



# HeuristicLab

A Paradigm-Independent and Extensible  
Environment for Heuristic Optimization

# System Identification and Data Mining with HeuristicLab

An Open Source Optimization Environment for  
Research and Education

M. Kommenda, A. Scheibenpflug

Heuristic and Evolutionary Algorithms Laboratory (HEAL)

School of Informatics/Communications/Media, Campus Hagenberg

University of Applied Sciences Upper Austria



**HEAL**

Heuristic and Evolutionary  
Algorithms Laboratory



Josef Ressel-Zentrum

**HEUREKA!**

# Instructor Biographies

- Michael Kommenda
  - Diploma in computer sciences (2007)  
Upper Austria University of Applied Sciences, Austria
  - MSc in bioinformatics (2011)  
Upper Austria University of Applied Sciences, Austria
  - Research associate at the Research Center Hagenberg
  - Joined HEAL in 2006
  - One of the main architects of HeuristicLab
  - <http://heal.heuristiclab.com/team/kommenda>
- Andreas Scheibenpflug
  - MSc in computer sciences (2011)  
Upper Austria University of Applied Sciences, Austria
  - Research assistant at the Research Center Hagenberg
  - Joined HEAL in 2010
  - One of the main architects of HeuristicLab
  - <http://heal.heuristiclab.com/team/scheibenpflug>



# HeuristicLab Team



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

Softwarepark 11  
A-4232 Hagenberg  
AUSTRIA

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



**HEAL**

Heuristic and Evolutionary  
Algorithms Laboratory



# 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
- Suggested Readings
- Bibliography
- Questions & Answers

# Objectives of the Tutorial

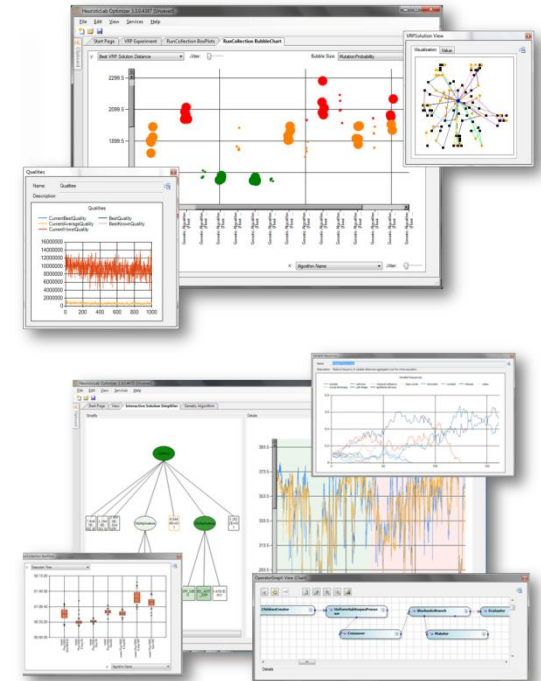


- Introduce general motivation and design principles of HeuristicLab
- Show where to get HeuristicLab
- Explain basic GUI usability concepts
- Introduce Metaheuristics and Evolutionary Algorithms
- Demonstrate basic features
- Demonstrate editing and analysis of optimization experiments
- Demonstrate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

# Introduction to HeuristicLab



- Motivation and Goals
  - graphical user interface
  - paradigm independence
  - multiple algorithms and problems
  - large scale experiments and analyses
  - parallelization
  - extensibility, flexibility and reusability
  - visual and interactive algorithm development
  - multiple layers of abstraction
- Facts
  - development of HeuristicLab started in 2002
  - based on Microsoft .NET and C#
  - used in research and education
  - second place at the *Microsoft Innovation Award 2009*
  - open source (GNU General Public License)
  - version 3.3.0 released on May 18th, 2010
  - latest version 3.3.5 released on July 9th, 2011



# Where to get HeuristicLab?

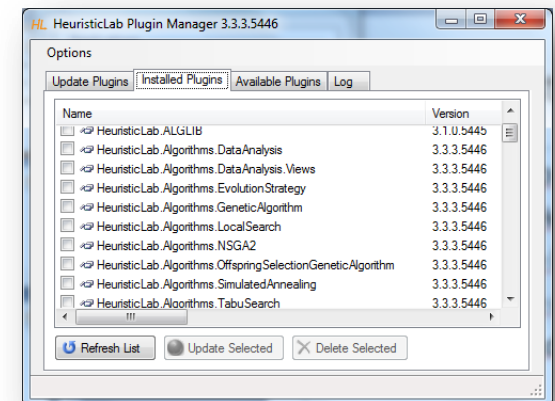
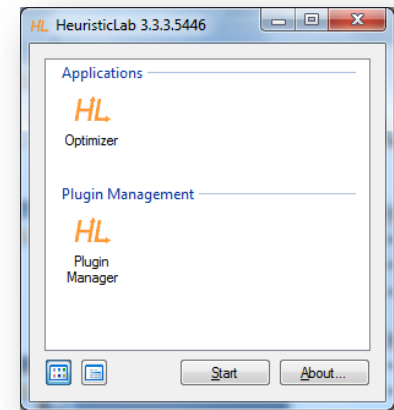


- Download binaries
  - deployed as ZIP archives
  - latest stable version 3.3.5
    - released on July 9th, 2011
  - daily trunk build
  - <http://dev.heuristiclab.com/download>
- Check out sources
  - SVN repository
  - HeuristicLab 3.3.5 tag
    - <http://dev.heuristiclab.com/svn/hl/core/tags/3.3.5>
  - current development trunk
    - <http://dev.heuristiclab.com/svn/hl/core/trunk>
- License
  - GNU General Public License (Version 3)
- System requirements
  - Microsoft .NET Framework 4.0 Full Version
  - enough RAM and CPU power ;-)



# Plugin Infrastructure

- HeuristicLab consists of many assemblies
  - 95 plugins in HeuristicLab 3.3.5
  - plugins can be loaded or unloaded at runtime
  - plugins can be updated via internet
  - application plugins provide GUI frontends
- Extensibility
  - developing and deploying new plugins is easy
  - dependencies are explicitly defined, automatically checked and resolved
  - automatic discovery of interface implementations (service locator pattern)
- Plugin Manager
  - GUI to check, install, update or delete plugins



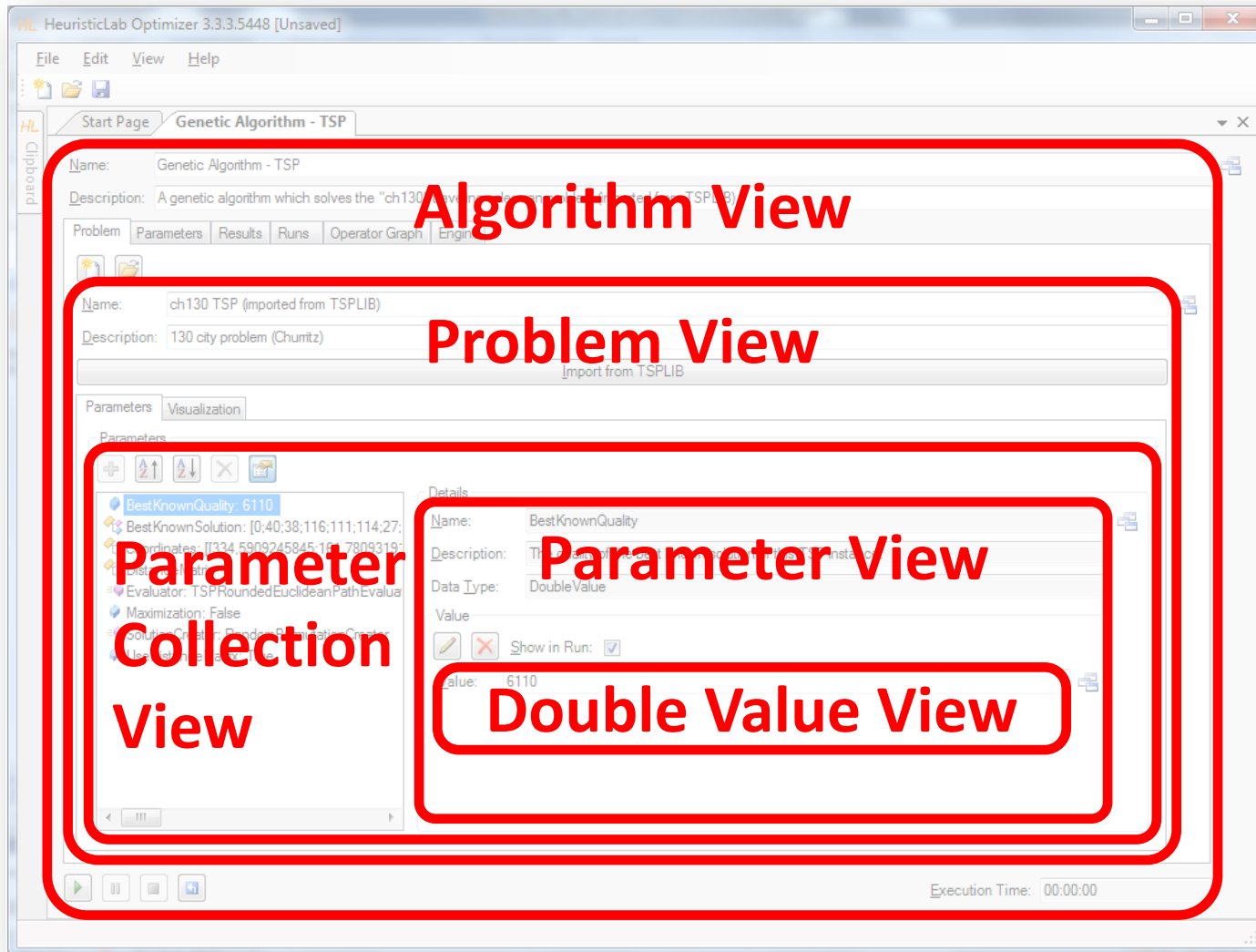


# Graphical User Interface



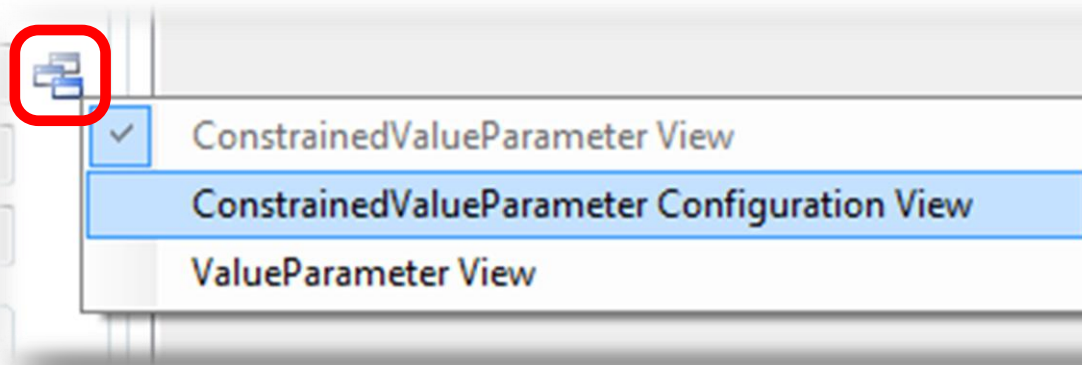
- HeuristicLab GUI is made up of views
  - views are visual representations of content objects
  - views are composed in the same way as their content
  - views and content objects are loosely coupled
  - multiple different views may exist for the same content
- Drag & Drop
  - views support drag & drop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

# Graphical User Interface



# Graphical User Interface

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



# Metaheuristics

- There are problems which can't be exactly solved in feasible time (e.g. NP-hard)
  - High complexity
  - Large search space
  - Curse of dimensionality/Combinatorial explosion
  - But: Solution evaluation is cheap
- → Try to generate a solution
- Check if you can generate a better solution
- Metaheuristics don't (always) find the best solution, but a good solution

# Evolutionary Algorithms

## – Idea:

- Simulation of natural evolution
- Adaptation of species considered as an optimization process
- Population-based optimization

## **Mechanisms of optimization:**

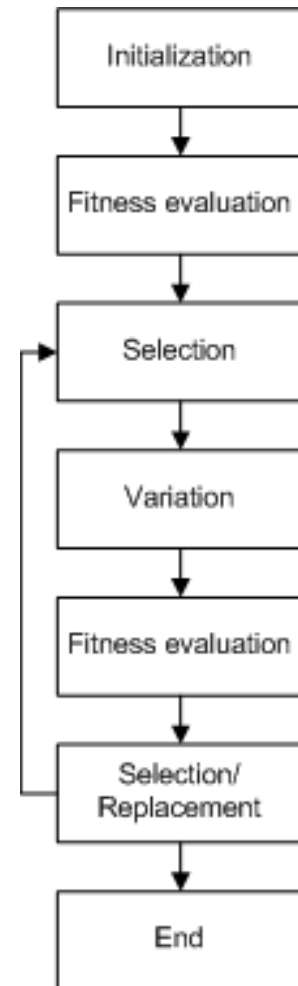
- Evaluation on the basis of fitness function
- Selection on the basis of fitness
- Solution manipulation
- Acceptance or rejection

## **Mechanisms of evolution:**

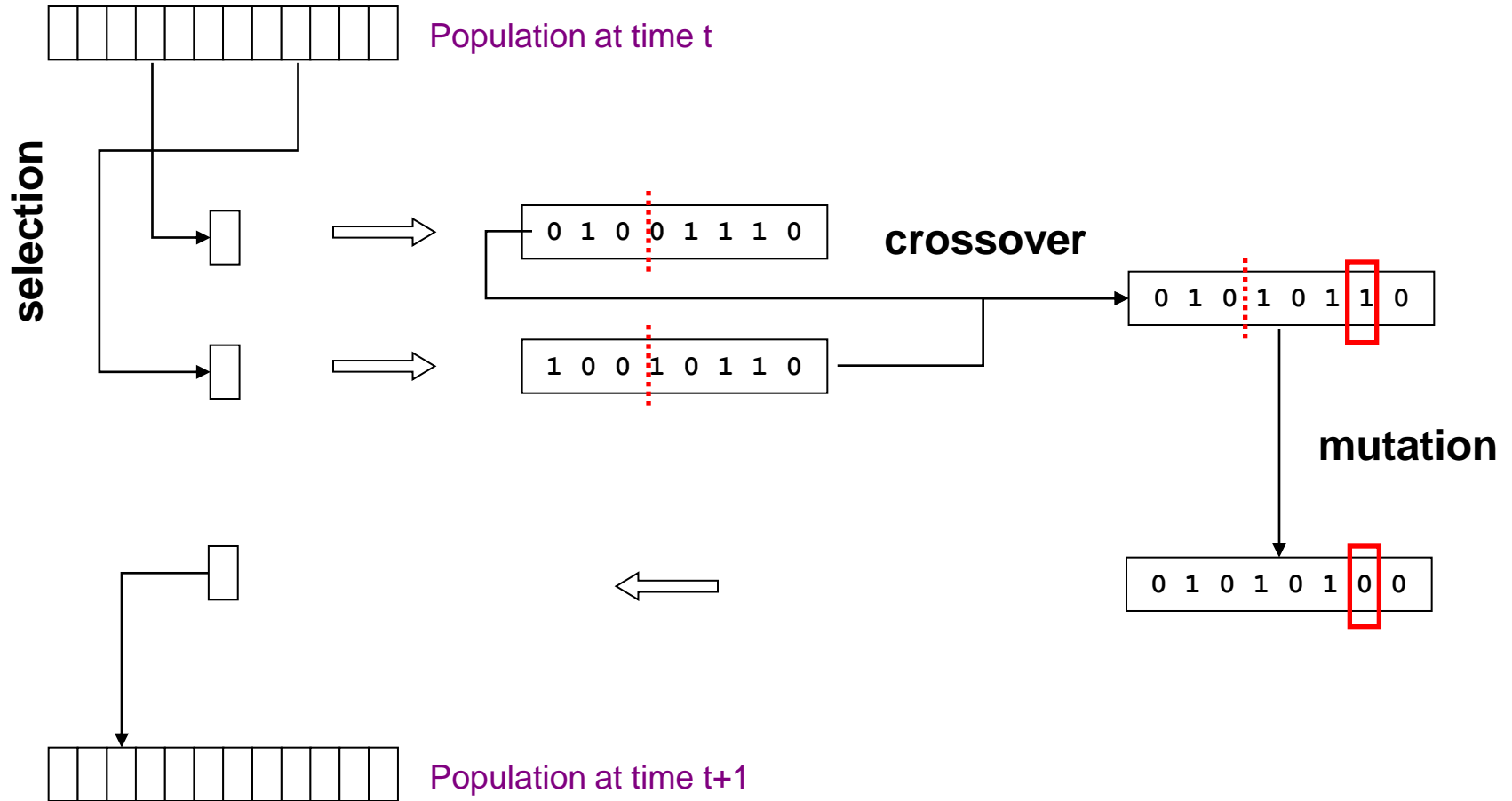
- Selection
- Variation
- Replication

# Flowchart of an EA

- Initial population of individuals is generated randomly or heuristically.
  - Individuals are evaluated and are assigned a certain fitness value.
  - While termination condition is not met
    - Fitter individuals are selected for reproduction and children are produced by applying crossover and mutation on the parents.
    - The new individuals are evaluated.
    - New population is generated from the new and/or old individuals.
- End while.

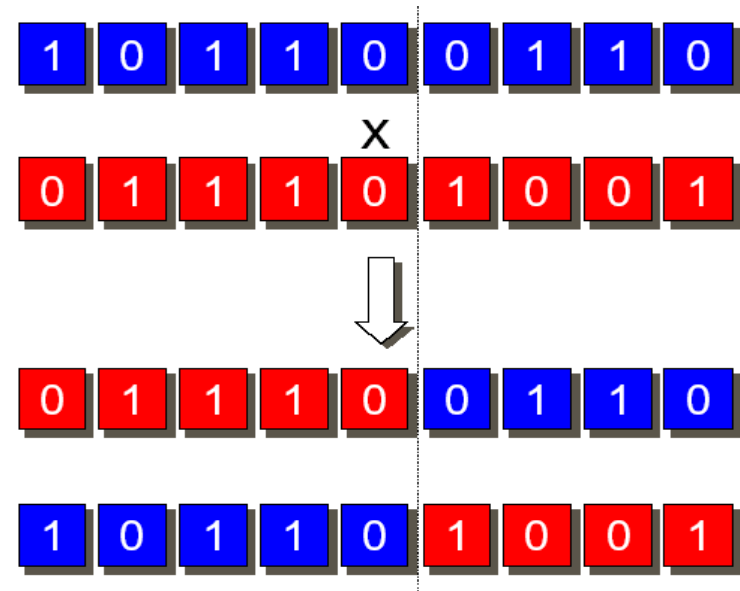


# Genetic Algorithms



# Crossover

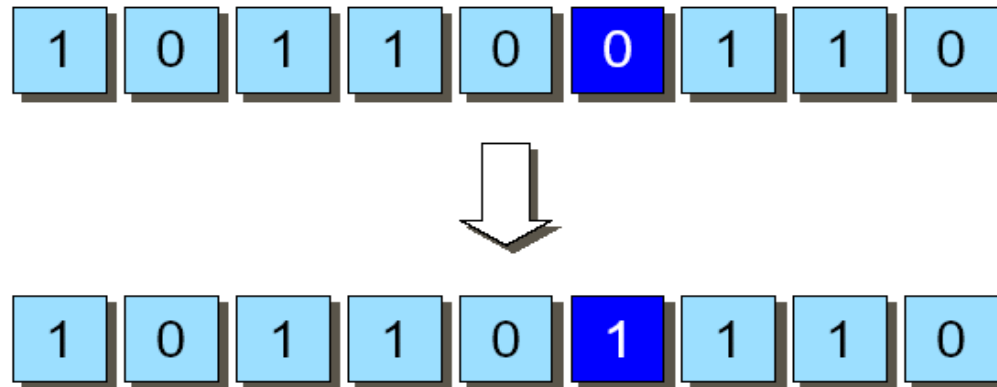
- Simple but effective:
  - Single-Point Crossover
- Generalizations:
  - Two-Point Crossover
  - n-Point Crossover
- Crossover points are set randomly





# Mutation

- Bit-Flip with little probability



- Generally one bit is changed in mutant (in case of very long chromosomes sometimes even more bits)
- Mutation rate defines probability of being mutated

# Available Algorithms & Problems



## Algorithms

- Genetic Algorithm
- Island Genetic Algorithm
- Offspring Selection Genetic Algorithm
- Island Offspring Selection Genetic Algorithm
- SASEGASA
- Evolution Strategy
- NSGA-II
- Particle Swarm Optimization
- Local Search
- Simulated Annealing
- Tabu Search
- Variable Neighborhood Search
- Linear Regression
- Linear Discriminant Analysis
- Support Vector Machine
- k-Means
- User-defined Algorithm

## Problems

- Single-Objective Test Function
- Traveling Salesman Problem
- Quadratic Assignment Problem
- Vehicle Routing Problem
- Scheduling
- Knapsack
- OneMax
- Data Analysis
- Regression
- Symbolic Regression
- Classification
- Symbolic Classification
- Clustering
- Artificial Ant
- External Evaluation Problem
- User-defined Problem

# Agenda

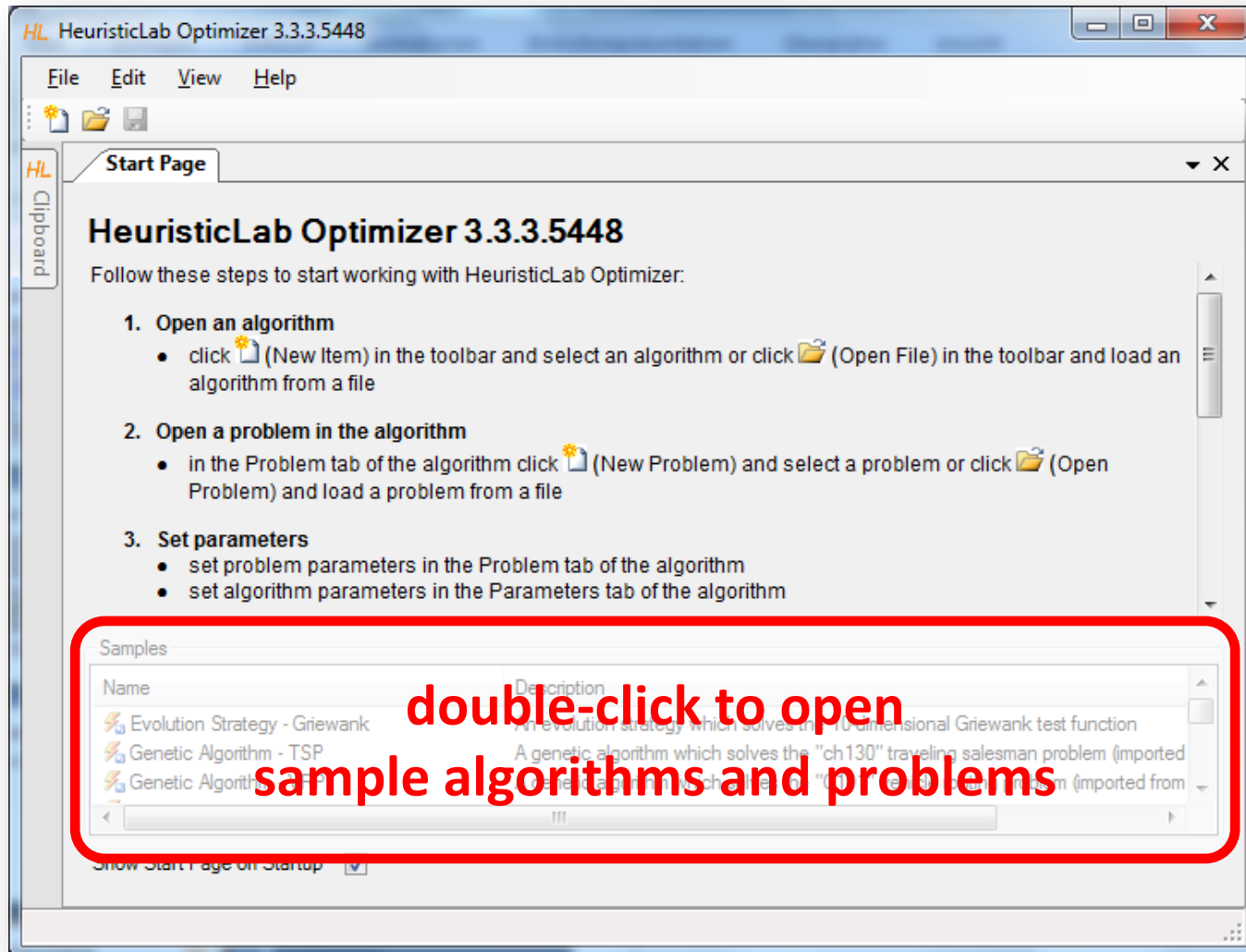
- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- **Demonstration Part I: Working with HeuristicLab**
- **Demonstration Part II: Data-based Modeling**
- Some Additional Features
- Planned Features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

# Demonstration Part I: Working with HeuristicLab

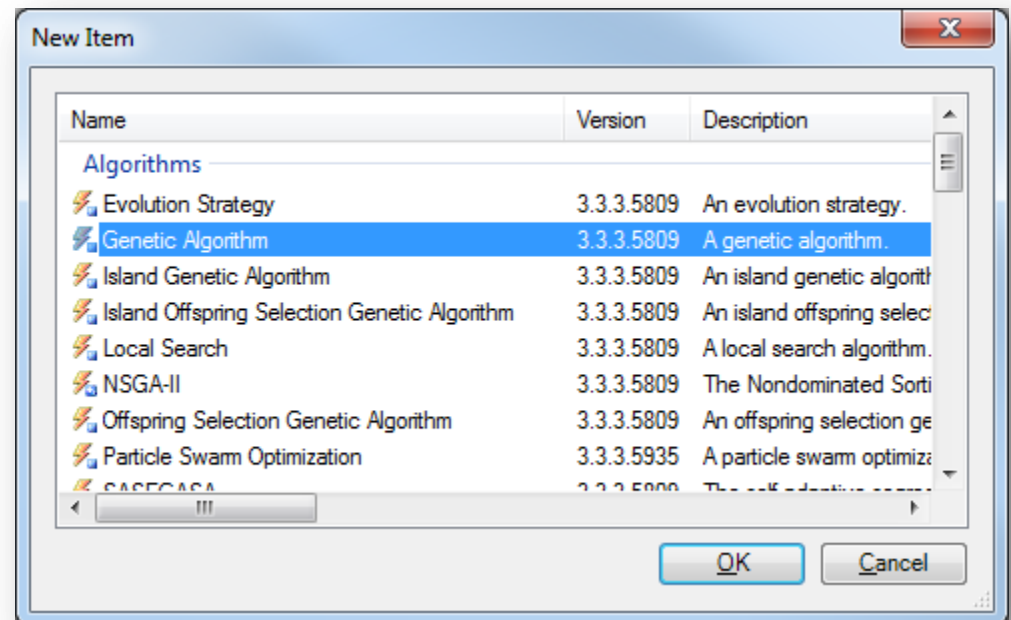
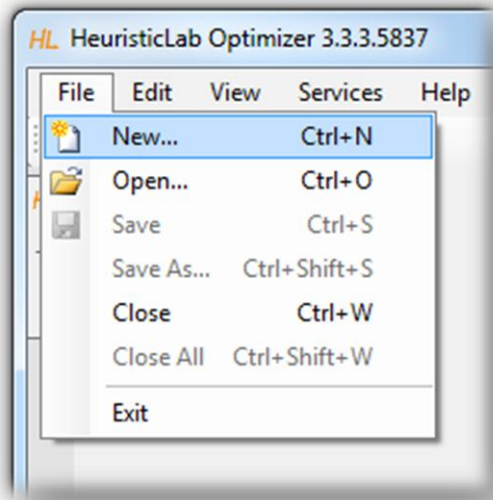


- Create, Parameterize and Execute Algorithms
- Save and Load Items
- Create Batch Runs and Experiments
- Multi-core CPUs and Parallelization
- Analyze Runs
- Analyzers
- Building User-Defined Algorithms

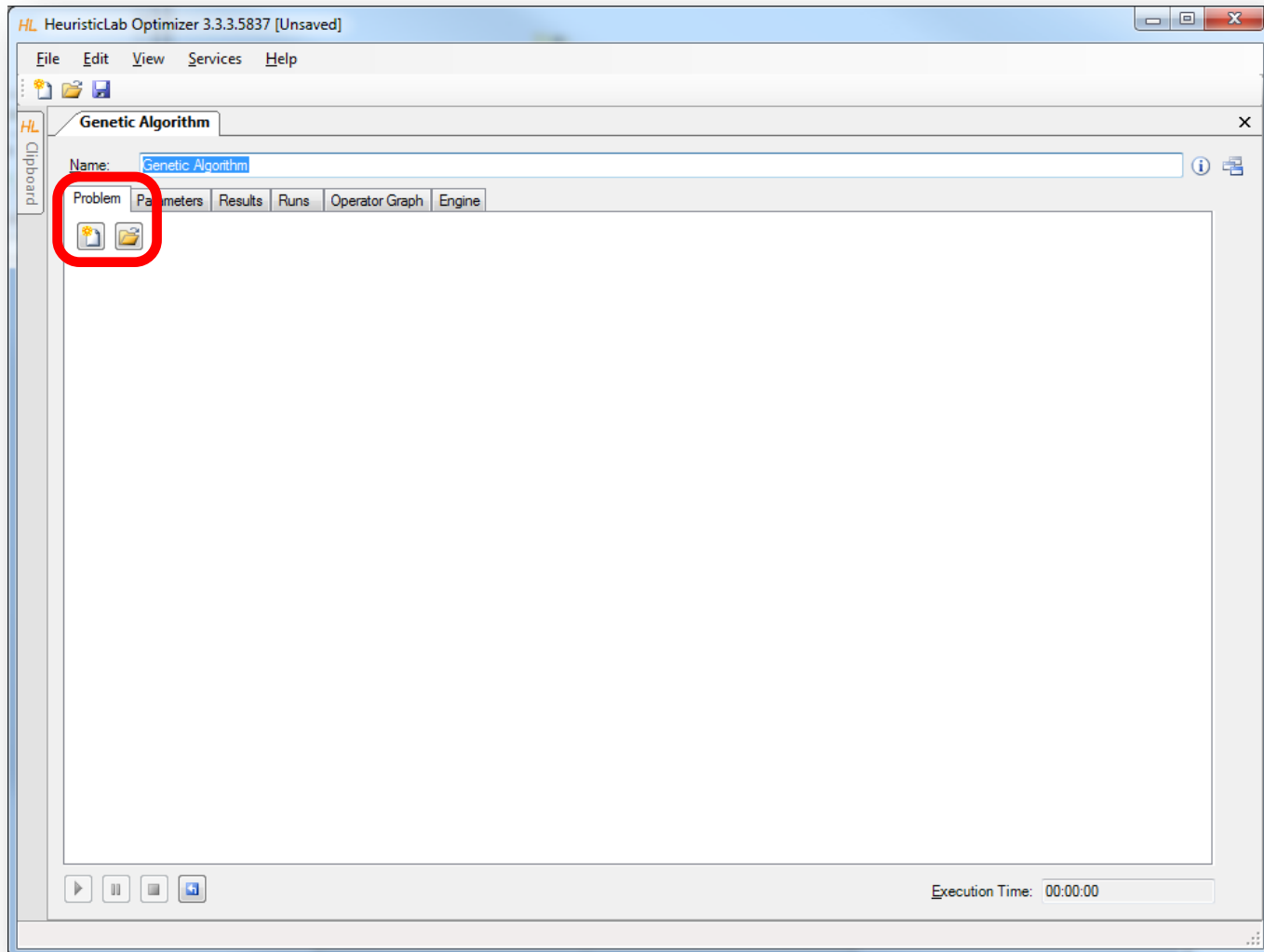
# HeuristicLab Optimizer



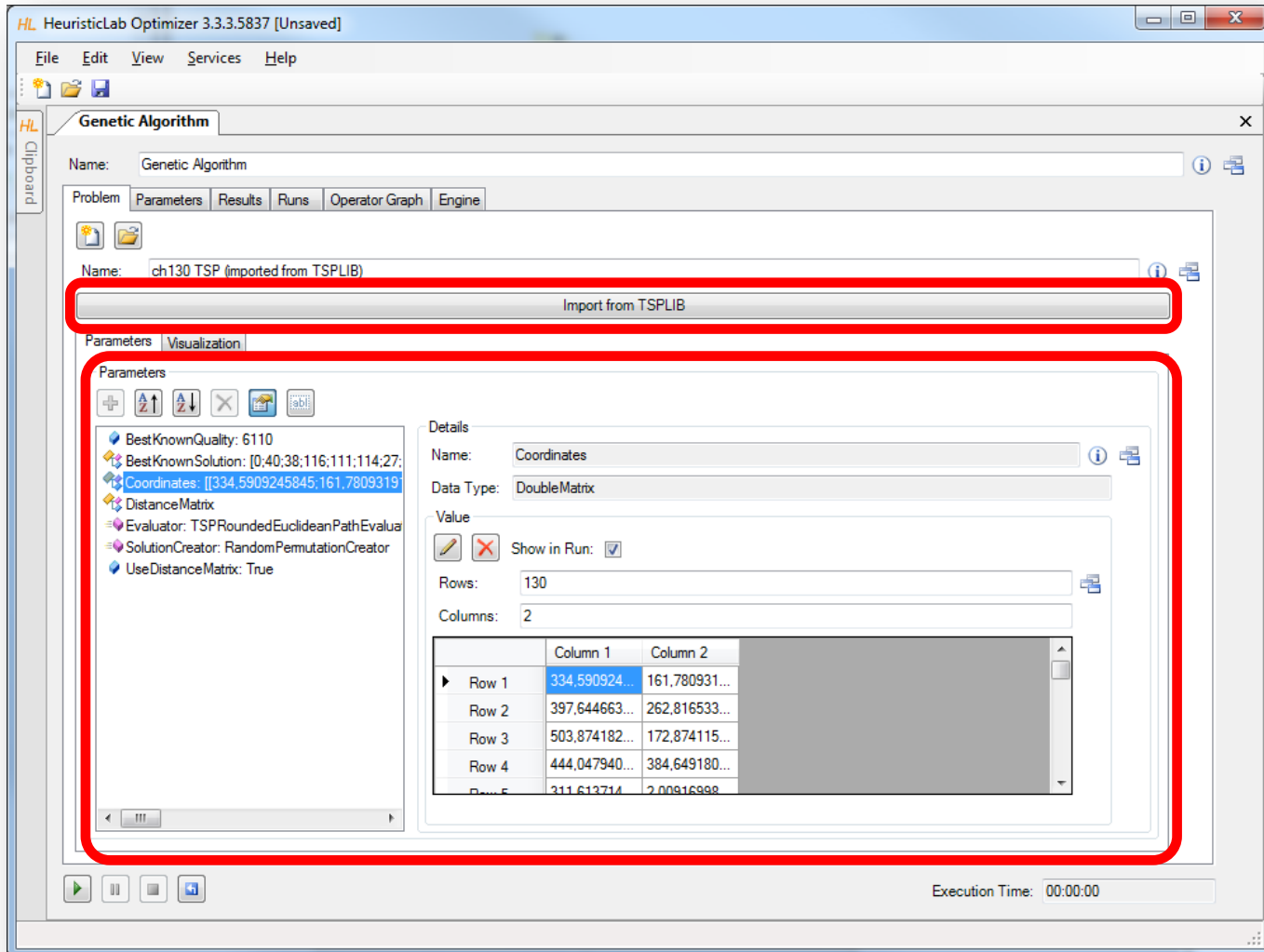
# Create Algorithm



# Create or Load Problem



# Import or Parameterize Problem Data



HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

Clipboard

Genetic Algorithm

Name: Genetic Algorithm

Problem Parameters Results Runs Operator Graph Engine

Name: ch130 TSP (imported from TSPLIB)

Import from TSPLIB

Parameters Visualization

Parameters

- BestKnownQuality: 6110
- BestKnownSolution: [0,40,38;116;111;114;27;
- Coordinates: [[334.5909245845;161.7809319
- DistanceMatrix
- Evaluator: TSPRoundedEuclideanPathEvalua
- SolutionCreator: RandomPermutationCreator
- UseDistanceMatrix: True

Details

Name: Coordinates

Data Type: DoubleMatrix

Value

Show in Run:

Rows: 130

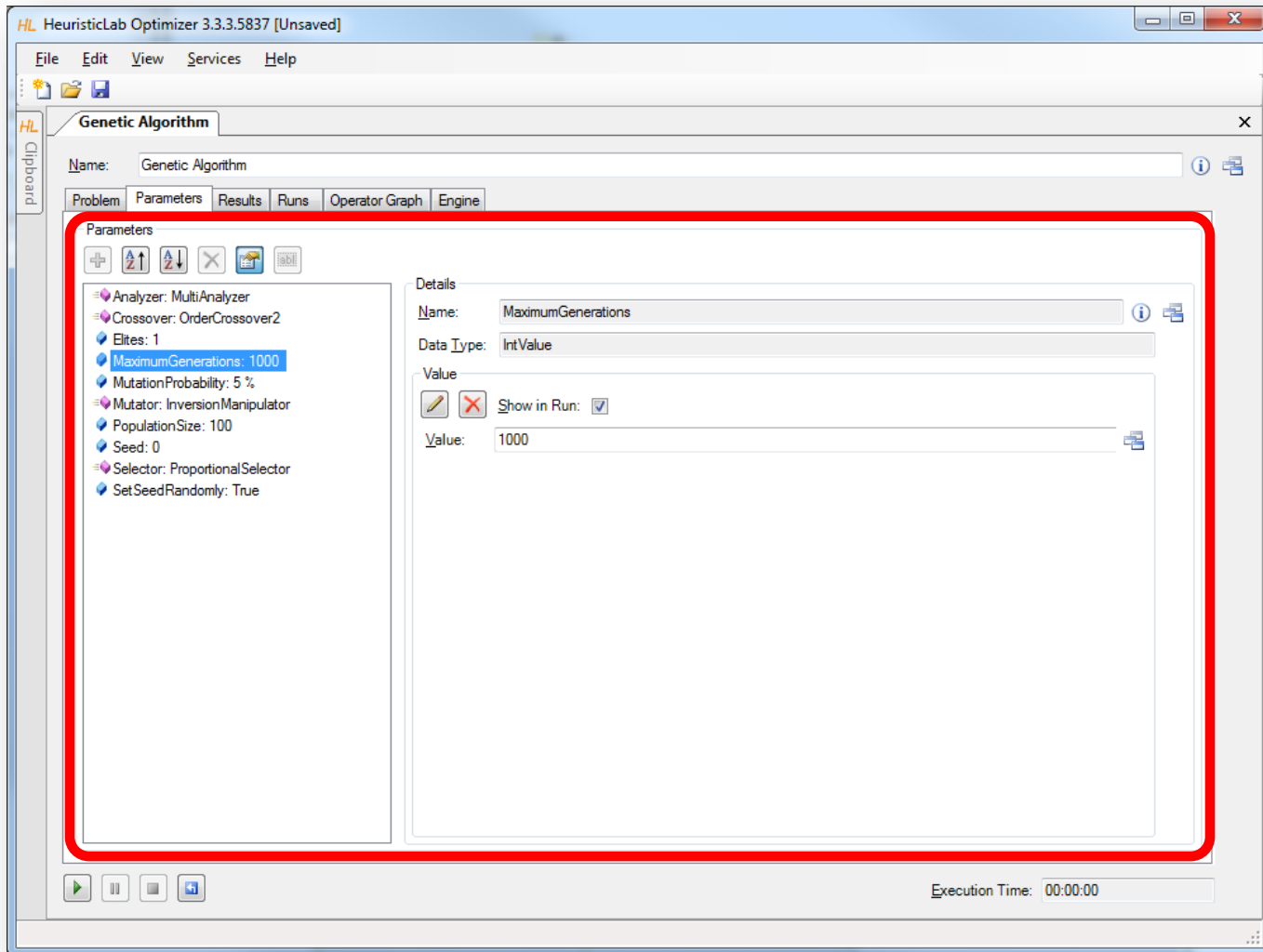
Columns: 2

	Column 1	Column 2
Row 1	334.590924...	161.780931...
Row 2	397.644663...	262.816533...
Row 3	503.874182...	172.874115...
Row 4	444.047940...	384.649180...
Row 5	311.613714...	2.00916998...

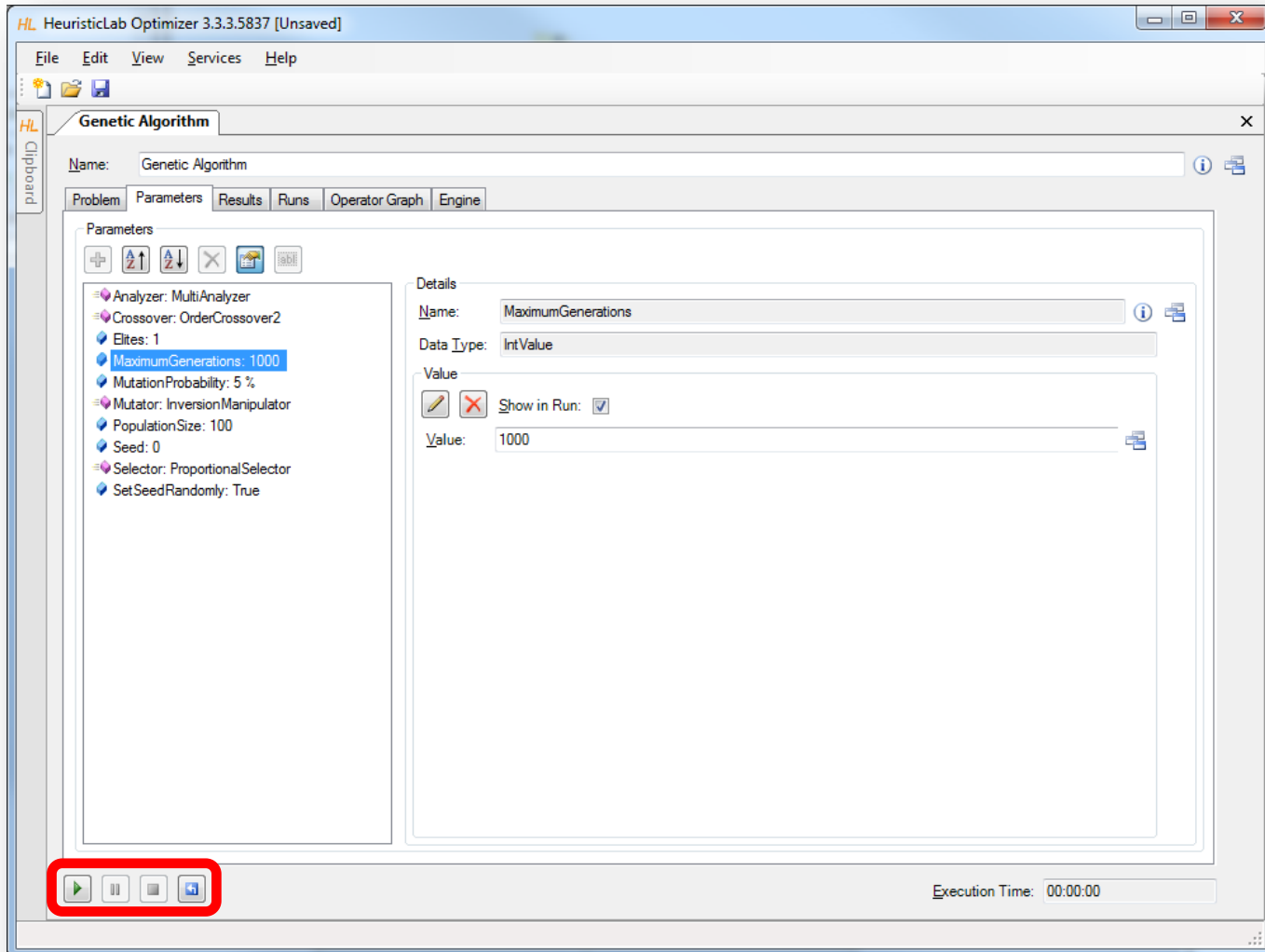
Execution Time: 00:00:00



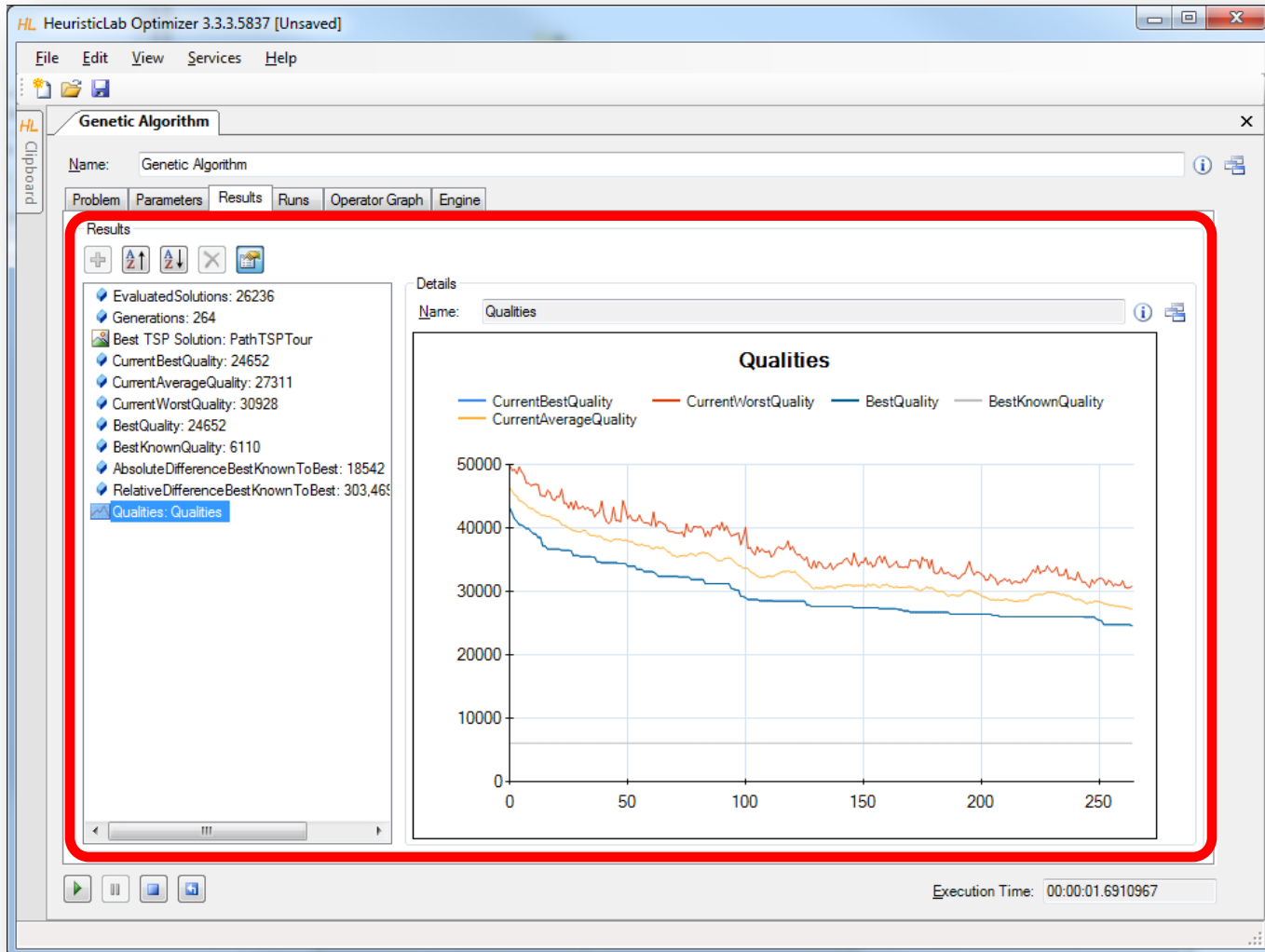
# Parameterize Algorithm



# Start, Pause, Resume, Stop and Reset

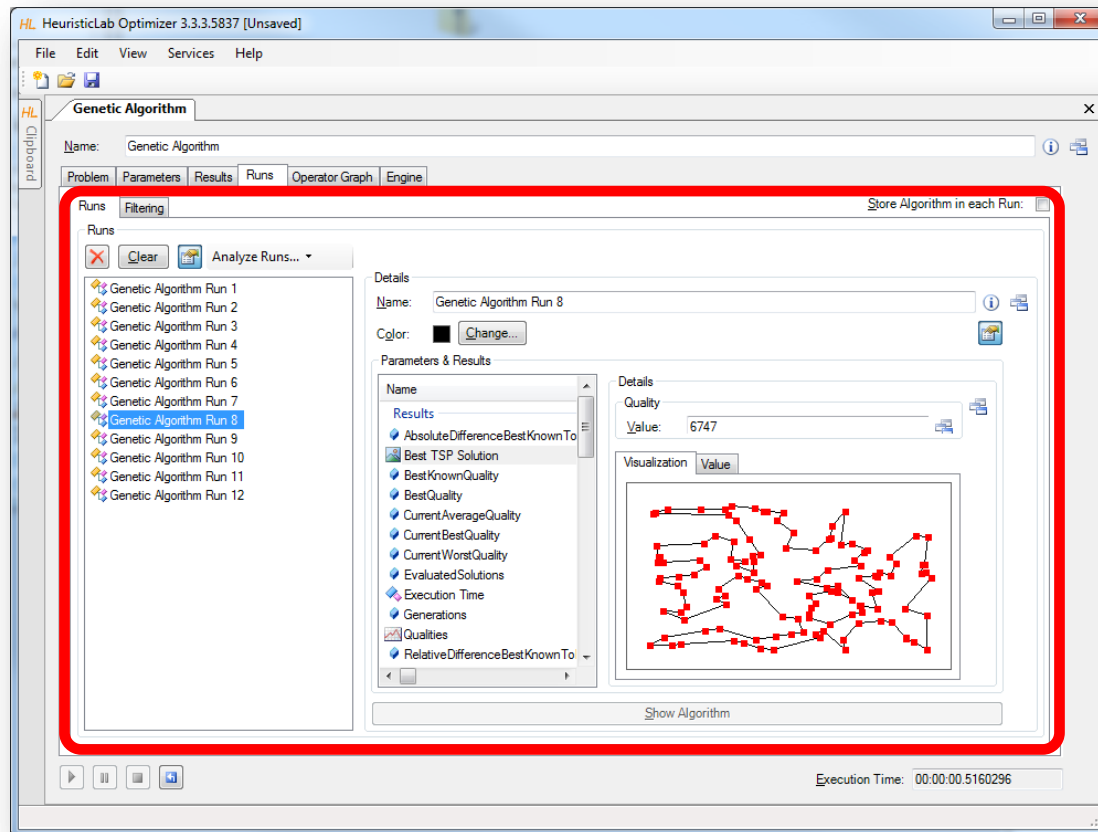


# Inspect Results



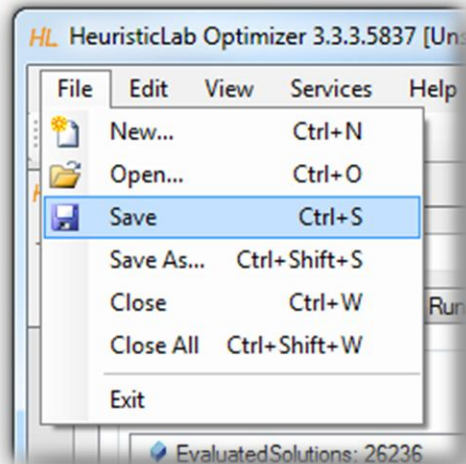
# Compare Runs

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



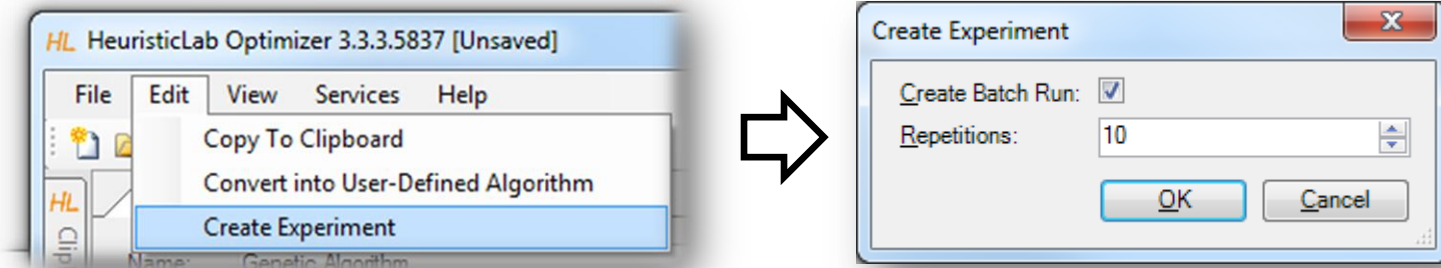
# Save and Load

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

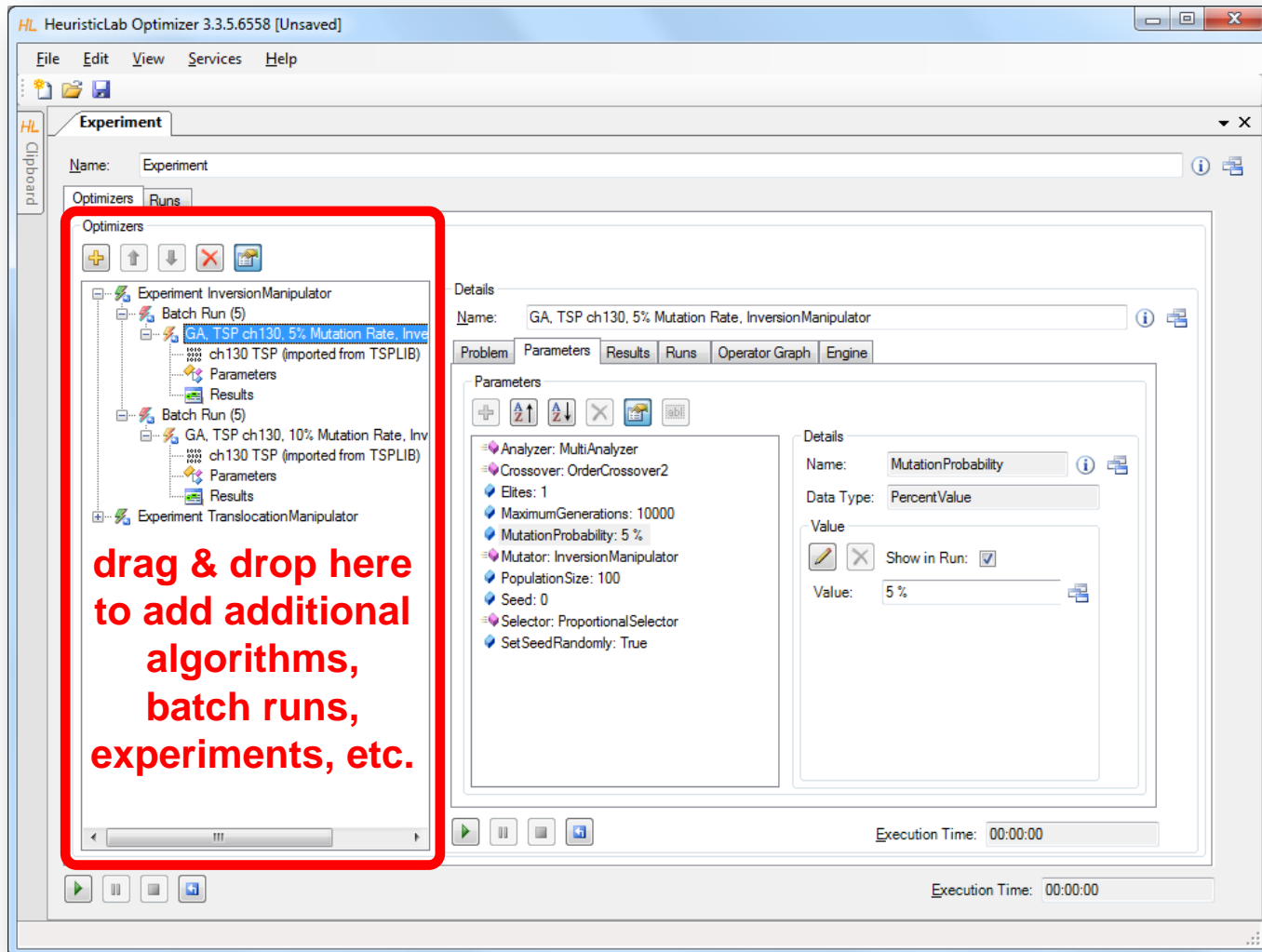


# Create Batch Runs and Experiments

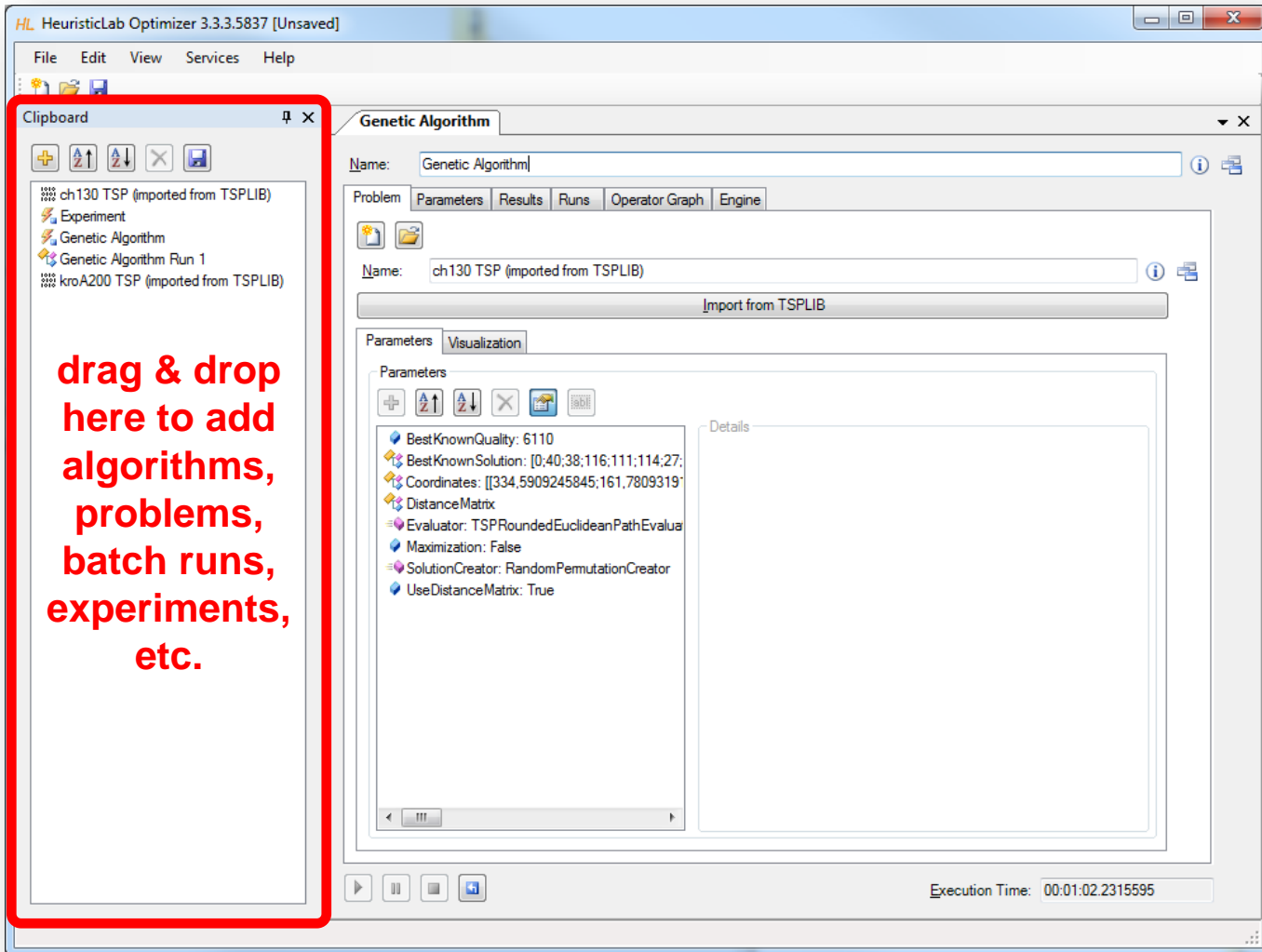
- Batch runs
  - execute the same optimizer (e.g. algorithm, batch run, experiment) several times
- Experiments
  - execute different optimizers
  - suitable for large scale algorithm comparison and analysis
- Experiments and batch runs can be nested
- Generated runs can be compared afterwards



# Create Batch Runs and Experiments



# Clipboard

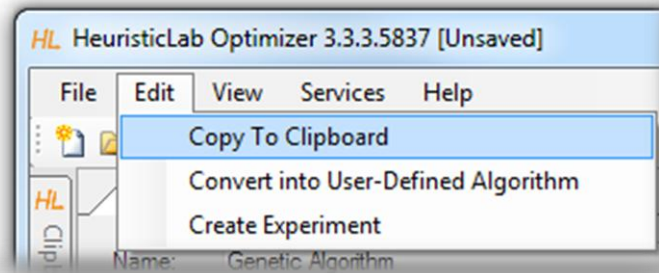


The screenshot shows the HeuristicLab Optimizer interface. A red box highlights the Clipboard window on the left, which contains a list of items: ch130 TSP (imported from TSPLIB), Experiment, Genetic Algorithm, Genetic Algorithm Run 1, and kroA200 TSP (imported from TSPLIB). A red text overlay on the clipboard window reads: "drag & drop here to add algorithms, problems, batch runs, experiments, etc." The main window displays the configuration for a Genetic Algorithm, including the problem name "ch130 TSP (imported from TSPLIB)" and various parameters such as BestKnownQuality, BestKnownSolution, Coordinates, DistanceMatrix, Evaluator, Maximization, SolutionCreator, and UseDistanceMatrix.



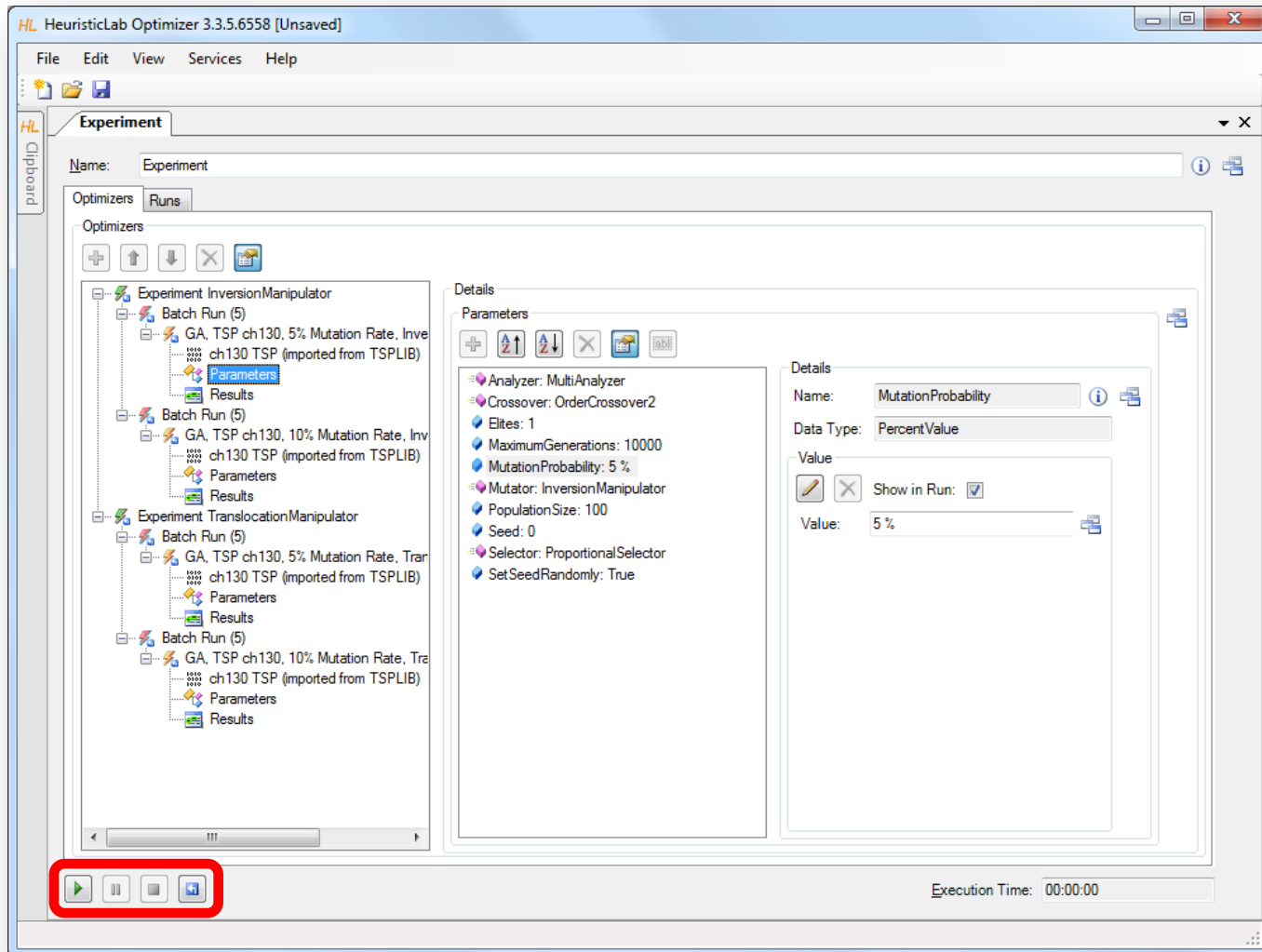
# Clipboard

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

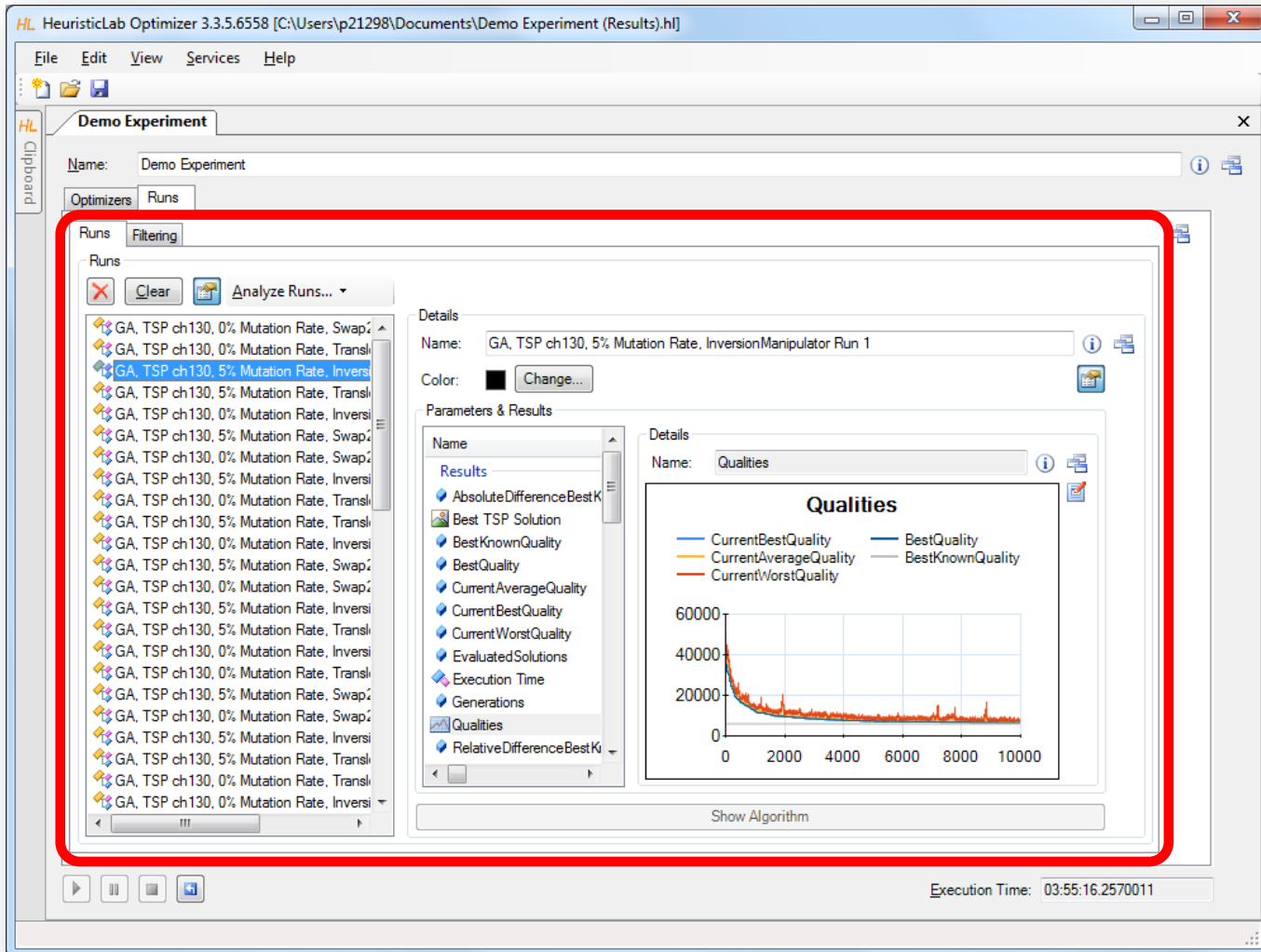


- Show items
  - double-click on an item in the clipboard to show its view
- Save and restore clipboard content
  - click on the save button to write the clipboard content to disk
  - clipboard is automatically restored when HeuristicLab is started the next time

# Start, Pause, Resume, Stop, Reset



# Compare Runs



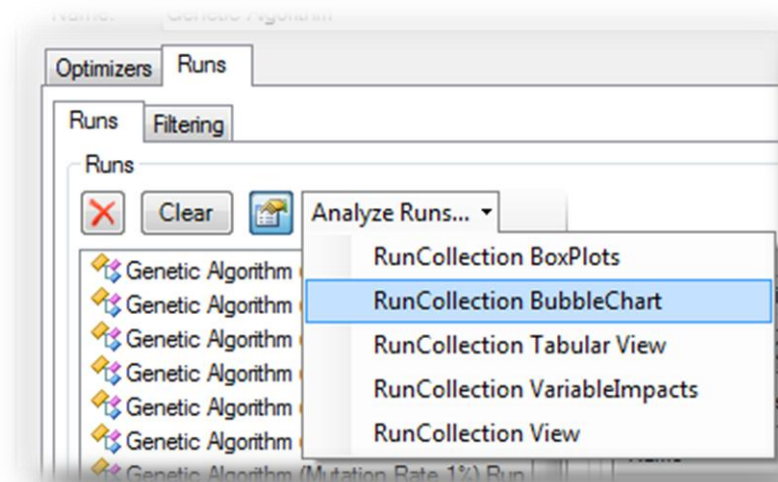
The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.5.6558 [C:\Users\p21298\Documents\Demo Experiment (Results).hl]". The interface is divided into several sections:

- Runs List:** A list of runs is shown, each with a small icon and a description. The runs are filtered by "GA, TSP ch130, 0% Mutation Rate, Swap?". The list is highlighted with a red border.
- Details Panel:** The details for a selected run are shown. The name is "GA, TSP ch130, 5% Mutation Rate, InversionManipulator Run 1". The color is set to black.
- Parameters & Results:** A list of results is shown, including "AbsoluteDifferenceBestK", "Best TSP Solution", "BestKnownQuality", "BestQuality", "CurrentAverageQuality", "CurrentBestQuality", "CurrentWorstQuality", "EvaluatedSolutions", "Execution Time", "Generations", "Qualities", and "RelativeDifferenceBestK".
- Qualities Graph:** A line graph titled "Qualities" showing the performance of the run. The x-axis represents "Generations" (0 to 10000) and the y-axis represents "Quality" (0 to 60000). The graph shows four data series: "CurrentBestQuality" (blue), "CurrentAverageQuality" (orange), "CurrentWorstQuality" (red), and "BestKnownQuality" (grey). The quality starts high and rapidly decreases, stabilizing around 10000 after approximately 2000 generations.

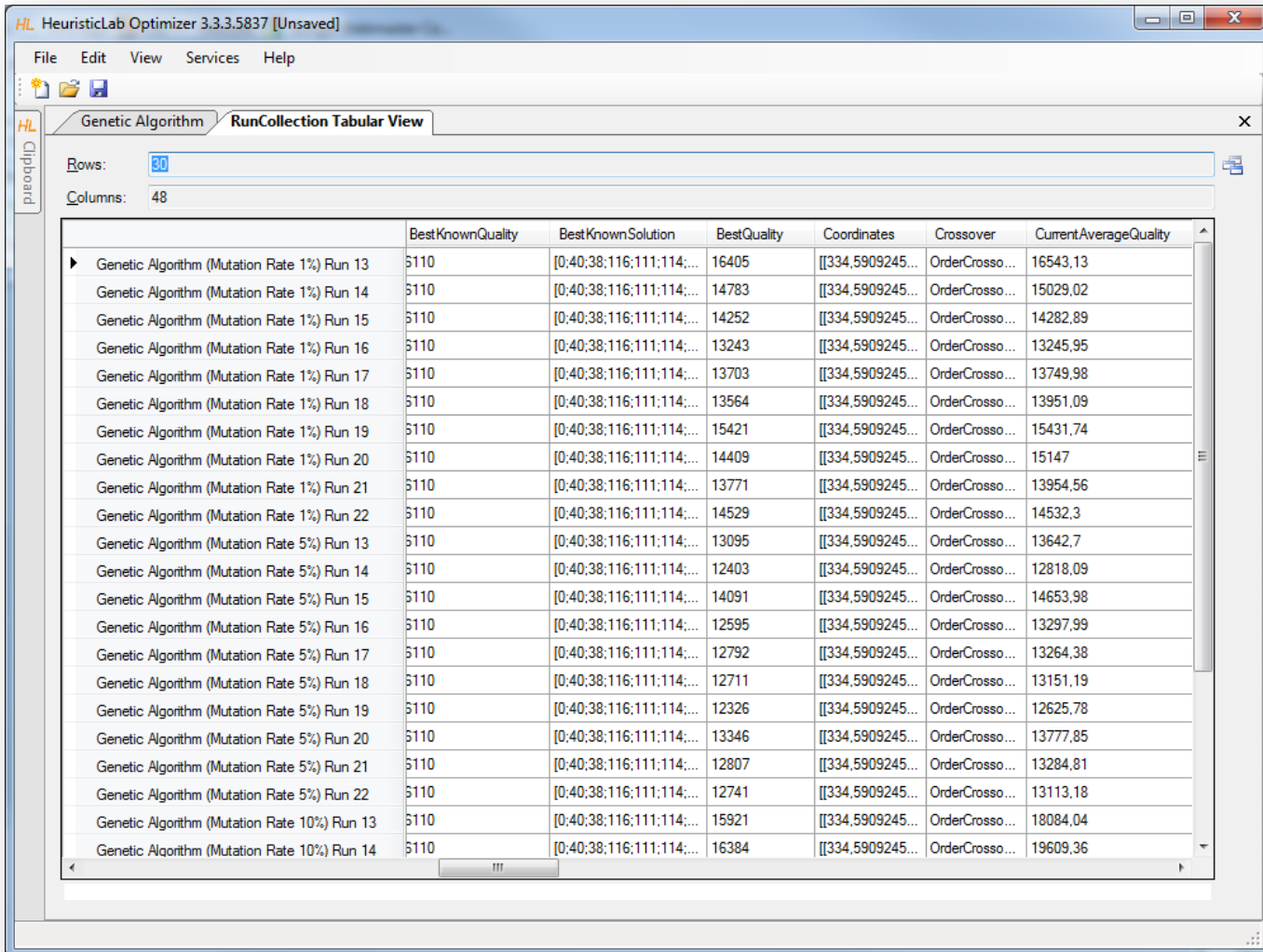
Execution Time: 03:55:16.2570011

# 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



# RunCollection Tabular View



HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

HL Clipboard

Genetic Algorithm RunCollection Tabular View

Rows: 30

Columns: 48

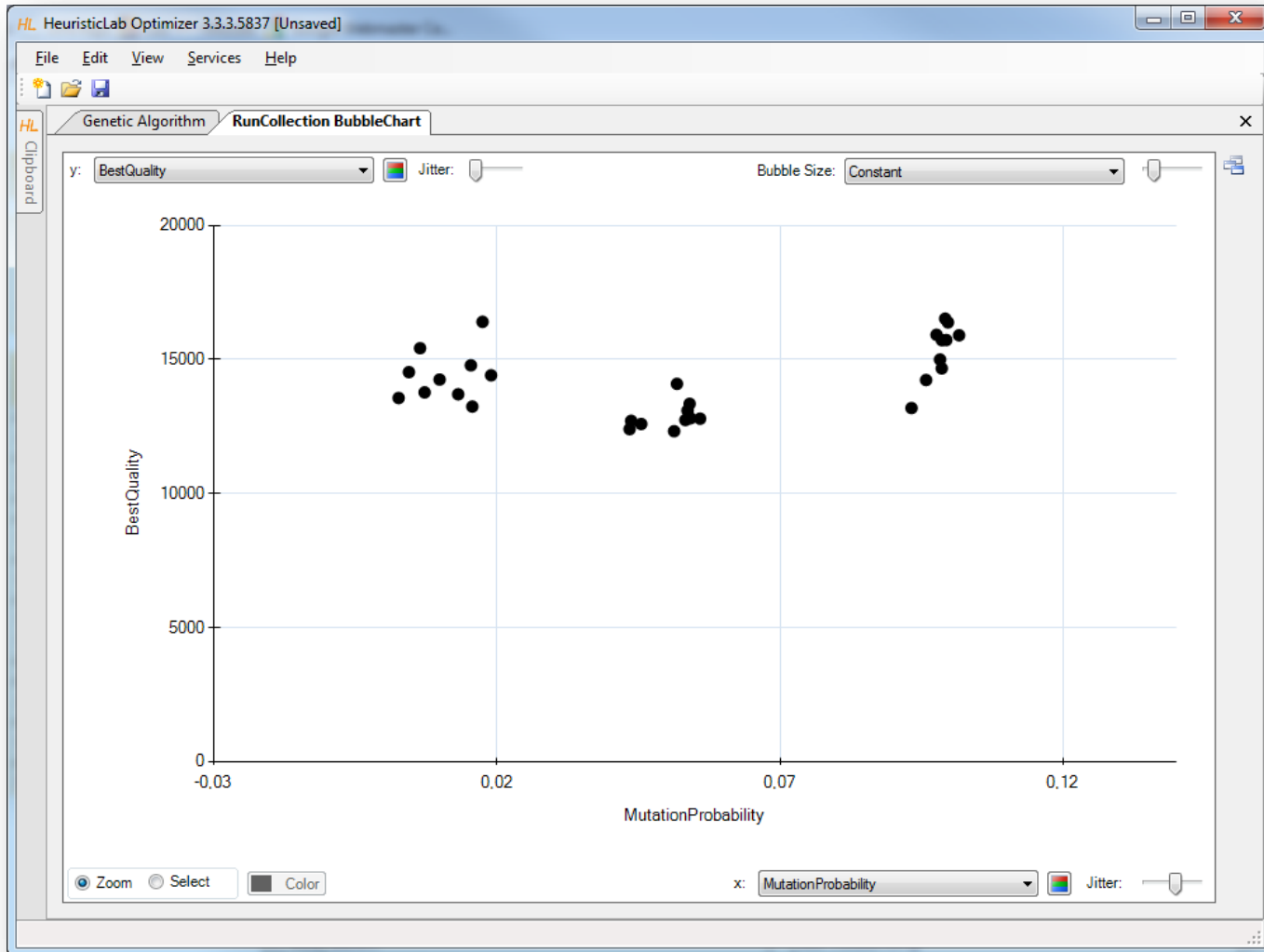
	BestKnownQuality	BestKnownSolution	BestQuality	Coordinates	Crossover	CurrentAverageQuality
▶ Genetic Algorithm (Mutation Rate 1%) Run 13	5110	[0;40;38;116;111;114;...	16405	[[334,5909245...	OrderCrosso...	16543,13
Genetic Algorithm (Mutation Rate 1%) Run 14	5110	[0;40;38;116;111;114;...	14783	[[334,5909245...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

# RunCollection Tabular View



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

# RunCollection BubbleChart

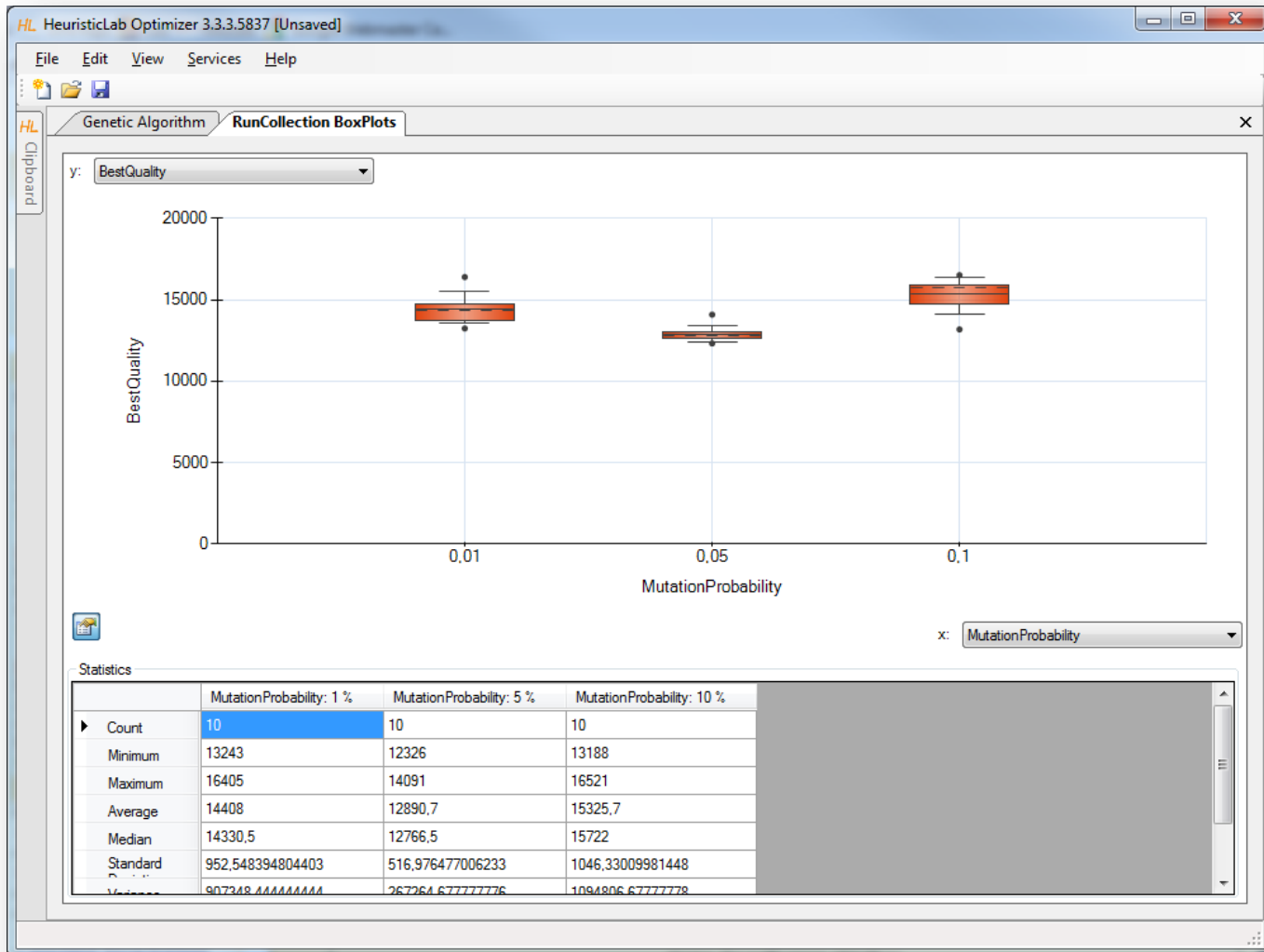


# RunCollection BubbleChart

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



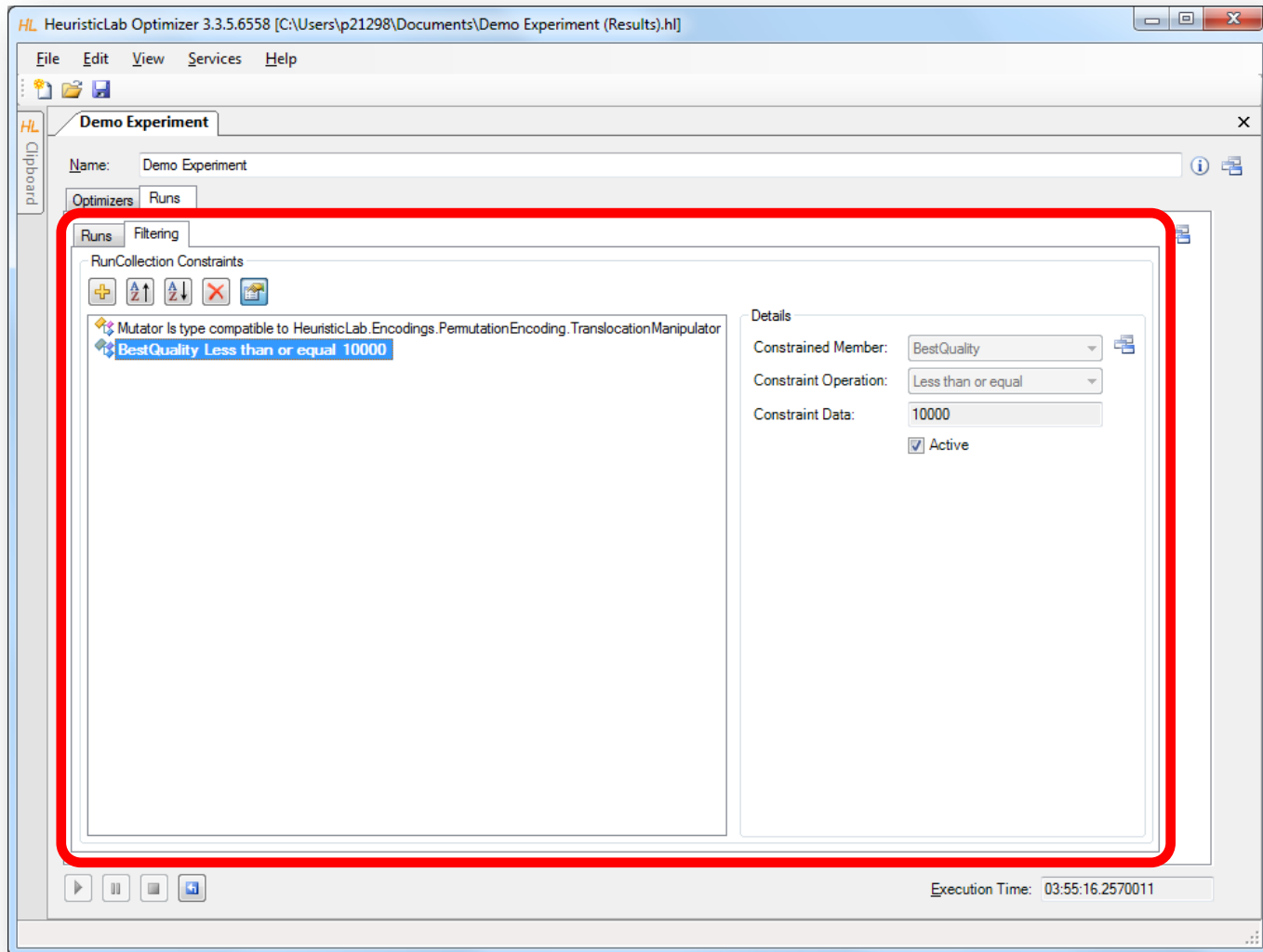
# RunCollection BoxPlots



# RunCollection BoxPlots

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

# Filter Runs

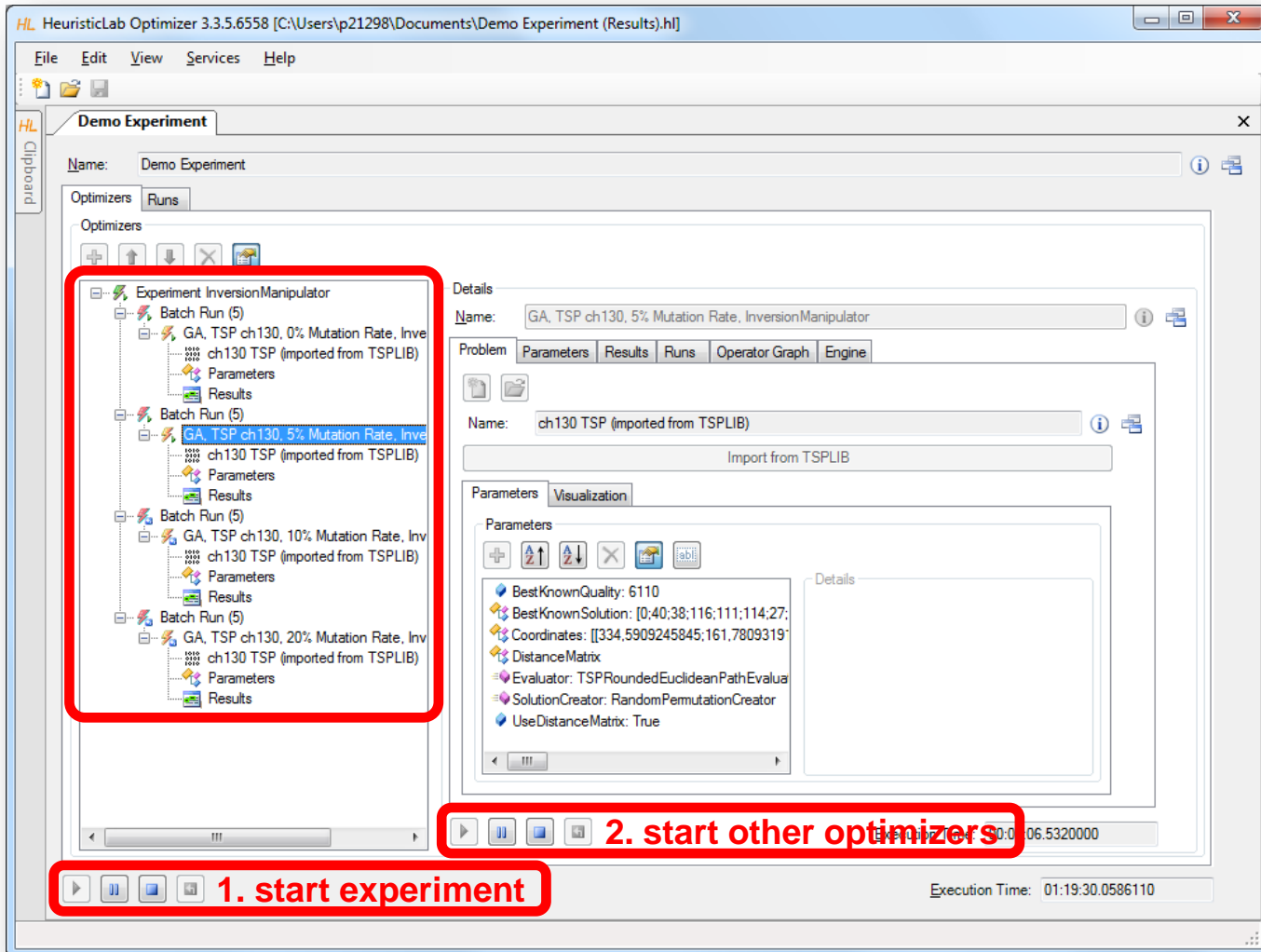


# Multi-core CPUs and Parallelization



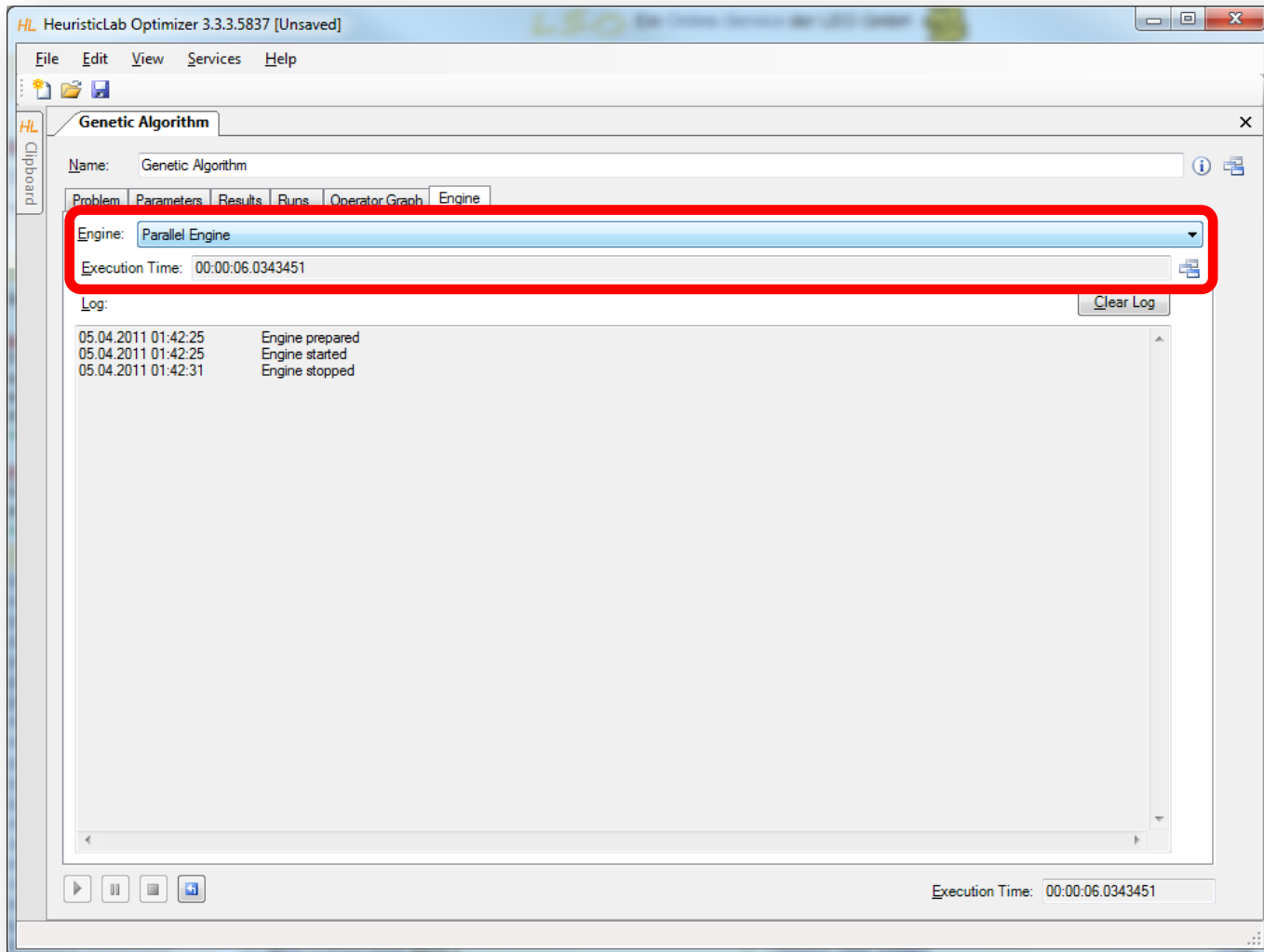
- Parallel execution of optimizers in experiments
  - optimizers in an experiment are executed sequentially from top to bottom per default
  - experiments support parallel execution of their optimizers
  - select a not yet executed optimizer and start it manually to utilize another core
  - execution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

# Parallel Execution of Experiments



The screenshot shows the HeuristicLab Optimizer interface. The main window is titled "Demo Experiment" and contains a tree view of optimizers and runs. A red box highlights the "Experiment InversionManipulator" node, which is expanded to show four "Batch Run (5)" entries, each representing a different mutation rate (0%, 5%, 10%, and 20%) for the "GA, TSP ch130" optimizer. A second red box highlights the "2. start other optimizers" button in the bottom right corner. A third red box highlights the "1. start experiment" button in the bottom left corner. The "Details" panel on the right shows the configuration for the selected optimizer, including the problem name "ch130 TSP (imported from TSPLIB)" and various parameters such as "BestKnownQuality: 6110" and "Evaluator: TSPRoundedEuclideanPathEvalua". The status bar at the bottom indicates the execution time as "01:19:30.0586110".

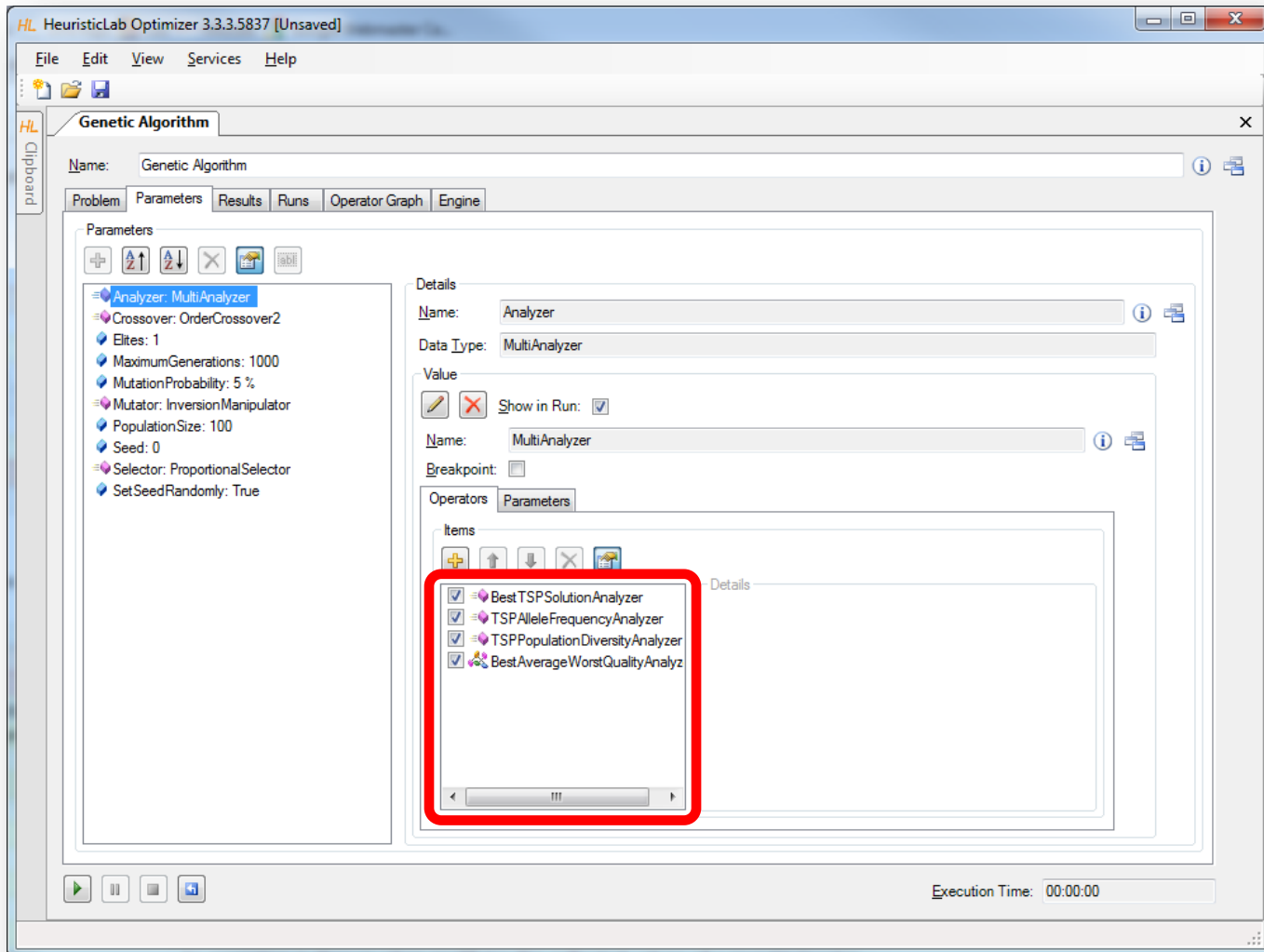
# Parallel Execution of Algorithms



# Analyzers

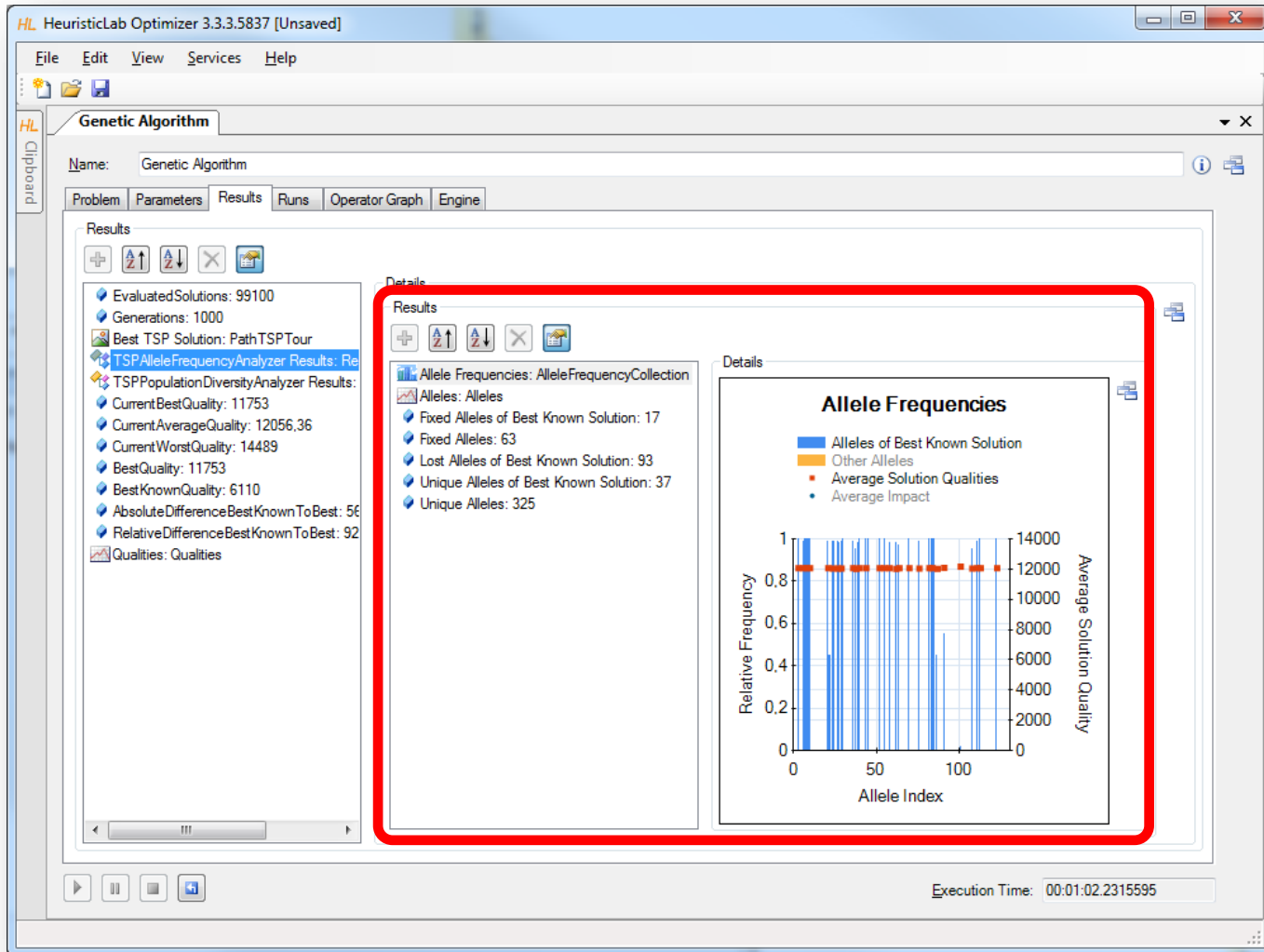
- Special operators for analysis purposes
  - are executed after each iteration
  - serve as general purpose extension points of algorithms
  - can be selected and parameterized in the algorithm
  - perform algorithm-specific and/or problem-specific tasks
  - some analyzers are quite costly regarding runtime and memory
  - implementing and adding custom analyzers is easy
- Examples
  - TSPAlleleFrequencyAnalyzer
  - TSPPopulationDiversityAnalyzer
  - SuccessfulOffspringAnalyzer
  - SymbolicDataAnalysisVariableFrequencyAnalyzer
  - SymbolicRegressionSingleObjectiveTrainingBestSolutionAnalyzer
  - ...

# Analyzers

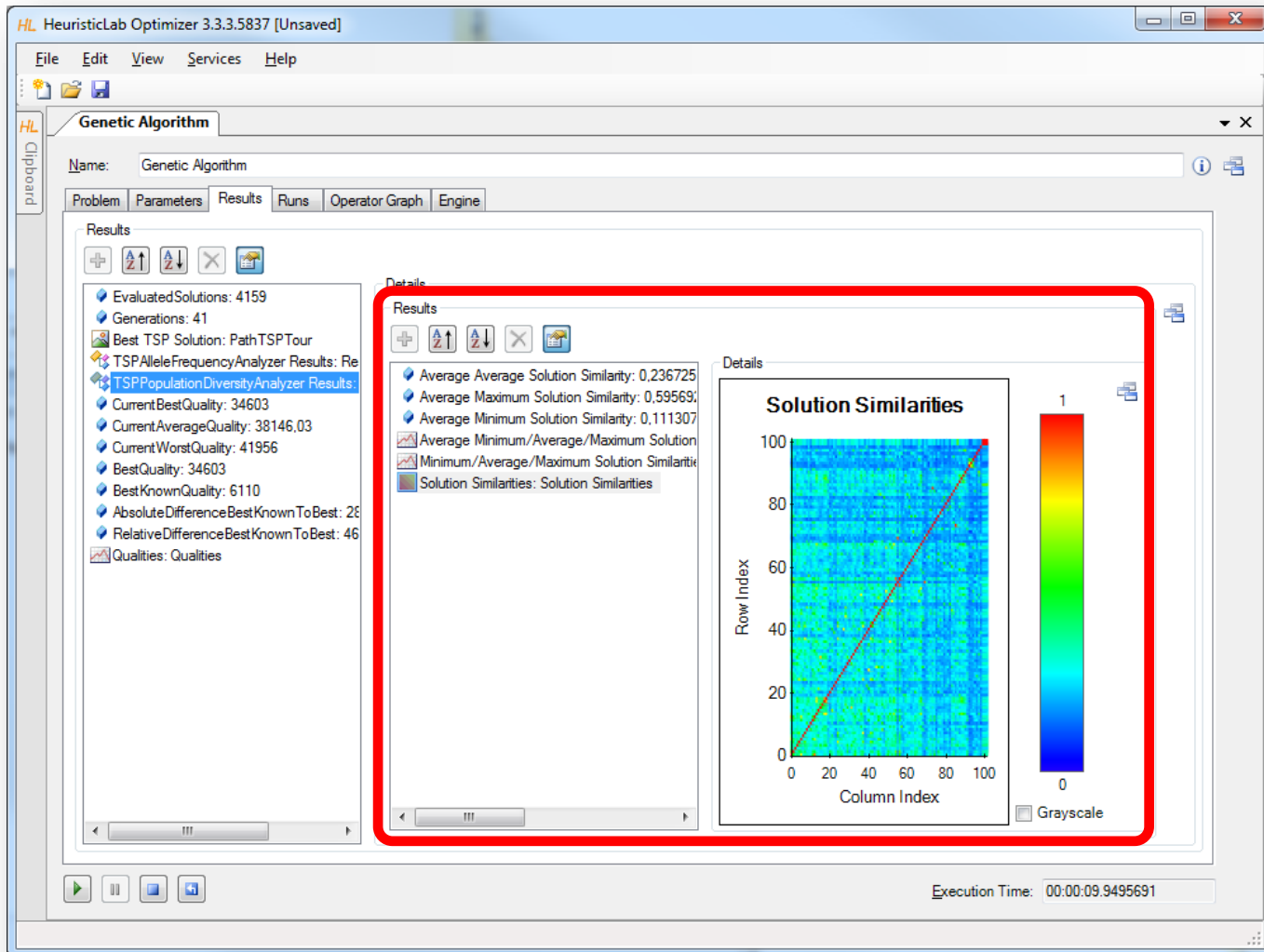




# TSPAlleleFrequencyAnalyzer

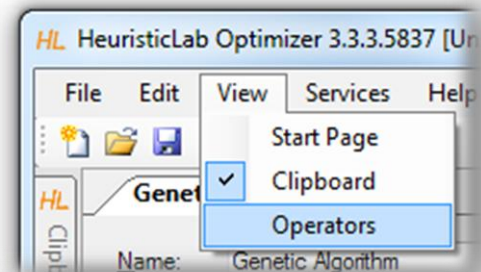
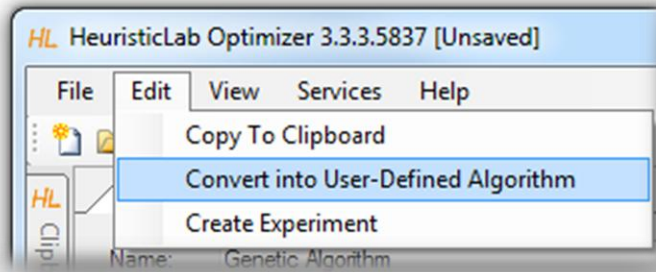


# TSP Population Diversity Analyzer



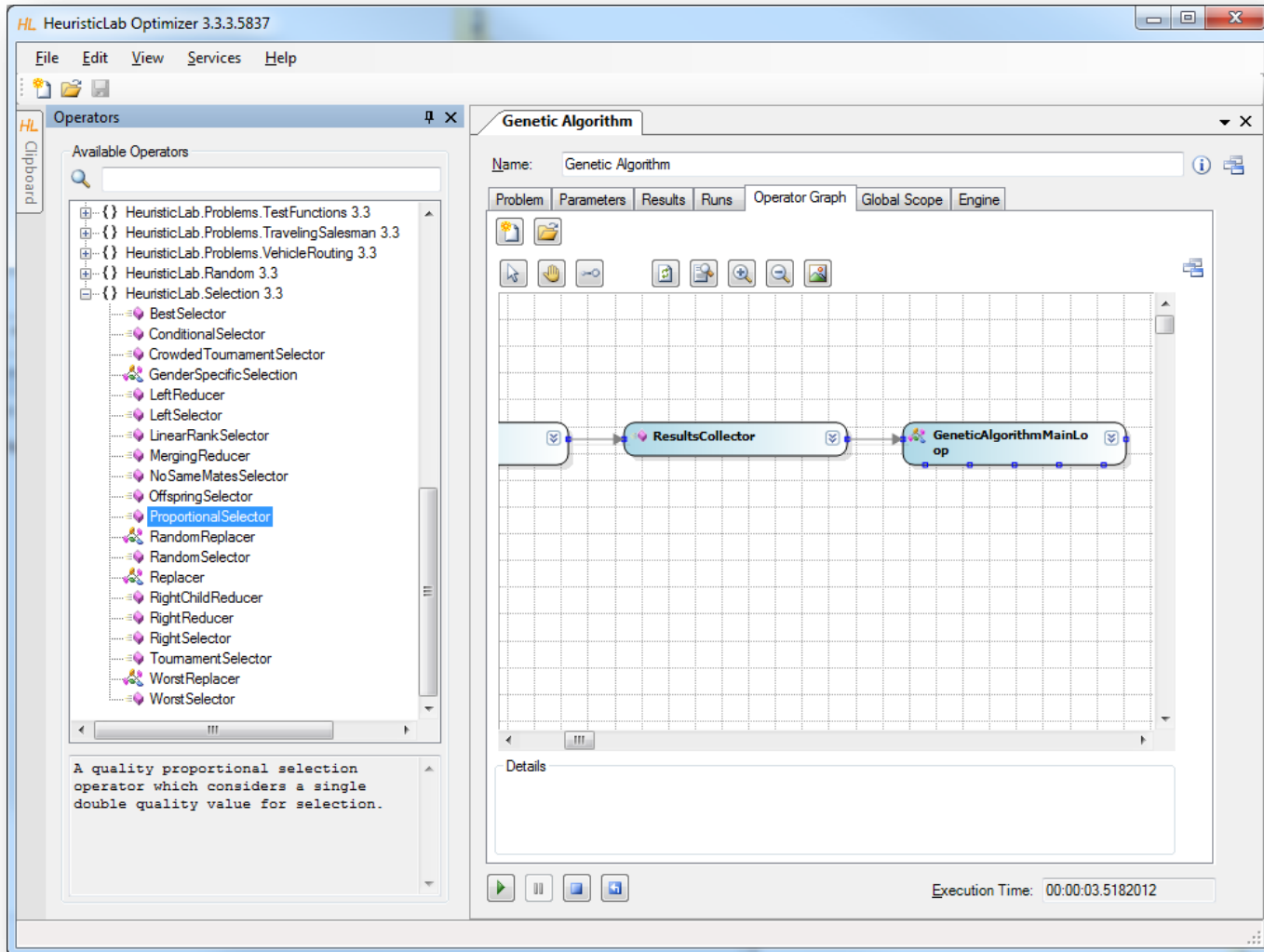
# Building User-Defined Algorithms

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



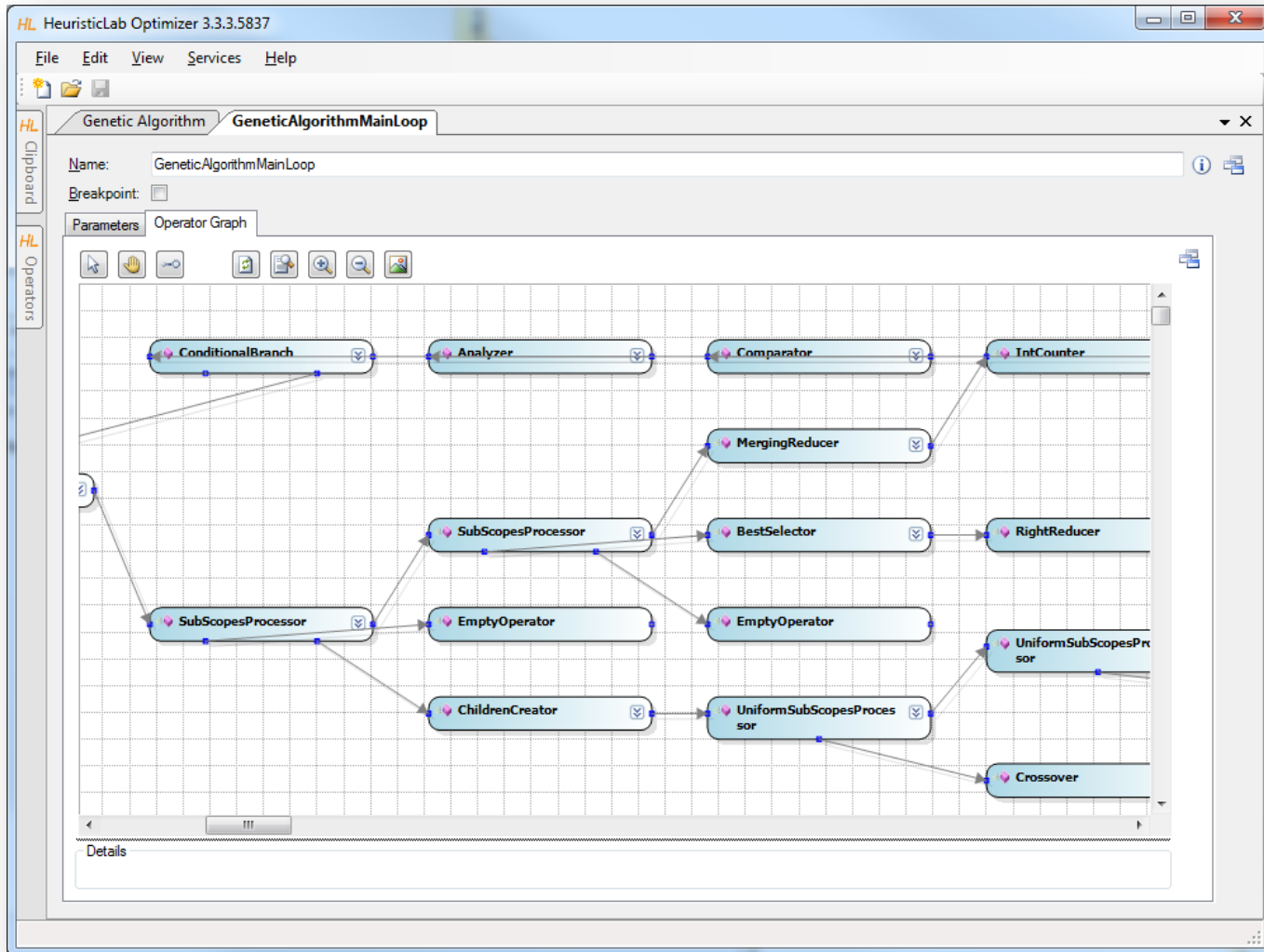
- Operators sidebar
  - drag & drop operators into an operator graph
- Programmable operators
  - add programmable operators in order to implement custom logic in an algorithm
  - no additional development environment needed
- Debug algorithms
  - use the debug engine to obtain detailed information during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

# Building User-Defined Algorithms

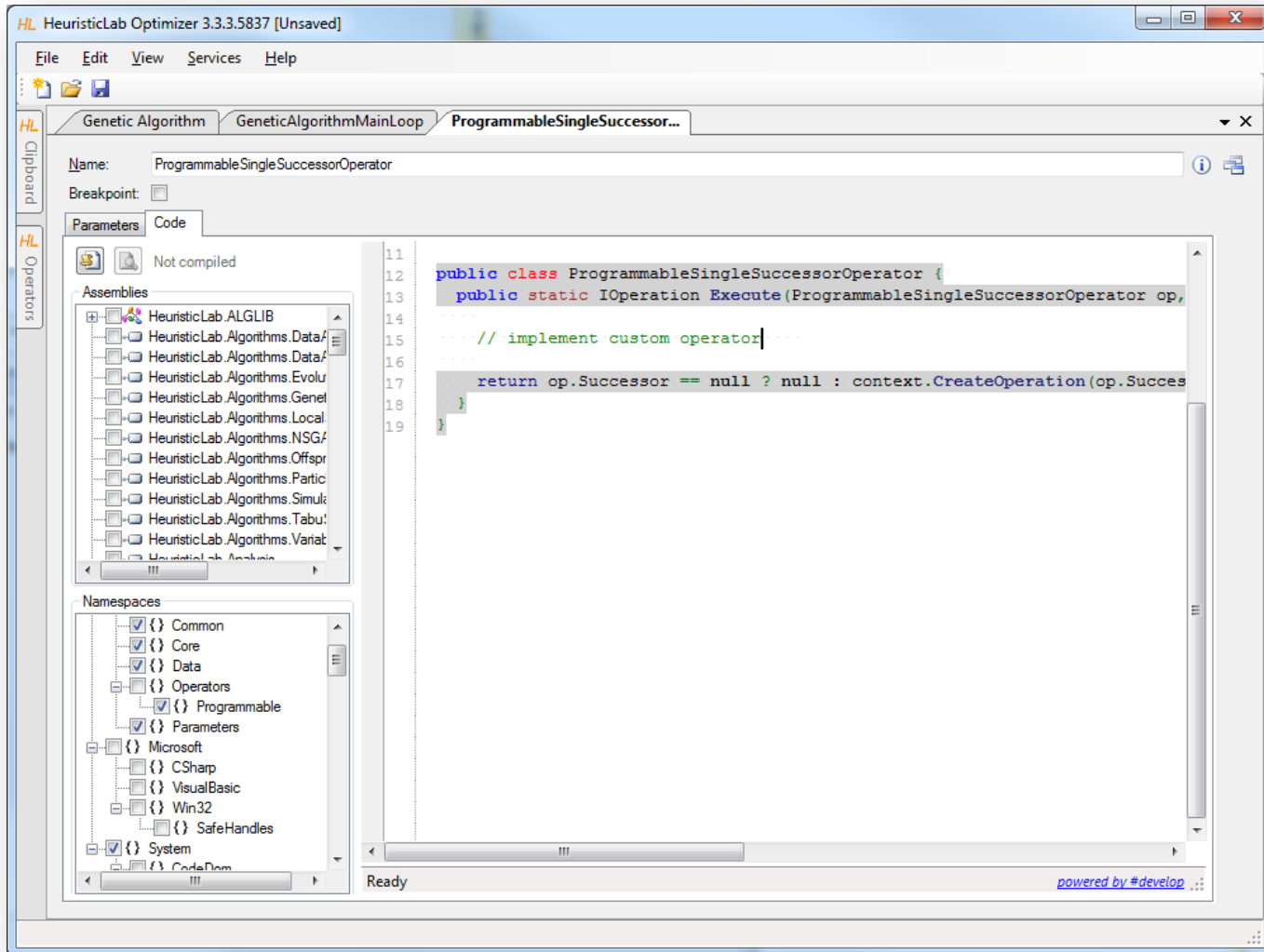


The screenshot displays the HeuristicLab Optimizer 3.3.3.5837 interface. The main window is titled "Genetic Algorithm" and shows a configuration panel with tabs for "Problem", "Parameters", "Results", "Runs", "Operator Graph", "Global Scope", and "Engine". The "Operator Graph" tab is active, showing a flow diagram with two operators: "ResultsCollector" and "GeneticAlgorithmMainLoop". The "Available Operators" list on the left includes various selection and reduction operators, with "ProportionalSelector" highlighted. A description for "ProportionalSelector" is visible at the bottom left: "A quality proportional selection operator which considers a single double quality value for selection." The "Execution Time" at the bottom right is 00:00:03.5182012.

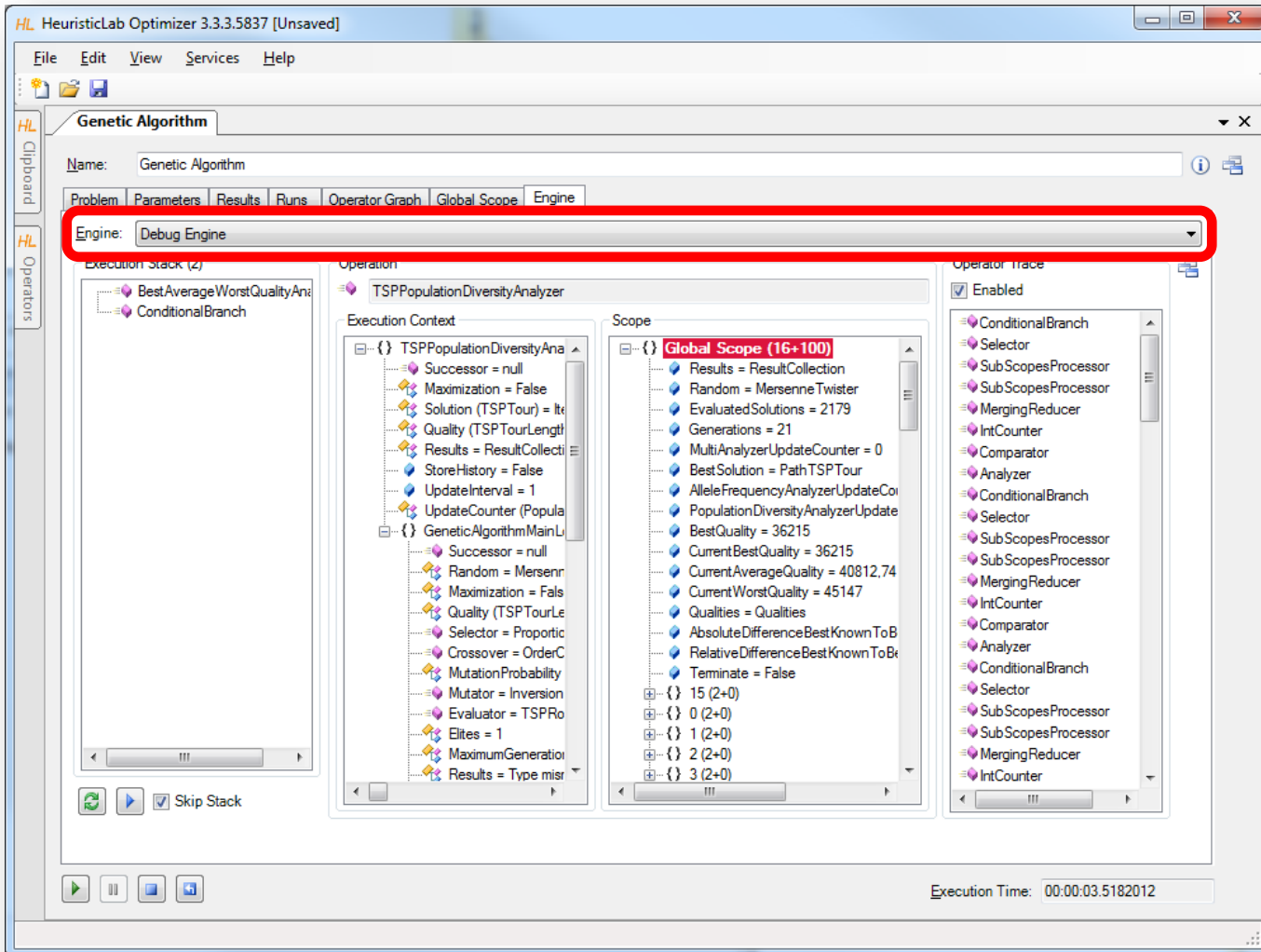
# Building User-Defined Algorithms



# Programmable Operators



# Debugging Algorithms



The screenshot displays the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "Genetic Algorithm" configuration is shown, with the "Engine" dropdown menu highlighted in red, set to "Debug Engine". The "Execution Stack (2)" panel shows the following components:

- BestAverageWorstQualityAnalyzer
- ConditionalBranch

The "Operation" panel shows the "TSPPopulationDiversityAnalyzer" configuration. The "Execution Context" and "Scope" panels are also visible. The "Scope" panel shows the following configuration:

- Global Scope (16+100)
- Results = ResultCollection
- Random = Mersenne Twister
- EvaluatedSolutions = 2179
- Generations = 21
- MultiAnalyzerUpdateCounter = 0
- BestSolution = PathTSPTour
- AlleleFrequencyAnalyzerUpdateCounter = 0
- PopulationDiversityAnalyzerUpdateCounter = 0
- BestQuality = 36215
- CurrentBestQuality = 36215
- CurrentAverageQuality = 40812.74
- CurrentWorstQuality = 45147
- Qualities = Qualities
- AbsoluteDifferenceBestKnownToBest = 0
- RelativeDifferenceBestKnownToBest = 0
- Terminate = False
- 15 (2+0)
- 0 (2+0)
- 1 (2+0)
- 2 (2+0)
- 3 (2+0)

The "Operator Trace" panel shows the following operators:

- Enabled
- ConditionalBranch
- Selector
- SubScopesProcessor
- MergingReducer
- IntCounter
- Comparator
- Analyzer
- ConditionalBranch
- Selector
- SubScopesProcessor
- MergingReducer
- IntCounter
- Comparator
- Analyzer
- ConditionalBranch
- Selector
- SubScopesProcessor
- MergingReducer
- IntCounter

The "Execution Time" is 00:00:03.5182012.

# 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
- Validation techniques

# Introduction to Data-based Modeling

$$y = f(x, w) + \varepsilon$$

	x1	x2	x3	x4	x5	x6	y
Training	52.0085	58497.7	26.9956	24.0391	12.9005	66.2633	4.027
	52.9874	59181.1	27.0061	23.7483	12.7239	66.8367	4.154
	54.0505	58879.2	26.8679	23.5869	12.5877	67.6773	4.047
	55.3182	58982.3	27.1852	23.7869	12.6229	67.7128	4.118
	46.5388	59270.3	26.4421	24.2601	12.5892	65.7425	4.743
	56.0167	58234.5	26.1072	23.4412	12.4224	67.9462	4.042
	57.0168	58021.7	26.2513	23.3085	12.3036	69.4392	3.721
	57.0139	59228.5	26.2285	23.4041	12.3685	68.3322	3.593
	62.5214	49855.7	26.3076	22.7812	12.0711	70.3331	3.785
	69.9885	53534.4	25.5916	21.3448	11.0299	73.7138	3.028
	70.0156	55058.2	25.5759	21.4617	11.0915	72.9846	3.137
Test	70.0446	56099.4	26.059	21.4652	11.0684	73.2351	?
	70.0335	55891.6	26.1834	21.3318	10.6288	74.8925	?
	70.0164	56841.1	26.1791	21.2119	10.0354	74.8419	?
	70.0147	58254.6	26.7381	21.3367	10.9244	73.2341	?
	70.0061	54444.9	26.5786	21.2395	10.6505	74.9187	?
	70.0325	48858.2	26.3799	21.4135	10.8558	74.0993	?

# Introduction to Data-based Modeling



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

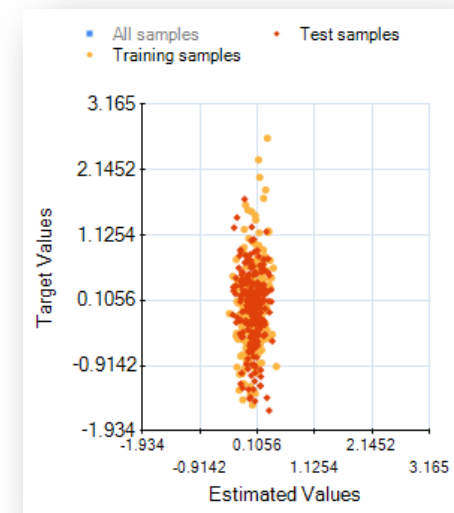
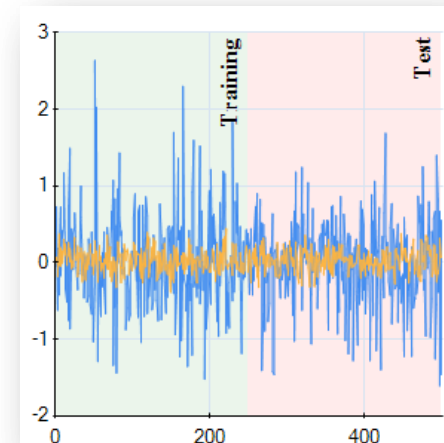
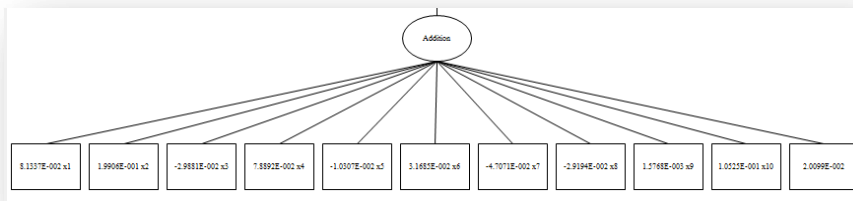
# Data-based Modeling Algorithms in HeuristicLab



- Symbolic regression and classification based on genetic programming
- External Libraries:
  - Support Vector Machines for Regression and Classification
  - Linear Regression
  - Linear Discriminate Analysis
  - K-Means clustering
  - Random Forests for Regression and Classification
  - Neural Networks

# 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

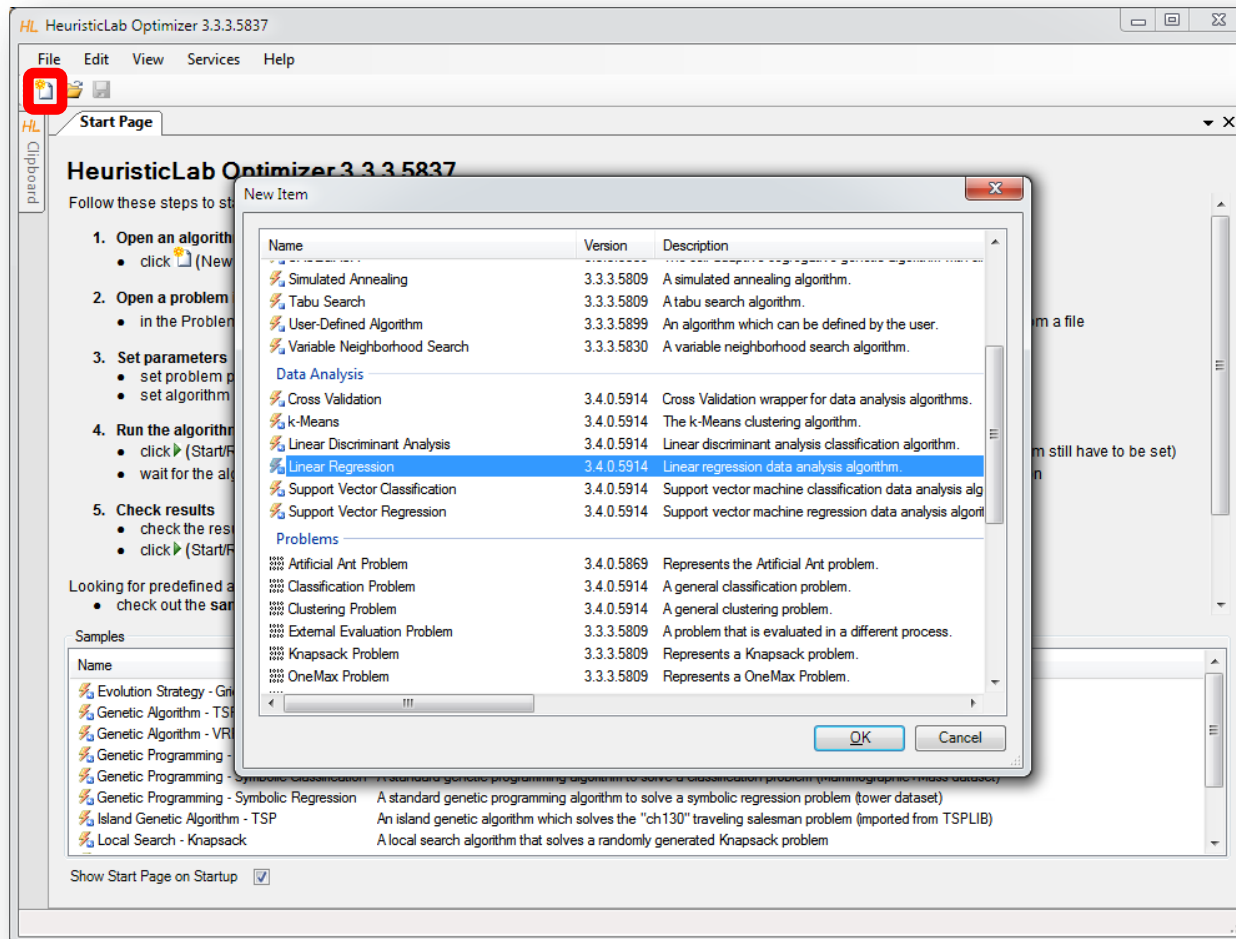


# Case Study: Regression

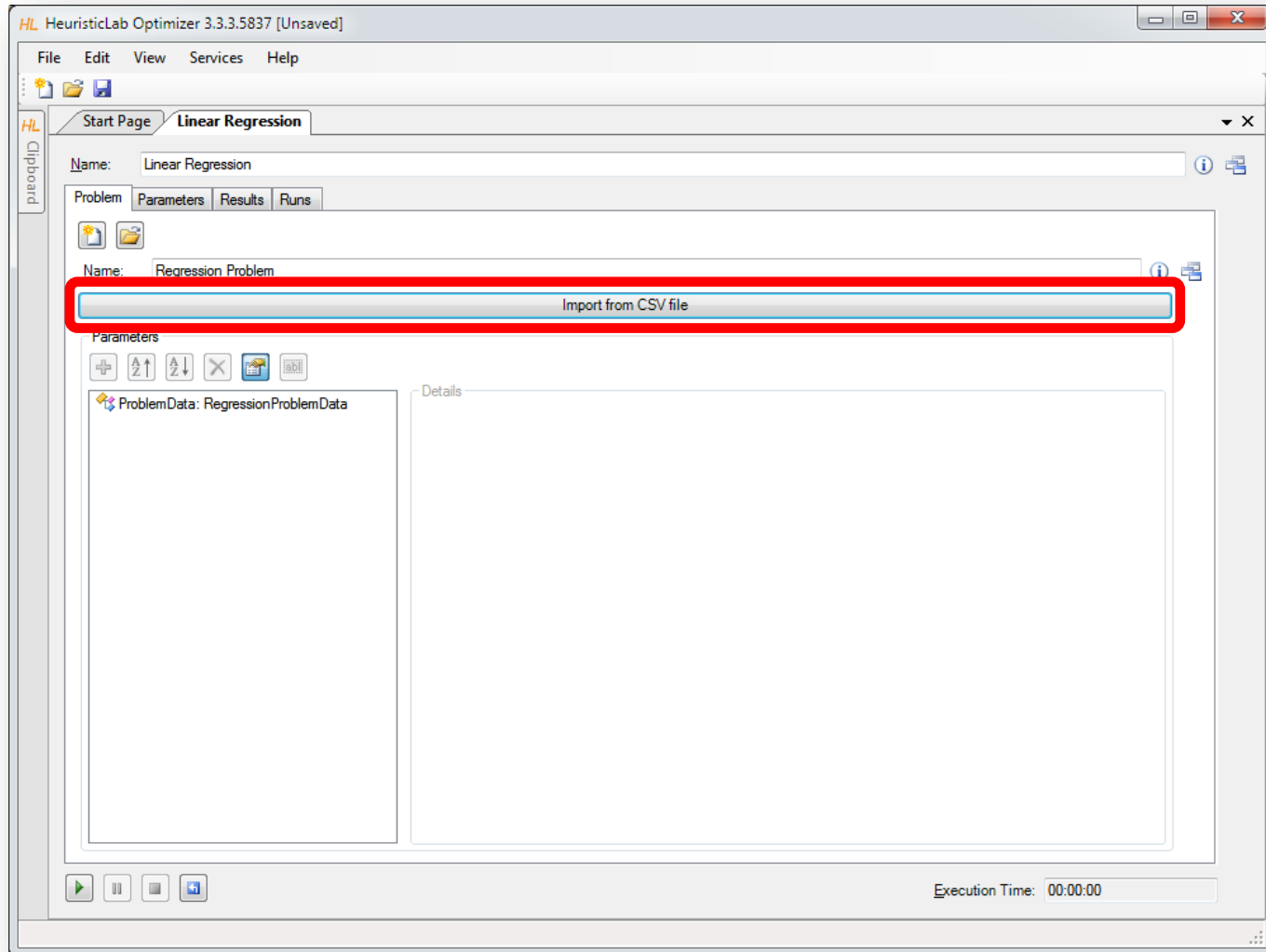
- Poly-10 benchmark problem dataset
  - 10 input variables  $x_1 \dots x_{10}$
  - $y = x_1 \cdot x_2 + x_3 \cdot x_4 + x_5 \cdot x_6 + x_1 \cdot x_7 \cdot x_9 + x_3 \cdot x_6 \cdot x_{10}$
  - non-linear modeling approach necessary
  - frequently used in GP literature
  - download  
<http://dev.heuristiclab.com/AdditionalMaterial#IMMM2011>

# Linear Regression

- Create new algorithm

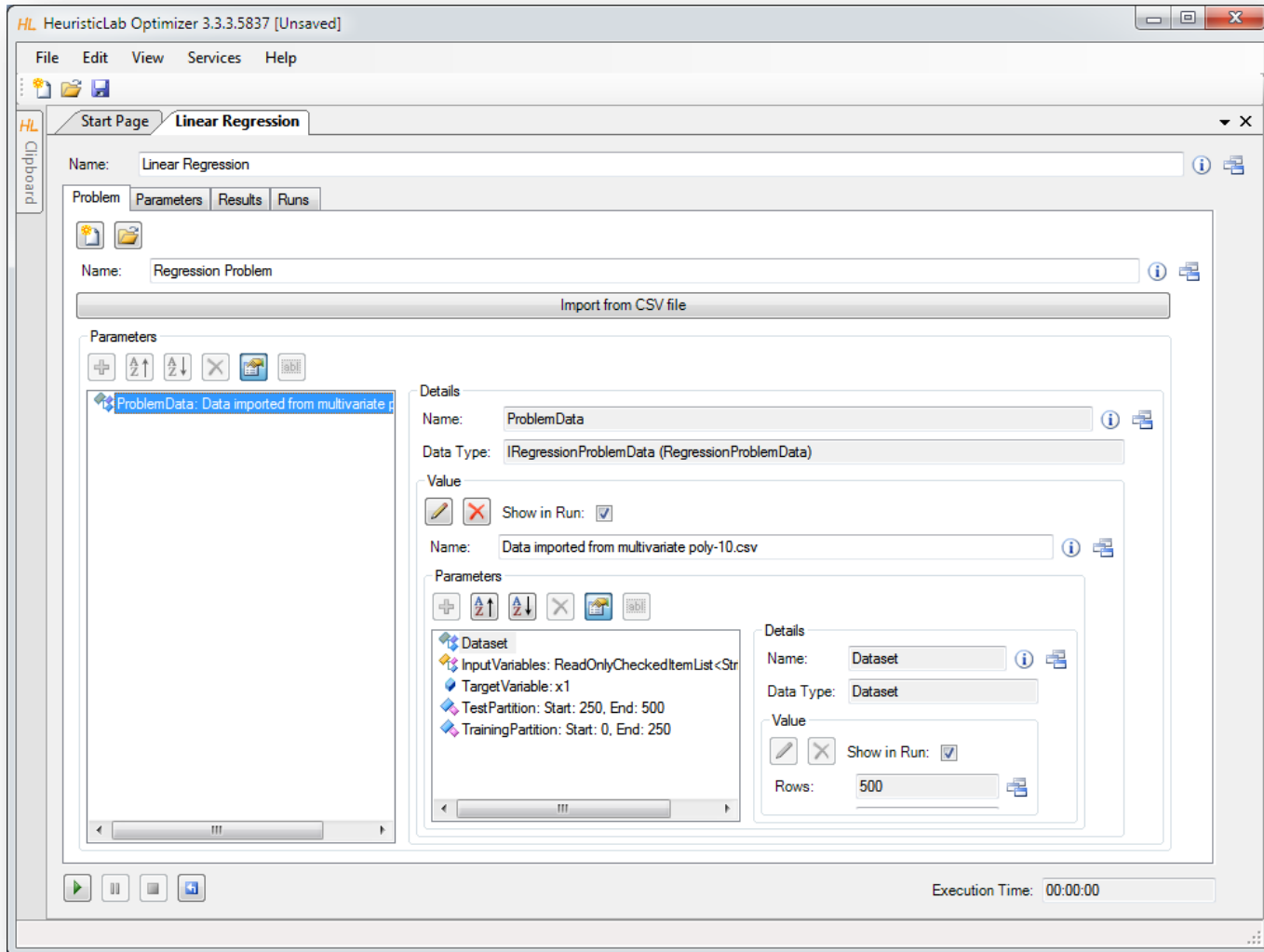


# Import Data from CSV-File

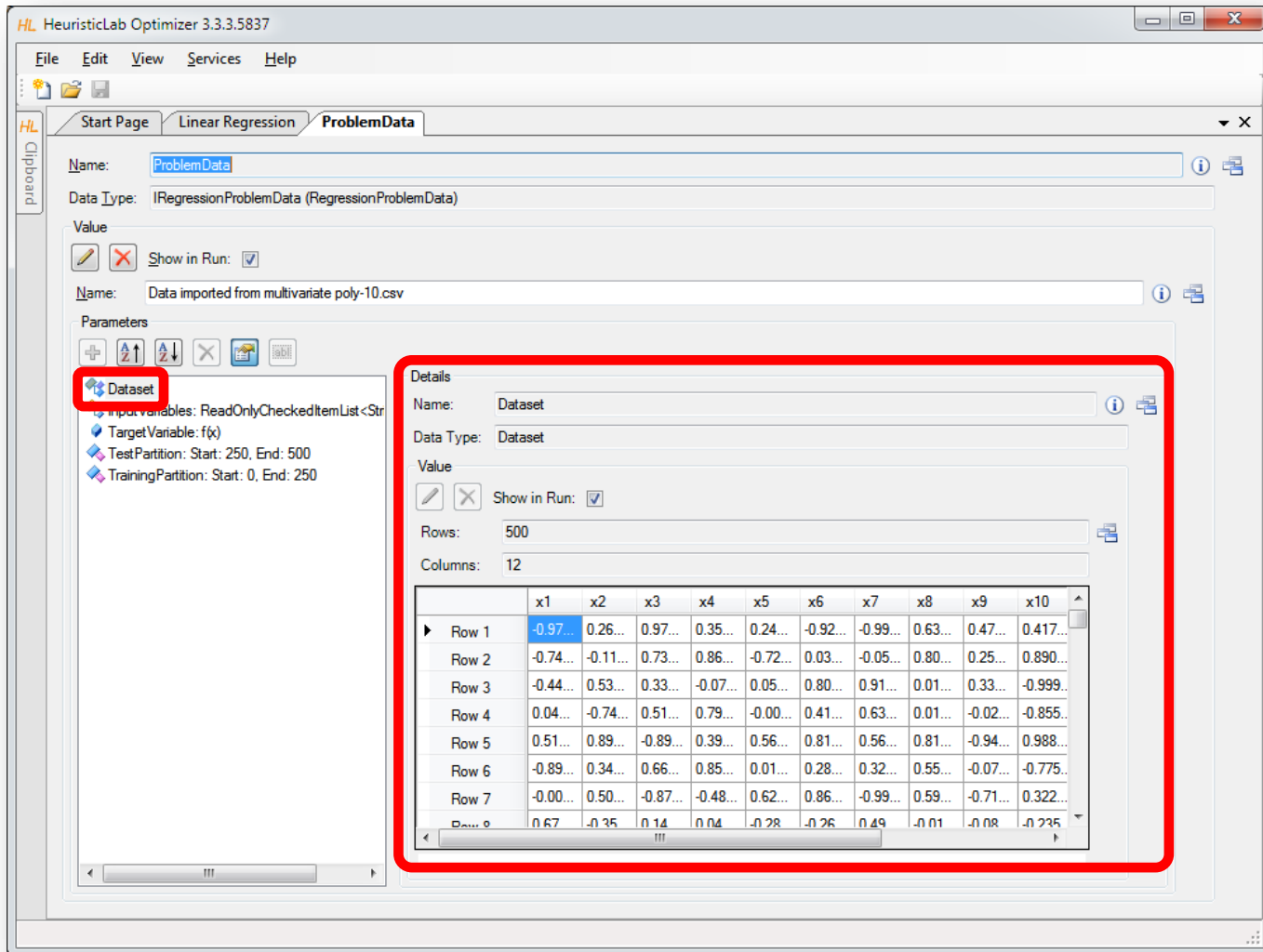




# Inspect and Configure Dataset



# Inspect Imported Data



HL HeuristicLab Optimizer 3.3.3.5837

File Edit View Services Help

Start Page Linear Regression **ProblemData**

Name: ProblemData

Data Type: IRegressionProblemData (RegressionProblemData)

Value

Show in Run:

Name: Data imported from multivariate poly-10.csv

Parameters

Dataset

Input variables: ReadOnlyCheckedItemList<Str

Target Variable: f(x)

TestPartition: Start: 250, End: 500

TrainingPartition: Start: 0, End: 250

Details

Name: Dataset

Data Type: Dataset

Value

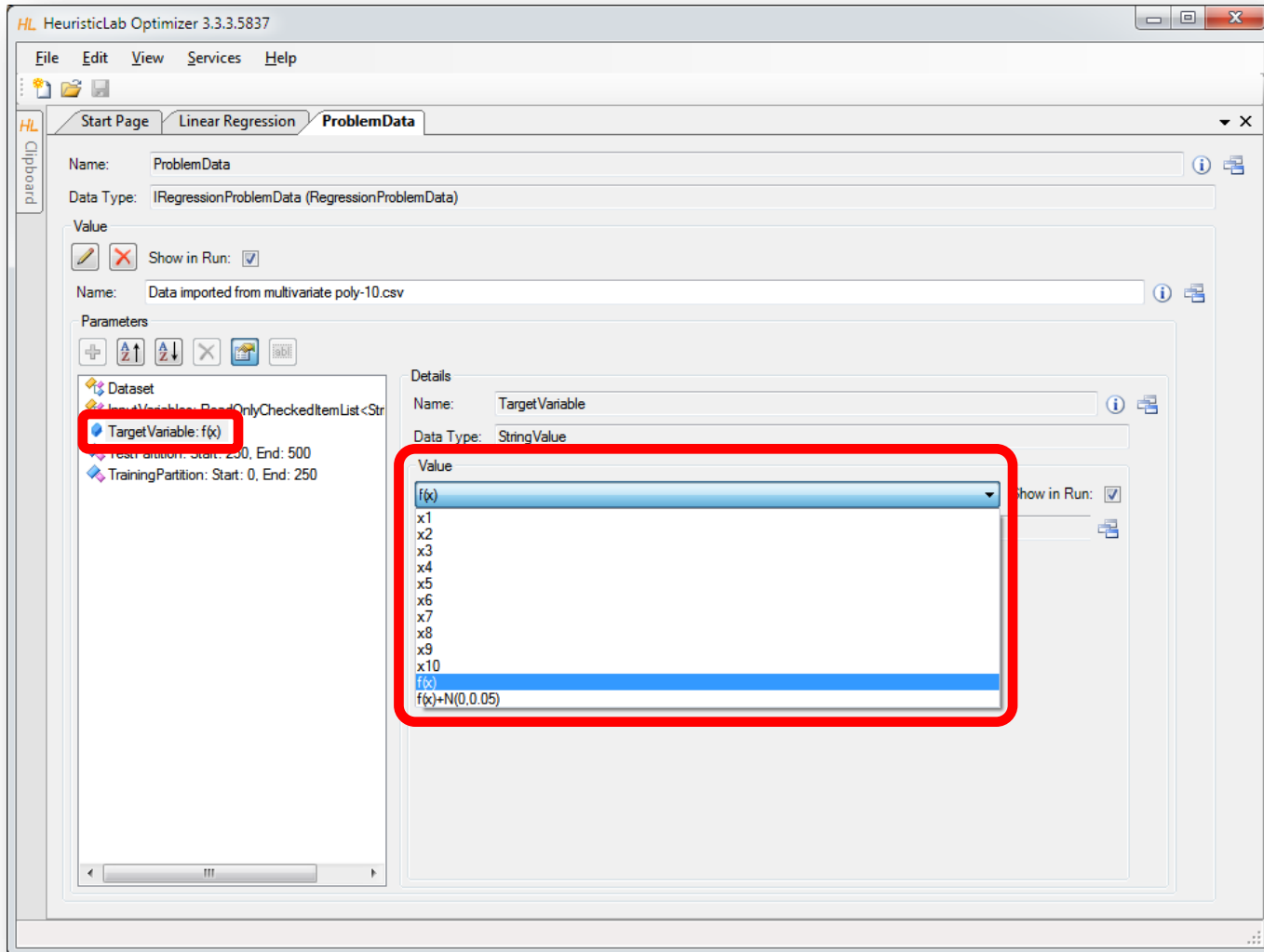
Show in Run:

Rows: 500

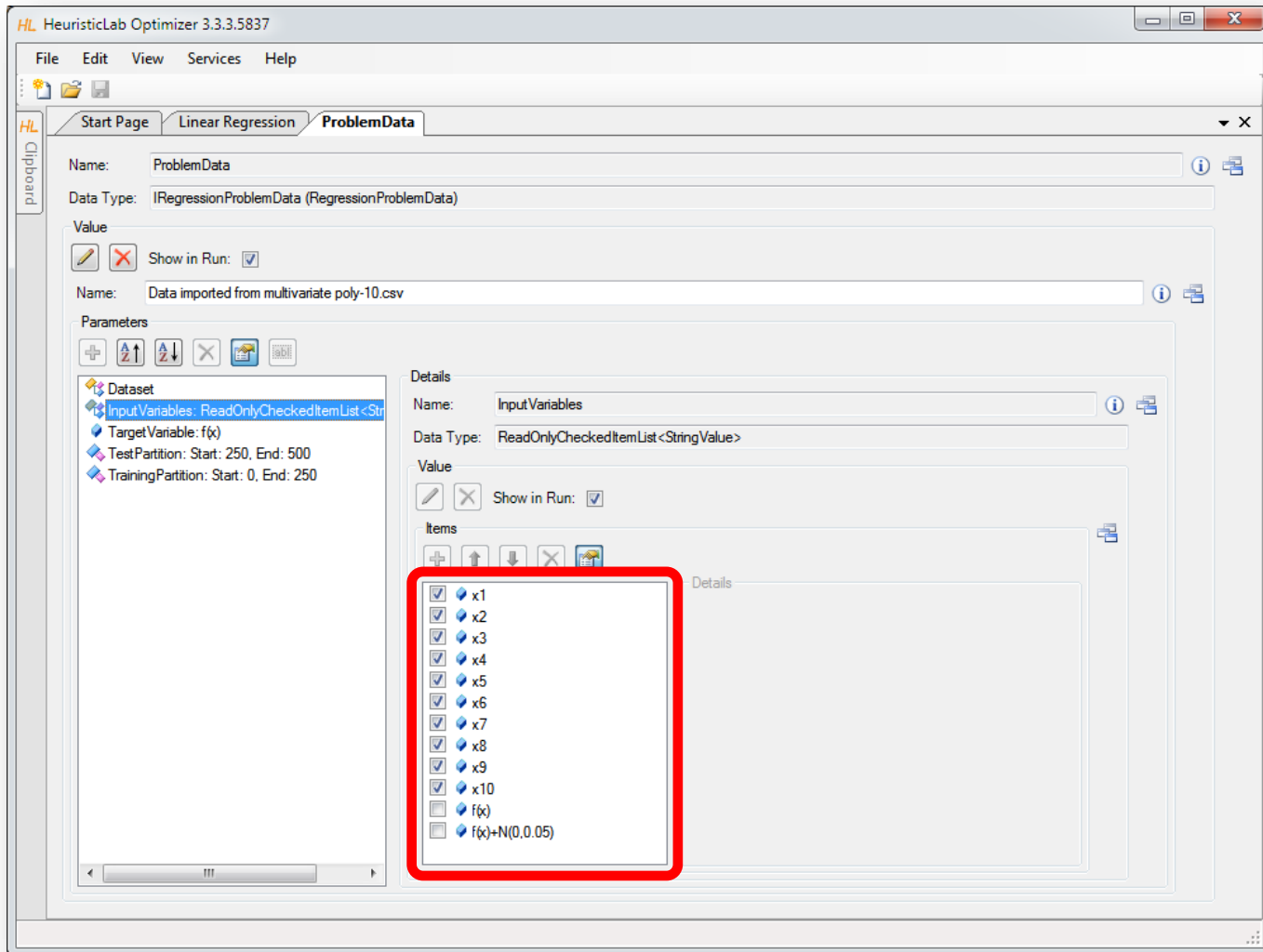
Columns: 12

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

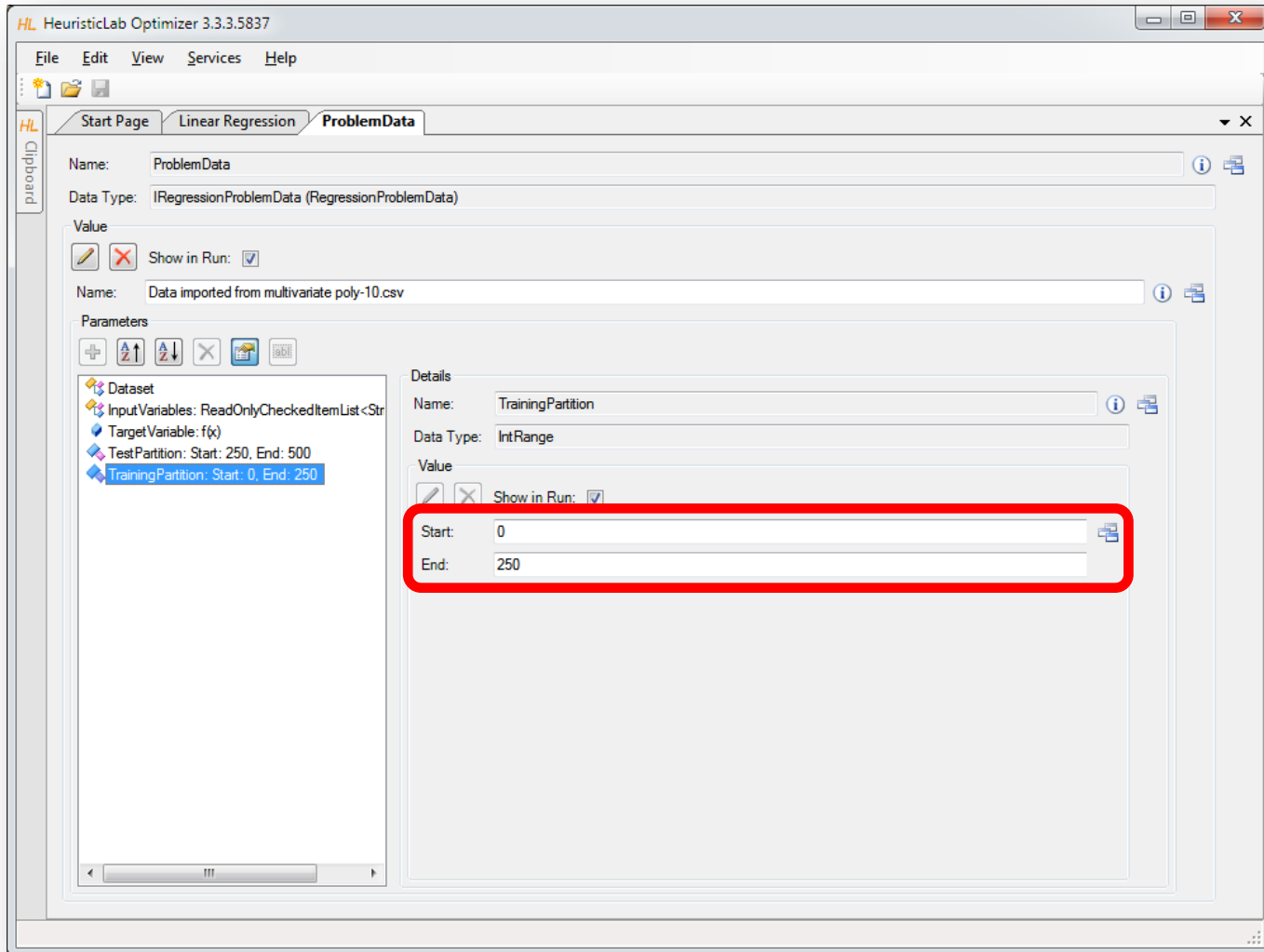
# Set Target Variable



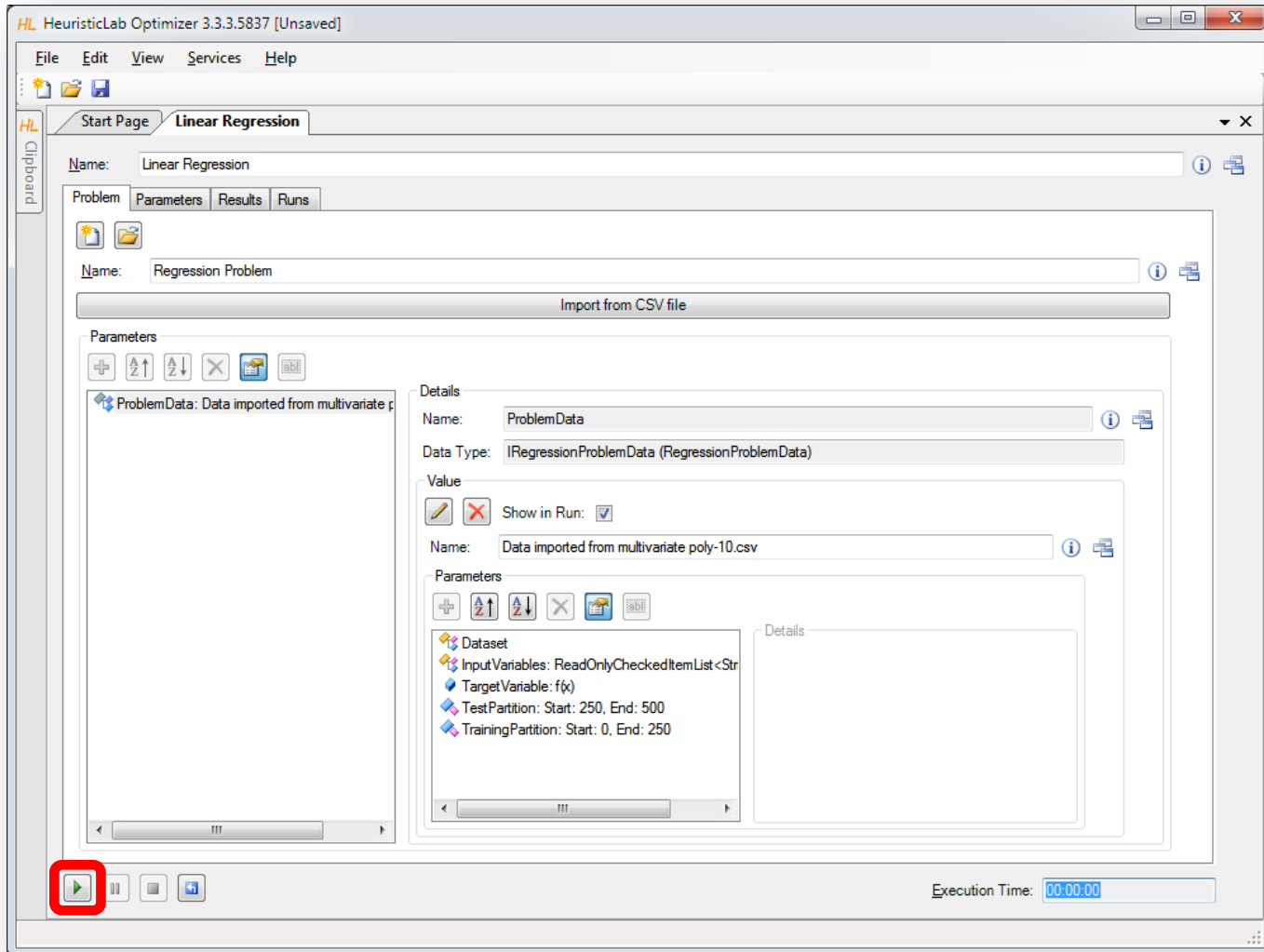
# Select Input Variables



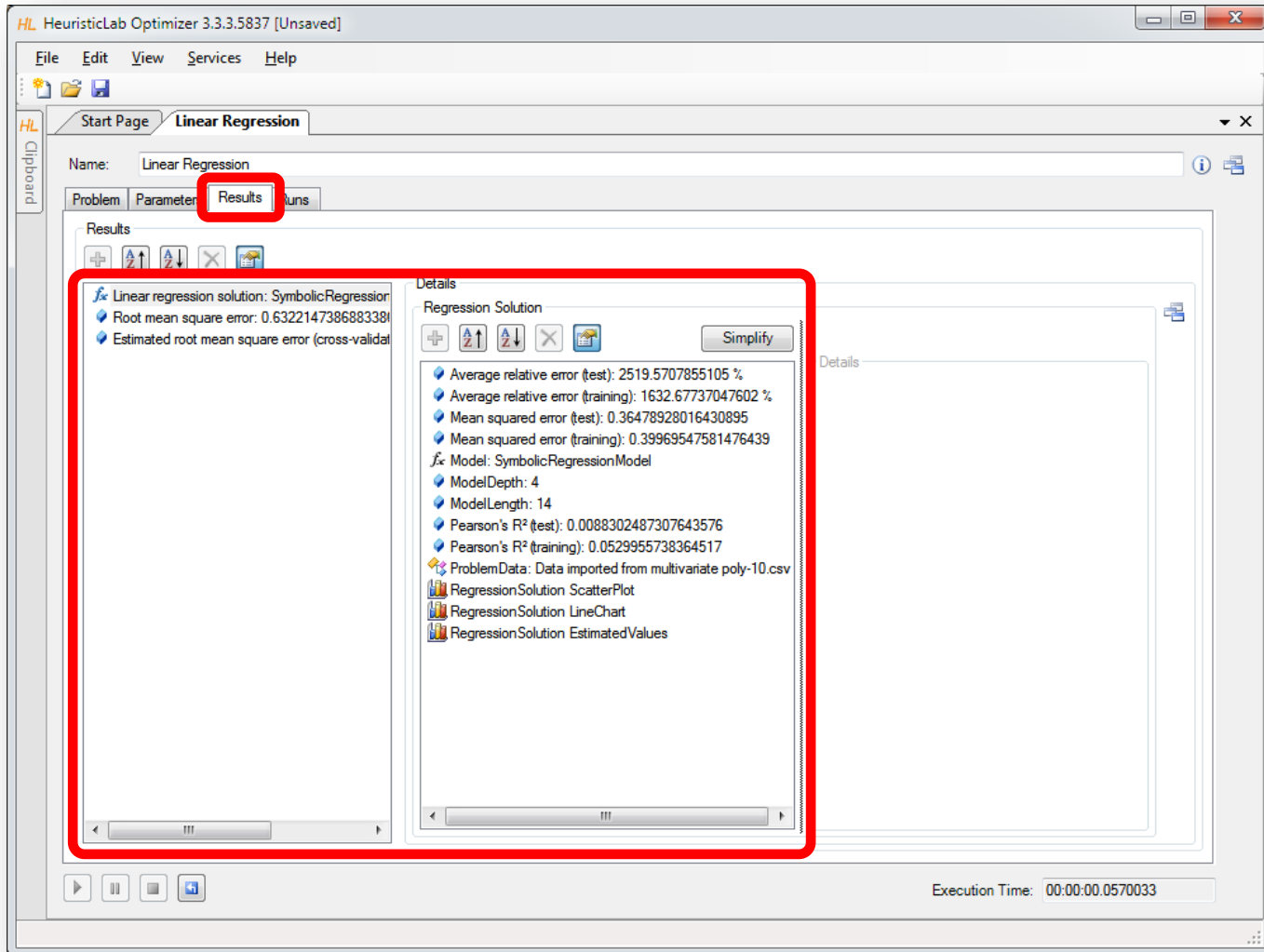
# Configure Training and Test Partitions



# Run Linear Regression



# Inspect Results



The screenshot displays the HeuristicLab Optimizer interface. The main window title is "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The menu bar includes "File", "Edit", "View", "Services", and "Help". The "Start Page" tab is active, and the "Linear Regression" problem is selected. The "Results" tab is highlighted with a red box. The "Results" panel shows the following information:

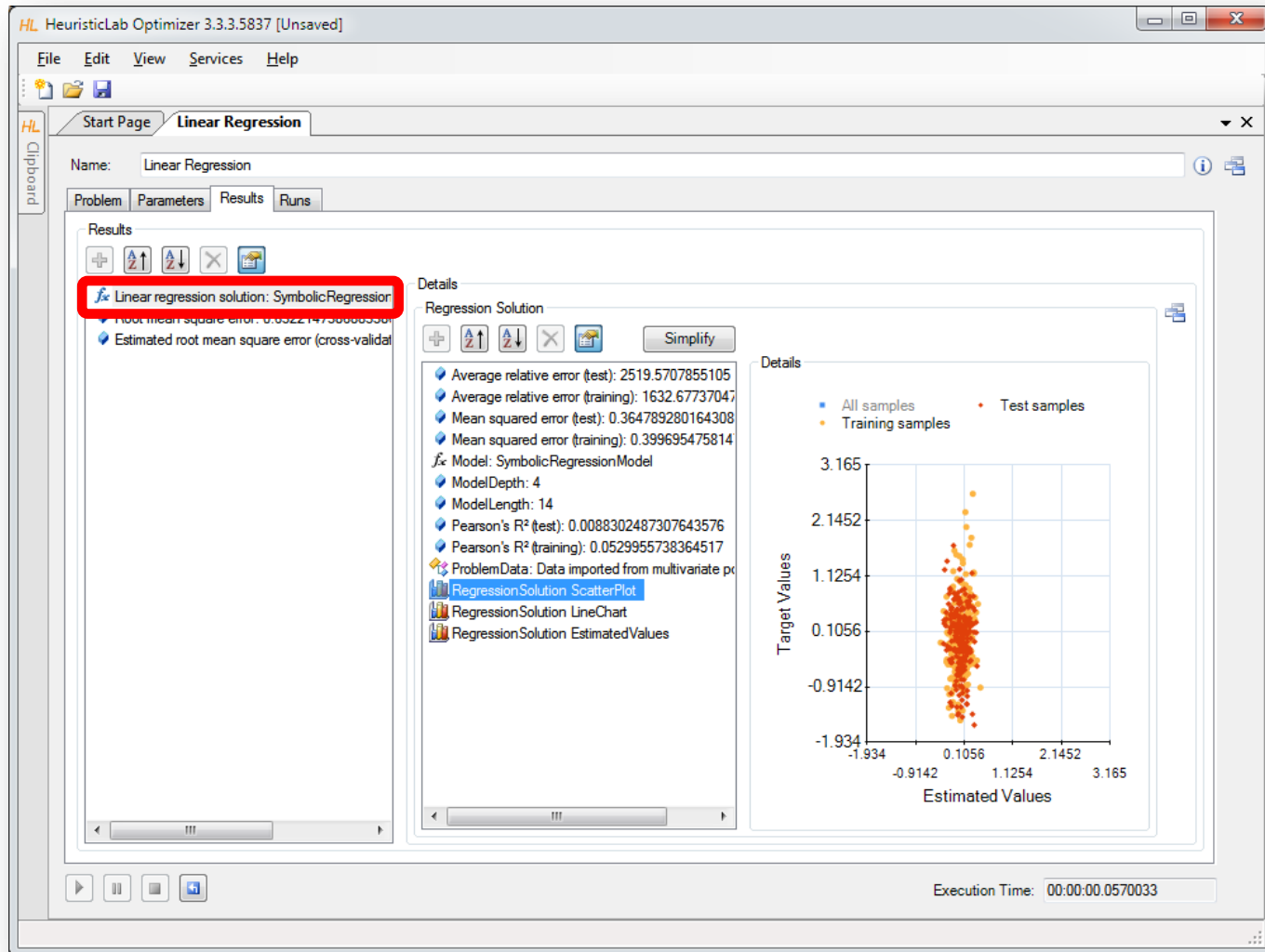
- Linear regression solution: SymbolicRegression
- Root mean square error: 0.632214738688338
- Estimated root mean square error (cross-validation): 0.632214738688338

The "Details" panel, also highlighted with a red box, provides a "Regression Solution" with the following metrics:

- Average relative error (test): 2519.5707855105 %
- Average relative error (training): 1632.67737047602 %
- Mean squared error (test): 0.36478928016430895
- Mean squared error (training): 0.39969547581476439
- Model: SymbolicRegressionModel
- ModelDepth: 4
- ModelLength: 14
- Pearson's R<sup>2</sup> (test): 0.0088302487307643576
- Pearson's R<sup>2</sup> (training): 0.0529955738364517
- ProblemData: Data imported from multivariate poly-10.csv
- RegressionSolution ScatterPlot
- RegressionSolution LineChart
- RegressionSolution EstimatedValues

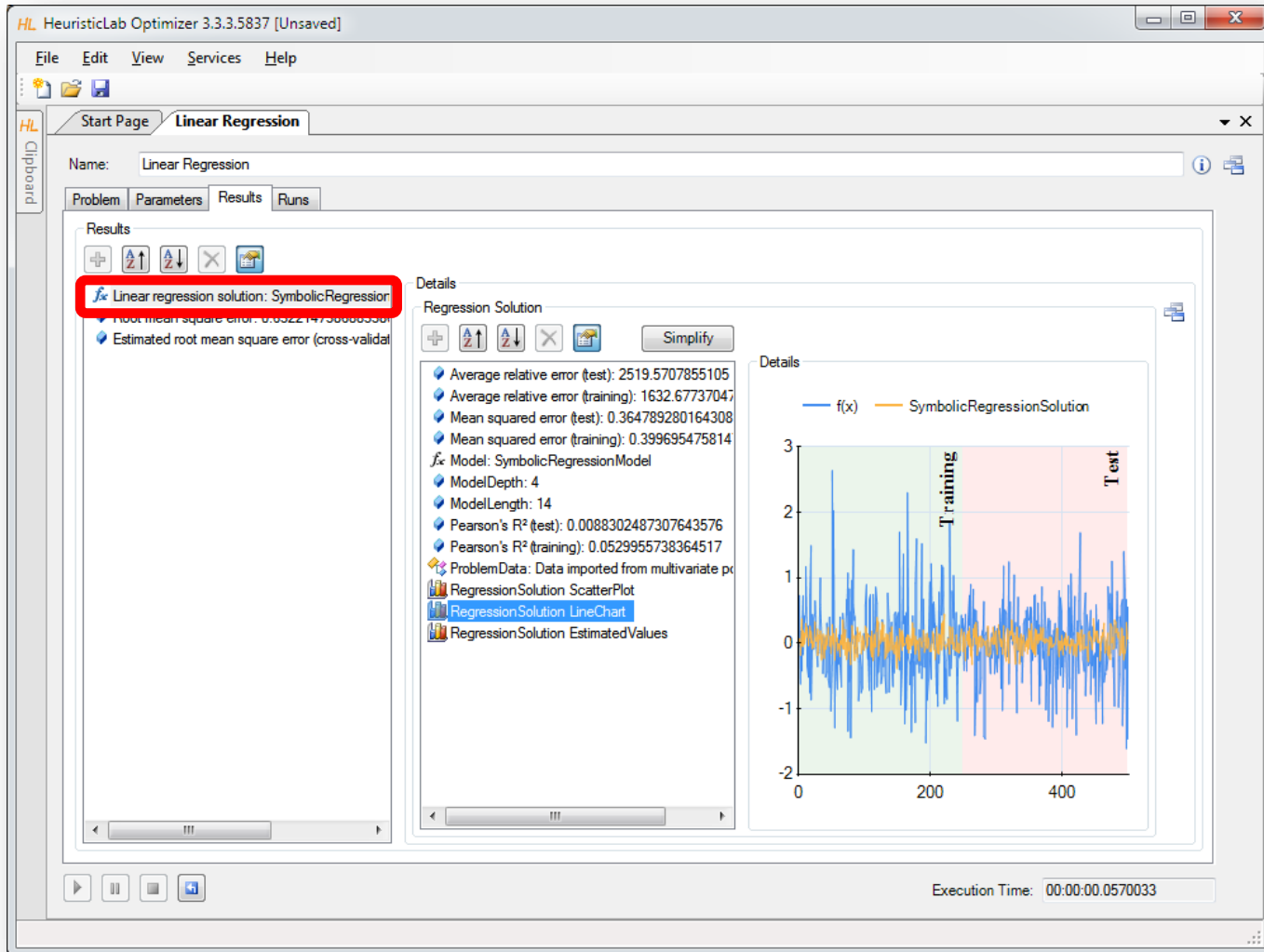
The "Execution Time" at the bottom right is 00:00:00.0570033.

# Inspect Scatterplot of Predicted and Target Values

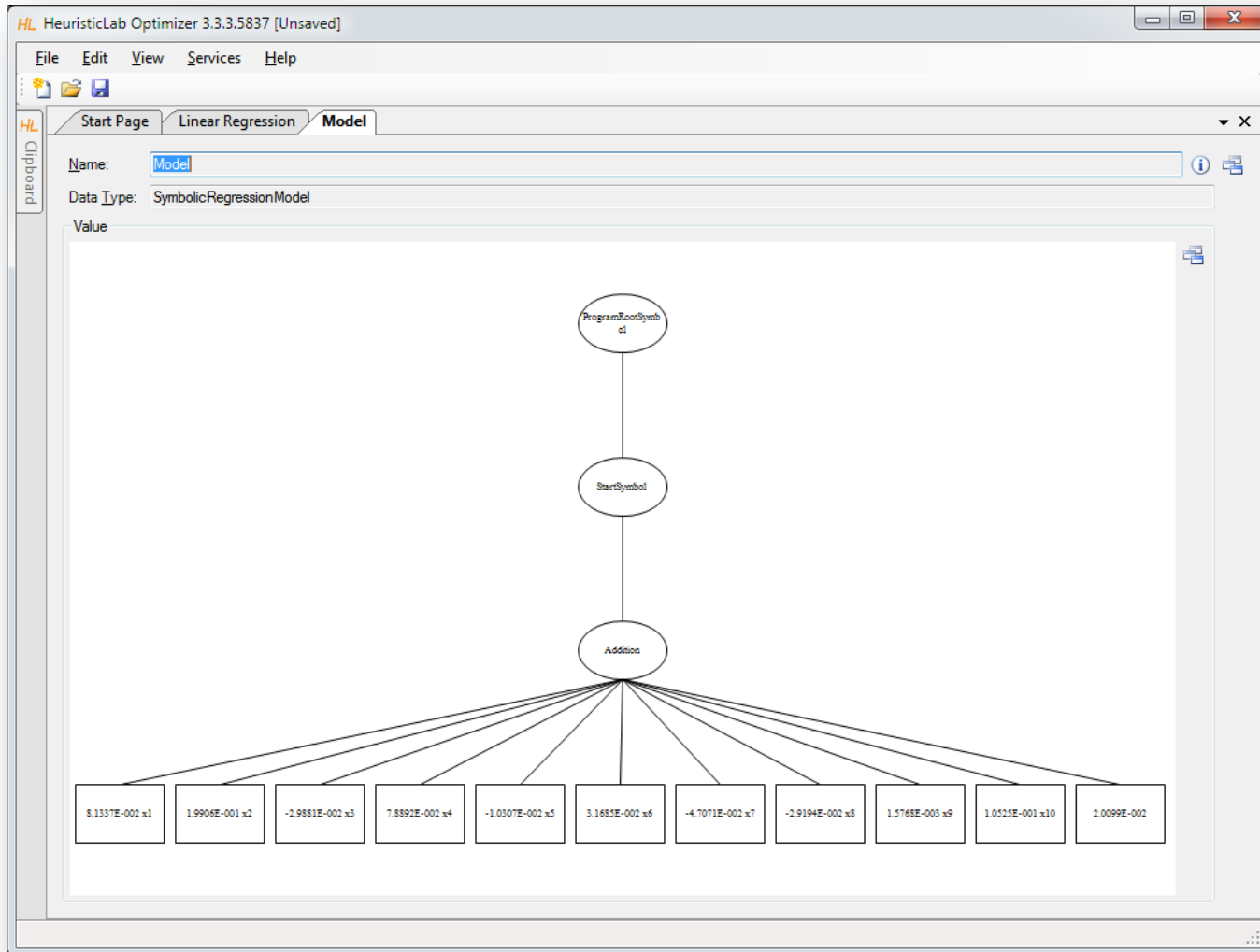




# Inspect Linechart



# Inspect Graphical Representation of Model



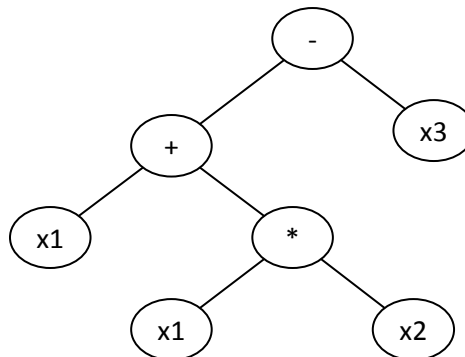
# Nonlinear Regression Methods



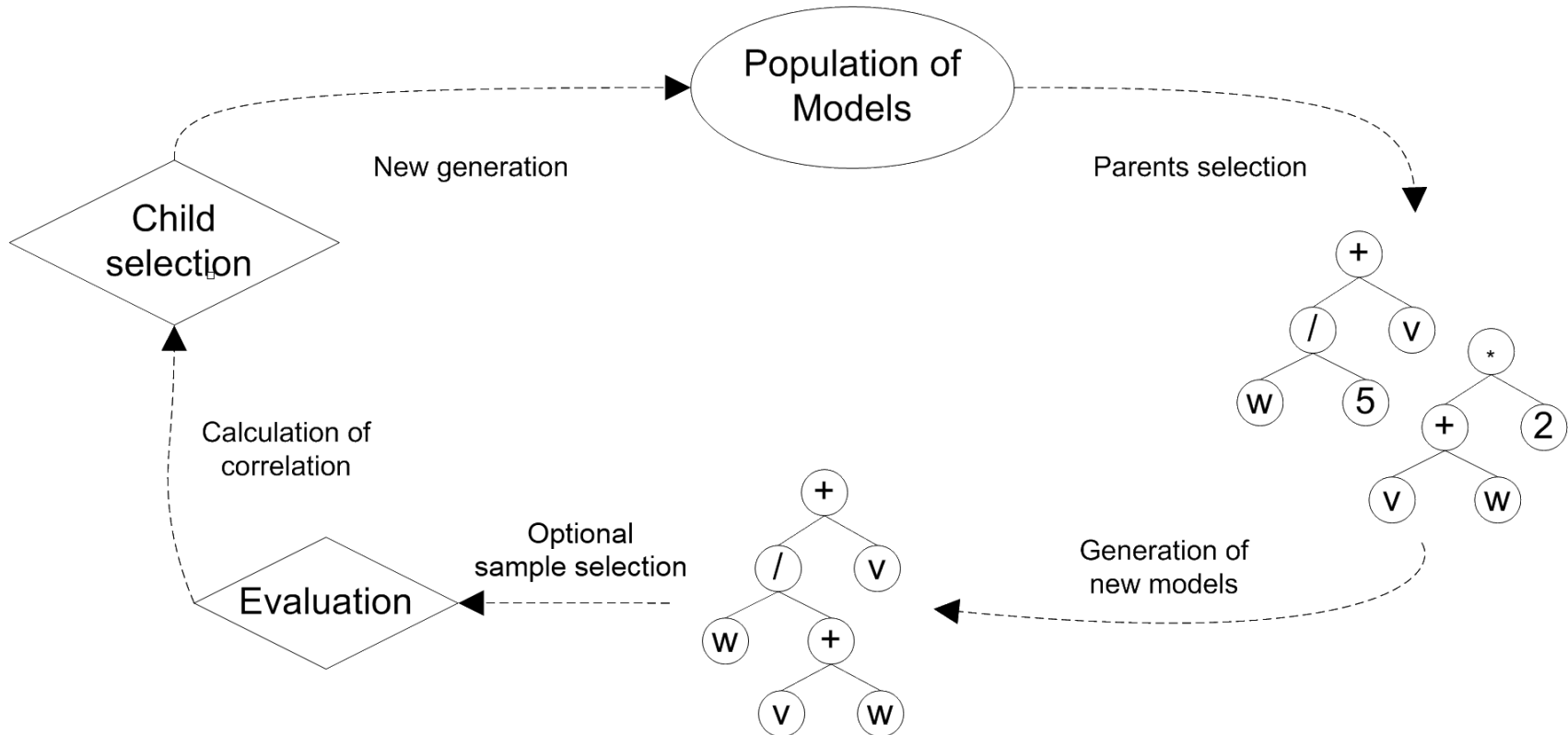
- Random Forests Regression
  - Learns multiple models on different sample subsets
- Support Vector Regression
  - Maps data to higher dimensional space
  - Calculates linear model
  - Transfers the data back to the input space

# Symbolic Regression with HeuristicLab

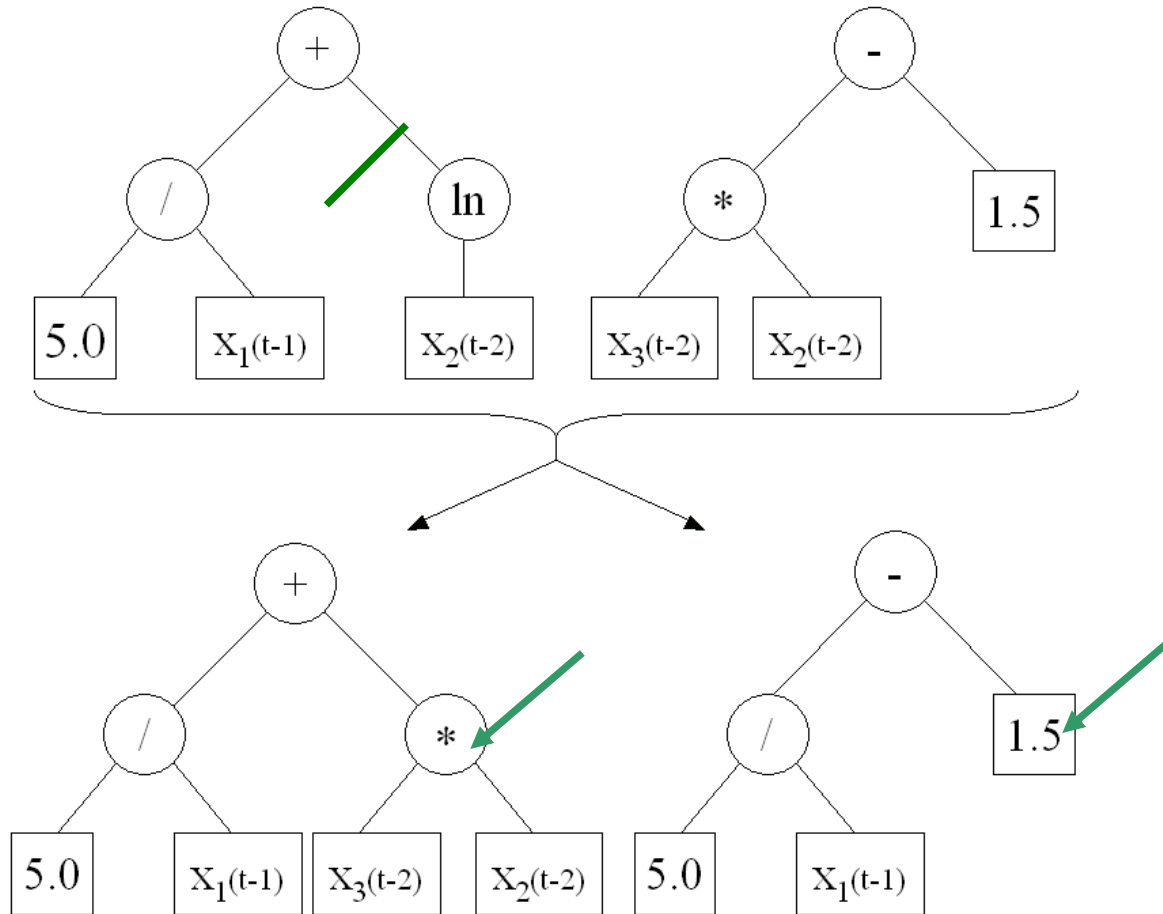
- Linear regression produced an inaccurate model.
- Nonlinear methods produces overfit models
- Next: produce a nonlinear symbolic regression model using genetic programming
- Genetic programming
  - evolve variable-length models
  - model representation: symbolic expression tree
  - structure and model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



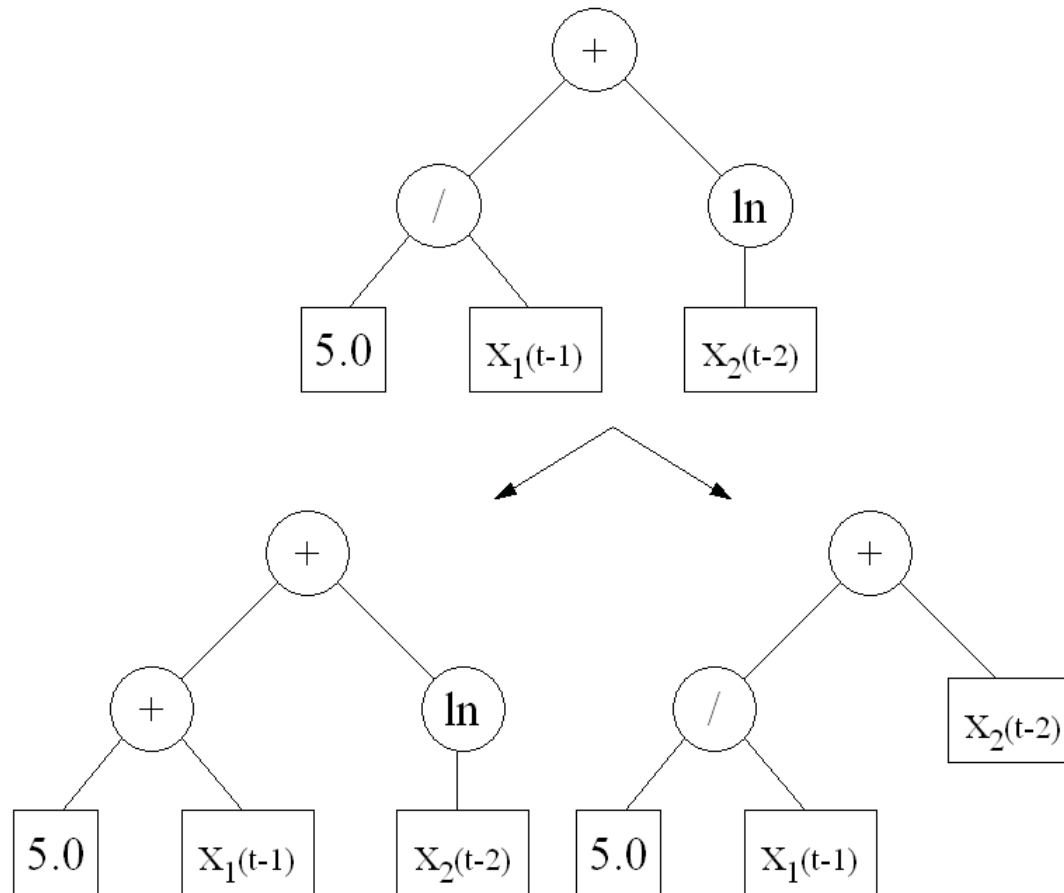
# Genetic Programming Life Cycle



# Generating new Models

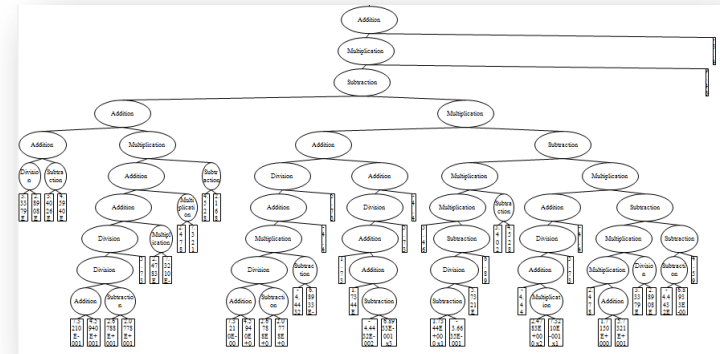


# Mutating models



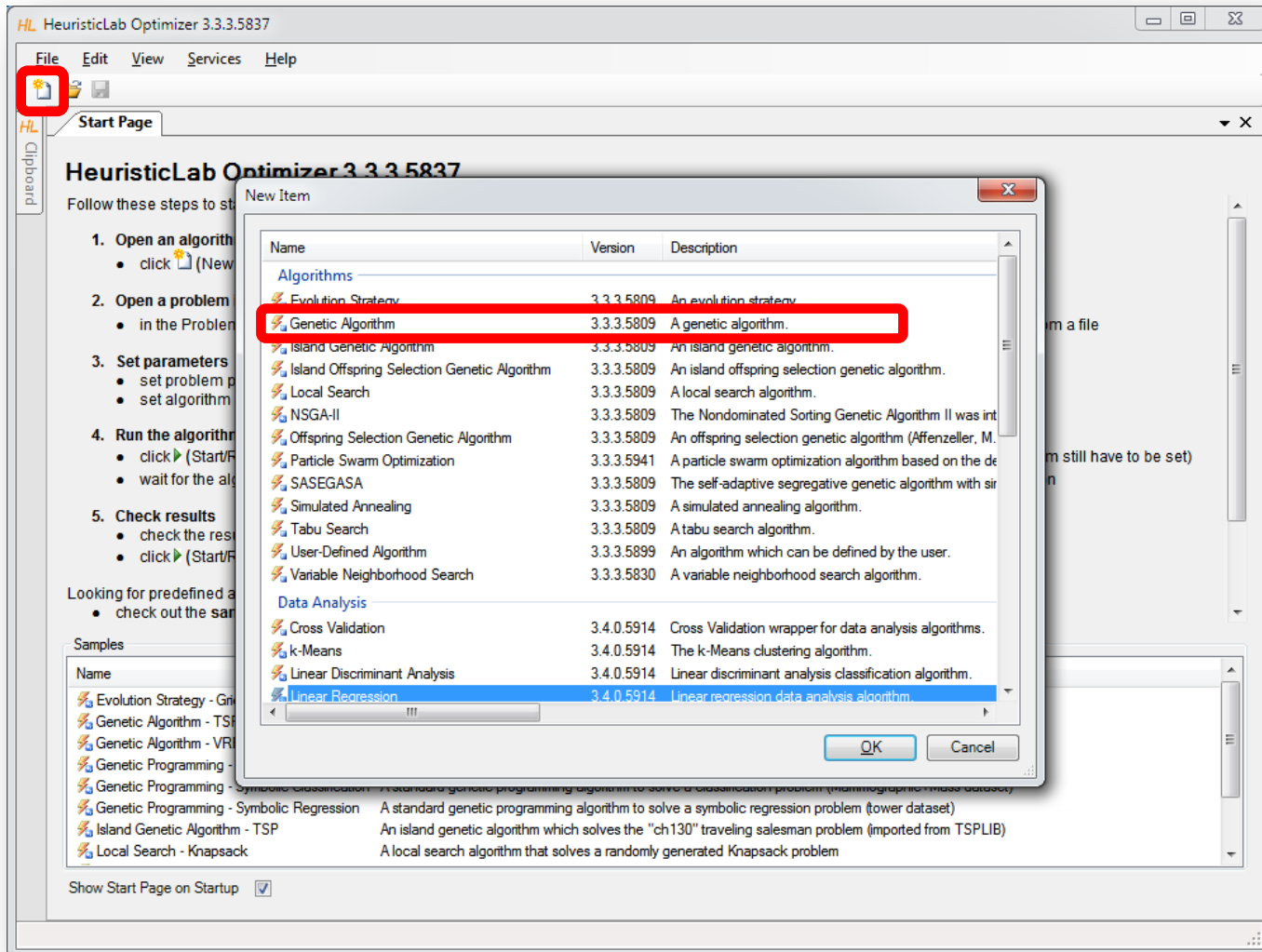
# Symbolic Regression with HeuristicLab

- 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

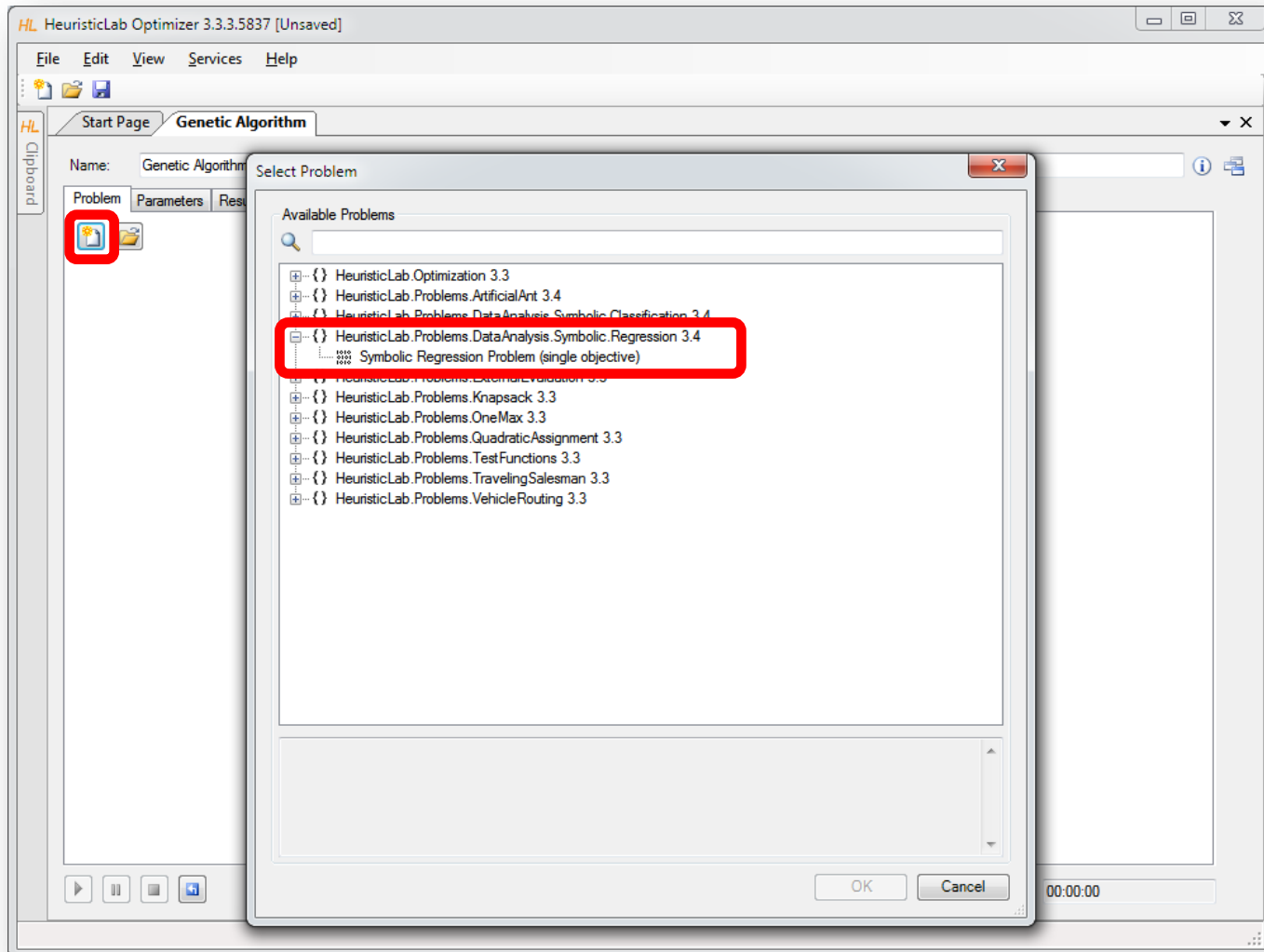




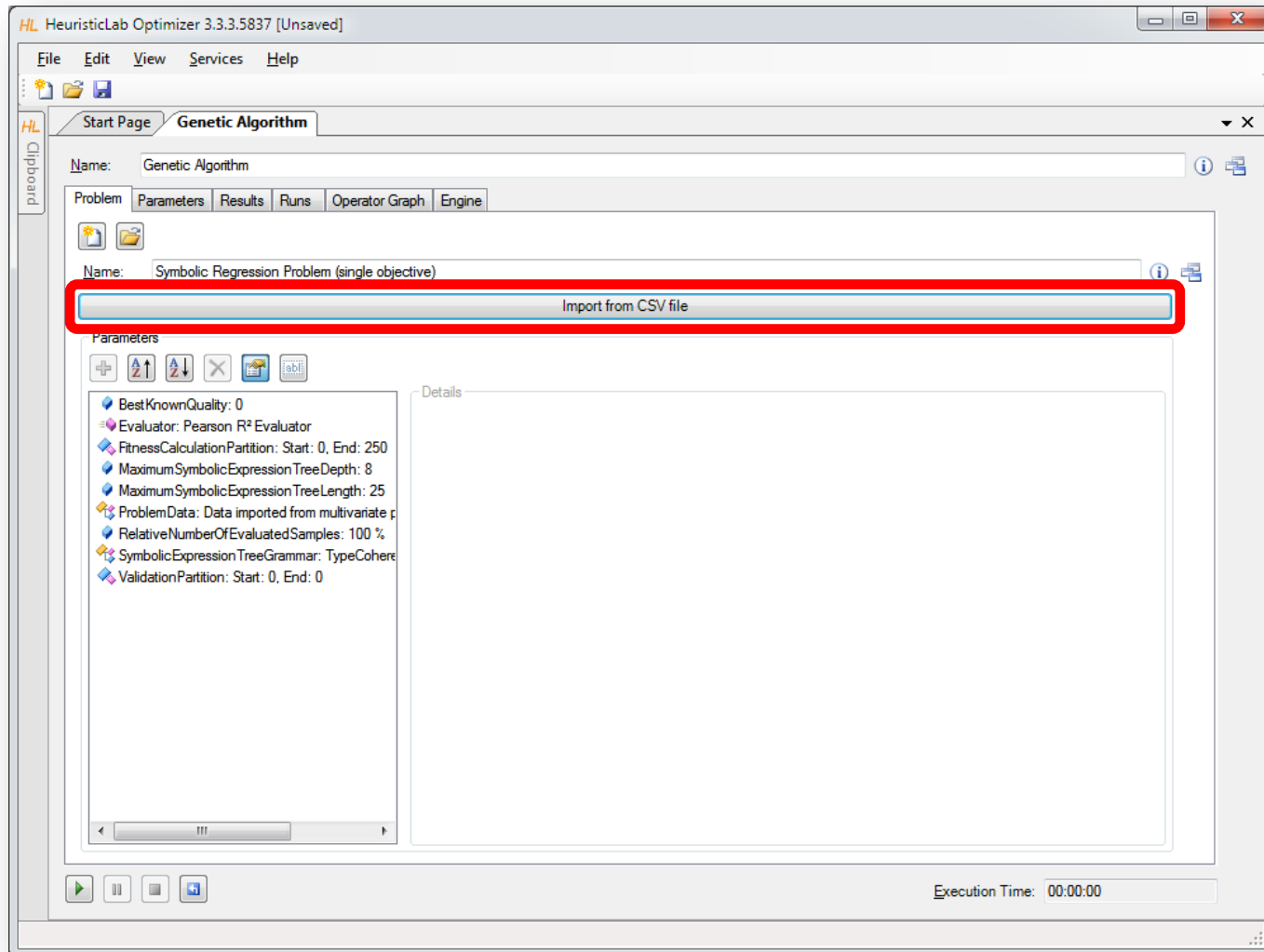
# Create New Genetic Algorithm



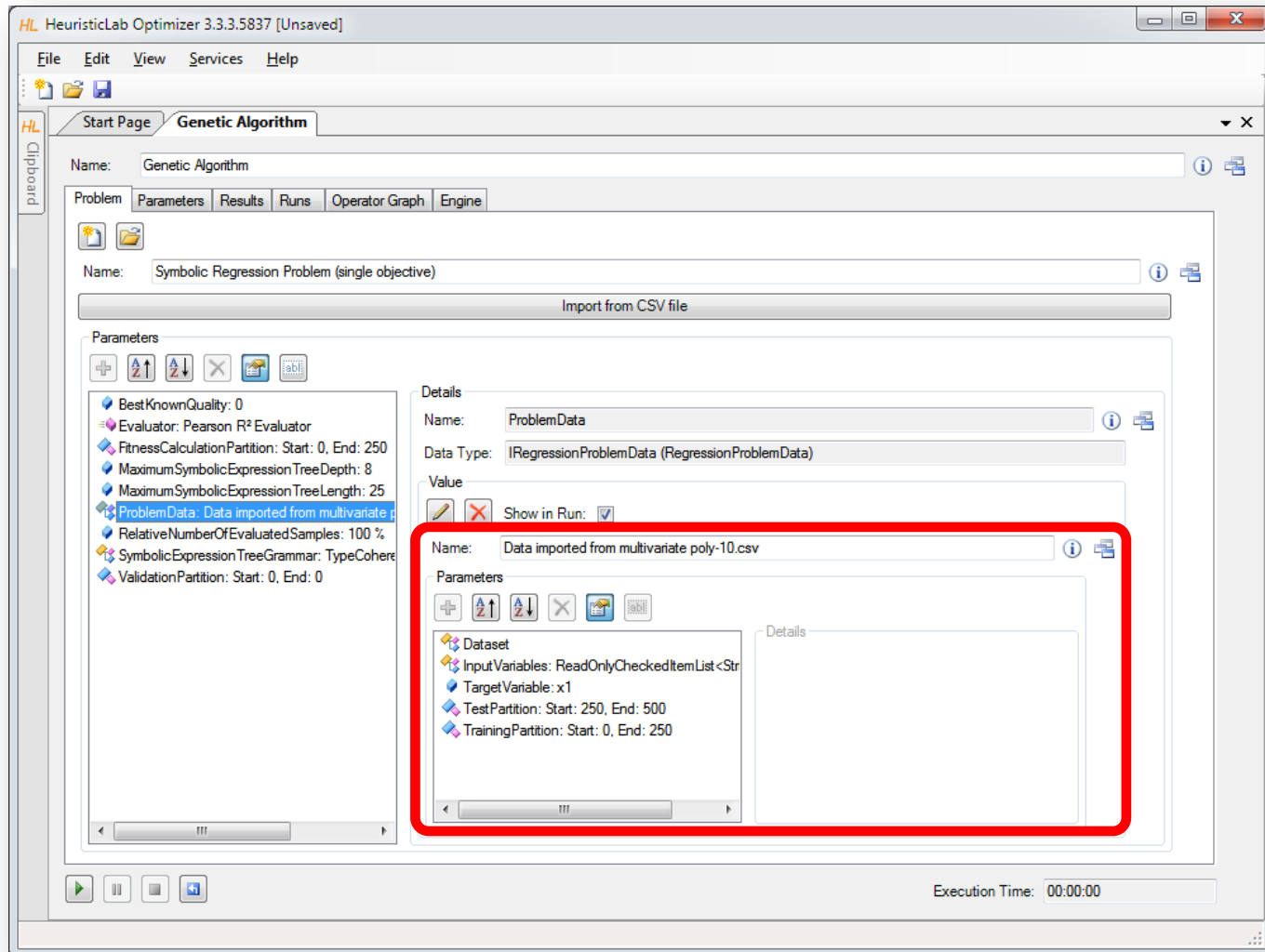
# Create New Symbolic Regression Problem



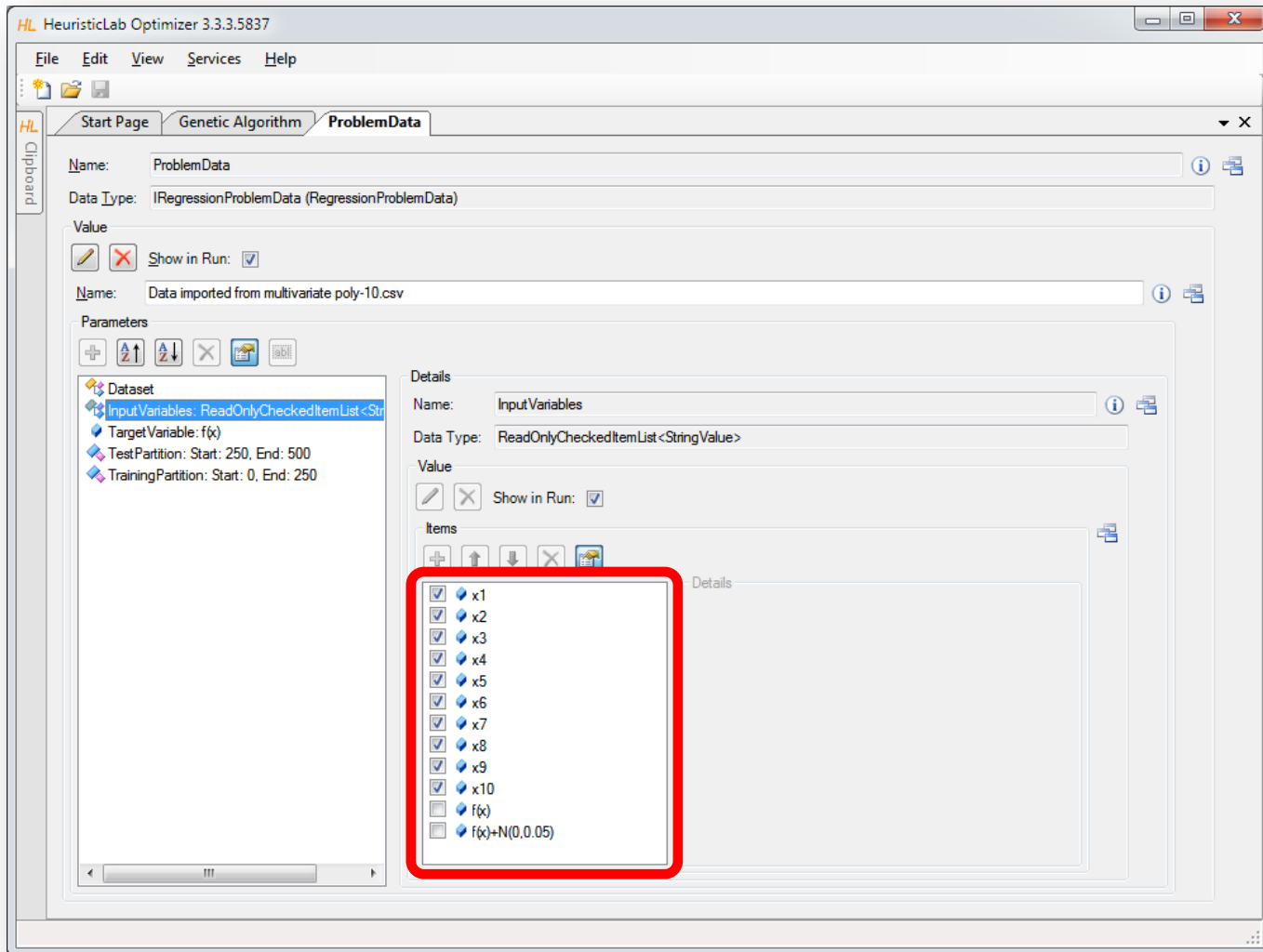
# Import Data



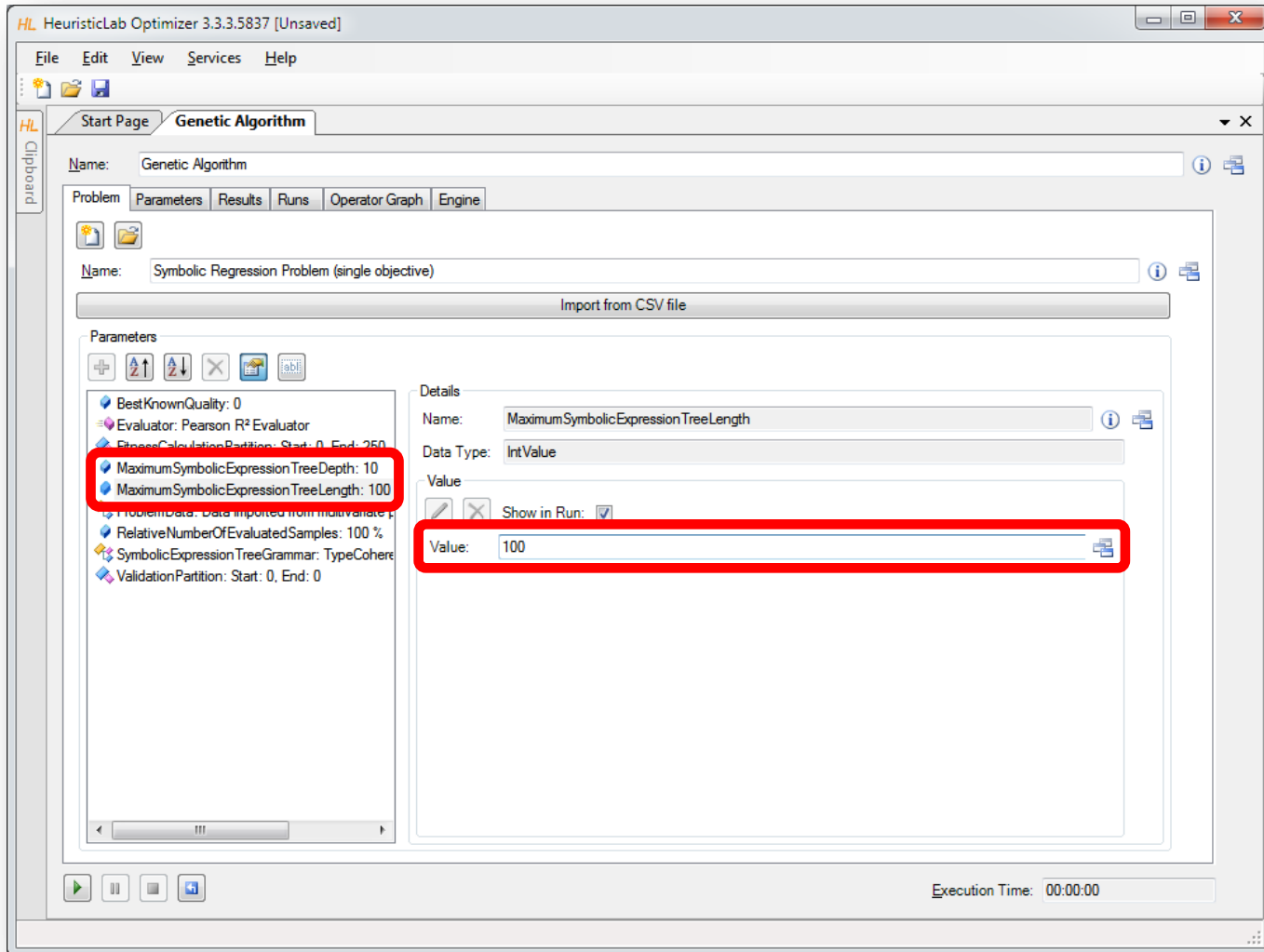
# Inspect Data and Configure Dataset



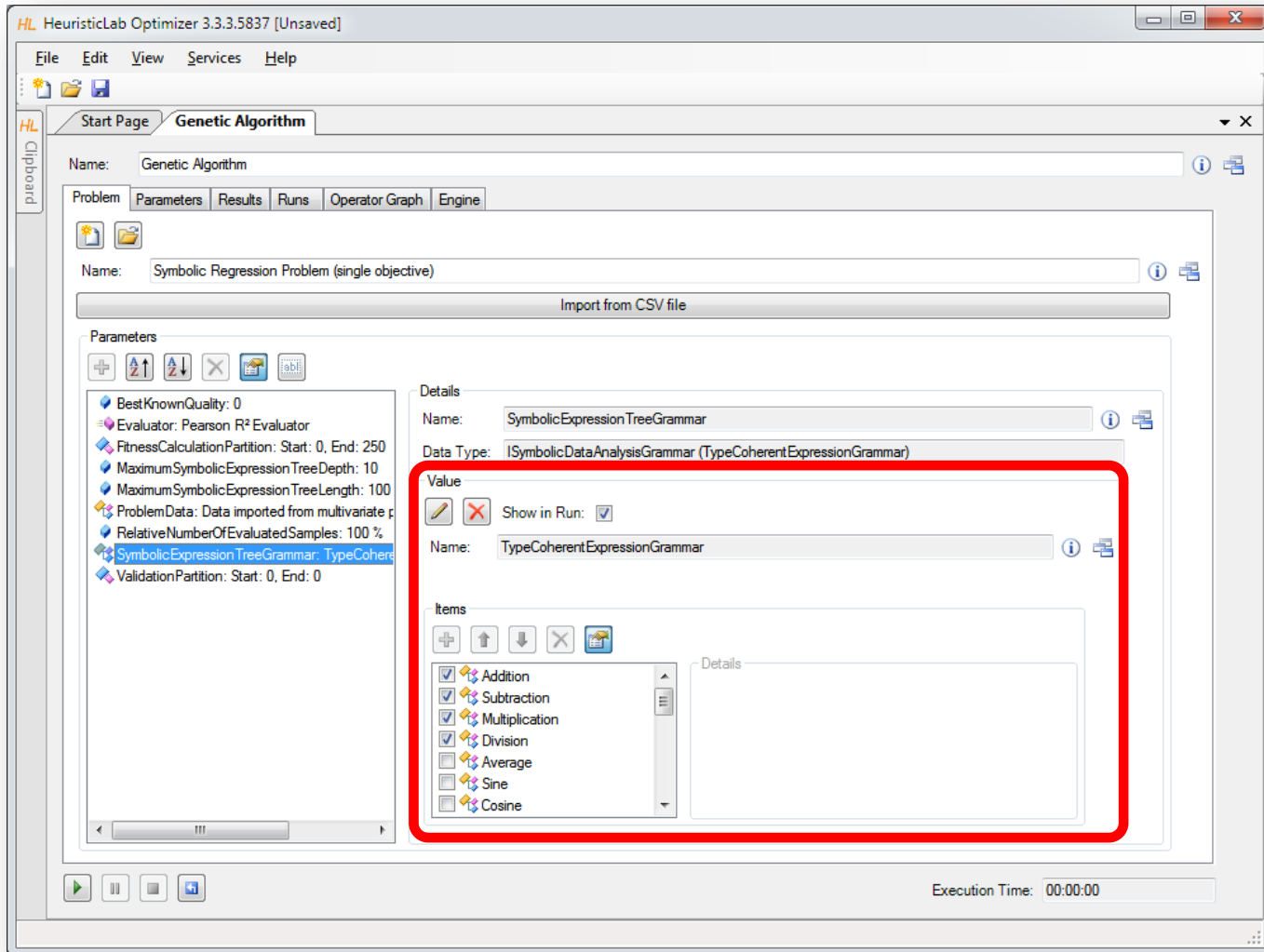
# Set Target and Input Variables



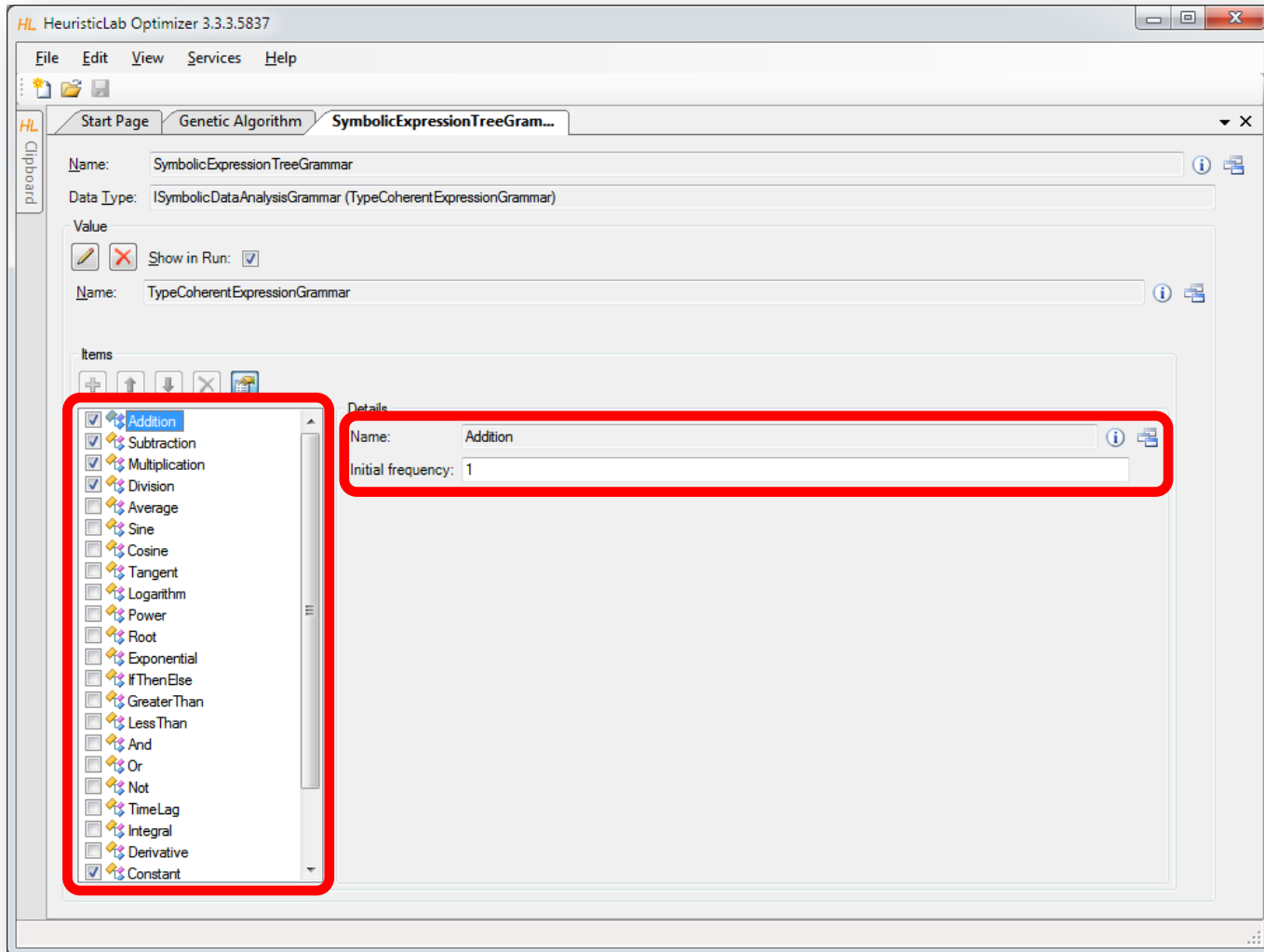
# Configure Maximal Model Depth and Length



# Configure Function Set (Grammar)

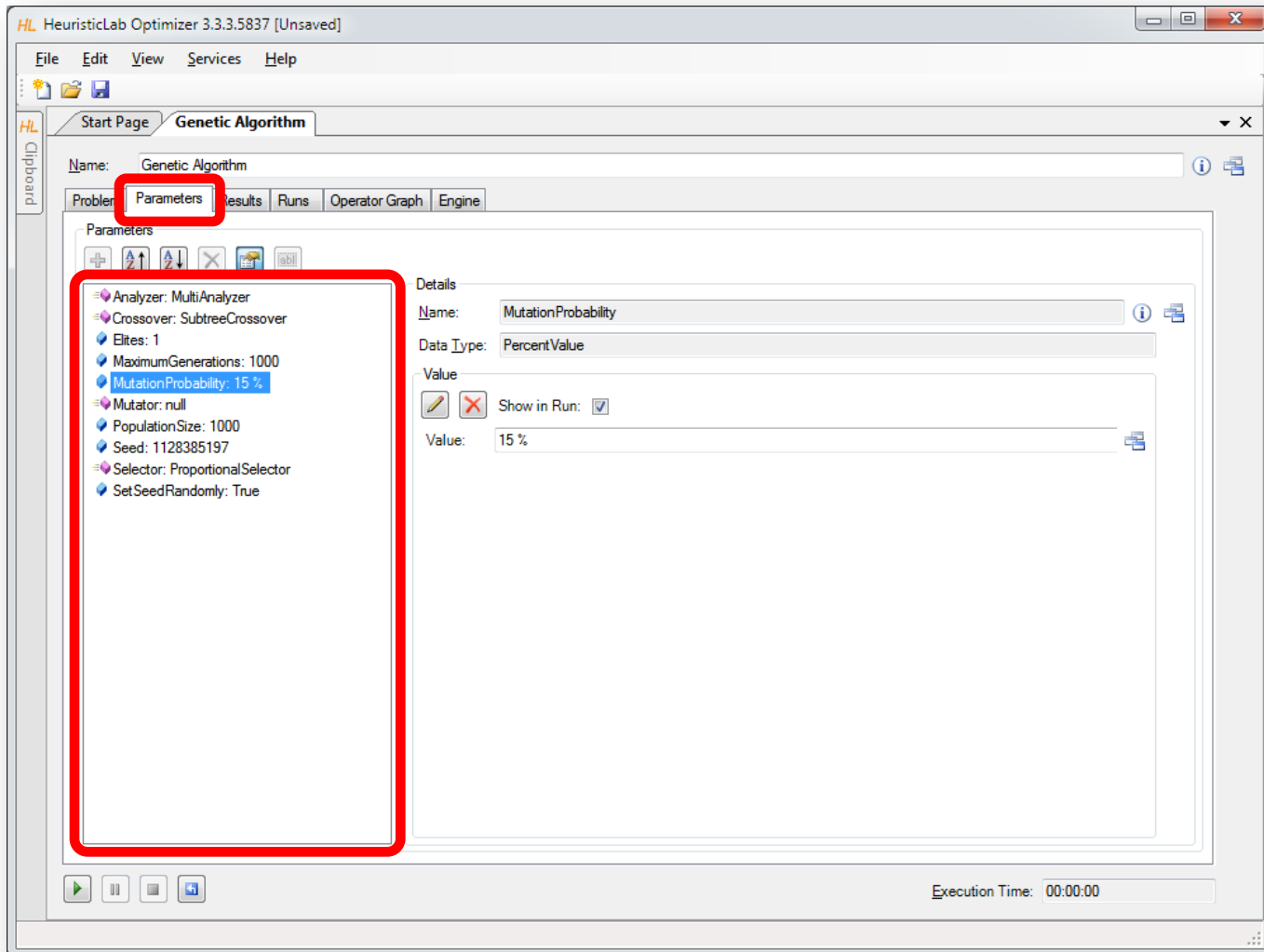


# Configure Function Set (Grammar)

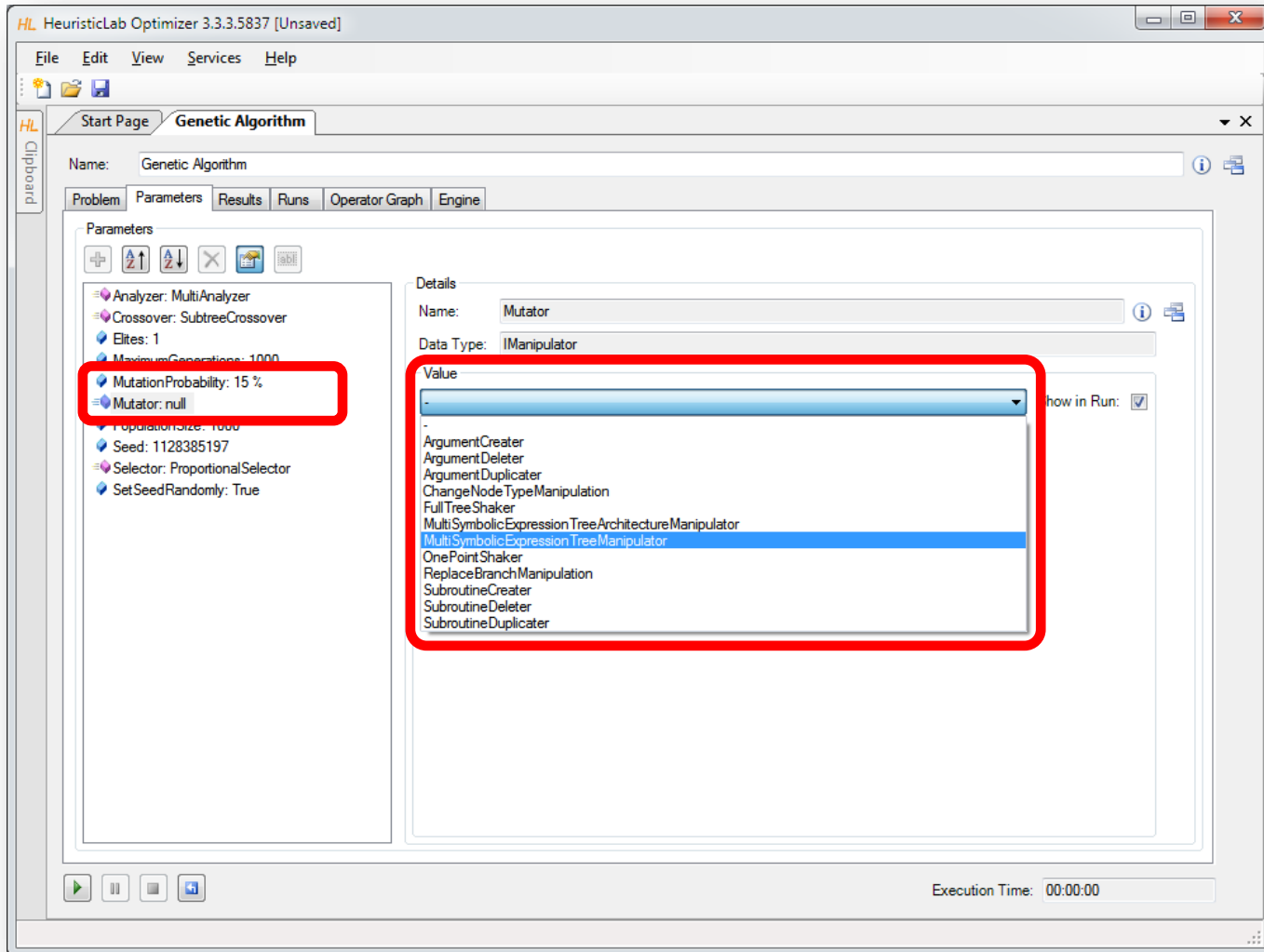




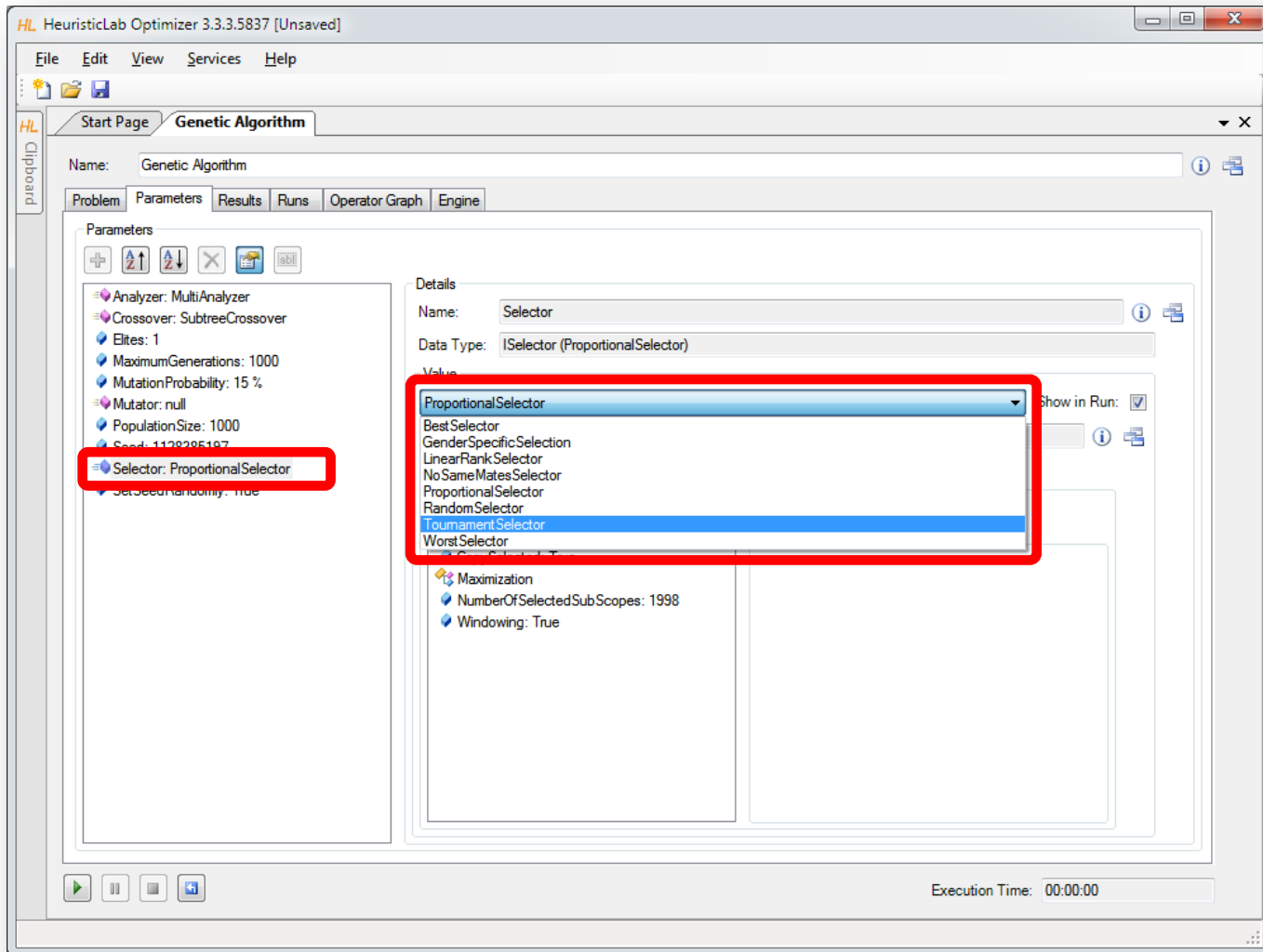
# Configure Algorithm Parameters



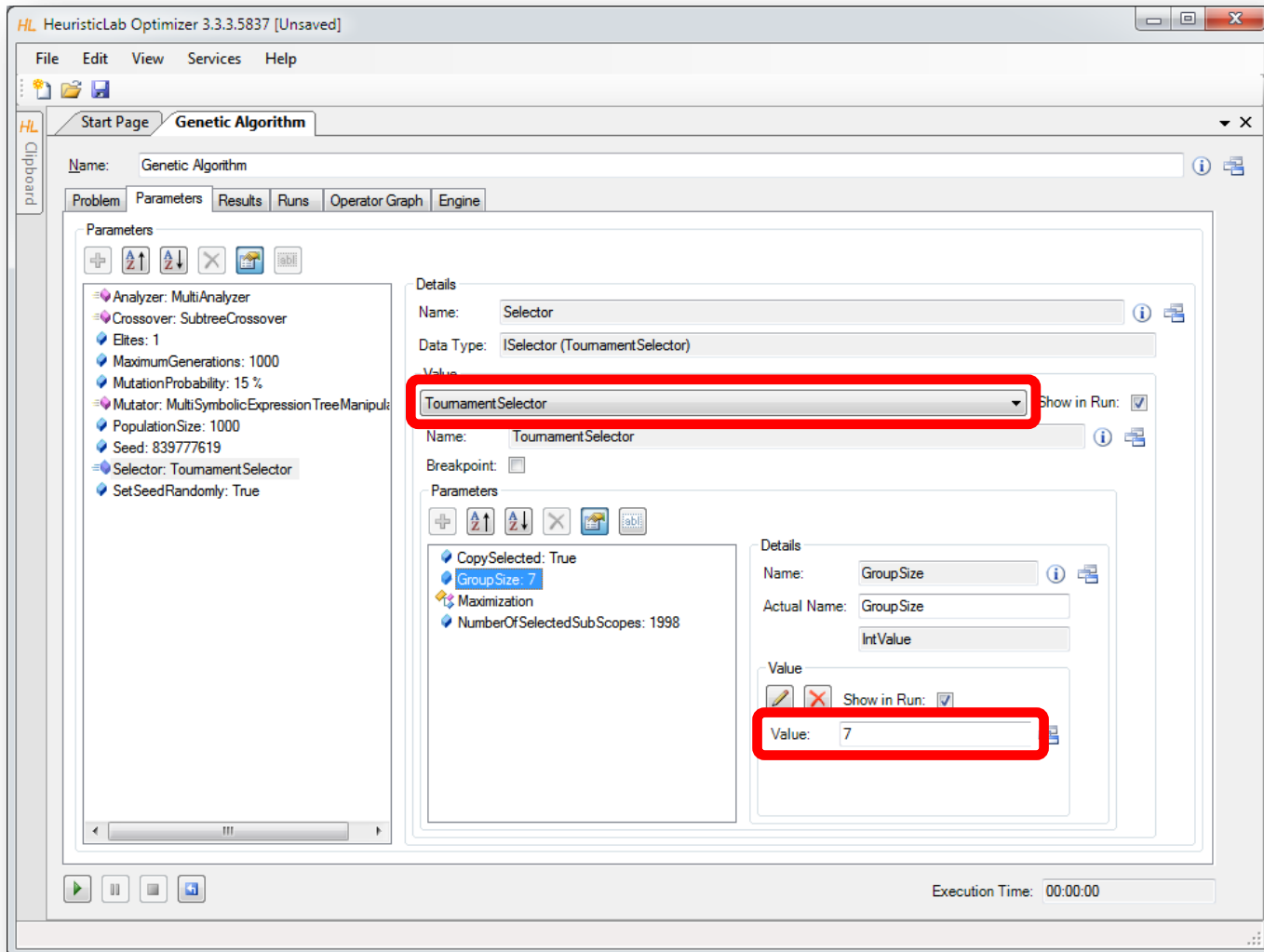
# Configure Mutation Operator



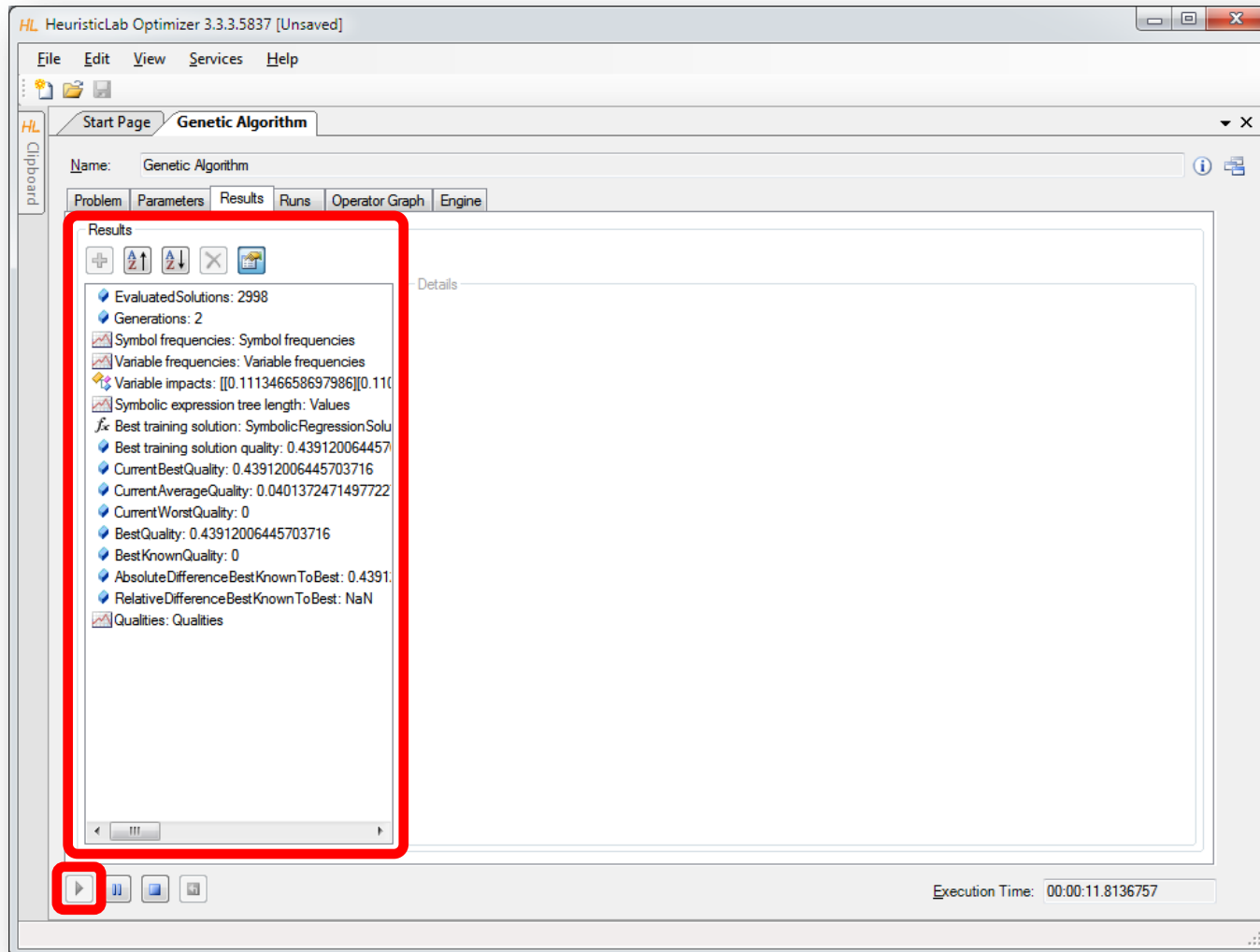
# Configure Selection Operator



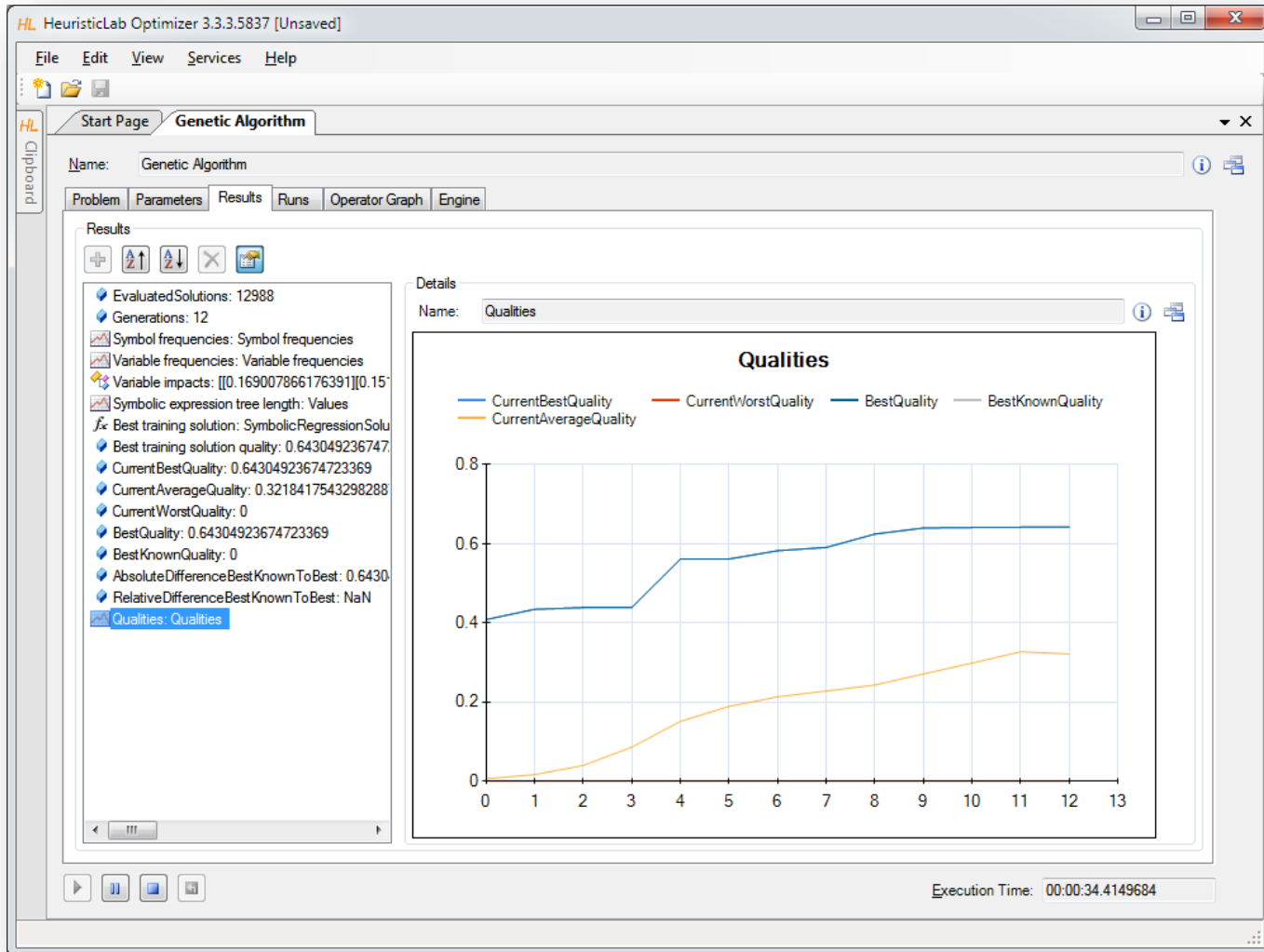
# Configure Tournament Group Size



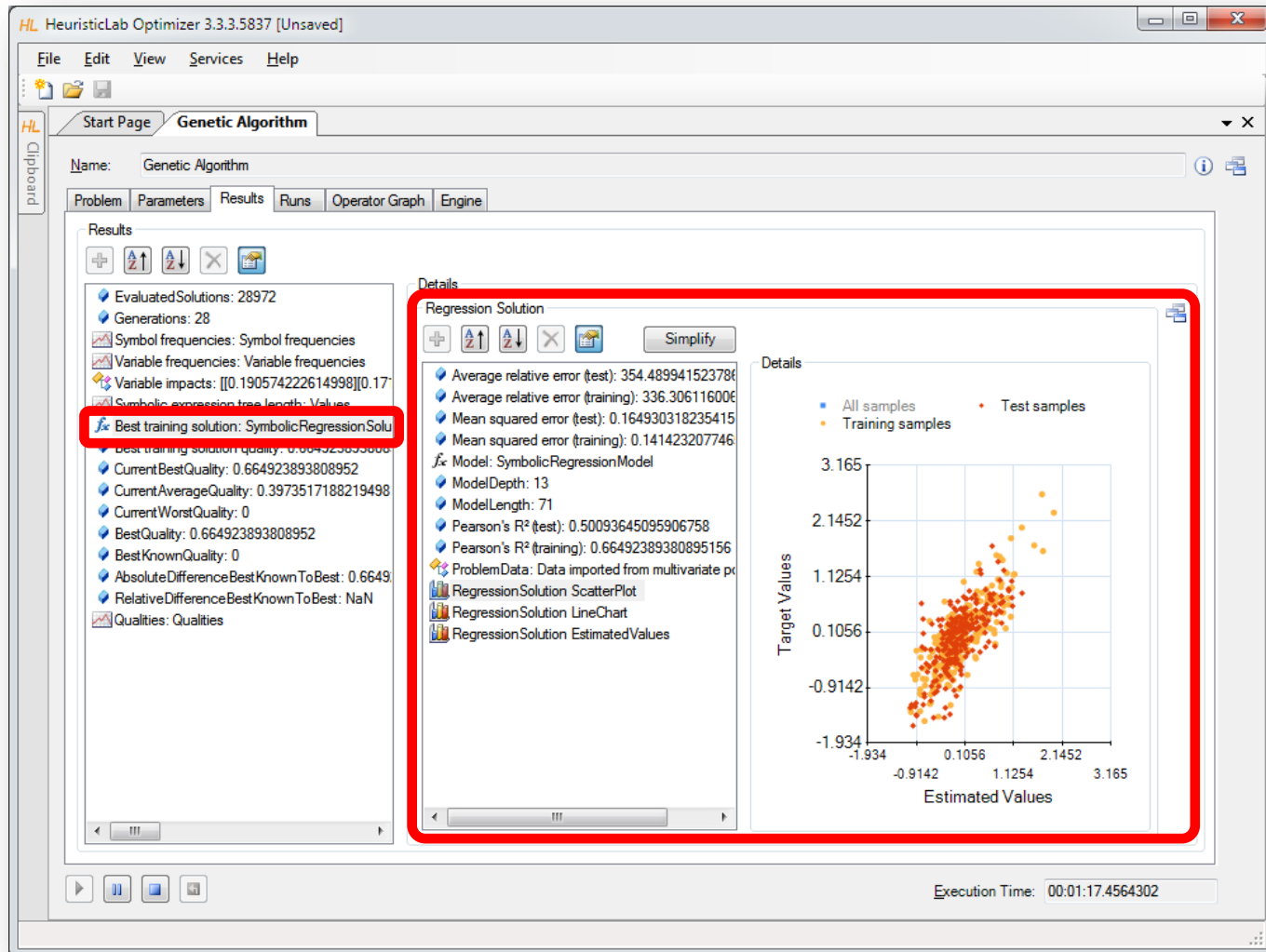
# Start Algorithm and Inspect Results



# Inspect Quality Chart



# Inspect Best Model on Training Partition

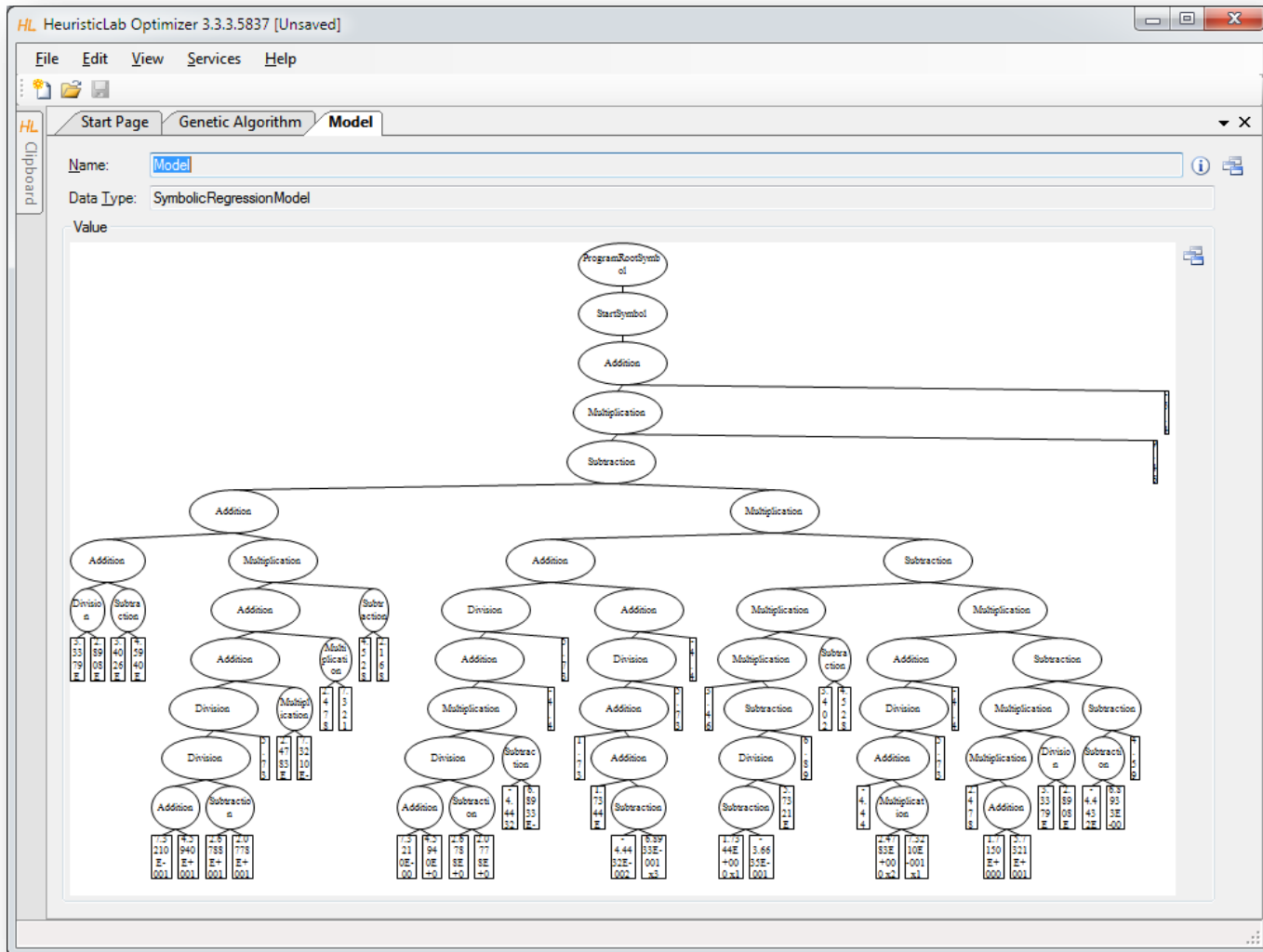


# Inspect Linechart of Best Model on Training Partition



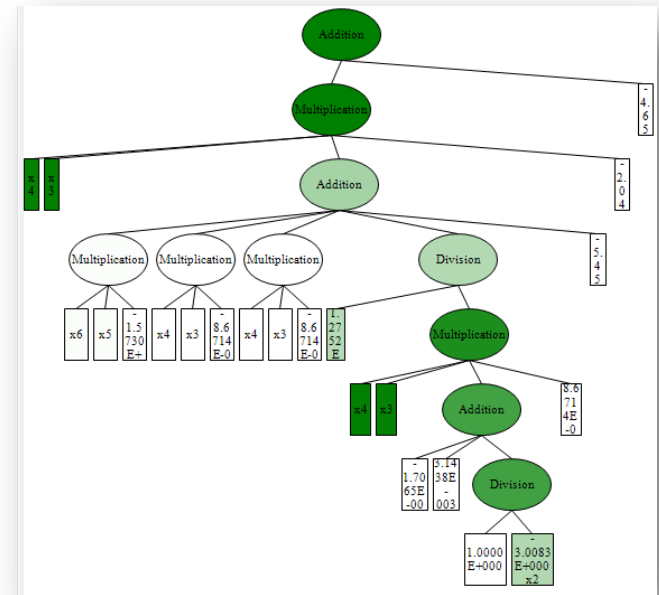


# Inspect Structure of Best Model on Training Partition



# Model Simplification and Export

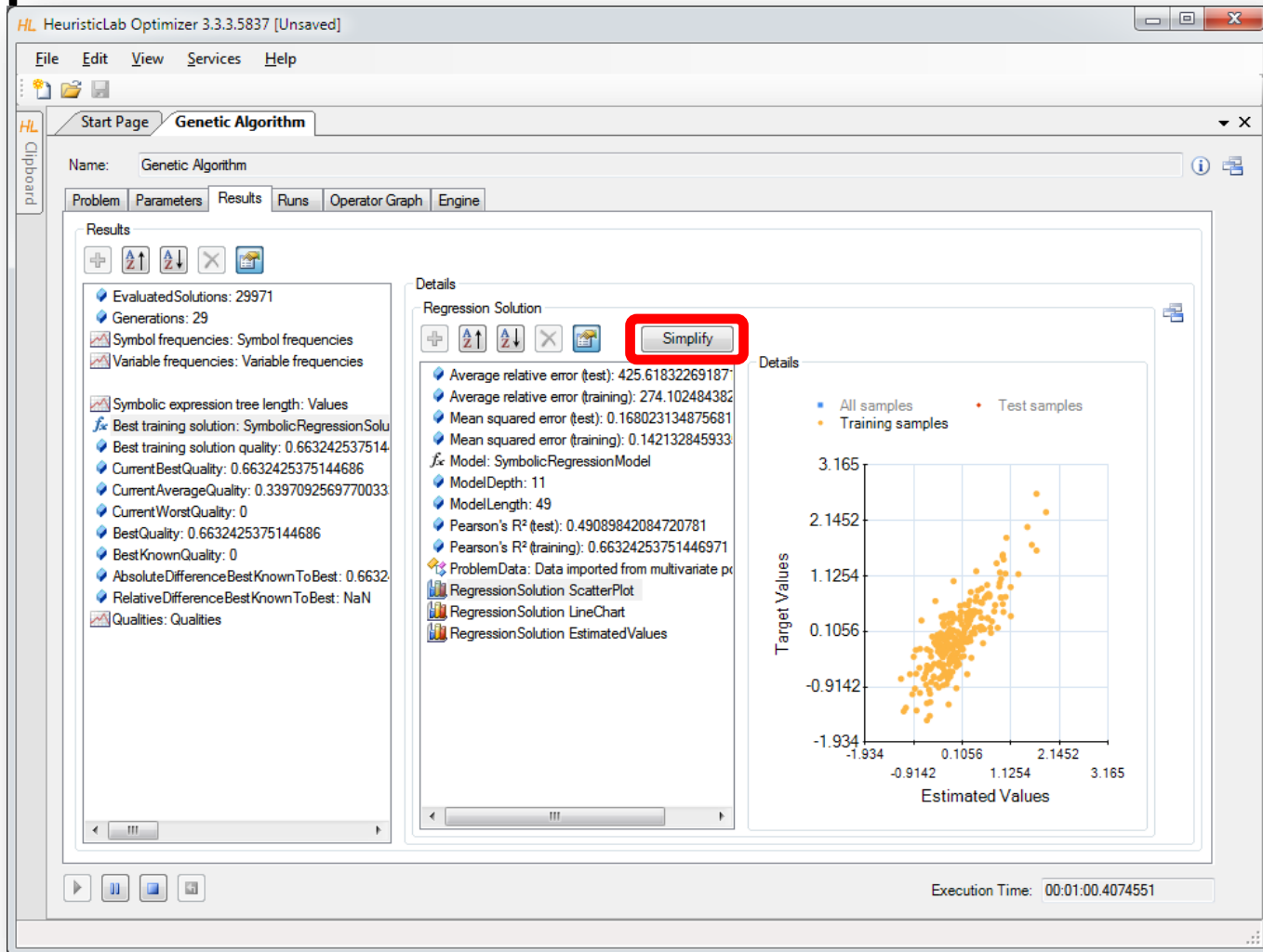
- Demonstration
  - automatic simplification
  - visualization of node impacts
  - manual simplification
    - online update of results
  - model export
    - MATLAB
    - LaTeX



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

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

# Detailed Model Analysis and Simplification



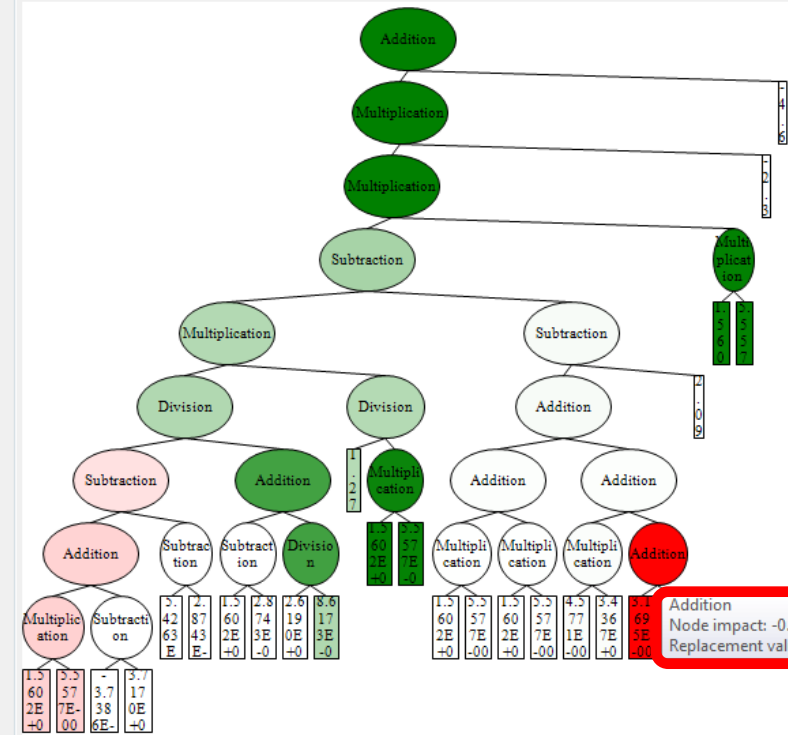
# Symbolic Simplification and Node Impacts

HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]

File Edit View Services Help

Start Page Genetic Algorithm Interactive Regression Solution...

Simplify



Details

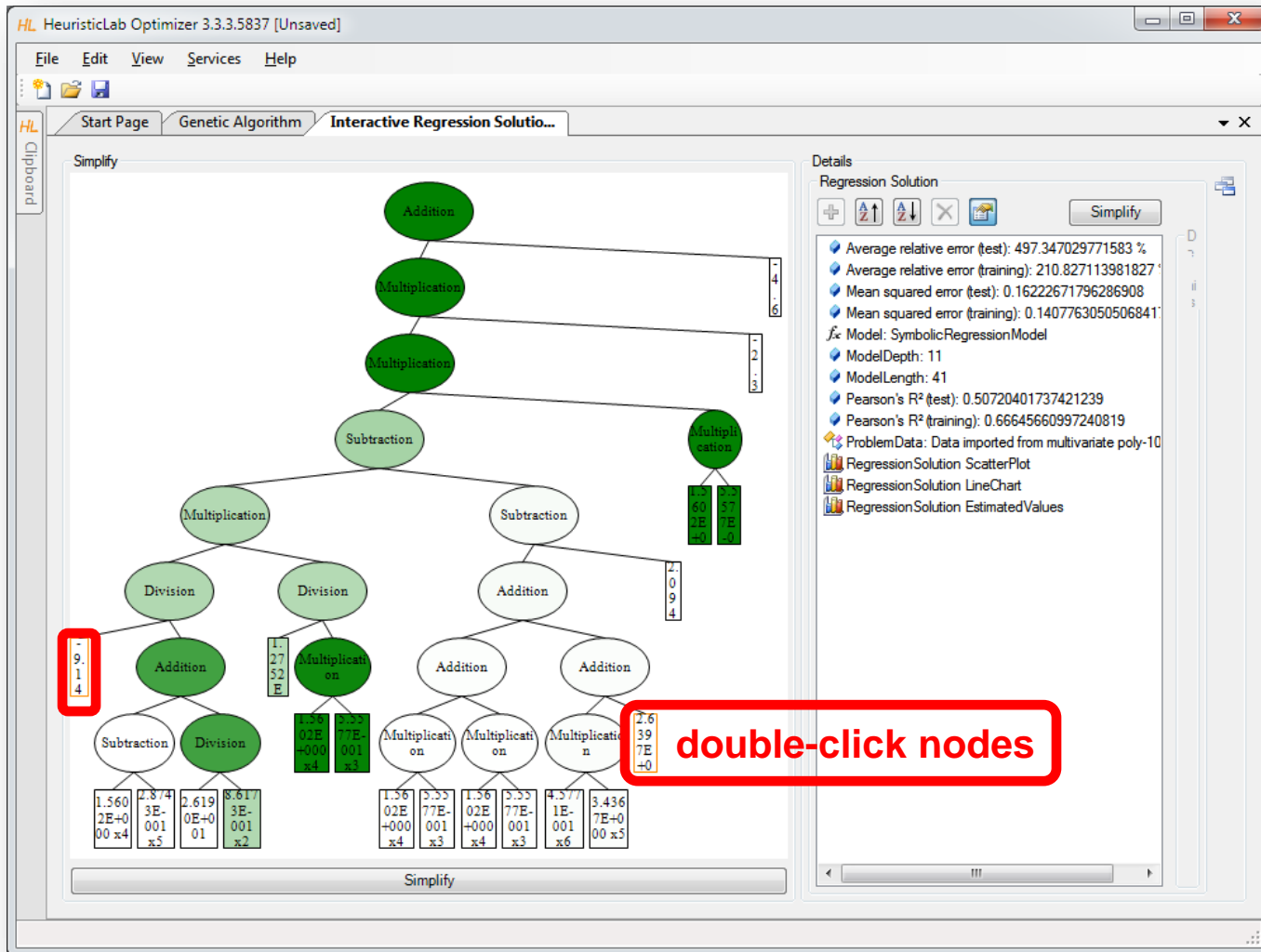
Regression Solution

Average relative error (test): 473.618715910248 %  
 Average relative error (training): 245.090781502651 %  
 Mean squared error (test): 0.163506091453717  
 Mean squared error (training): 0.1415659913306669  
 $f_x$  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

Addition  
 Node impact: -0.00166473209841678  
 Replacement value: 26.3974611821139

Simplify

# Manual Simplification



The screenshot displays the HeuristicLab Optimizer interface. The main window shows a 'Simplify' view of a regression solution tree. The tree consists of nodes representing mathematical operations: Addition, Multiplication, Subtraction, and Division. Some nodes are highlighted in green, indicating they are part of the current solution. A red box highlights a node with the value  $-9.14$ . Another red box highlights a node with the value  $2.6397E+0$ , with the text 'double-click nodes' written next to it. The right-hand side of the window shows a 'Details' panel for the 'Regression Solution', containing various statistical metrics and model information.

HL HeuristicLab Optimizer 3.3.5837 [Unsaved]

File Edit View Services Help

Start Page Genetic Algorithm Interactive Regression Solution...

Simplify

Details

Regression Solution

Average relative error (test): 497.347029771583 %  
Average relative error (training): 210.827113981827 %  
Mean squared error (test): 0.16222671796286908  
Mean squared error (training): 0.1407763050506841

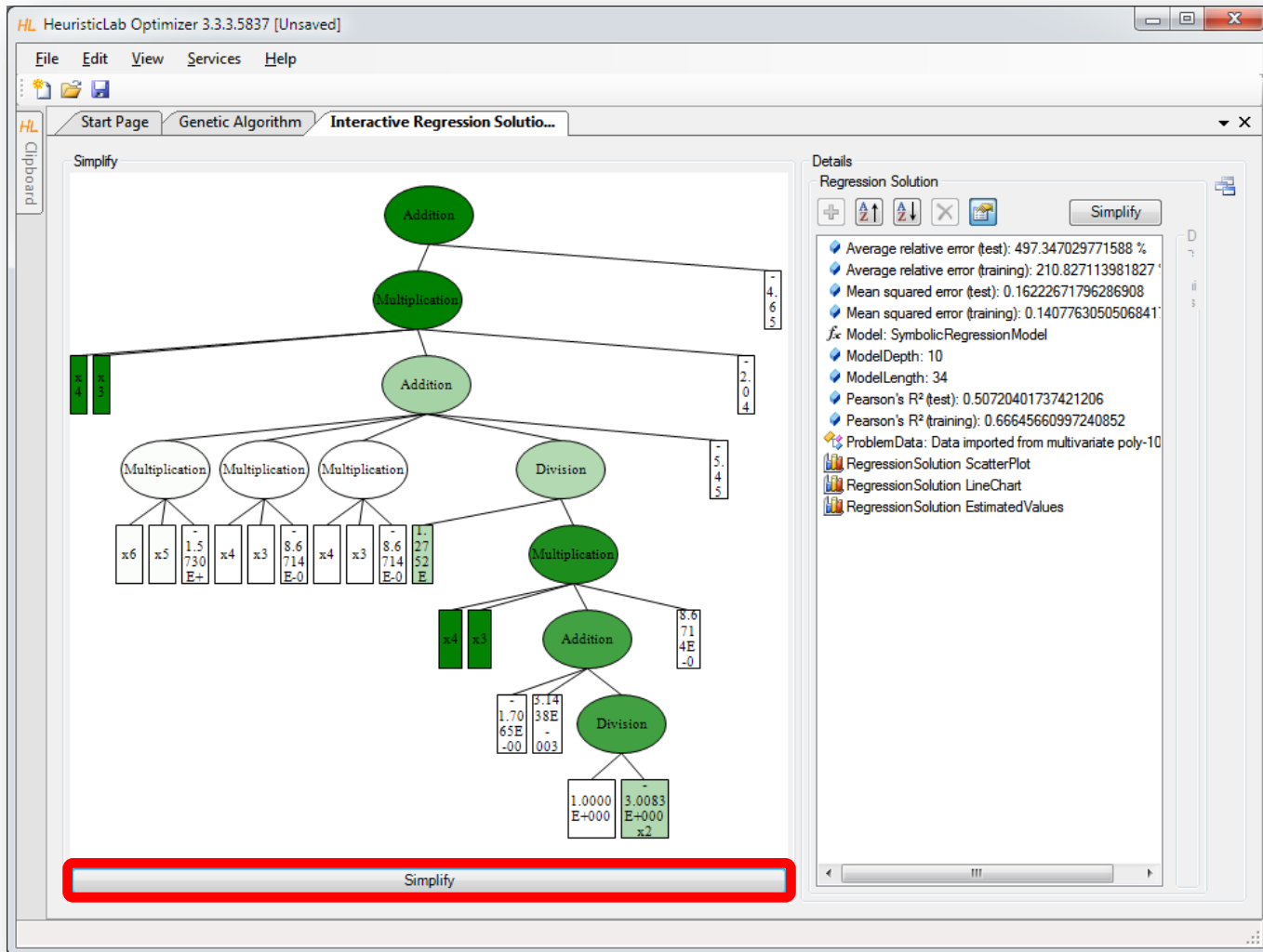
f<sub>x</sub> Model: SymbolicRegressionModel  
ModelDepth: 11  
ModelLength: 41  
Pearson's R<sup>2</sup> (test): 0.50720401737421239  
Pearson's R<sup>2</sup> (training): 0.66645660997240819

ProblemData: Data imported from multivariate poly-10

RegressionSolution ScatterPlot  
RegressionSolution LineChart  
RegressionSolution EstimatedValues

double-click nodes

# 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: Further sub-nodes including Multiplication, Addition, and Division.
- Level 5: Leaf nodes with numerical values and variables like  $x_6$ ,  $x_5$ ,  $x_4$ ,  $x_3$ ,  $x_2$ .

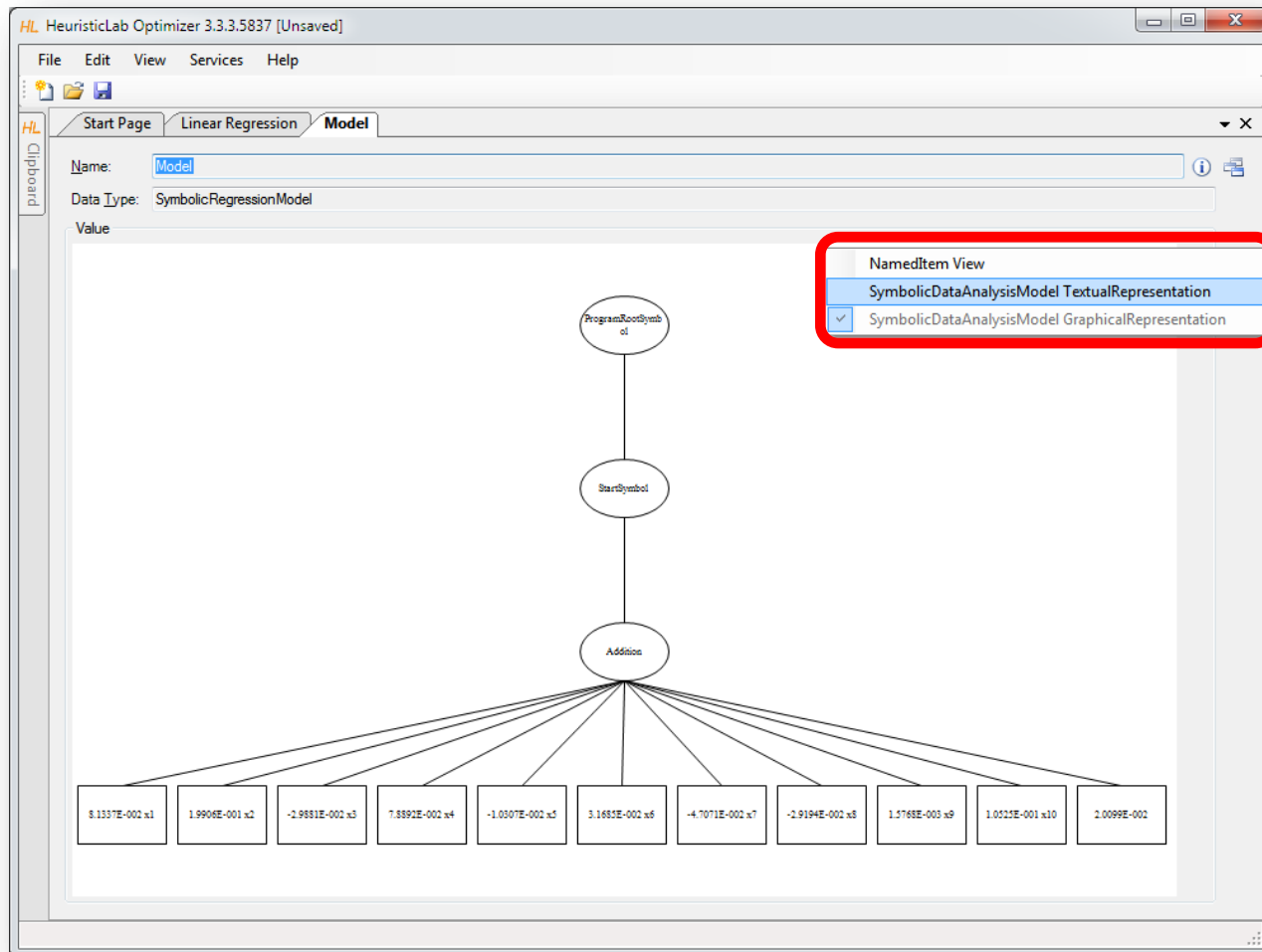
The details panel on the right provides the following information:

- 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

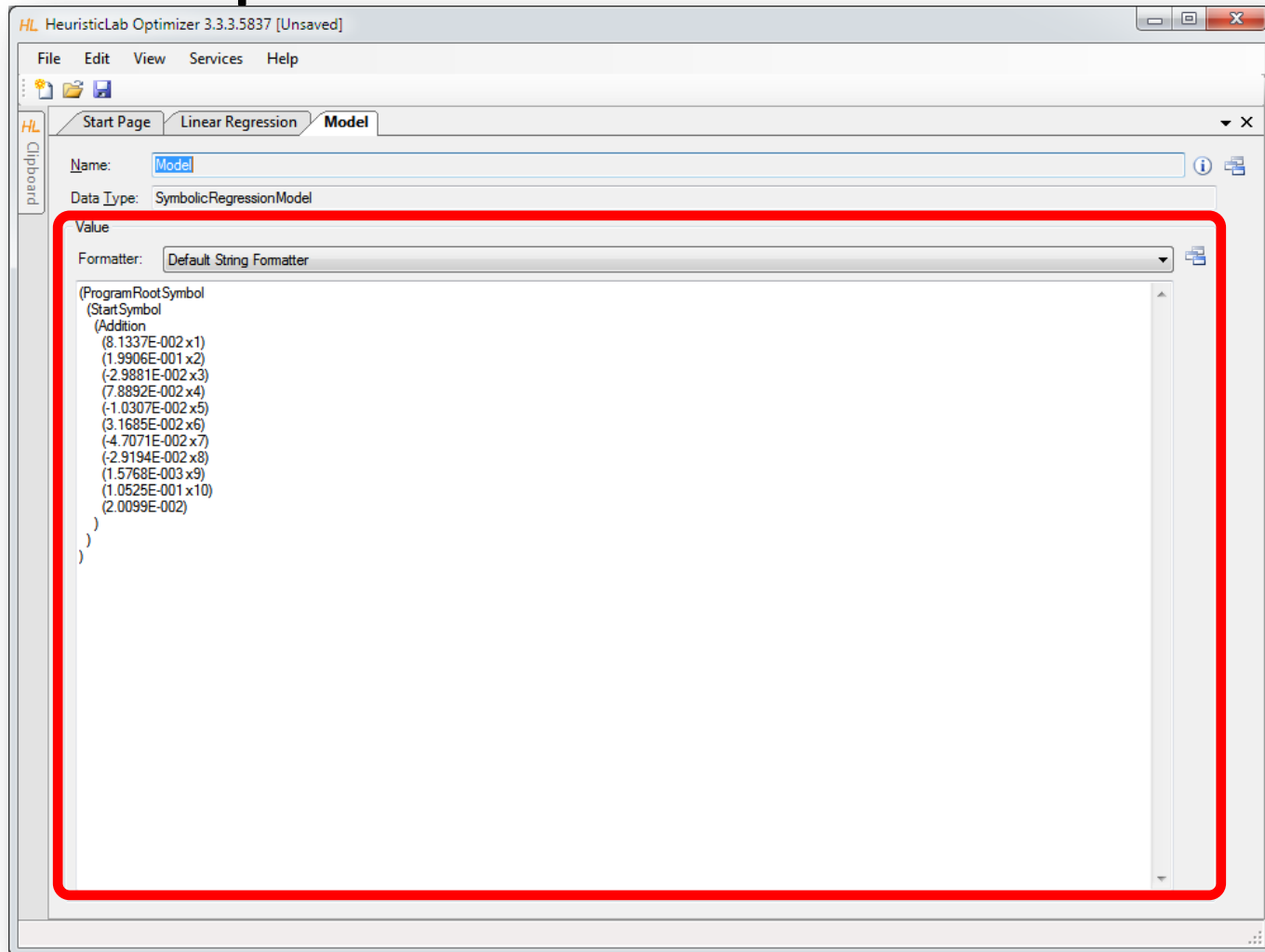
A red box highlights the "Simplify" button at the bottom of the main window.

# Textual Representations Are Also Available

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

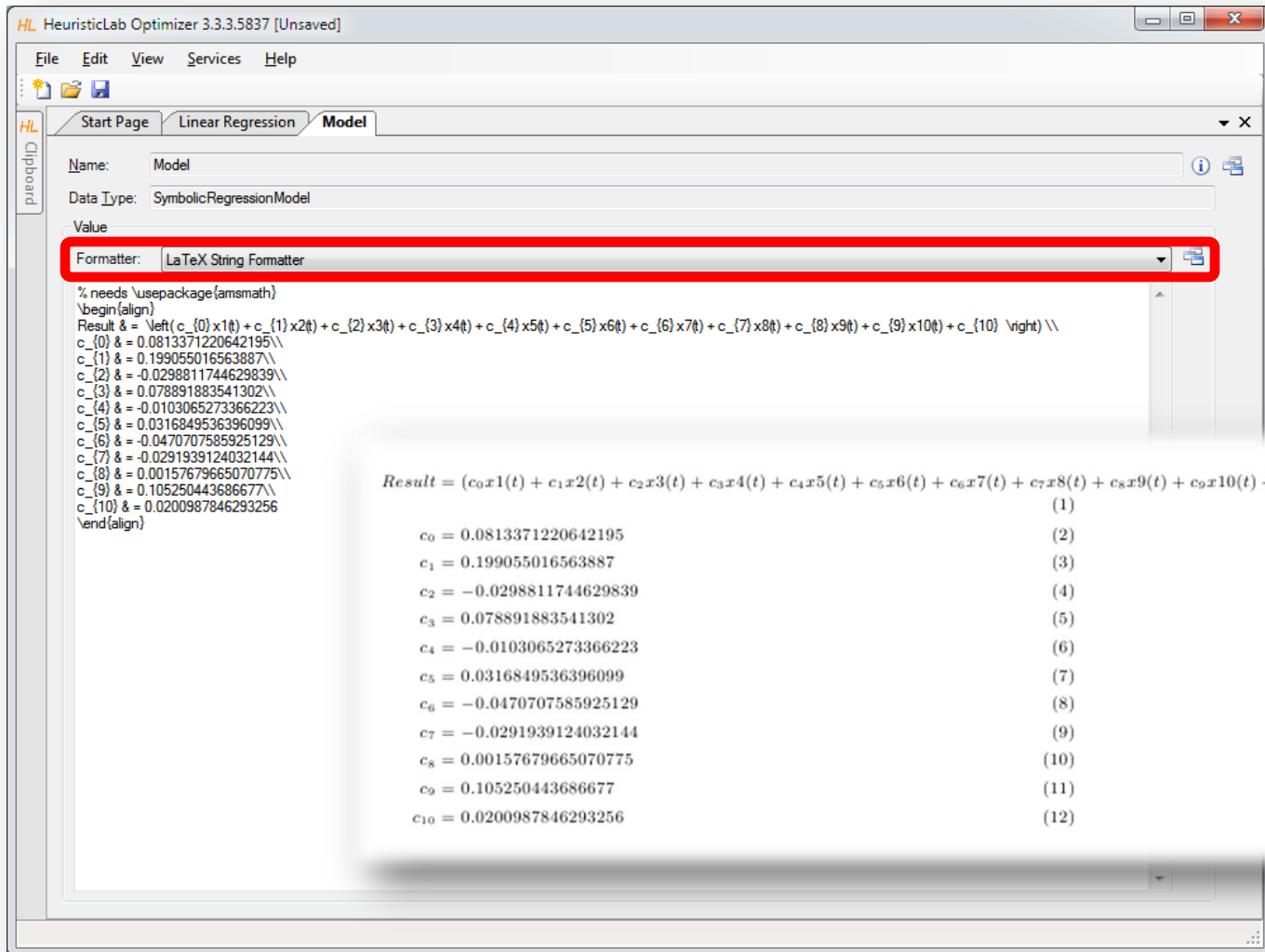


# Default Textual Representation for Model Export





# Textual Representation for Export to LaTeX



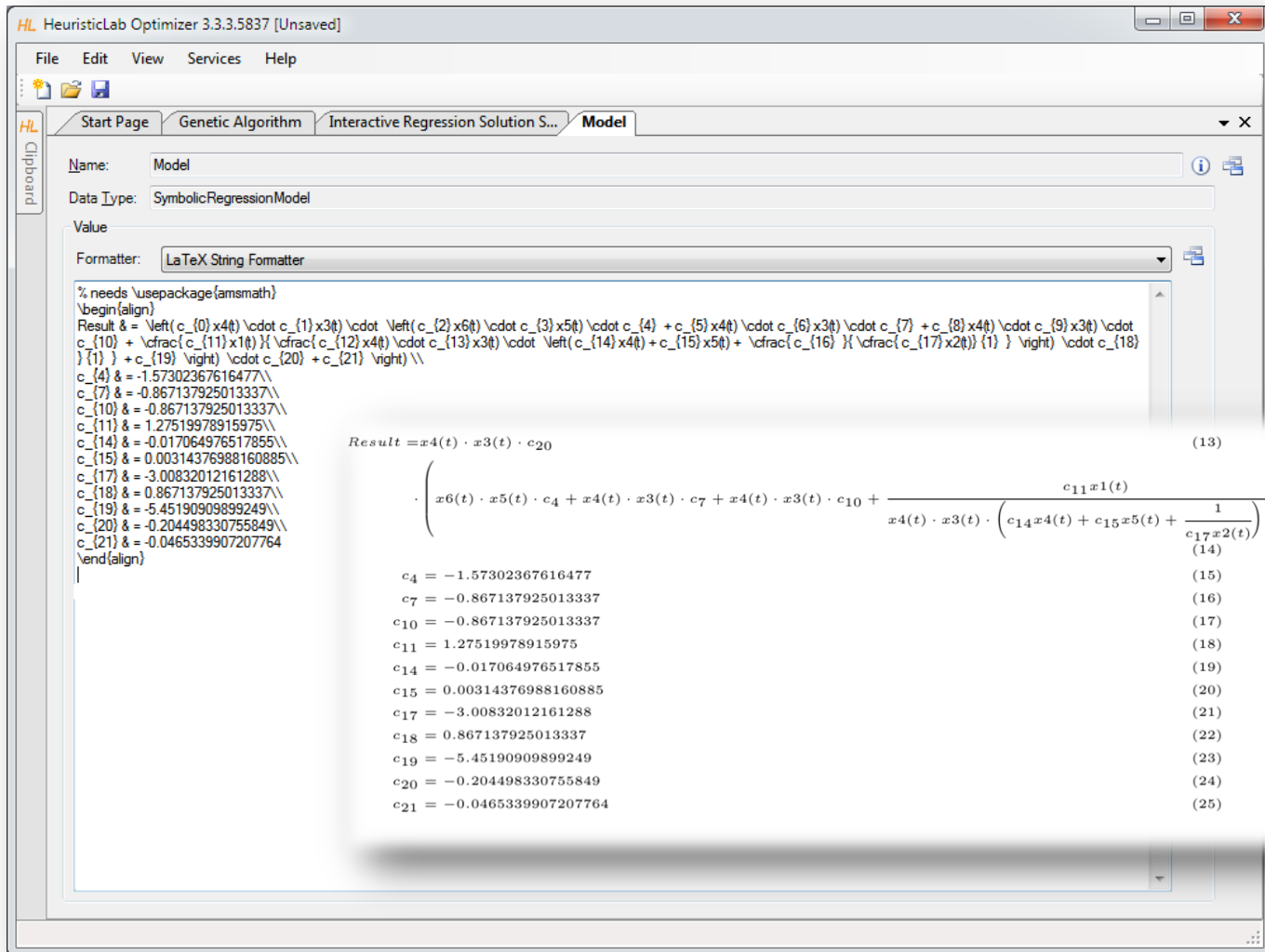
The screenshot shows the HeuristicLab Optimizer interface. The 'Model' tab is active, and the 'Formatter' dropdown menu is highlighted with a red box, showing 'LaTeX String Formatter' selected. Below the dropdown, the LaTeX code for the result is displayed. A callout box highlights the resulting LaTeX output, which includes the equation for the result and the values for the coefficients  $c_0$  through  $c_{10}$ .

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

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

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

# LaTeX Export



The screenshot shows the HeuristicLab Optimizer interface. The main window displays the 'Model' tab with a 'LaTeX String Formatter' selected. The left pane shows the LaTeX code for the model, and the right pane shows the rendered LaTeX output.

**LaTeX Code (Left Pane):**

```

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

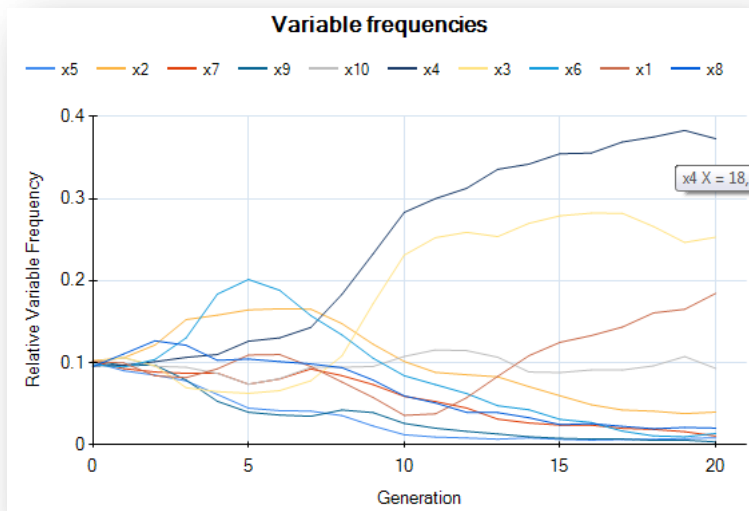
```

**Rendered LaTeX Output (Right Pane):**

$$\begin{aligned}
 Result &= x_4(t) \cdot x_3(t) \cdot c_{20} & (13) \\
 &\cdot \left( x_6(t) \cdot x_5(t) \cdot c_4 + x_4(t) \cdot x_3(t) \cdot c_7 + x_4(t) \cdot x_3(t) \cdot c_{10} + \frac{c_{11} x_1(t)}{x_4(t) \cdot x_3(t) \cdot \left( c_{14} x_4(t) + c_{15} x_5(t) + \frac{1}{c_{17} x_2(t)} \right) \cdot c_{18}} + c_{19} \right) + c_{21} \\
 c_4 &= -1.57302367616477 & (15) \\
 c_7 &= -0.867137925013337 & (16) \\
 c_{10} &= -0.867137925013337 & (17) \\
 c_{11} &= 1.27519978915975 & (18) \\
 c_{14} &= -0.017064976517855 & (19) \\
 c_{15} &= 0.00314376988160885 & (20) \\
 c_{17} &= -3.00832012161288 & (21) \\
 c_{18} &= 0.867137925013337 & (22) \\
 c_{19} &= -5.45190909899249 & (23) \\
 c_{20} &= -0.204498330755849 & (24) \\
 c_{21} &= -0.0465339907207764 & (25)
 \end{aligned}$$

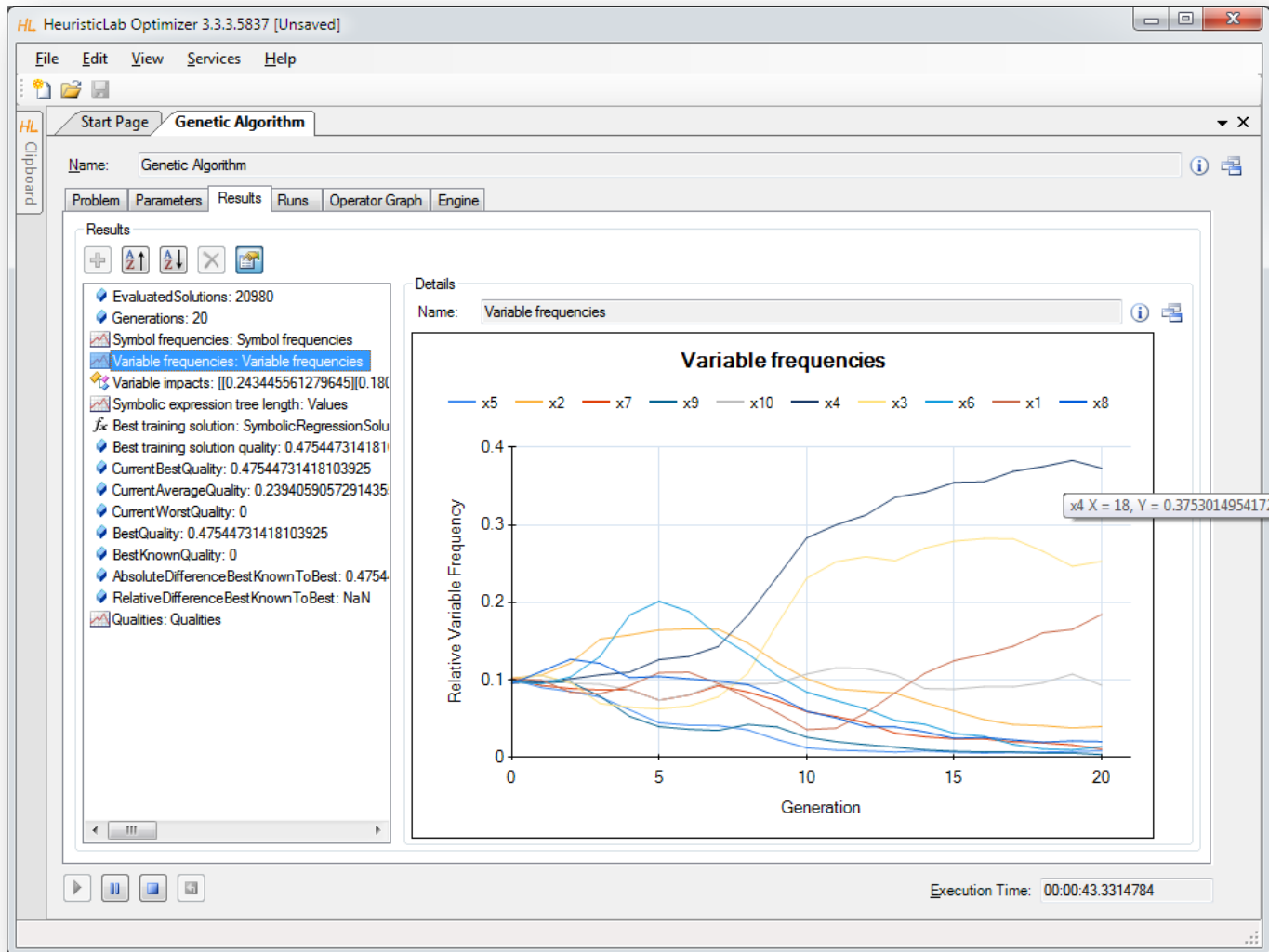
# Variable Relevance Analysis

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

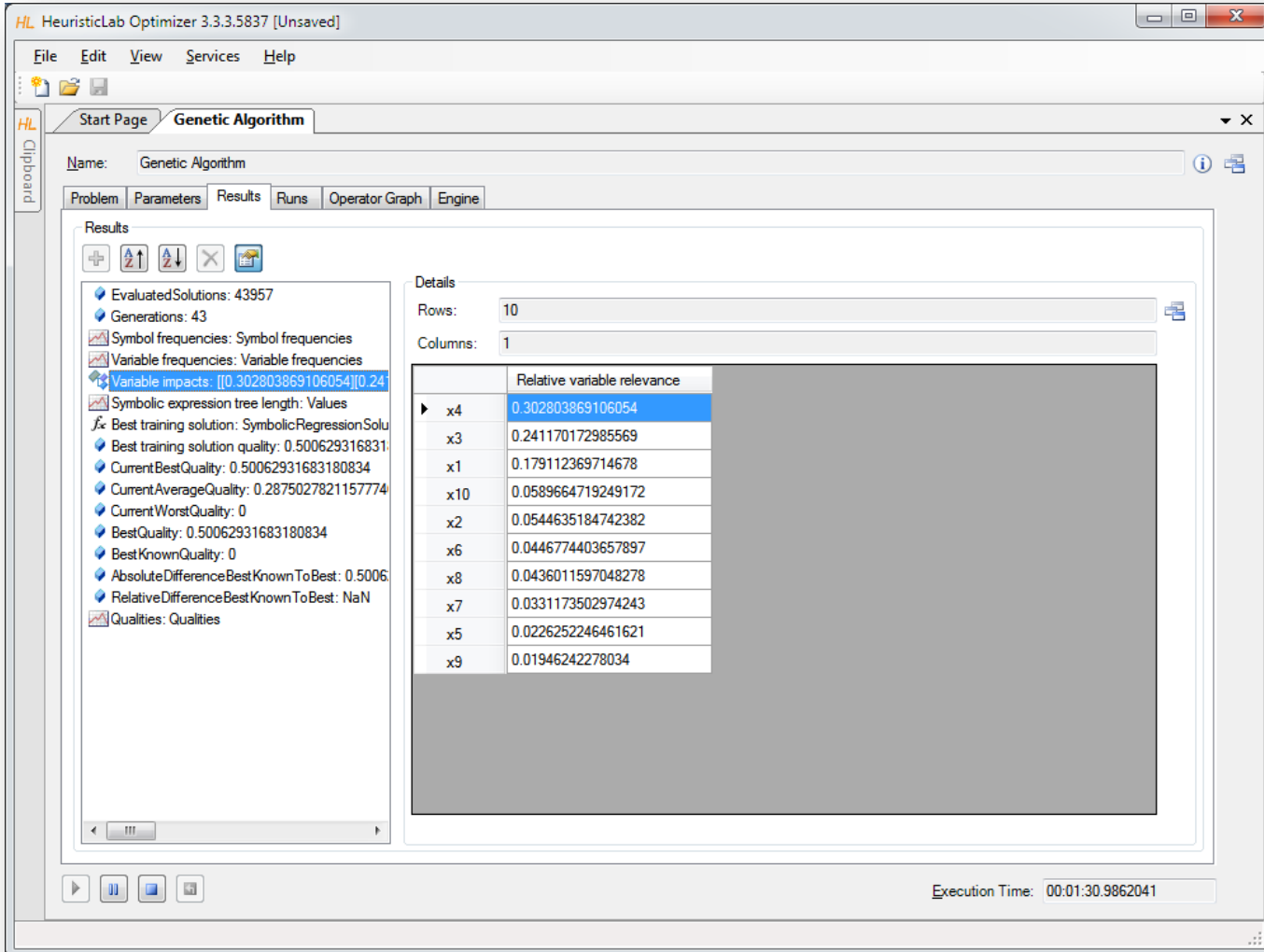


	Relative variable relevance
x4	0.302803869106054
x3	0.241170172985569
x1	0.179112369714678
x10	0.0589664719249172
x2	0.0544635184742382
x6	0.0446774403657897
x8	0.0436011597048278
x7	0.0331173502974243
x5	0.0226252246461621
x9	0.01946242278034

# Inspect Variable Frequency Chart



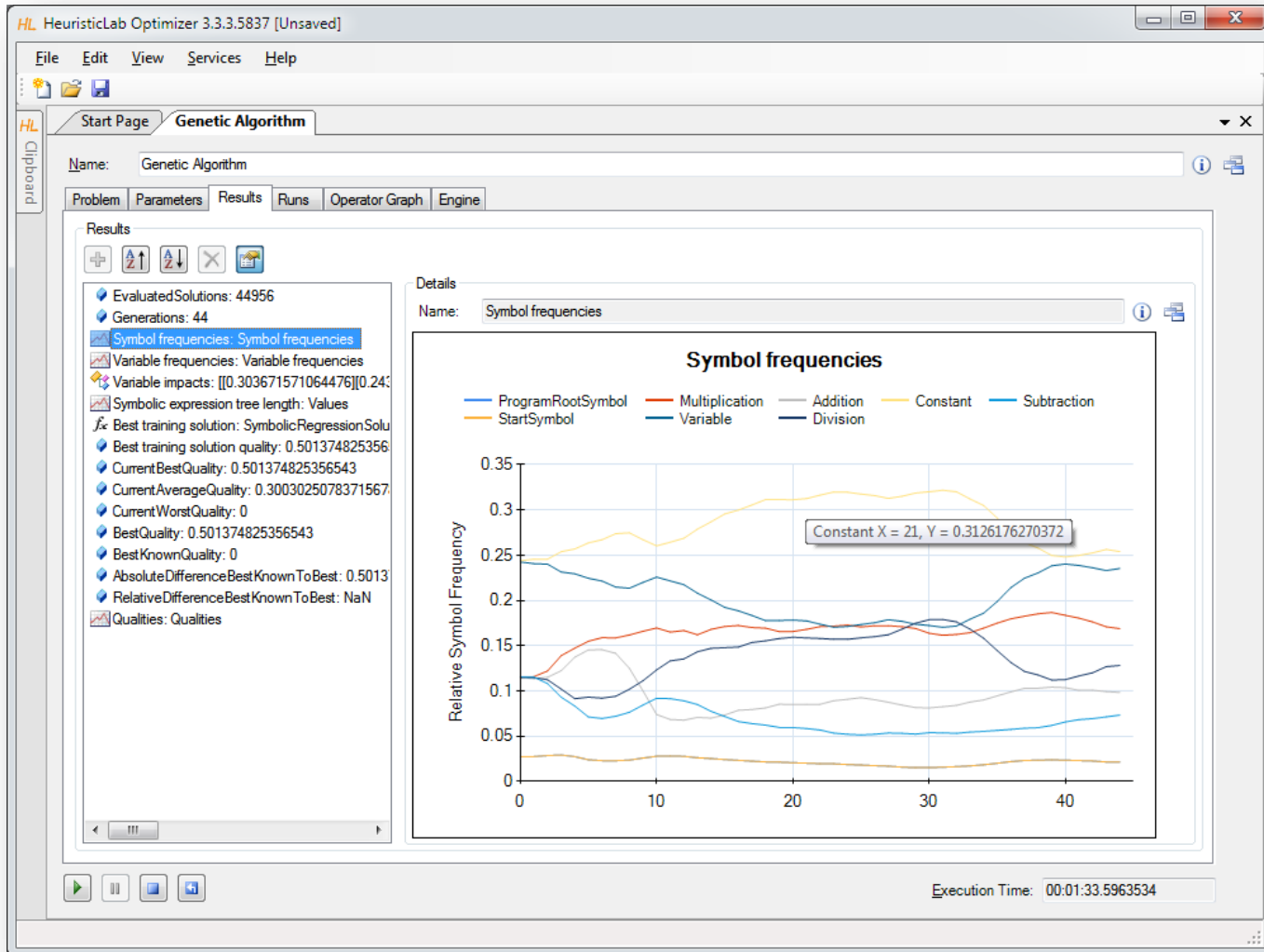
# Inspect Variable Impacts



The screenshot shows the HeuristicLab Optimizer interface. The main window is titled "HL HeuristicLab Optimizer 3.3.3.5837 [Unsaved]". The "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.

	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

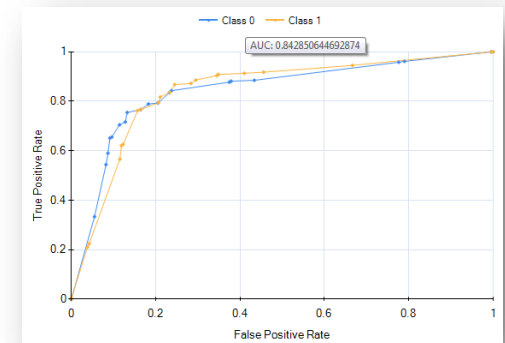
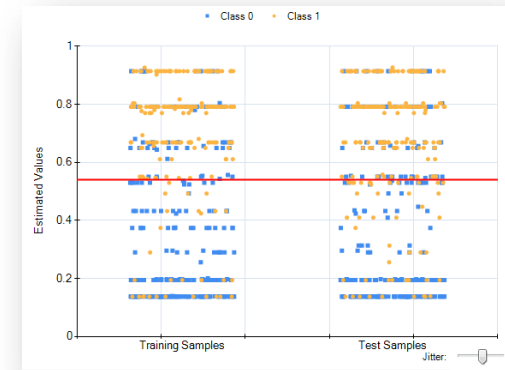
# Inspect Symbol Frequencies



# 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



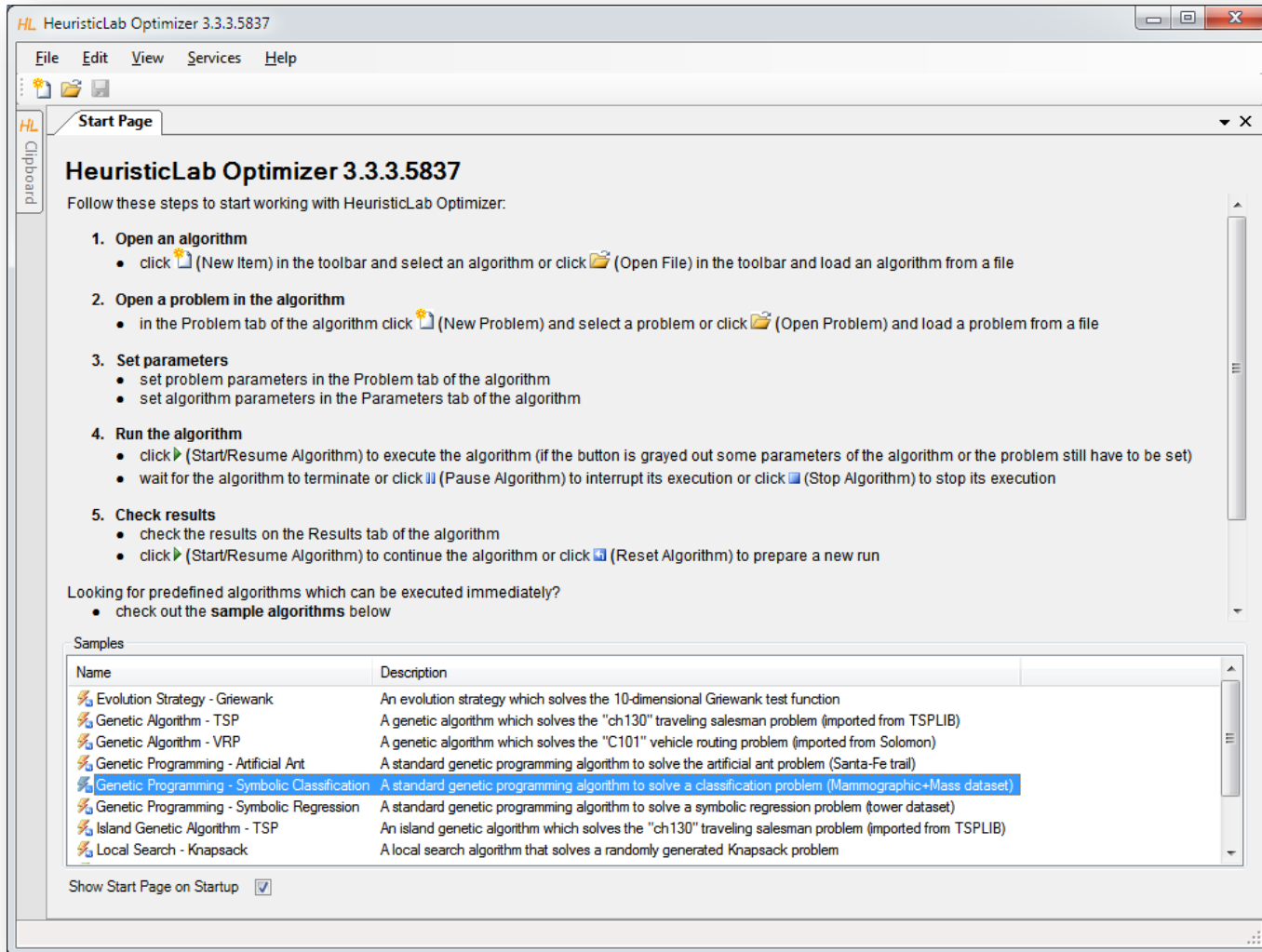
	Actual Class 0	Actual Class 1
Predicted Class 0	197	29
Predicted Class 1	64	190

# 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)
- download  
<http://dev.heuristiclab.com/AdditionalMaterial#IMMM2011>



# Open Sample












HL HeuristicLab Optimizer 3.3.3.5837

File Edit View Services Help

HL 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?

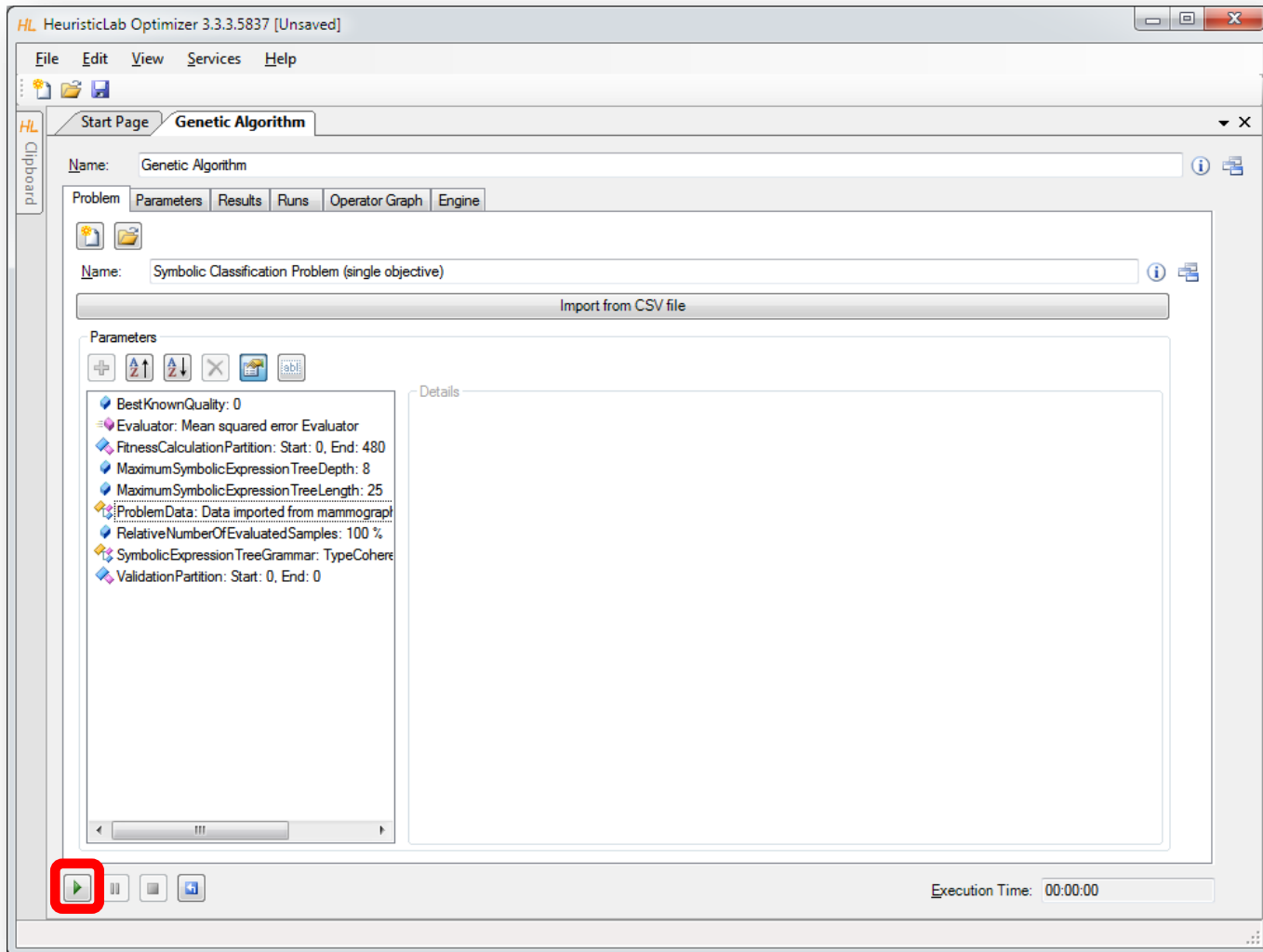
- check out the **sample algorithms** below

Samples

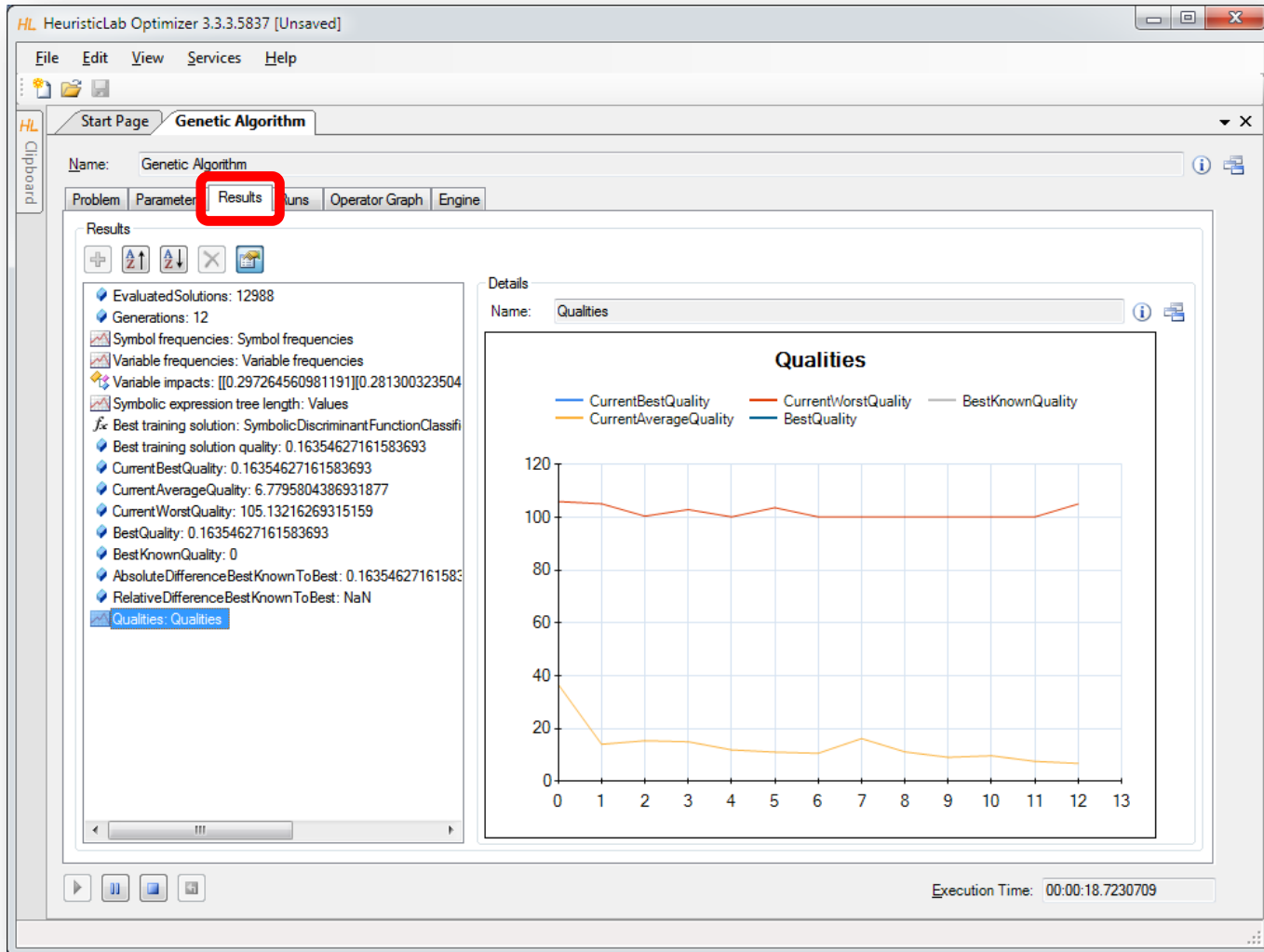
Name	Description
 Evolution Strategy - Griewank	An evolution strategy which solves the 10-dimensional Griewank test function
 Genetic Algorithm - TSP	A genetic algorithm which solves the "ch130" traveling salesman problem (imported from TSPLIB)
 Genetic Algorithm - VRP	A genetic algorithm which solves 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

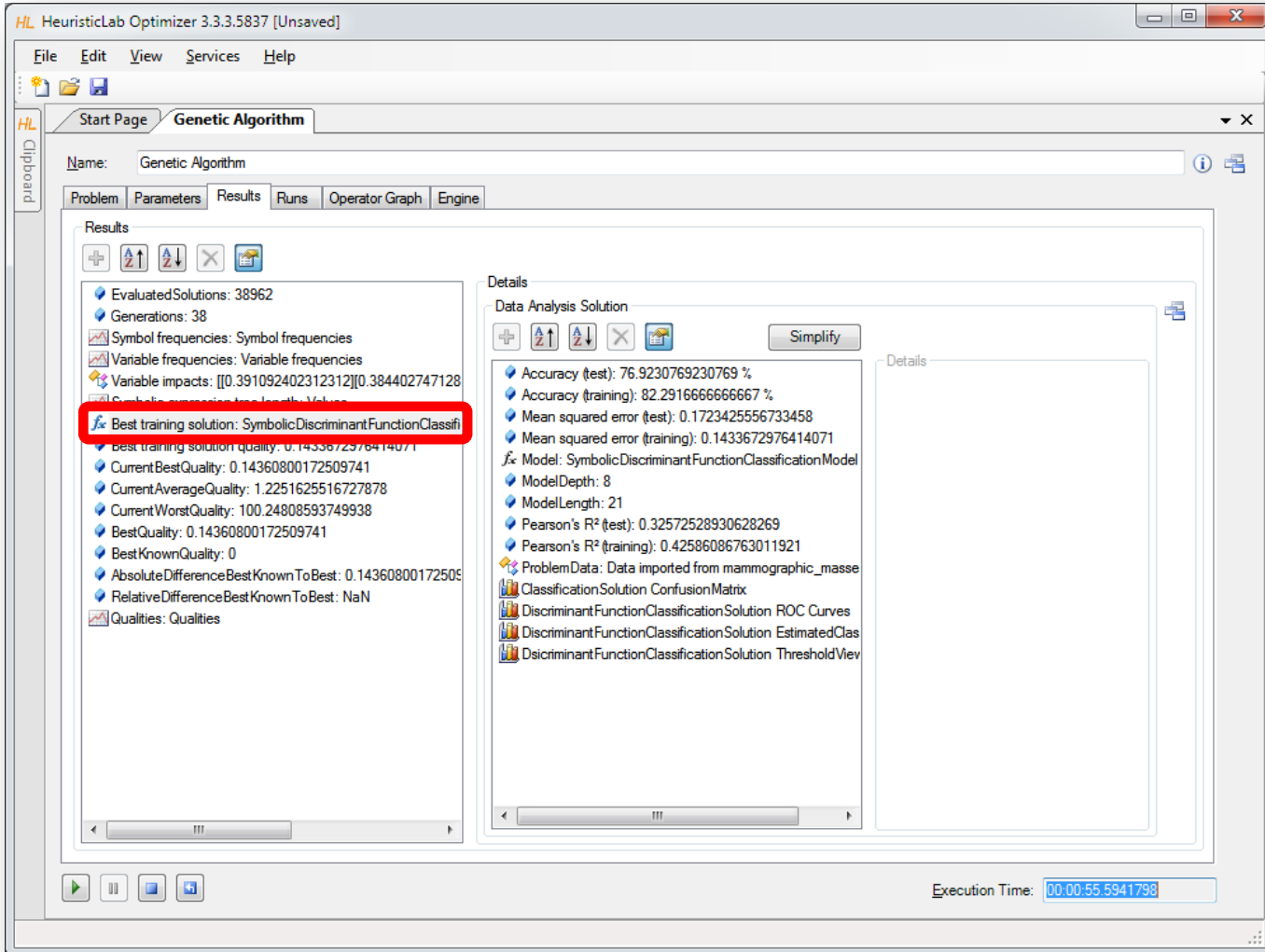
# Configure and Run Algorithm



# Inspect Quality Linechart



# Inspect Best Training Solution



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

**Results**

- Evaluated Solutions: 38962
- Generations: 38
- Symbol frequencies: Symbol frequencies
- Variable frequencies: Variable frequencies
- Variable impacts: [[0.391092402312312][0.384402747128
- Symbol frequencies: Symbol frequencies
- Best training solution: SymbolicDiscriminantFunctionClassifi**
- Best training solution quality: 0.14360800172509741
- Current Best Quality: 0.14360800172509741
- Current Average Quality: 1.2251625516727878
- Current Worst Quality: 100.24808593749938
- Best Quality: 0.14360800172509741
- Best Known Quality: 0
- Absolute Difference Best Known To Best: 0.14360800172509741
- Relative Difference Best Known To Best: NaN
- Qualities: Qualities

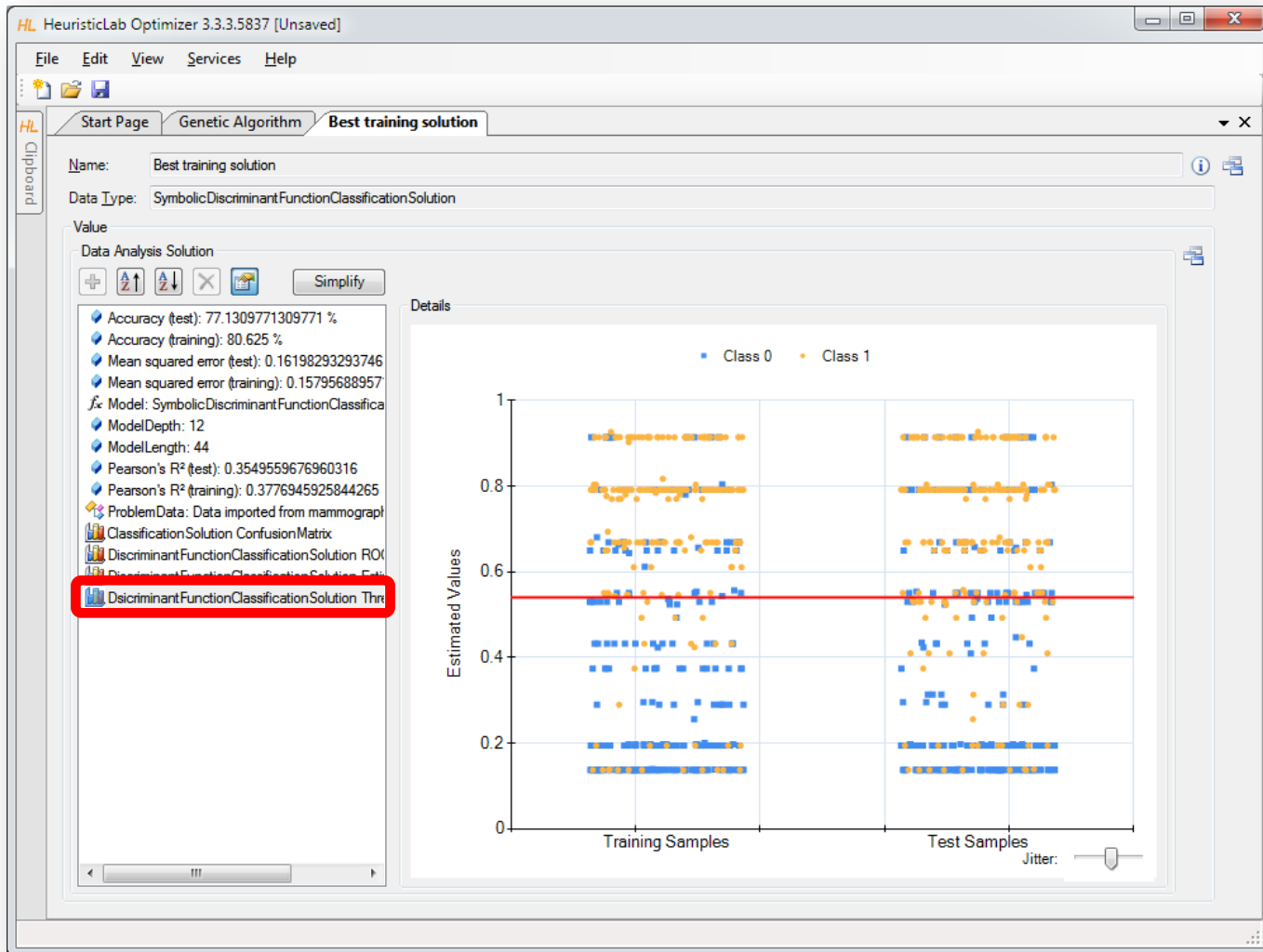
**Details**

Data Analysis Solution

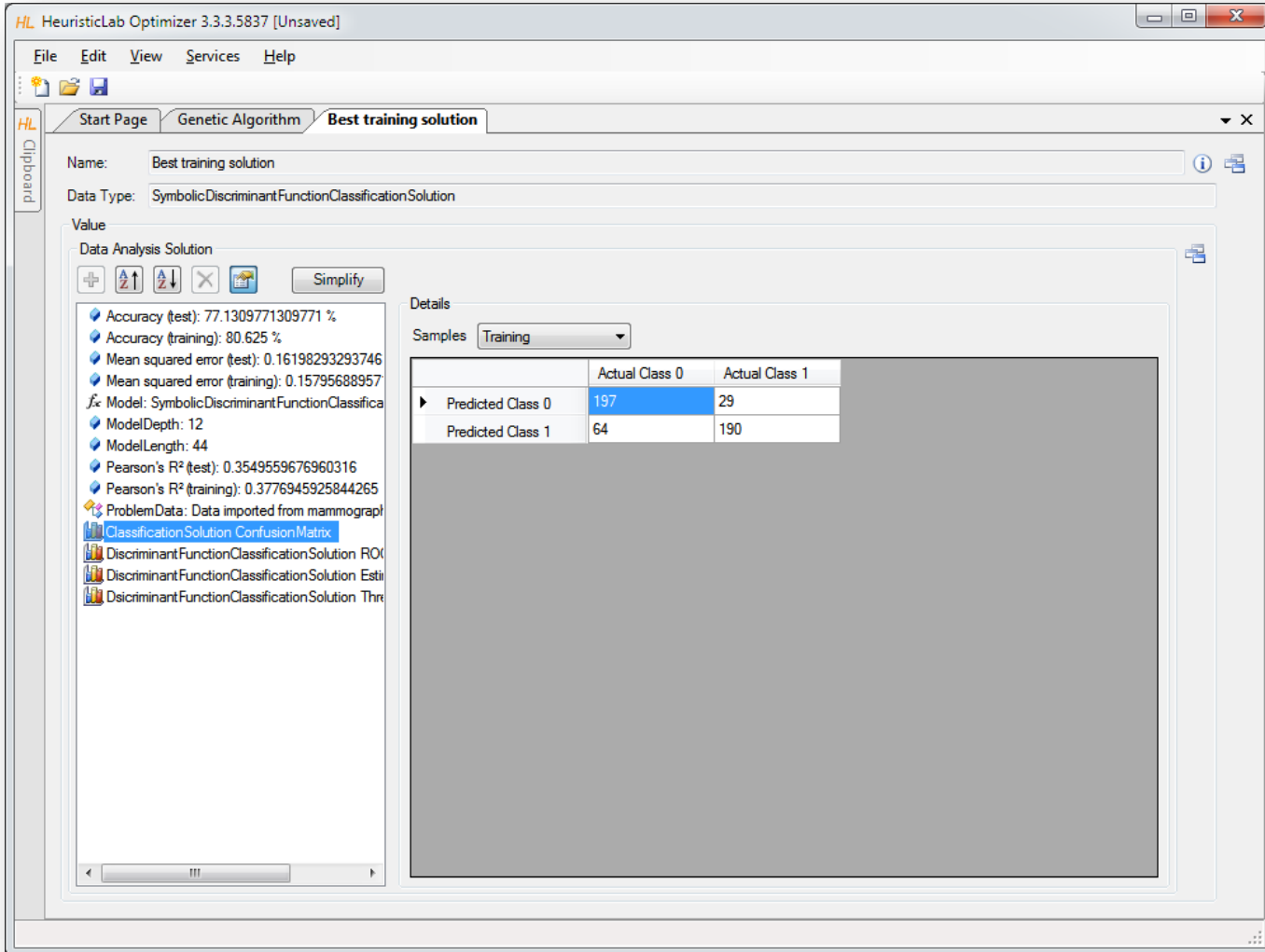
- Accuracy (test): 76.9230769230769 %
- Accuracy (training): 82.2916666666667 %
- Mean squared error (test): 0.1723425556733458
- Mean squared error (training): 0.1433672976414071
- Model: SymbolicDiscriminantFunctionClassificationModel
- Model Depth: 8
- Model Length: 21
- Pearson's R<sup>2</sup> (test): 0.32572528930628269
- Pearson's R<sup>2</sup> (training): 0.42586086763011921
- Problem Data: Data imported from mammographic\_masse
- Classification Solution Confusion Matrix
- DiscriminantFunctionClassificationSolution ROC Curves
- DiscriminantFunctionClassificationSolution Estimated Class
- DiscriminantFunctionClassificationSolution Threshold View

Execution Time: 00:00:55.5941798

# Inspect Model Output and Thresholds



# Inspect Confusion Matrix



The screenshot shows the HeuristicLab Optimizer interface. The main window displays the 'Best training solution' for a Genetic Algorithm. The 'Value' section shows a 'Data Analysis Solution' with various performance metrics. The 'Details' section shows a confusion matrix for the 'Training' samples.

**Value**

Data Analysis Solution

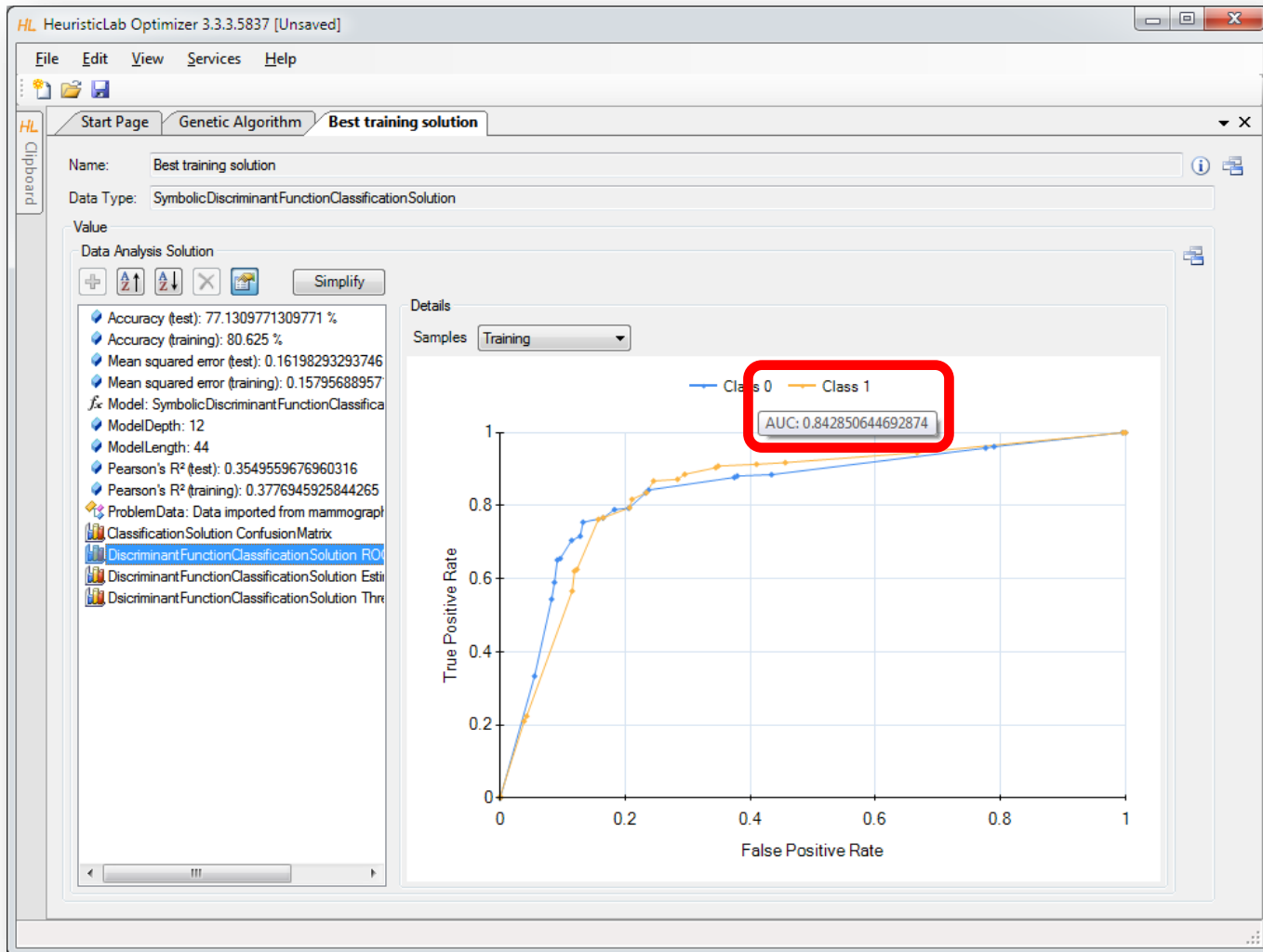
Accuracy (test): 77.1309771309771 %  
Accuracy (training): 80.625 %  
Mean squared error (test): 0.16198293293746  
Mean squared error (training): 0.15795688957  
Model: SymbolicDiscriminantFunctionClassifica  
ModelDepth: 12  
ModelLength: 44  
Pearson's R<sup>2</sup> (test): 0.3549559676960316  
Pearson's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

**Details**

Samples: Training

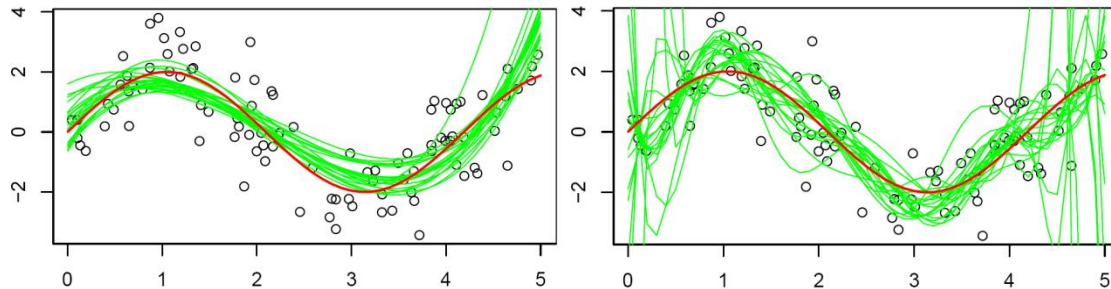
	Actual Class 0	Actual Class 1
Predicted Class 0	197	29
Predicted Class 1	64	190

# Inspect ROC Curve



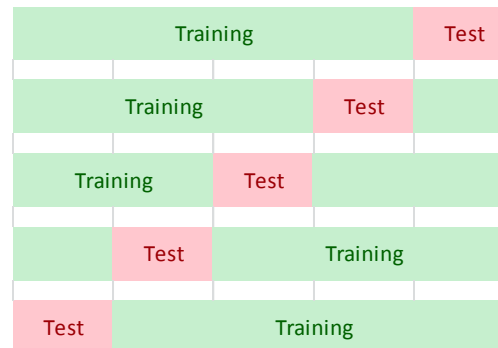
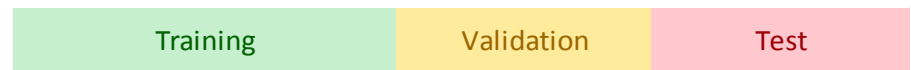
# Validation of Results

- Overfitting = memorizing data



- Strategies to reduce overfitting

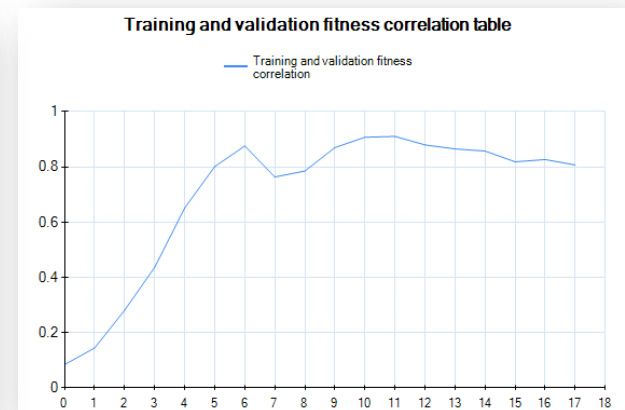
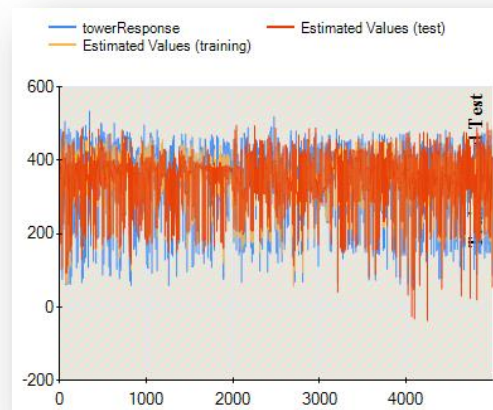
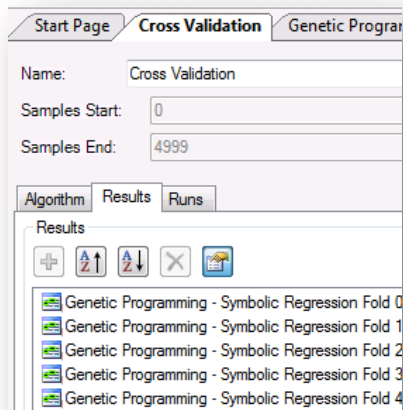
- validation partition
- cross-validation



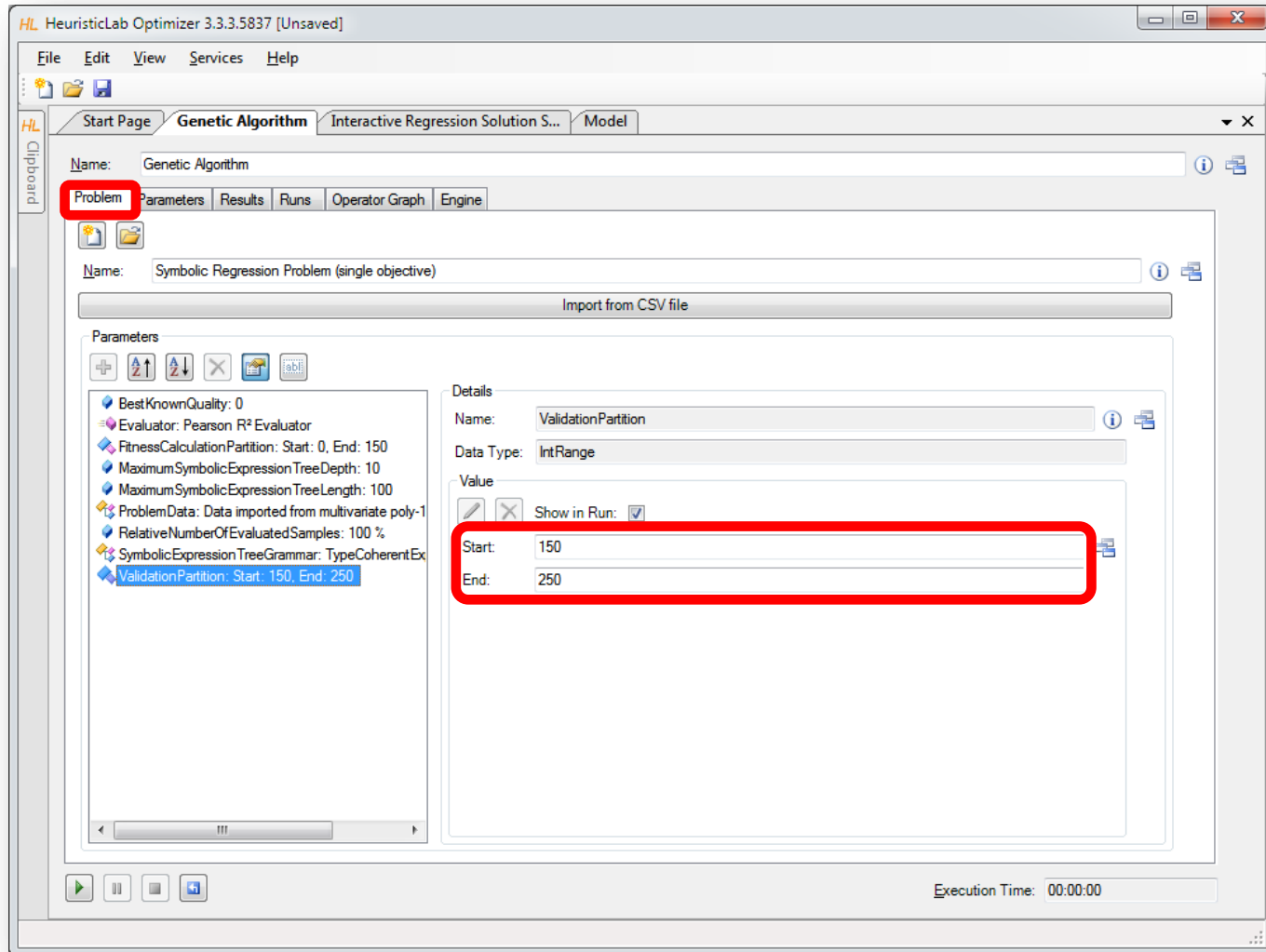


# Validation of Results

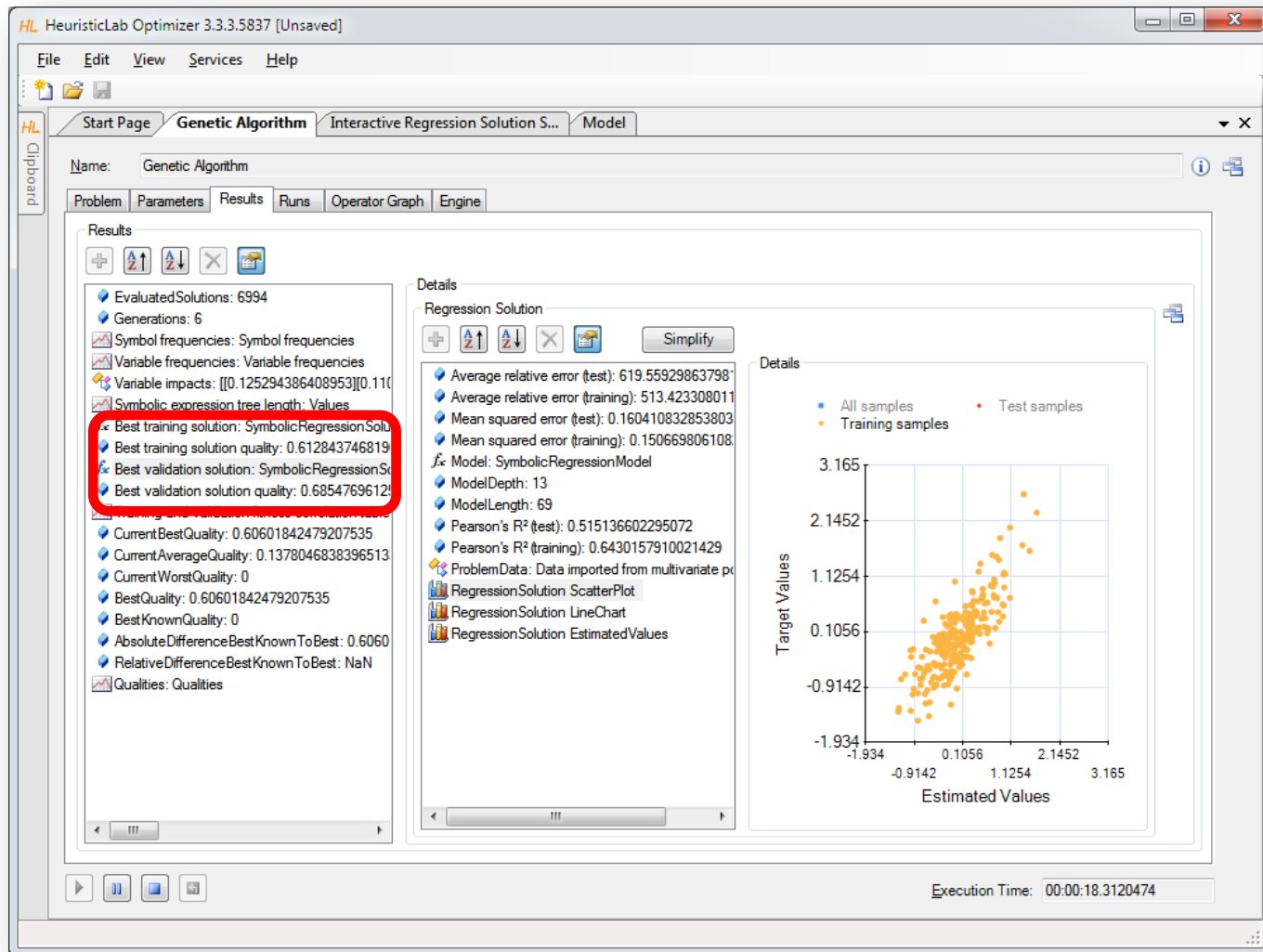
- Demonstration
  - Configuration of a validation set
  - Inspection of best solution on validation set
  - Analysis of training- and validation fitness correlation
  - Cross-validation
    - Configuration
    - Analysis of results



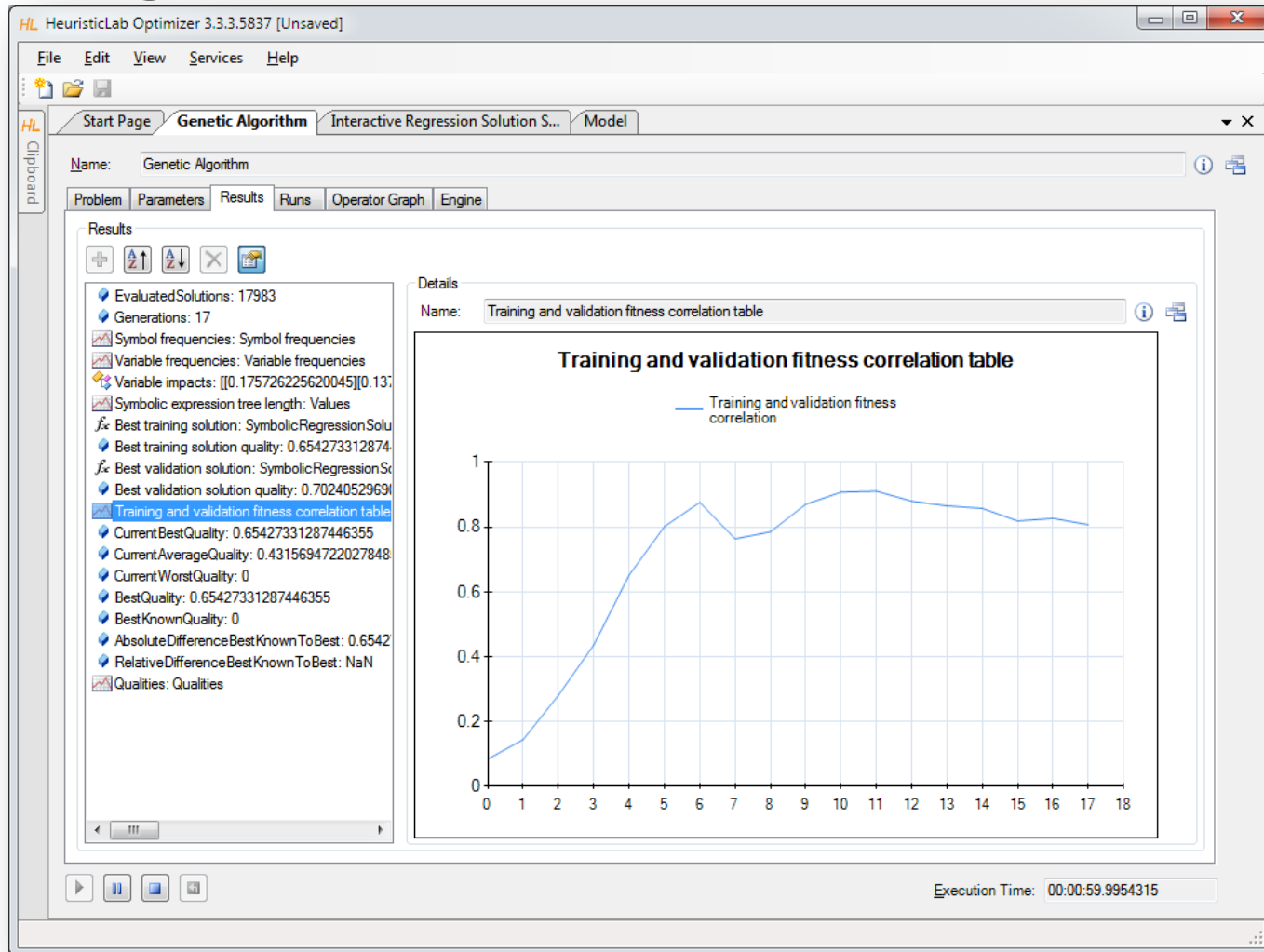
# Configuration of Validation Partition



# Inspect Best Model on Validation Partition



# Inspect Linechart of Correlation of Training and Validation Fitness



# Agenda

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

# Some Additional Features

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

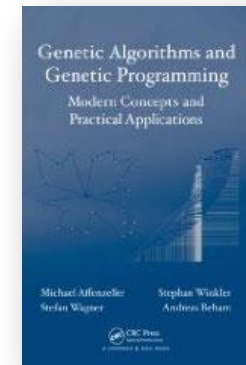
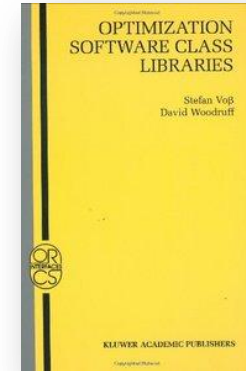


# Planned Features

- Algorithms & Problems
  - steady-state genetic algorithm
  - unified tabu search for vehicle routing
  - scatter search
  - ...
- Cloud Computing
  - port HeuristicLab Hive to Windows Azure
- Linux
  - port HeuristicLab to run on Mono and Linux machines
- Have a look at the HeuristicLab roadmap
  - <http://dev.heuristiclab.com/trac/hl/core/roadmap>
- Any other ideas, requests or recommendations?
  - please write an e-mail to [support@heuristiclab.com](mailto:support@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
- Detailed list of all publications of the HEAL research group: <http://research.fh-ooe.at/de/orgunit/detail/356#showpublications>

# Questions & Answers



<http://dev.heuristiclab.com>

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

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