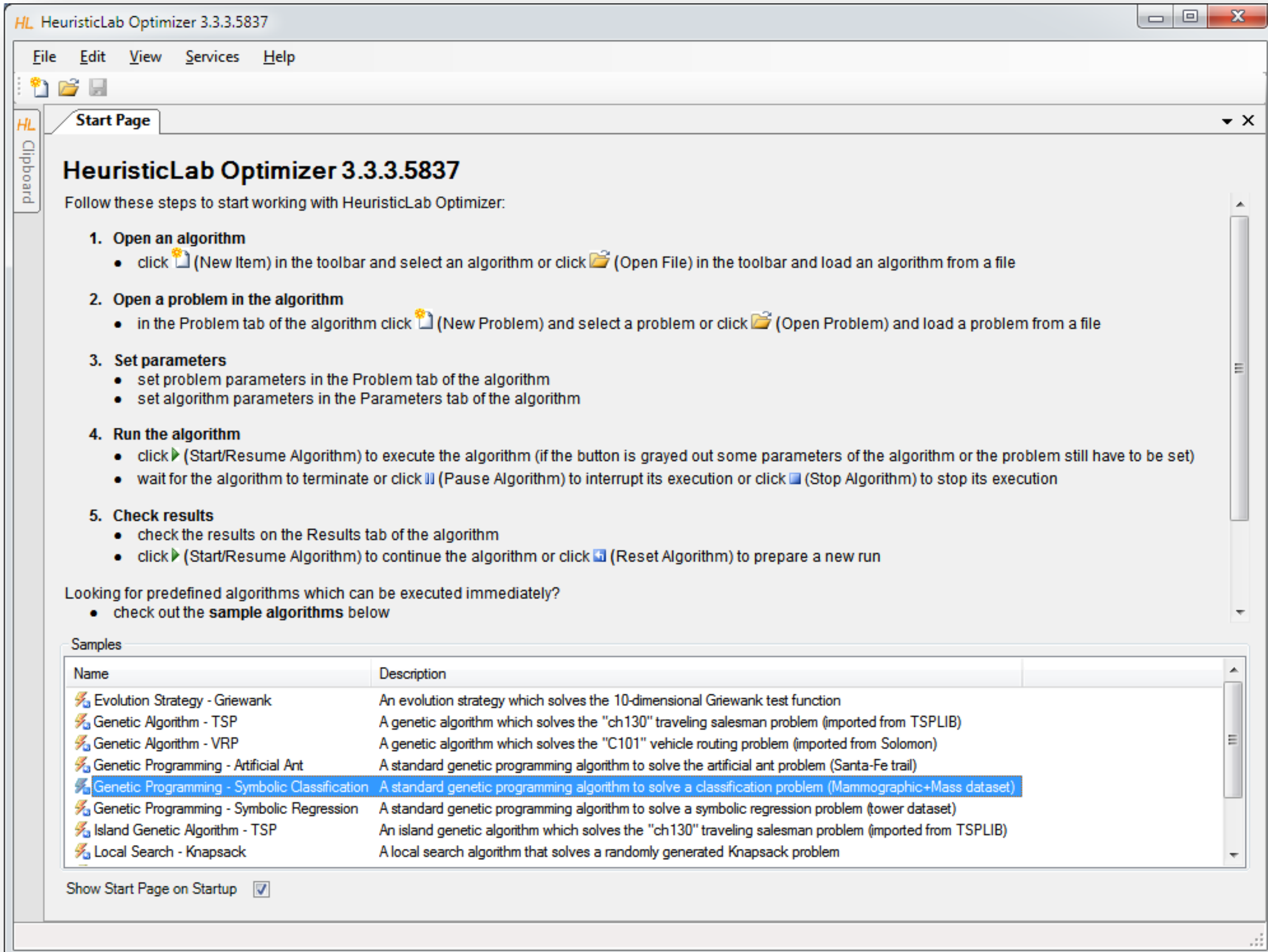


	Actual Class 0	Actual Class 1
Predicted Class 0	197	29
Predicted Class 1	64	190

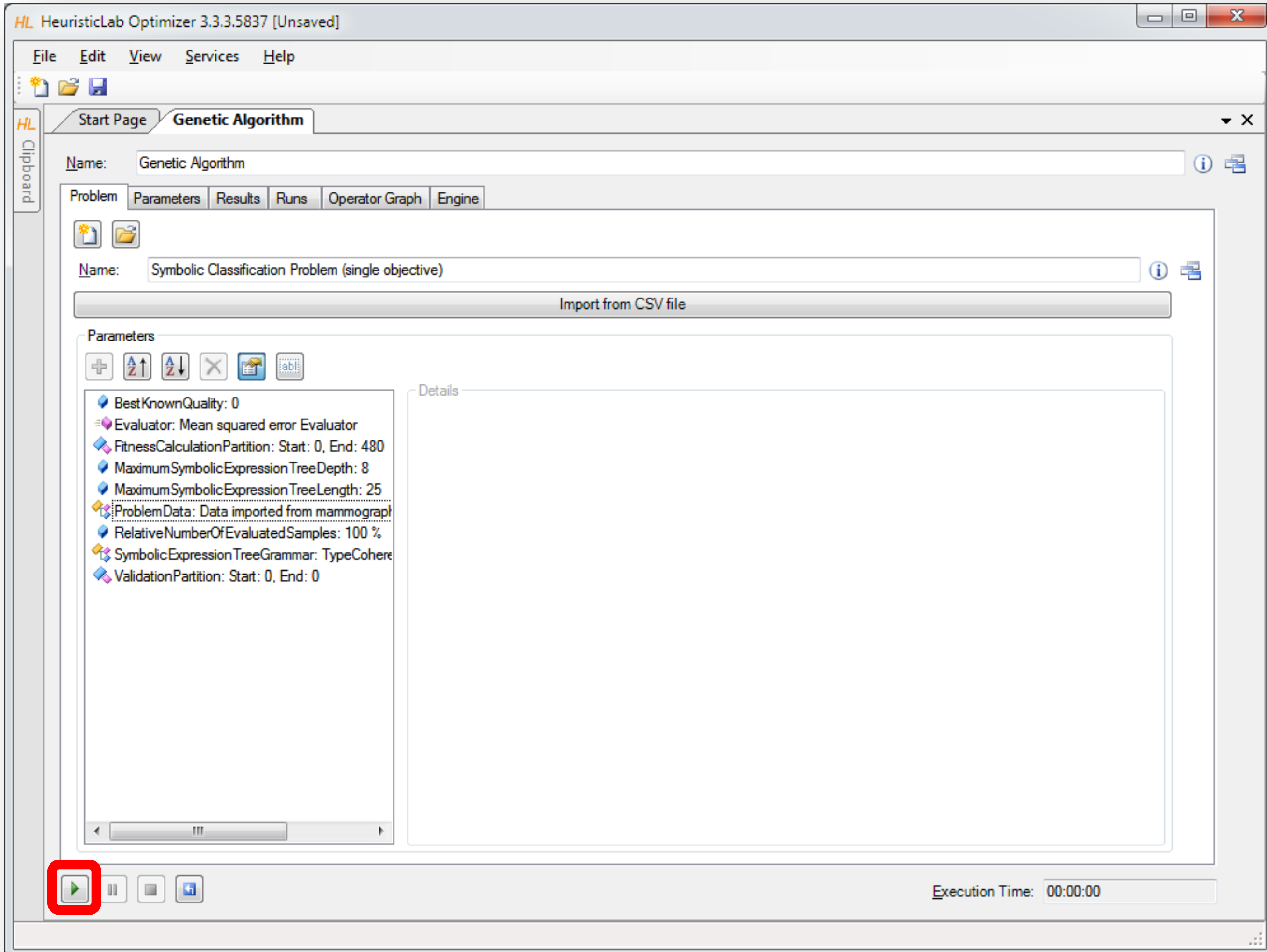
Case Study: Classification

- 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#GECCO2012

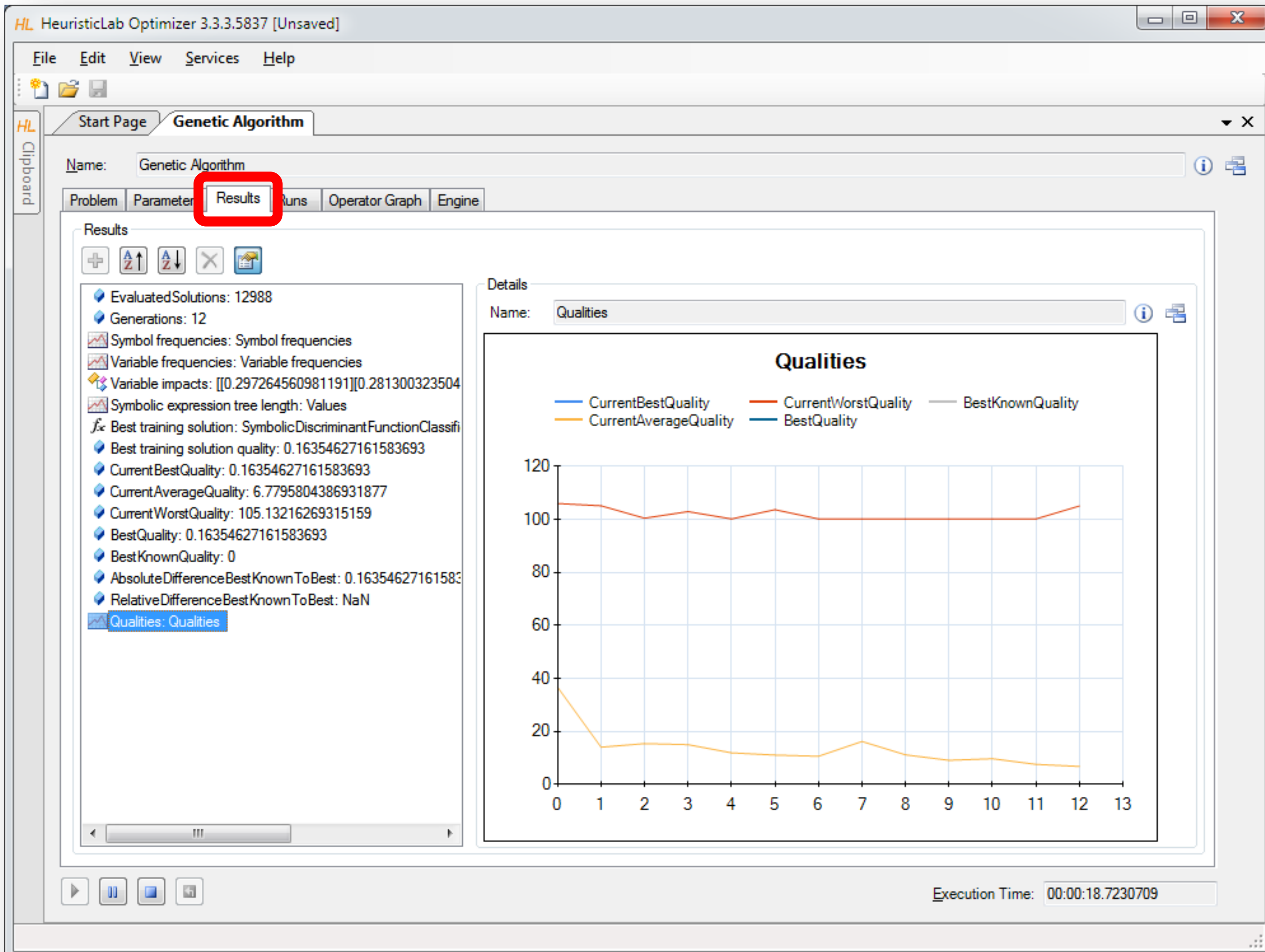
Open Sample


HeuristicLab Optimizer 3.3.3.5837
File Edit View Services Help
Clipboard
Start Page
HeuristicLab Optimizer 3.3.3.5837
Follow these steps to start working with HeuristicLab Optimizer:
1. Open an algorithm
click (New Item) in the toolbar and select an algorithm or click (Open File) in the toolbar and load an algorithm from a file
2. Open a problem in the algorithm
in the Problem tab of the algorithm click (New Problem) and select a problem or click (Open Problem) and load a problem from a file
3. Set parameters
set problem parameters in the Problem tab of the algorithm
set algorithm parameters in the Parameters tab of the algorithm
4. Run the algorithm
click (Start/Resume Algorithm) to execute the algorithm (if the button is grayed out some parameters of the algorithm or the problem still have to be set)
wait for the algorithm to terminate or click (Pause Algorithm) to interrupt its execution or click (Stop Algorithm) to stop its execution
5. Check results
check the results on the Results tab of the algorithm
click (Start/Resume Algorithm) to continue the algorithm or click (Reset Algorithm) to prepare a new run
Looking for predefined algorithms which can be executed immediately?
check out the sample algorithms below
Samples
Name
Description
Evolution Strategy - Griewank
An evolution strategy which solves the 10-dimensional Griewank test function
Genetic Algorithm - TSP
A genetic algorithm which solves the "ch130" traveling salesman problem (imported from TSPLIB)
Genetic Algorithm - VRP
A genetic algorithm which solves the "C101" vehicle routing problem (imported from Solomon)
Genetic Programming - Artificial Ant
A standard genetic programming algorithm to solve the artificial ant problem (Santa-Fe trail)
Genetic Programming - Symbolic Classification
A standard genetic programming algorithm to solve a classification problem (Mammographic+Mass dataset)
Genetic Programming - Symbolic Regression
A standard genetic programming algorithm to solve a symbolic regression problem (tower dataset)
Island Genetic Algorithm - TSP
An island genetic algorithm which solves the "ch130" traveling salesman problem (imported from TSPLIB)
Local Search - Knapsack
A local search algorithm that solves a randomly generated Knapsack problem
Show Start Page on Startup

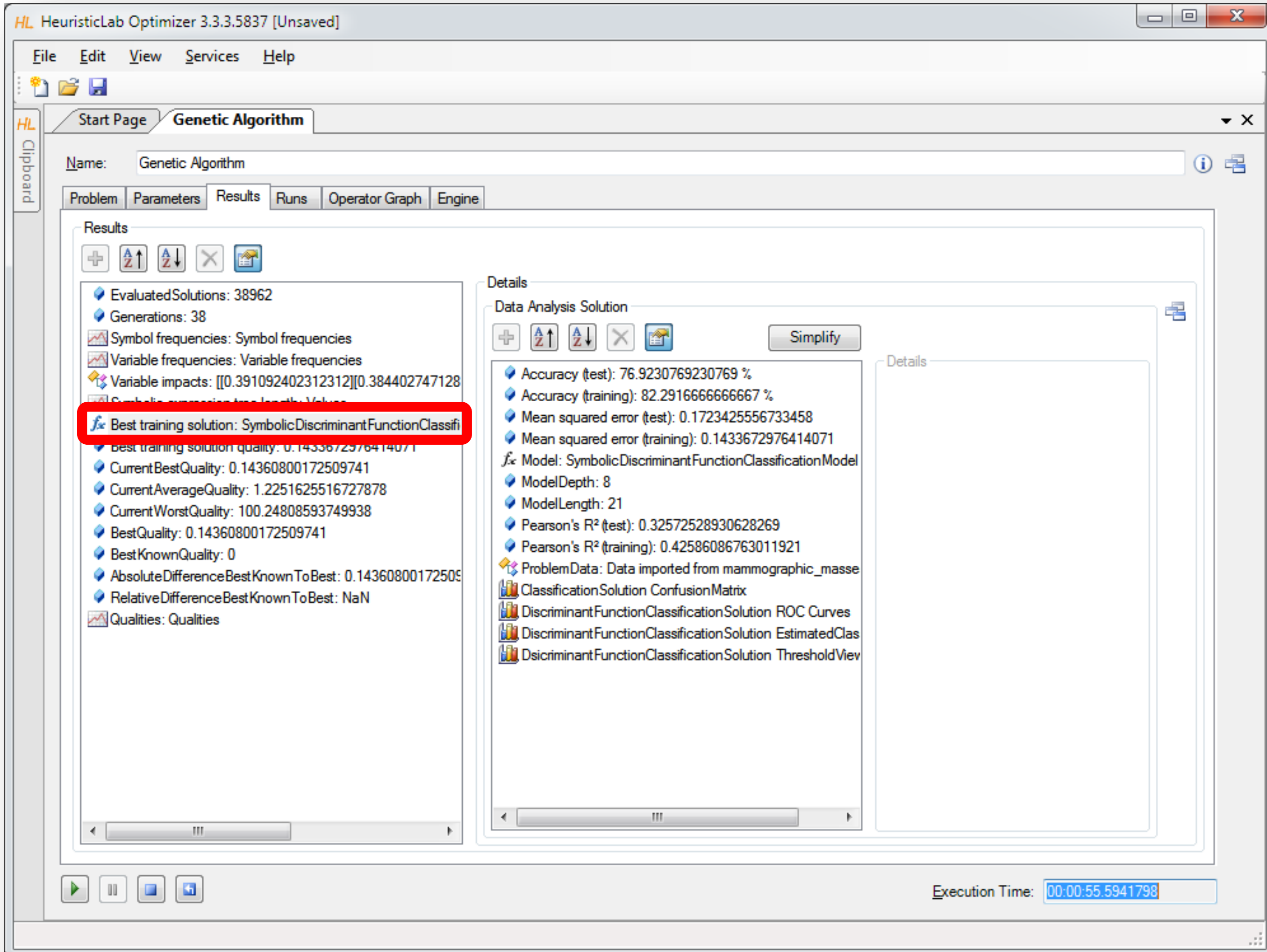
Configure and Run Algorithm



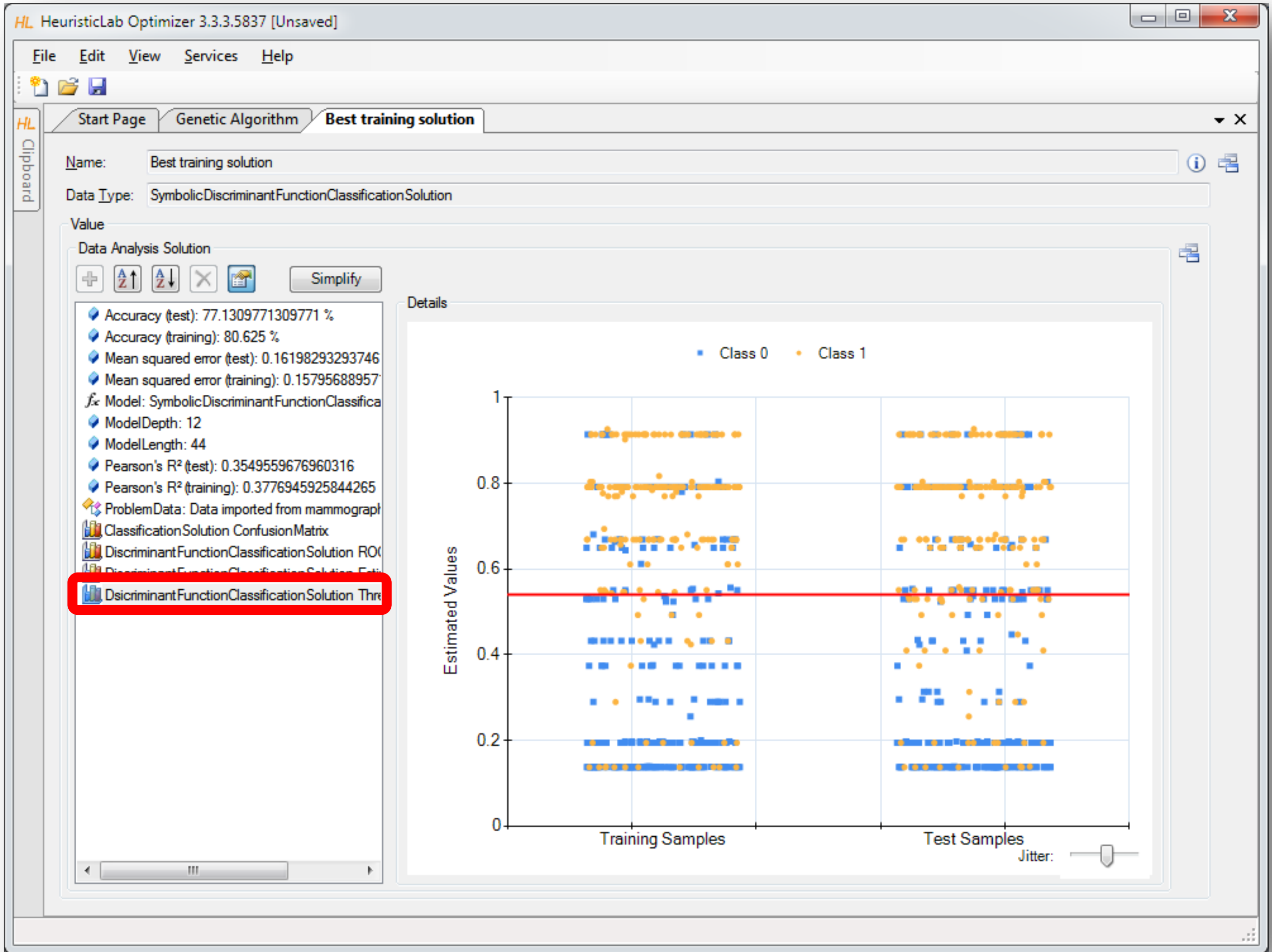
Inspect Quality Linechart



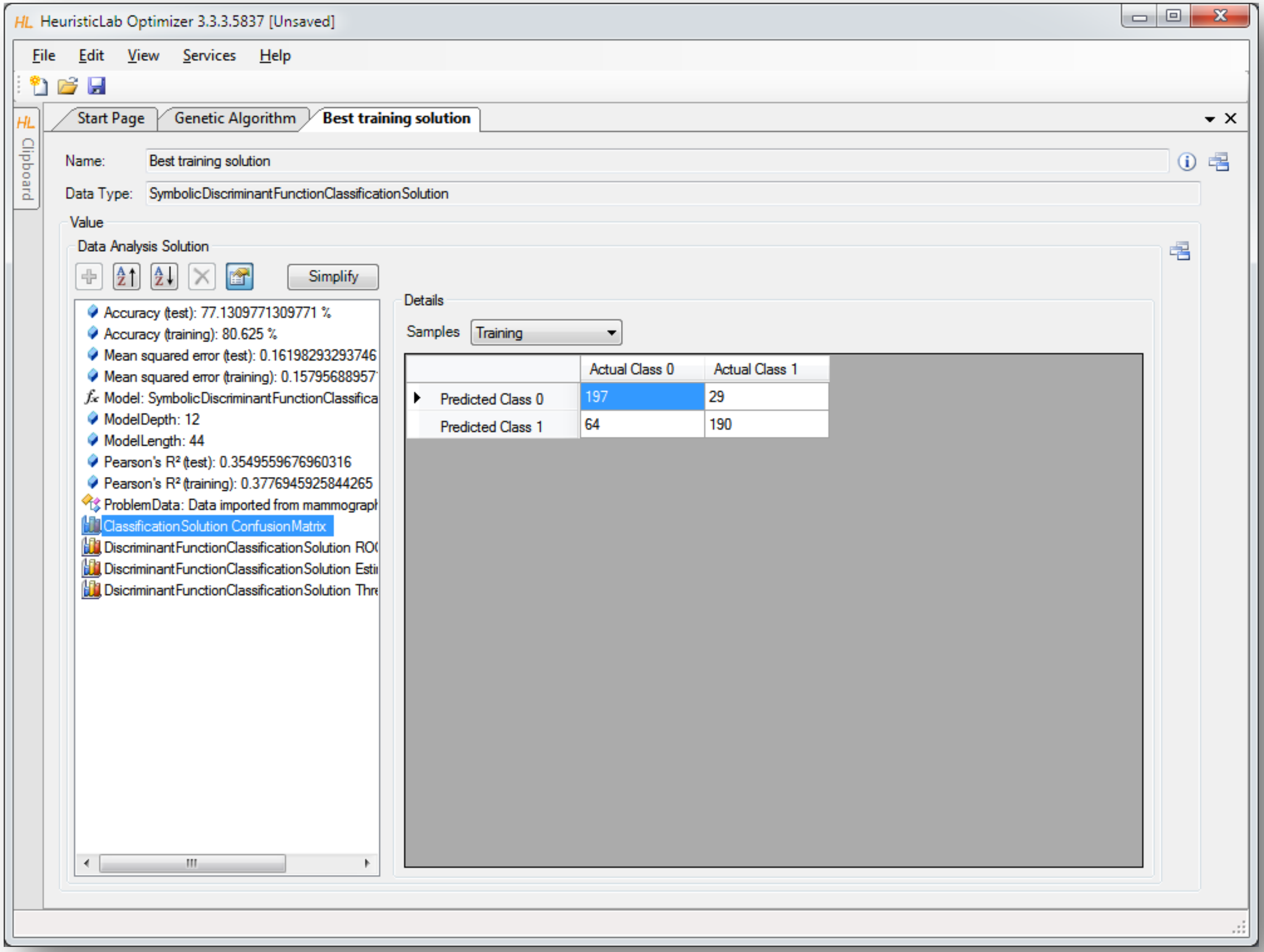
Inspect Best Training Solution



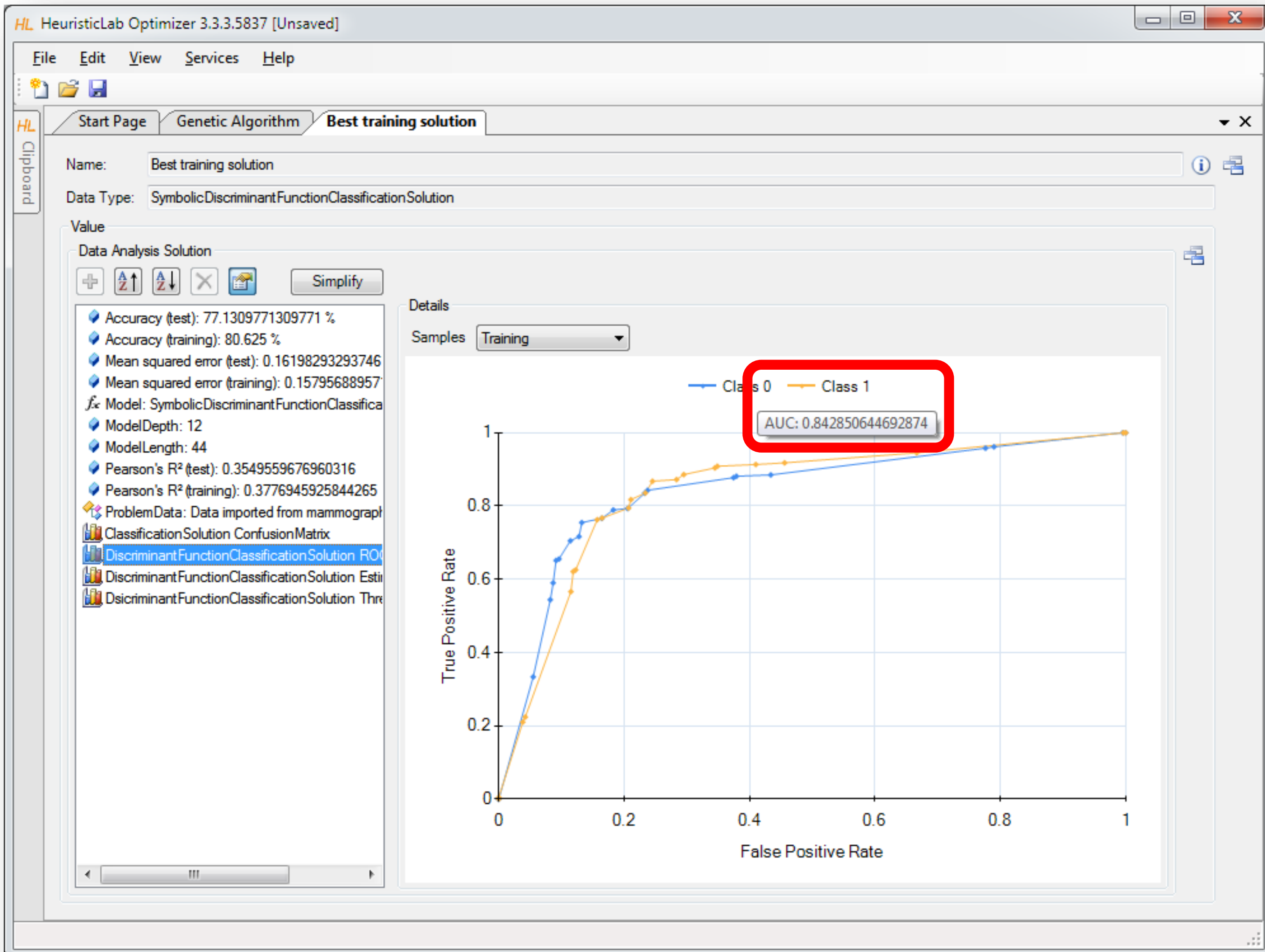
Inspect Model Output and Thresholds



Inspect Confusion Matrix

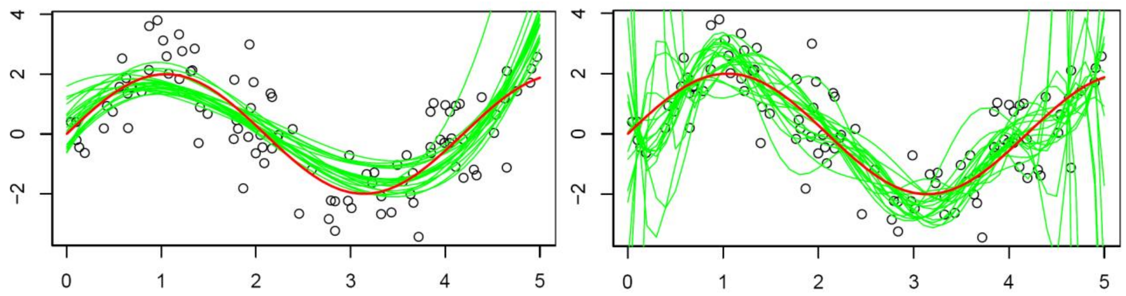


Inspect ROC Curve

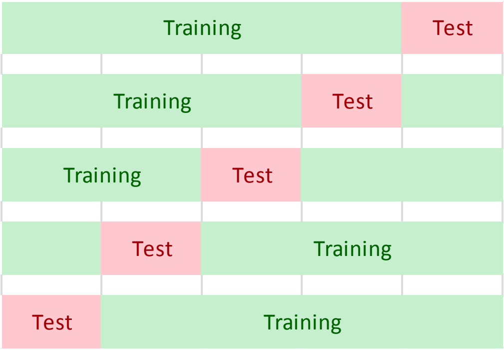


Validation of Results

- Overfitting = memorizing data

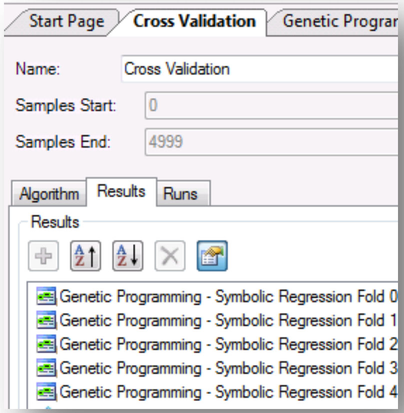


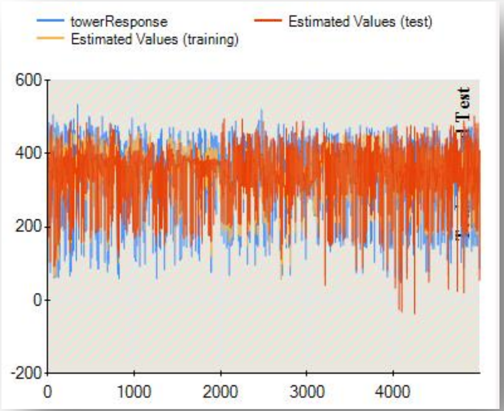
- Strategies to reduce overfitting
  - validation partition
  - cross-validation

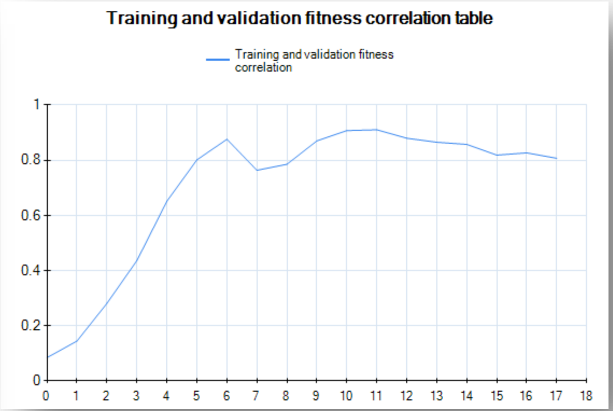


Validation of Results

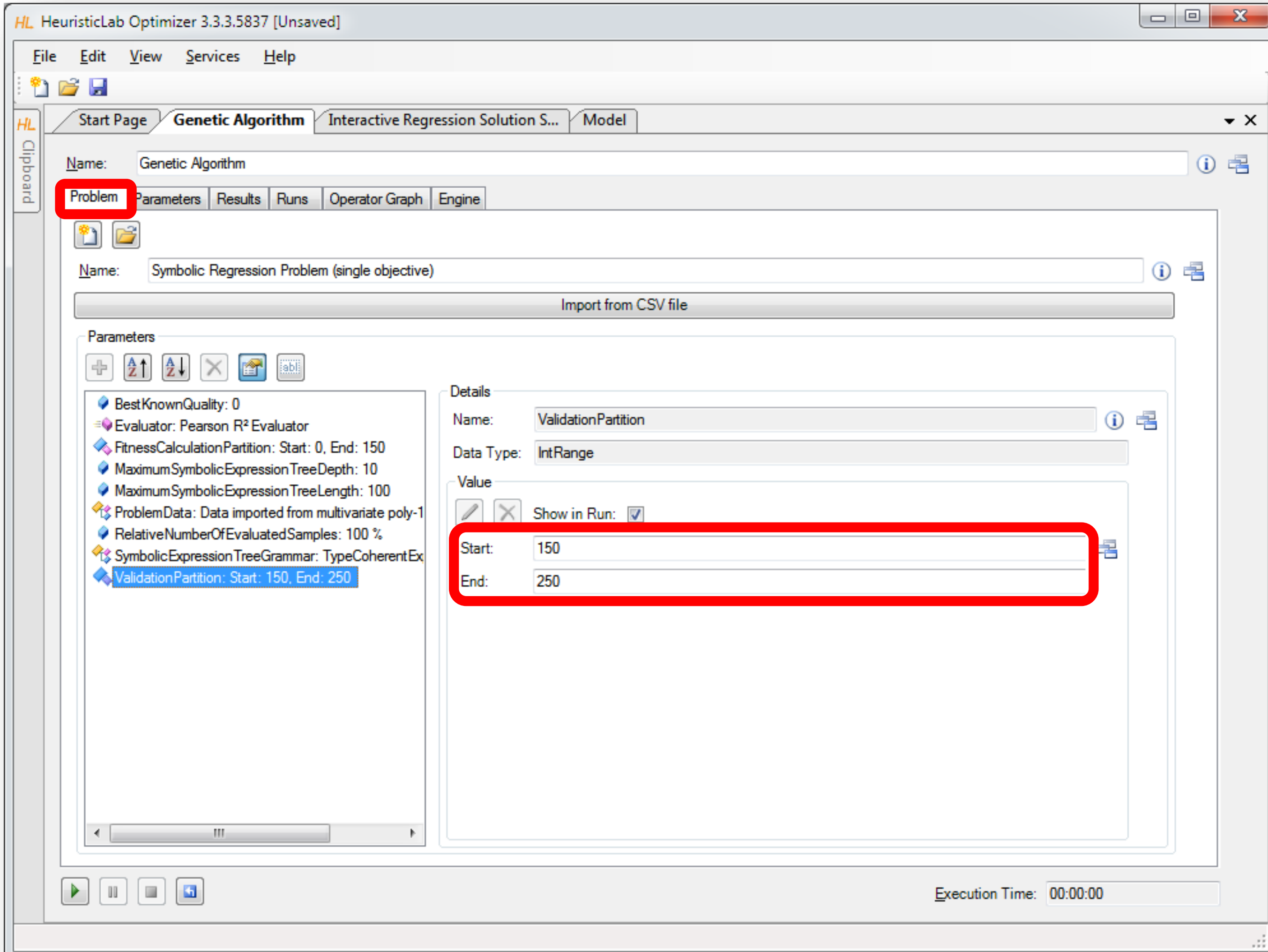
- Demonstration
  - Configuration of a validation set
  - Inspection of best solution on validation set
  - Analysis of training- and validation fitness correlation
  - Cross-validation
    - Configuration
    - Analysis of results



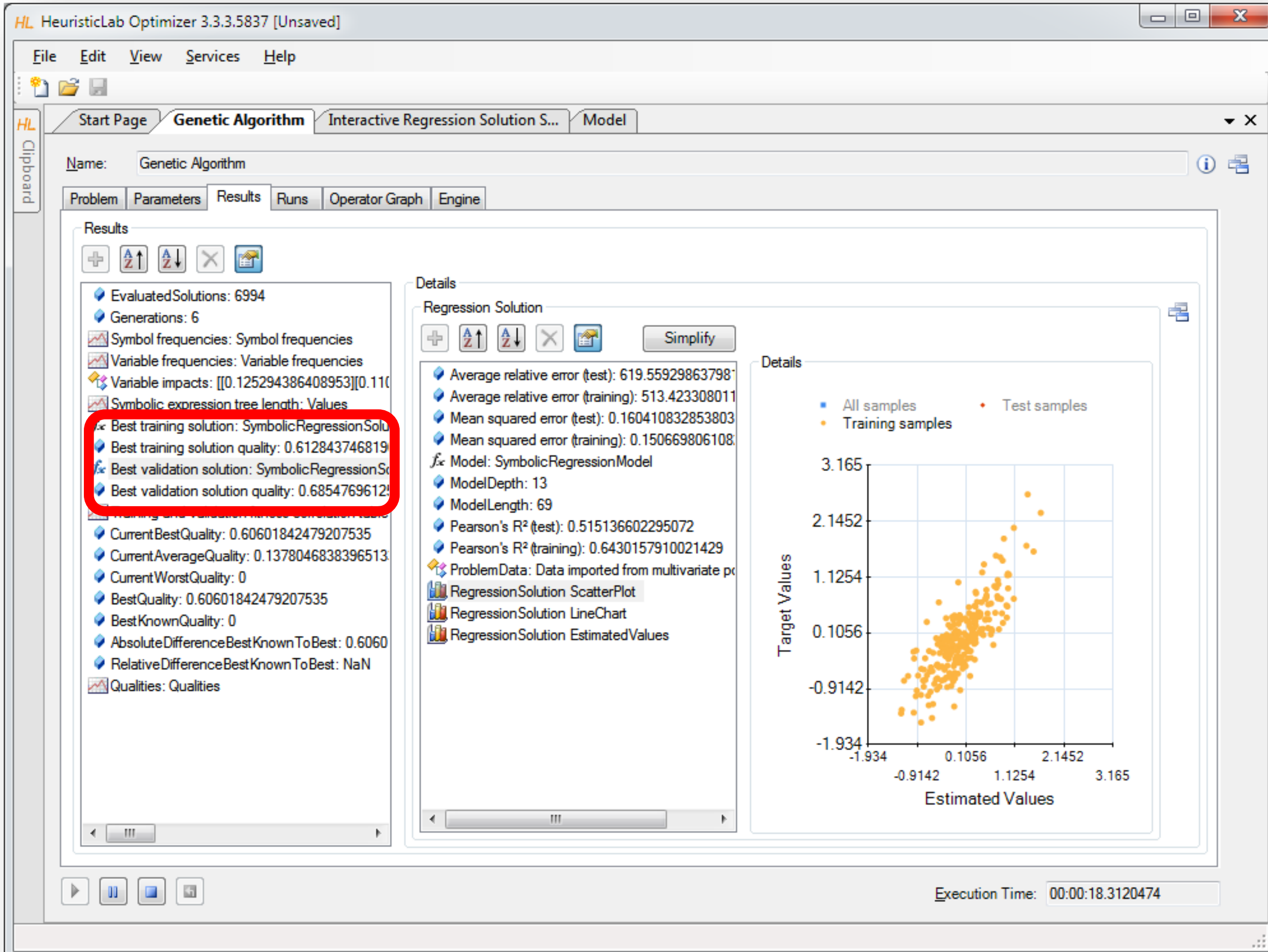




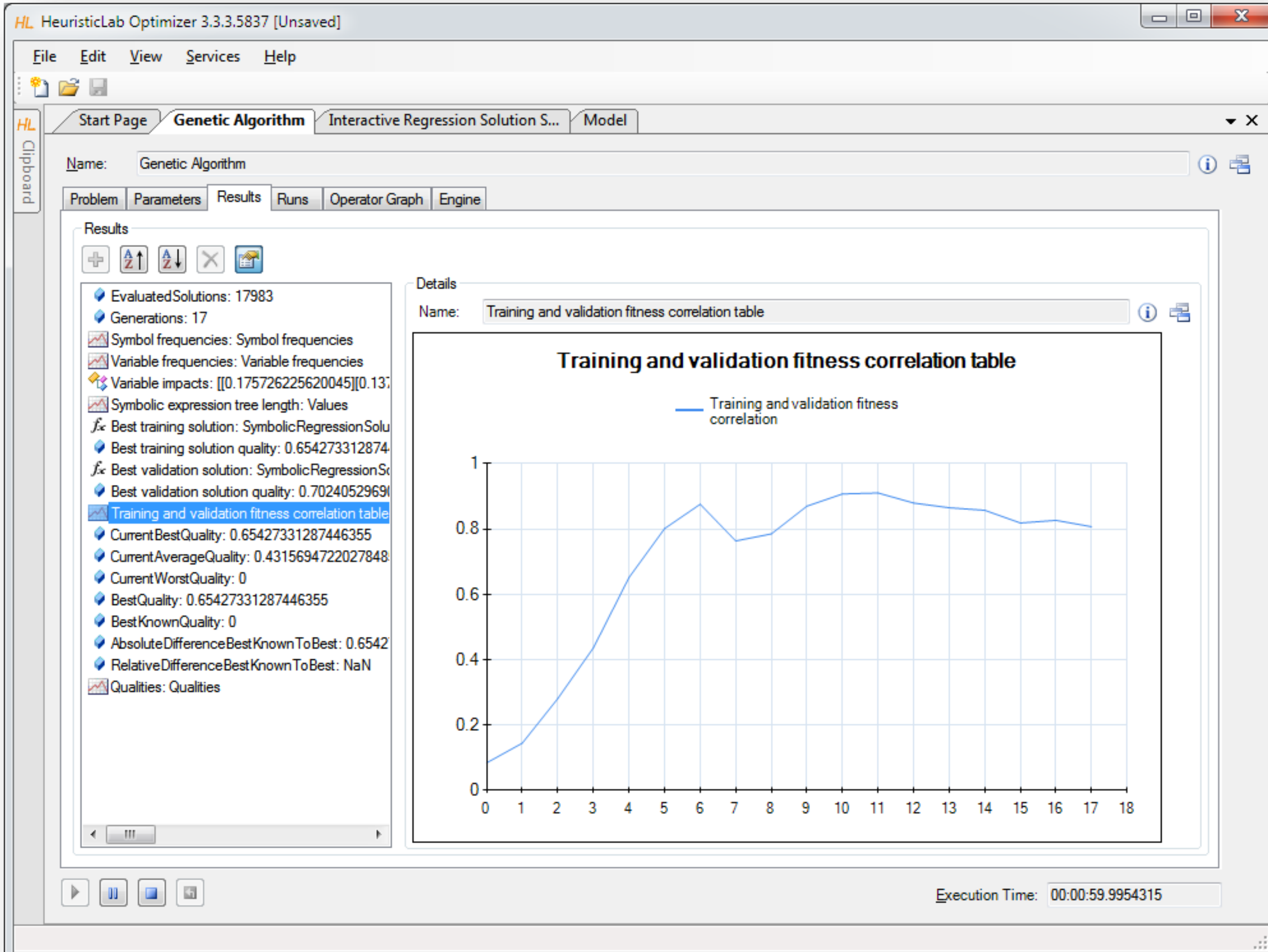
Configuration of Validation Partition



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
  - estimation of distribution algorithms
  - evolution of arbitrary code (Robocode, controller, etc.)
  - …
- Cloud Computing
  - port HeuristicLab Hive to Windows Azure
- Statistics
  - implement statistical tests and automated statistical analysis
- 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
  - write an e-mail to support@heuristiclab.com

HeuristicLab Team

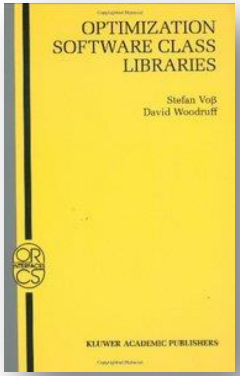
Heuristic and Evolutionary Algorithms Laboratory (HEAL)
School of Informatics, Communications and Media
University of Applied Sciences Upper Austria

Softwarepark 11
A-4232 Hagenberg
AUSTRIA

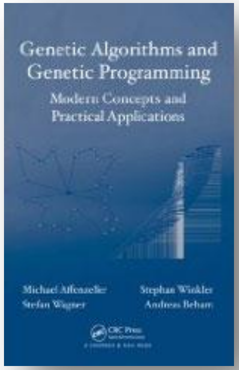
WWW: http://heal.heuristiclab.com

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

http://www.youtube.com/heuristiclab

http://www.facebook.com/heuristiclab