



HeuristicLab

A Paradigm-Independent and Extensible
Environment for Heuristic Optimization

Optimizing External Applications with HeuristicLab

A. Beham, M. Kommenda

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



HEAL

Heuristic and Evolutionary
Algorithms Laboratory



Heuristic
Optimization in
Production and
Logistics

Instructor Biographies

- Andreas Beham
 - Research Associate (since 2007)
 - Team: Combinatorial and Simulation-based Optimization
 - Graduate of Johannes Kepler University
 - PhD in progress
 - Member of the HEAL research group
 - Architect of HeuristicLab
 - <http://heal.heuristiclab.com/team/beham>
- Michael Kommenda
 - Research Associate (since 2008)
 - Team: System Identification and Data Analysis
 - Graduate of University of Applied Sciences Upper Austria
 - PhD in progress
 - Member of the HEAL research group
 - Architect of HeuristicLab
 - <http://heal.heuristiclab.com/team/kommenda>



Latest Version of this Tutorial



- An up-to-date version of this tutorial can be downloaded from the HeuristicLab website
 - <http://dev.heuristiclab.com/trac.fcgi/browser/trunk/documentation/Tutorials>

Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- **Demonstration Part I: External Evaluation Problem**
- **Demonstration Part II: MATLAB and Scilab Parameter Optimization Problem**
- **Demonstration Part III: Programmable Problem**
- 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 optimization of parameters in external applications
- Demonstrate optimization of parameters in MATLAB and Scilab
- Demonstrate optimization of custom problem definitions
- Outline some additional features

Introduction



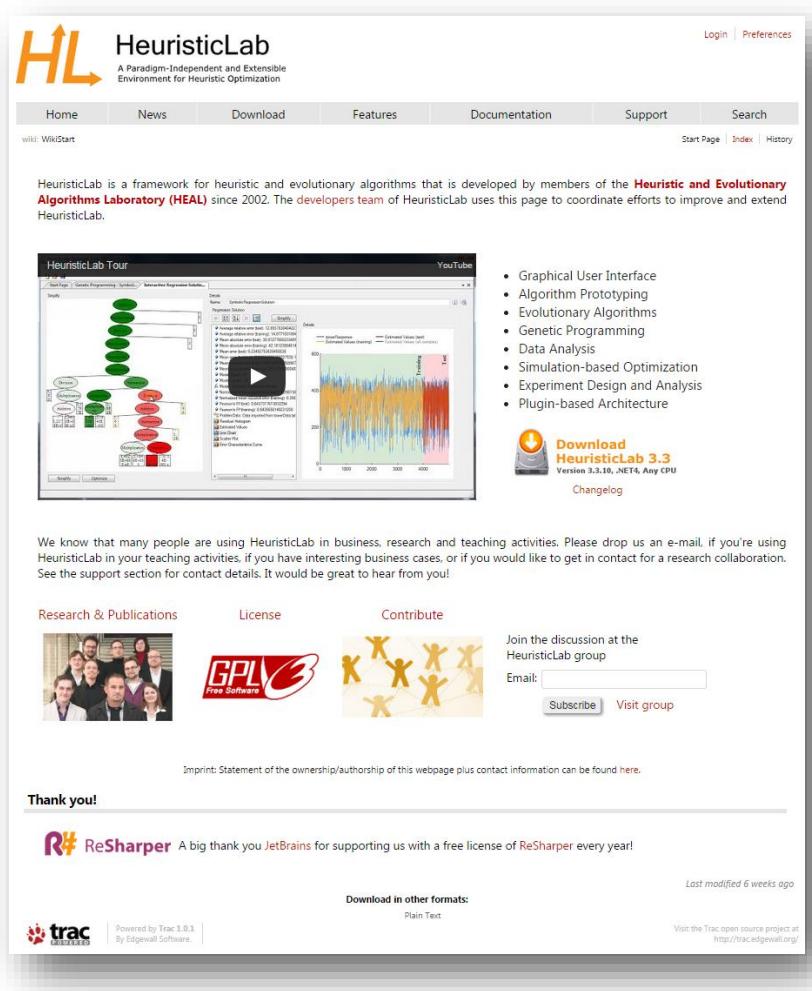
- Motivation and Goals
 - graphical user interface
 - paradigm independence
 - multiple algorithms and problems
 - large scale experiments and analyses
 - parallelization
 - extensibility, flexibility and reusability
 - visual and interactive algorithm development
 - multiple layers of abstraction
- 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 3.3.14 “Denver”
 - released on July 24th, 2016



Where to get HeuristicLab?



- Download binaries
 - deployed as ZIP archives
 - latest stable version 3.3.14 “Denver”
 - released on July 24th, 2016
 - daily trunk builds
 - <http://dev.heuristiclab.com/download>
- Check out sources
 - SVN repository
 - HeuristicLab 3.3.14 tag
 - <http://svn.heuristiclab.com/svn/core/tags/3.3.14>
 - Stable development version
 - <http://svn.heuristiclab.com/svn/core/stable>
- License
 - GNU General Public License (Version 3)
- System requirements
 - Microsoft .NET Framework 4.5
 - enough RAM and CPU power ;-)

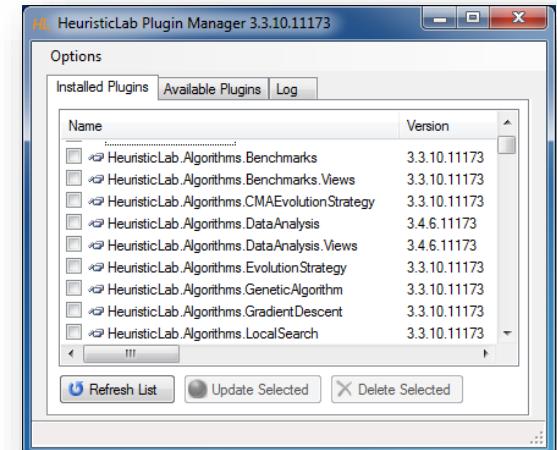
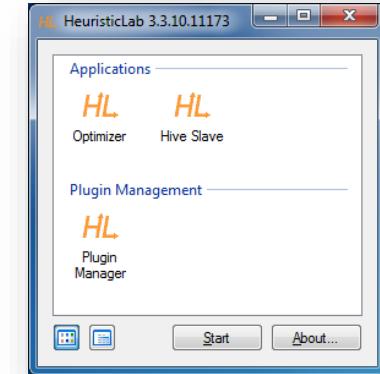


The screenshot shows the HeuristicLab website homepage. At the top right is the HeuristicLab logo and navigation links for Login, Preferences, Home, News, Download, Features, Documentation, Support, and Search. Below the header is a banner with the text: "HeuristicLab is a framework for heuristic and evolutionary algorithms that is developed by members of the **Heuristic and Evolutionary Algorithms Laboratory (HEAL)** since 2002. The developers team of HeuristicLab uses this page to coordinate efforts to improve and extend HeuristicLab." To the left is a "WikiStart" link. The main content area features a "HeuristicLab Tour" video player showing a genetic programming interface with a tree diagram and a scatter plot. To the right of the video is a list of features: Graphical User Interface, Algorithm Prototyping, Evolutionary Algorithms, Genetic Programming, Data Analysis, Simulation-based Optimization, Experiment Design and Analysis, and Plugin-based Architecture. Below the tour is a "Download HeuristicLab 3.3" section with a download button and a changelog link. Further down is a "Research & Publications" section with a photo of the development team, a "License" section with the GPL logo, and a "Contribute" section with a yellow stick figure icon. On the right is a sidebar for joining the discussion group with fields for Email and Subscribe, and a "Visit group" link. At the bottom are links for Imprint, Thank you!, ReSharper, Download in other formats, and Trac information.

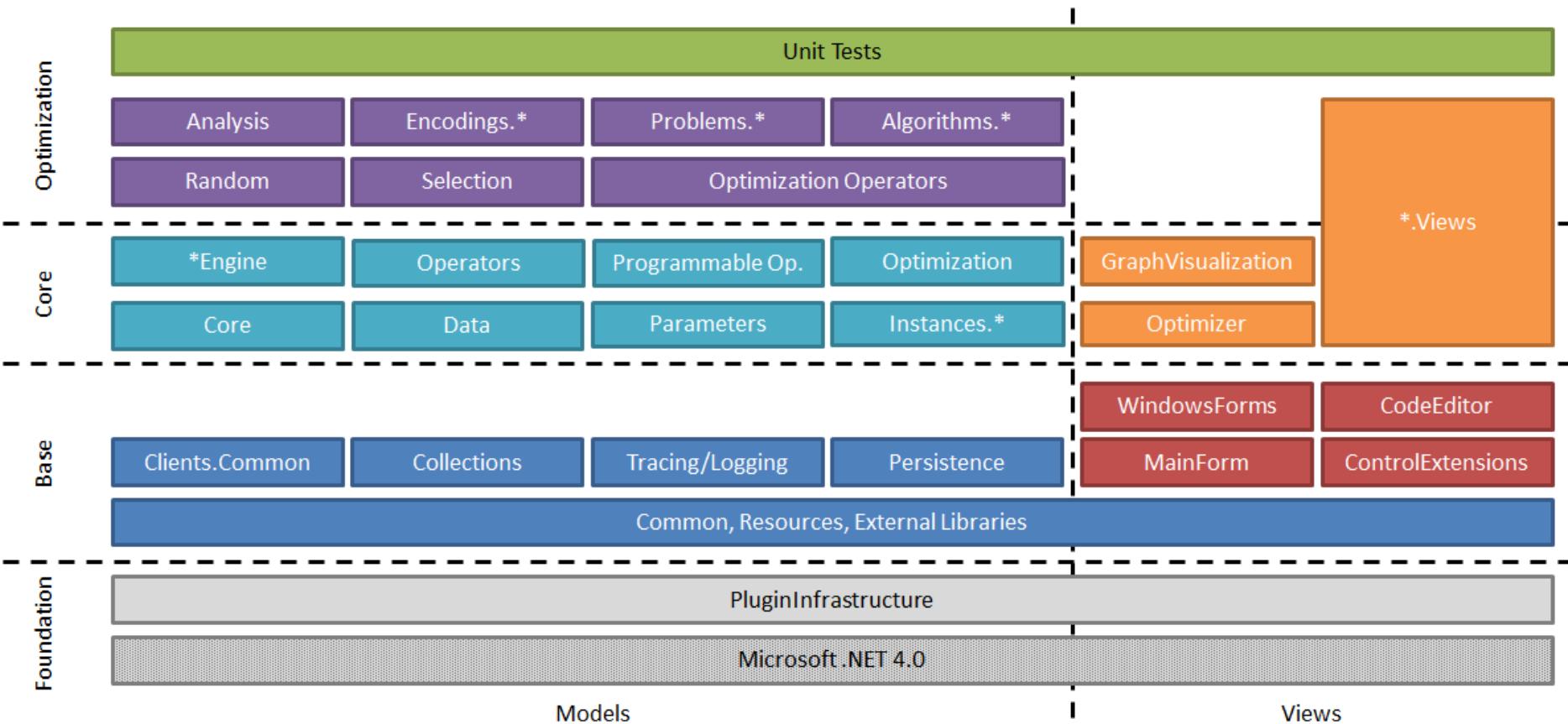
Plugin Infrastructure



- HeuristicLab consists of many assemblies
 - 150+ plugins in HeuristicLab 3.3.14
 - 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

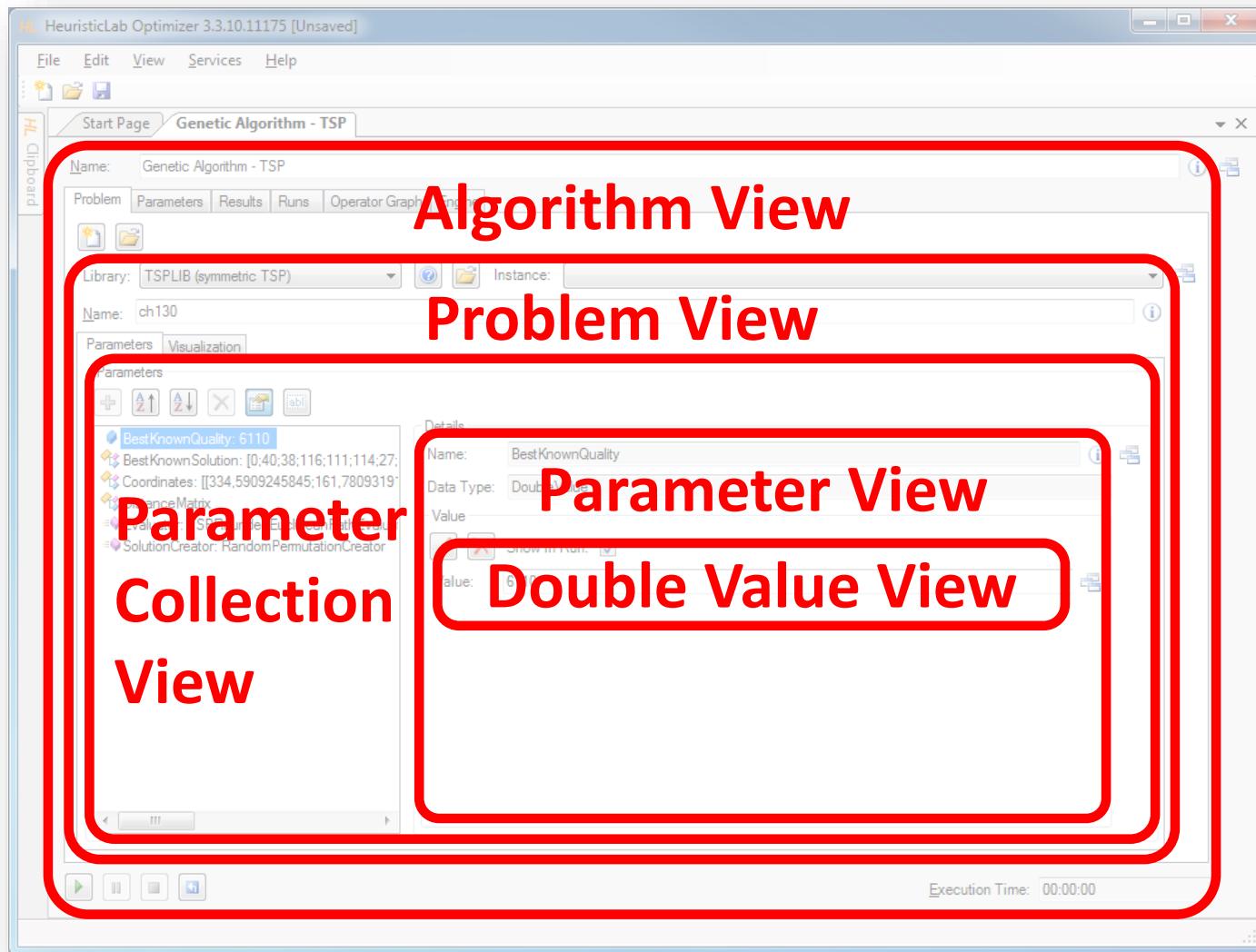


Graphical User Interface



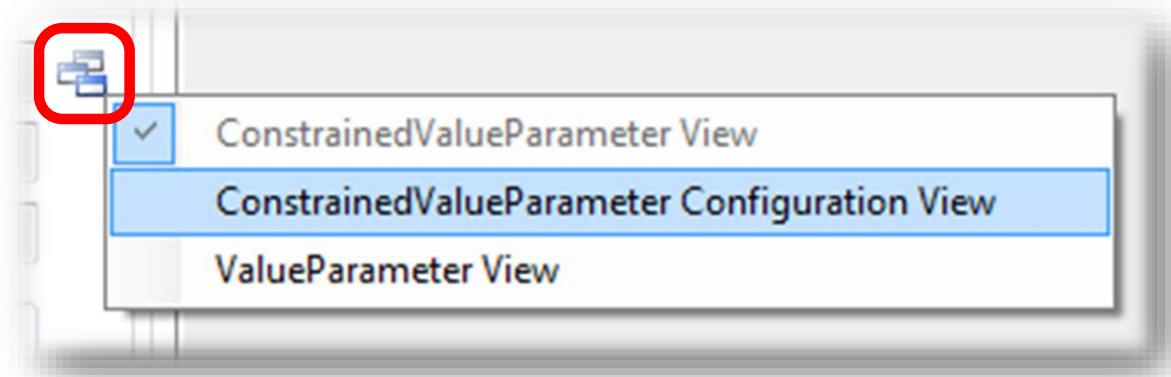
- HeuristicLab GUI is made up of views
 - views are visual representations of content objects
 - views are composed in the same way as their content
 - views and content objects are loosely coupled
 - multiple different views may exist for the same content
- Drag & Drop
 - views support drag & drop operations
 - 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 (OSGA)
- Island Genetic Algorithm
- Island Offspring Selection Genetic Algorithm
- Parameter-less Population Pyramid (P3)
- SASEGASA
- Relevant Alleles Preserving GA (RAPGA)
- Aged-Layered Population Structure (ALPS)
- 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 (NCA)
- Artificial Neural Networks
- Random Forests
- Support Vector Machines
- Gaussian Processes
- Gradient Boosted Trees
- Gradient Boosted Regression

Additional Algorithms

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

Available Problems

Combinatorial Problems

- Test Problems (1-max, NK, HIFF, Deceptive Trap)
- Traveling Salesman
- Probabilistic Traveling Salesman
- Vehicle Routing (MDCVRPTW, PDPTW, ...)
- Knapsack
- Bin Packing
- Job Shop Scheduling
- Linear Assignment
- Quadratic Assignment
- Orienteering

Genetic Programming Problems

- Test Problems (Even Parity, MUX)
- Symbolic Classification
- Symbolic Regression
- Symbolic Time-Series Prognosis
- Artificial Ant
- Lawn Mower
- Robocode
- Grammatical Evolution

Additional Problems

- Single-/Multi-Objective Test Functions
 - Ackley, Griewank, Rastrigin, Sphere, etc.
- Programmable Problem
- External Evaluation Problem
(generic TCP/IP, Scilab, MATLAB)
- Regression, Classification, Clustering
- Trading

Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- Demonstration Part I: External Evaluation Problem
- Demonstration Part II: MATLAB and Scilab Parameter Optimization Problem
- Demonstration Part III: Programmable Problem
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

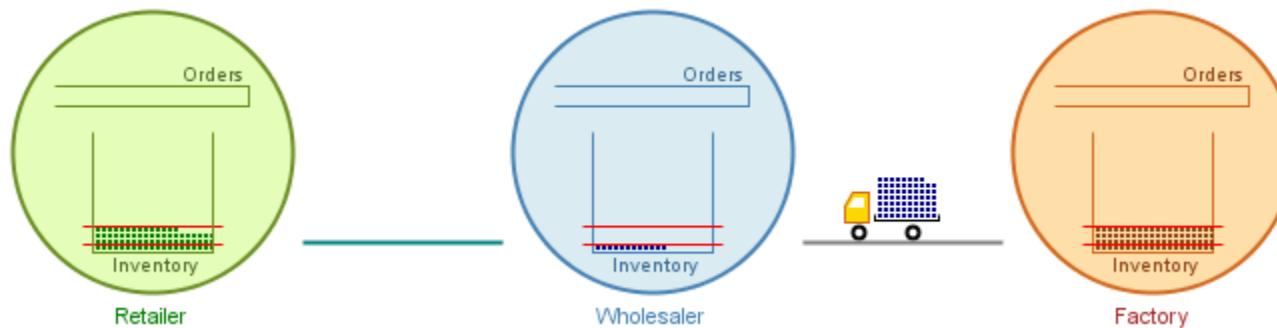
Demonstration Part I: External Evaluation Problem



- Optimize parameters of an existing simulation model
- Implement necessary steps to talk with HeuristicLab

Supply Chain Simulation

- (s, S) Order Policy
 - 3 Echelons and 2 Parameters per Echelon
 - Minimize Inventory and Ordering Costs
 - Maintain a minimum service level
 - Bound on the waiting time due to „out of stock“ situations



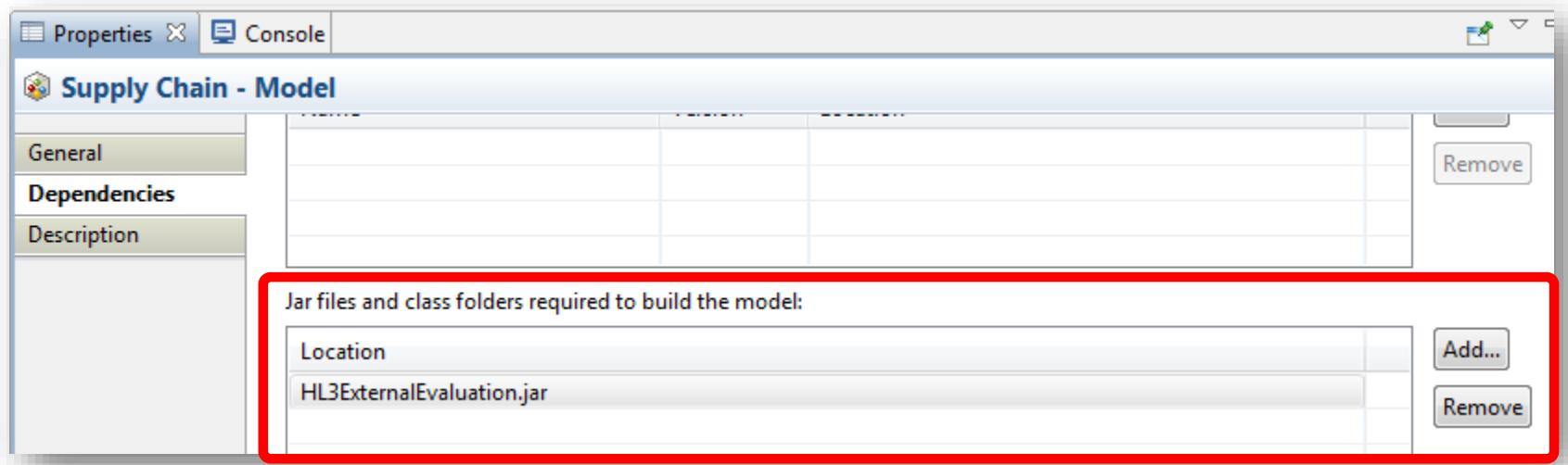
Supply Chain Simulation



- Create a new simulation experiment for evaluating parameters from HeuristicLab
- Create the problem definition in HeuristicLab
- Optimize

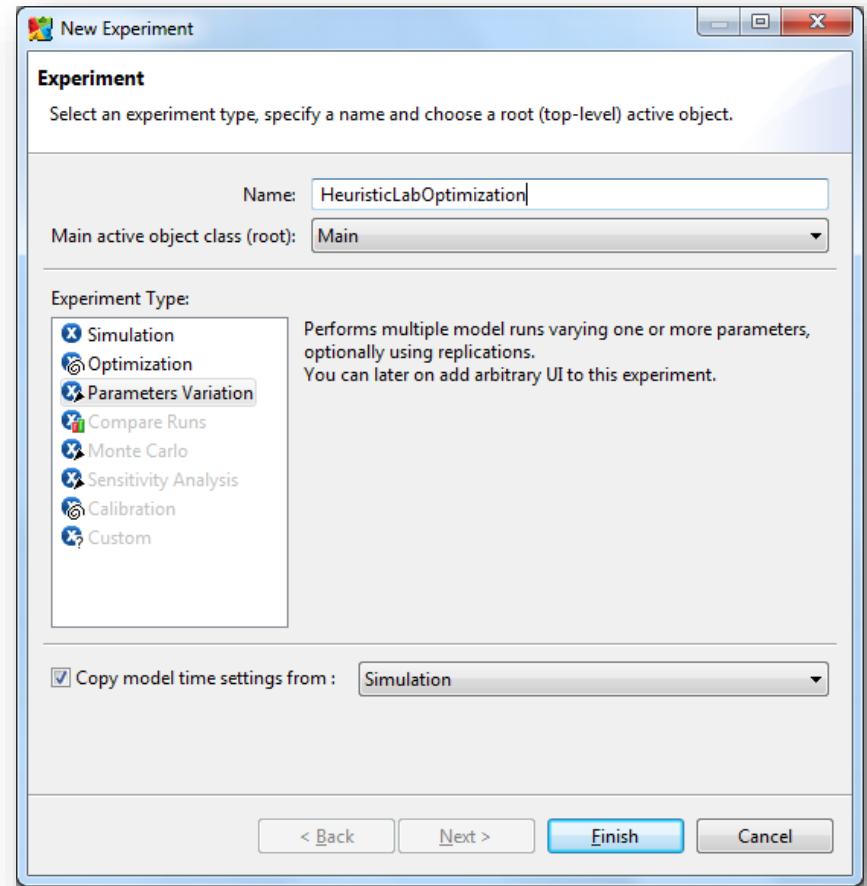
Supply Chain Simulation

- Download communication helper library
 - <http://dev.heuristiclab.com/trac.cgi/wiki/Documentation/Howto/OptimizeExternalApplications#no1>
- Add library as dependency to the model



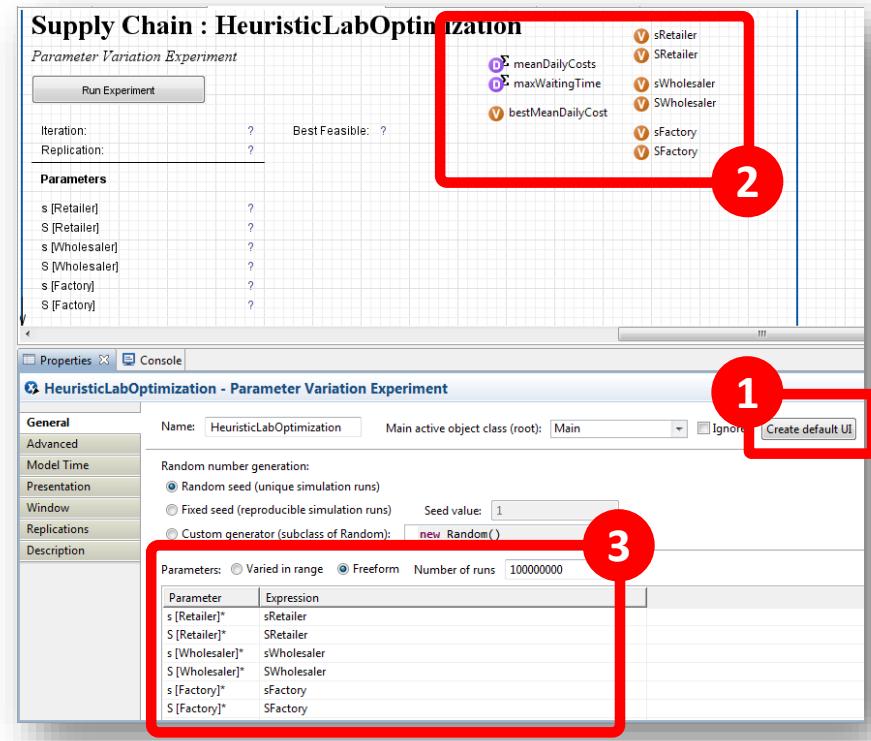
Supply Chain Simulation

- The model is included as a sample in AnyLogic
- Open the model and create a new experiment of type „Parameters Variation“



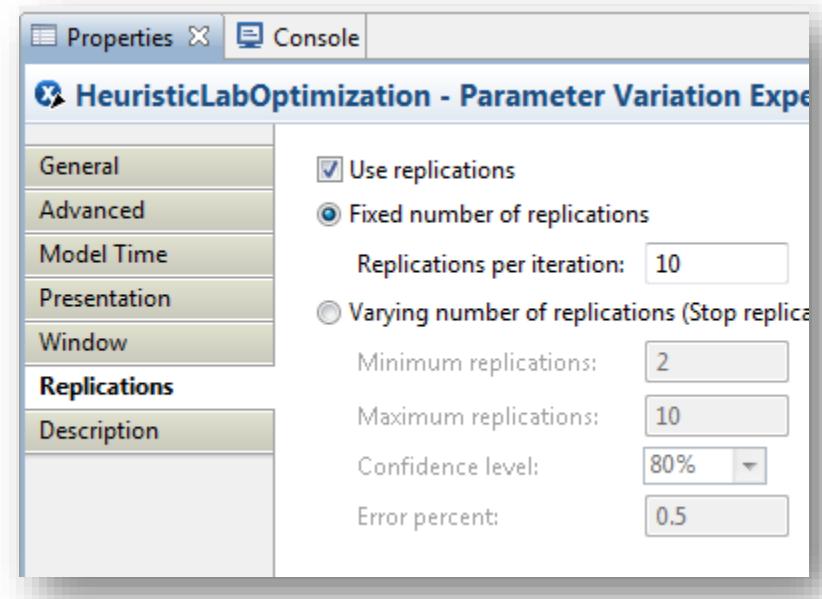
Supply Chain Simulation

- Configure the experiment
- 1. Create the default user interface by clicking the button
- 2. Add one variable for each parameter and add statistics for collecting quality relevant information over multiple replications
- 3. Select to vary parameters Freeform, use an arbitrary, but high number of runs and specify the variable that is assigned to each parameter
- 4. Optional: remember the best identified solution so far



Supply Chain Simulation

- Switch to the Replications tab
- Check the box „Use replications“
- Use a fixed number of 10 replications



Supply Chain Simulation



- How do parameter variation experiments work in AnyLogic?
- In an experiment run there are N iterations
- In each iteration R runs execute
- N = „Number of runs“ on the General tab
- R = „Replications per iteration“

Supply Chain Simulation



- Switch to the Advanced tab, here we need to enter some code

1. Initialize the communication
2. Receive parameters from HeuristicLab
3. Collect statistics after each run
4. Send back the averaged results and continue with step 2

The screenshot shows the HeuristicLab Optimization interface with the 'Advanced' tab selected. The main window displays a Java code editor for a 'Parameter Variation Experiment'. The code imports various Java and HeuristicLab classes and defines a class with methods for handling messages and setting up the experiment. The interface also includes tabs for 'Properties' and 'Console', and sections for application options like memory and command-line arguments.

```
import java.io.*;
import java.text.*;
import com.heuristiclab.problems.externalevaluation.*;
import com.heuristiclab.problems.externalevaluation.ExternalEvaluationMessages.*;

Additional class code:
int repCounter = 0;

com.heuristiclab.problems.externalevaluation.PollService commDriver;
SolutionMessage currentSolution = null;

private void getMessage() {
    currentSolution = commDriver.getSolution();
    if (currentSolution.getIntegerArrayVarsCount() > 0) {
        SolutionMessage.IntegerArrayVariable vector = currentSolution.getIntegerArrayVars(0);
        sRetailer = vector.getData(0);
        sRetailer = sRetailer + vector.getData(1);
        sWholesaler = vector.getData(2);
        sWholesaler = sWholesaler + vector.getData(3);
        sFactory = vector.getData(4);
        sFactory = sFactory + vector.getData(5);
    }
}

Initial experiment setup:
commDriver = new com.heuristiclab.problems.externalevaluation.PollService(new ServerSocketListenerFactory(2112),1);
commDriver.start();
Before each experiment run:
getMessage();
Before simulation run:
```

Supply Chain Simulation

- Imports section:

```
import java.io.*;  
import java.text.*;  
import com.heuristiclab.problems.externalevaluation.*;  
import com.heuristiclab.problems.externalevaluation.ExternalEvaluationMessages.*;
```

- Additional class code:

```
com.heuristiclab.problems.externalevaluation.PollService commDriver;  
SolutionMessage currentSolution = null;  
  
private void getMessage() {  
    currentSolution = commDriver.getSolution();  
    if (currentSolution.getIntegerArrayVarsCount() > 0) {  
        SolutionMessage.IntegerArrayVariable vector = currentSolution.getIntegerArrayVars(0);  
        sRetailer = vector.getData(0);  
        SRetailer = sRetailer + vector.getData(1);  
        sWholesaler = vector.getData(2);  
        SWholesaler = sWholesaler + vector.getData(3);  
        sFactory = vector.getData(4);  
        SFactory = sFactory + vector.getData(5);  
    }  
}
```

Supply Chain Simulation



- Initial experiment setup:

```
commDriver = new com.heuristiclab.problems.externalevaluation.PollService(new  
ServerSocketListenerFactory(2112),1);  
commDriver.start();
```

- Before each experiment run:

```
getMessage();
```

- After simulation run:

```
meanDailyCosts.add(root.meanDailyCost());  
maxWaitingTime.add(root.histWaitingTime.max());
```

Supply Chain Simulation

- After iteration:

```
double fitness = 0;
Boolean isFeasible = maxWaitingTime.max() < 0.001;

if (isFeasible) {
    fitness = meanDailyCosts.mean();
    if (meanDailyCosts.mean() < bestMeanDailyCost)
        bestMeanDailyCost = meanDailyCosts.mean();
} else {
    fitness = 2000 + maxWaitingTime.max();
}

try {
    commDriver.sendQuality(currentSolution, fitness);
} catch (IOException e) { /* handle error */ }

meanDailyCosts.reset();
maxWaitingTime.reset();

getMessage();
```

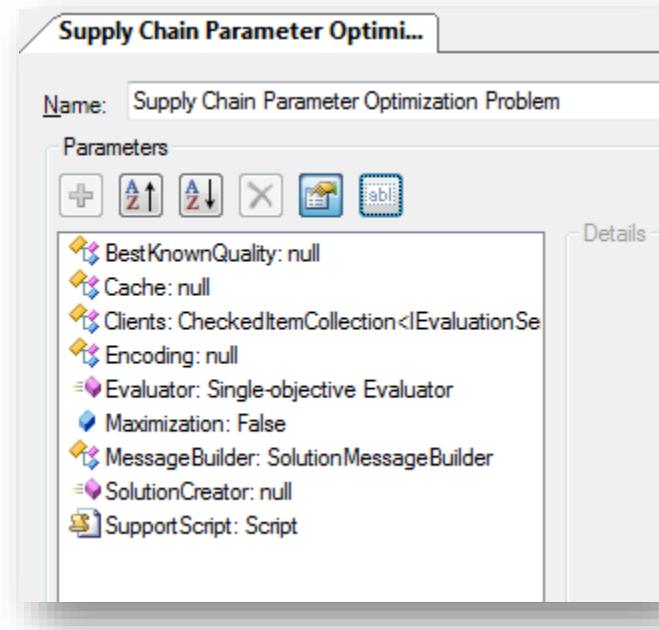
Supply Chain Simulation



- Create a new simulation experiment for evaluating parameters from HeuristicLab
- Create the problem definition in HeuristicLab
- Optimize

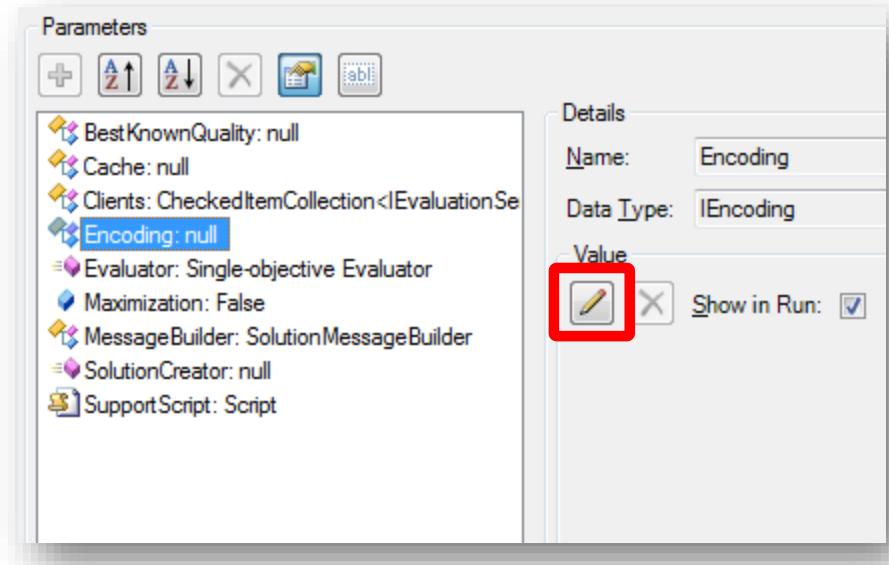
Supply Chain Simulation

- **Cache:** optional parameter that caches solutions and their corresponding quality
- **Clients:** the simulation model instances that can evaluate parameters
- **Encoding:** The encoding that describes a solution, e.g. a vector of integer values
- **Evaluator:** The operator that will extract the parameters and transmit them to the model
- **Maximization:** Whether the returned fitness is to be minimized or maximized
- **SolutionCreator:** The operator that will construct the initial solutions
- **SupportScript:** Additional code for analyzing solution candidates as well as ability to define a neighborhood function



Supply Chain Simulation

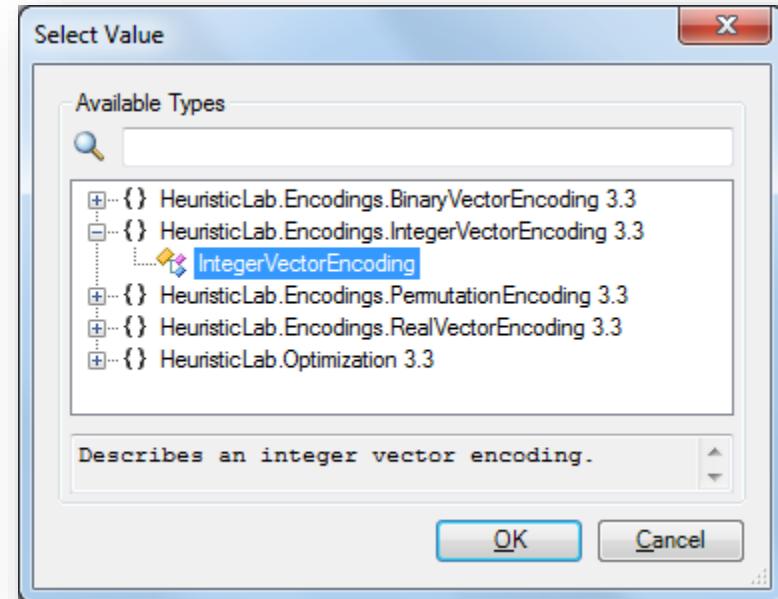
- Click on the Encoding parameter
- Click on the pencil icon to choose an encoding



Supply Chain Simulation

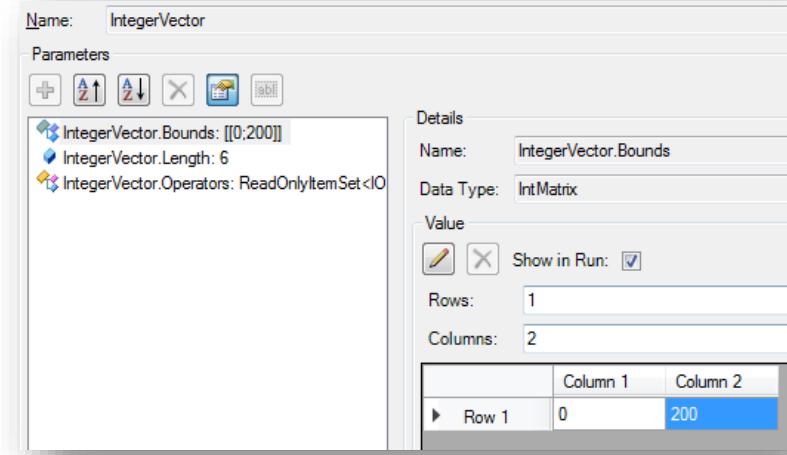
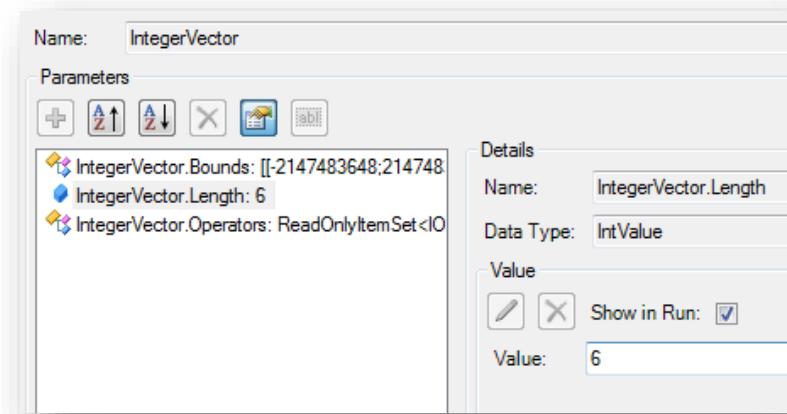


- Select the *IntegerVectorEncoding* from the list of available types
- Click OK



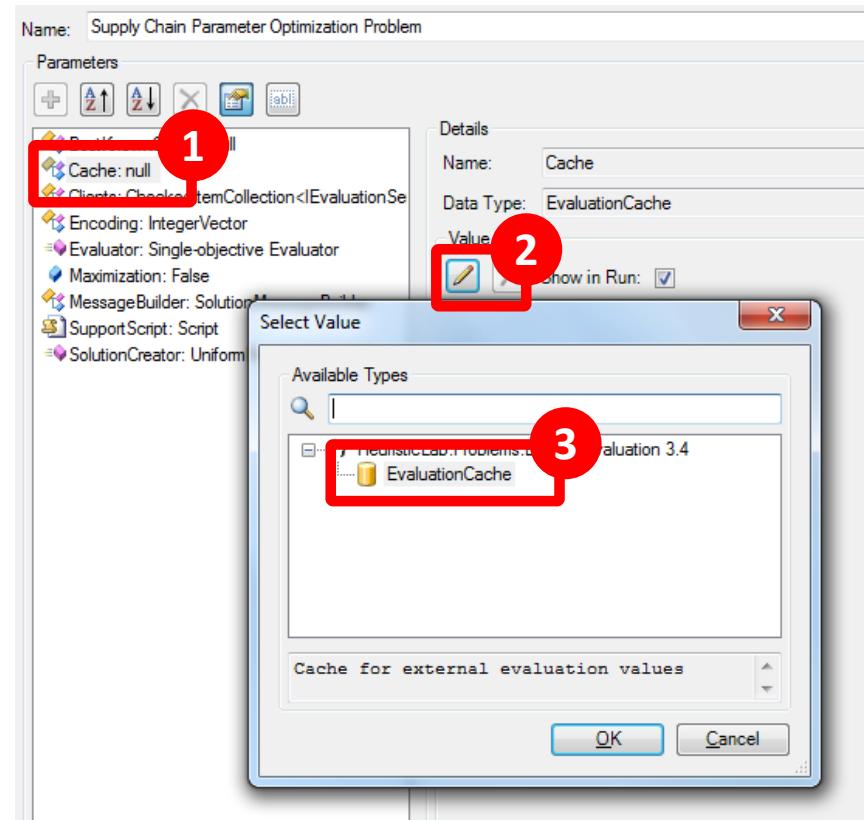
Supply Chain Simulation

- Adjust the parameters of the Encoding
 - Change the Length to 6
 - Change the Bounds to be 0 and 200 for all 6 dimensions
- The columns in the bounds specify lower and upper bound for each dimension respectively
 - If you have less rows in the bounds than dimensions in the vector, the bounds will be cycled



Supply Chain Simulation

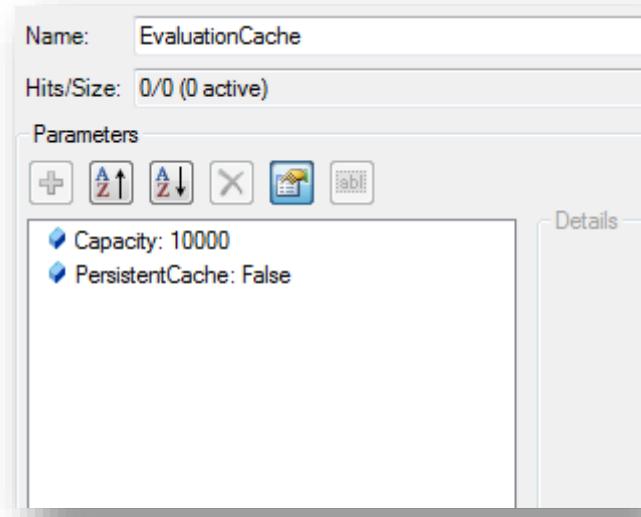
- Click on the Cache parameter
- Click on the pencil icon to assign a value
 - A dialog will pop up
- In the dialog select the EvaluationCache
- Click Ok



Supply Chain Simulation

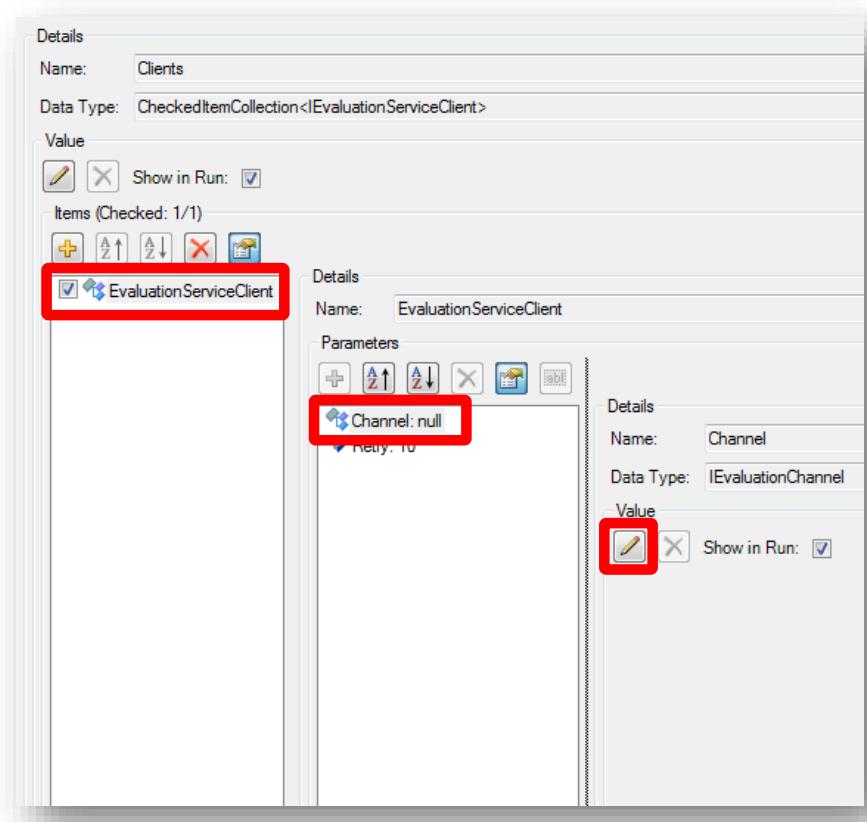
- The cache will store the solution message and the returned quality
- New solution messages will be compared against those already inside the cache

- The capacity of the cache can be adjusted
- PersistentCache determines if the cache should be stored when the problem or algorithm is saved



Supply Chain Simulation

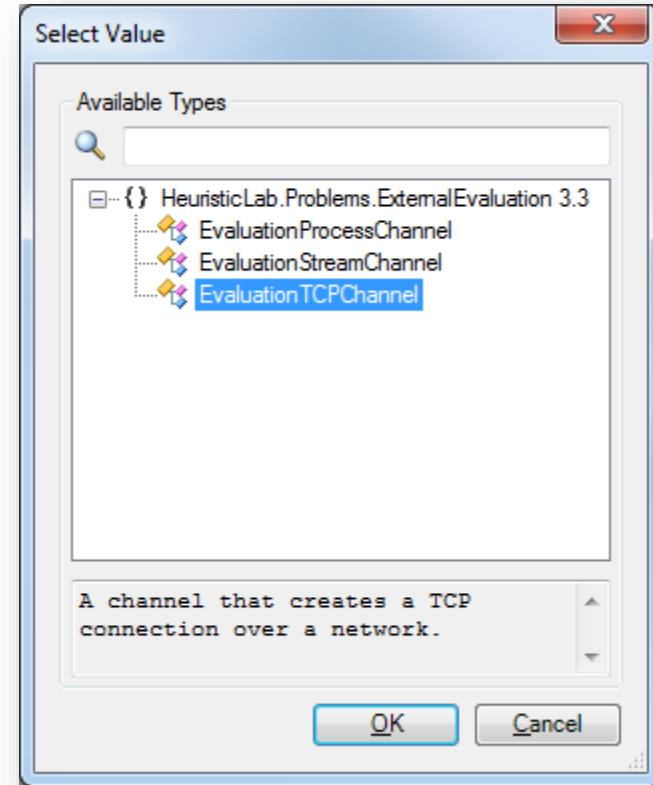
- Now set up the client machine that runs the simulation model
- Click on the Client parameter
- Click on EvaluationServiceClient
- Click on the Channel parameter
- Click the pencil icon to set a new channel



Supply Chain Simulation

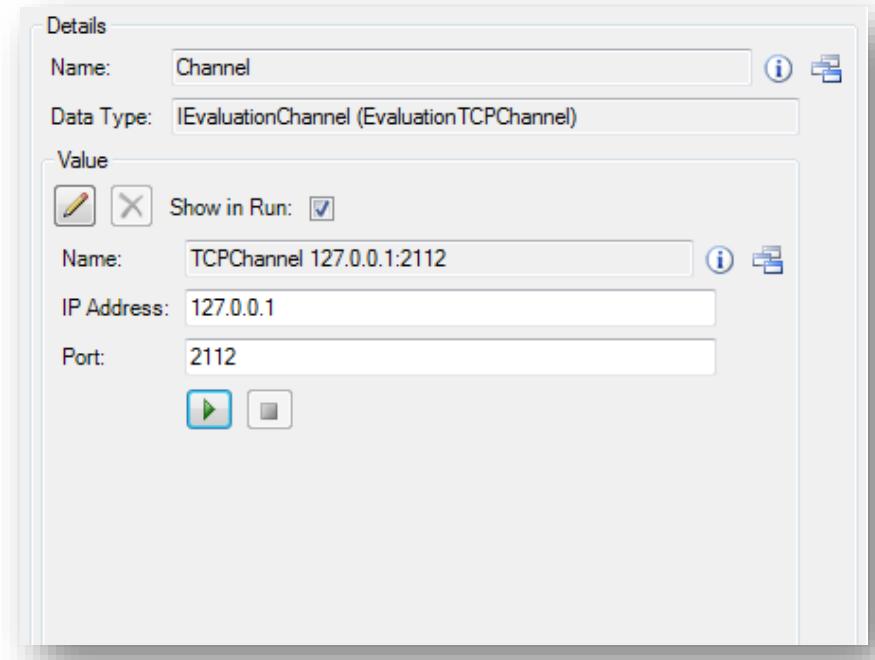


- Select the EvaluationTCPChannel for communicating over a TCP/IP connection
- Click OK



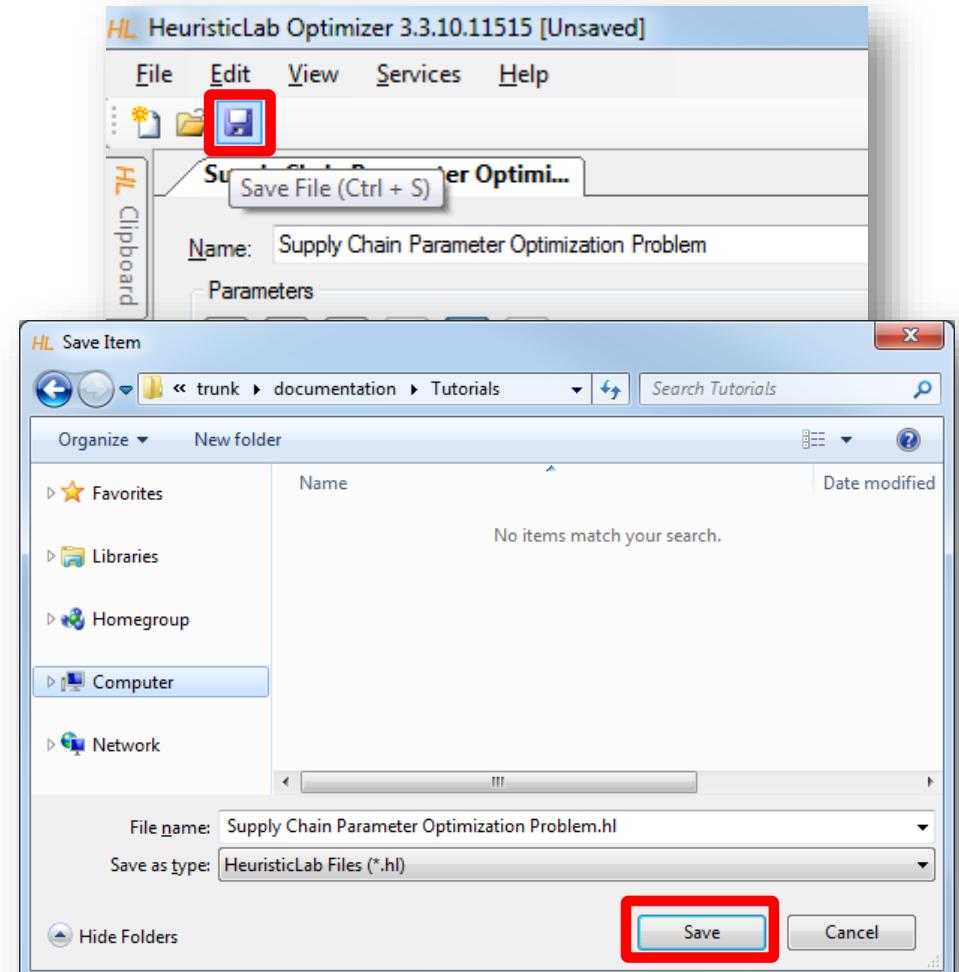
Supply Chain Simulation

- Enter the IP address of the machine that the model runs on, use 127.0.0.1 if the model runs on the same machine
- We configured our model to listen on port 2112 so we enter this information as Port



Supply Chain Simulation

- The problem is now configured
- Click the save button to store the external problem definition into a file



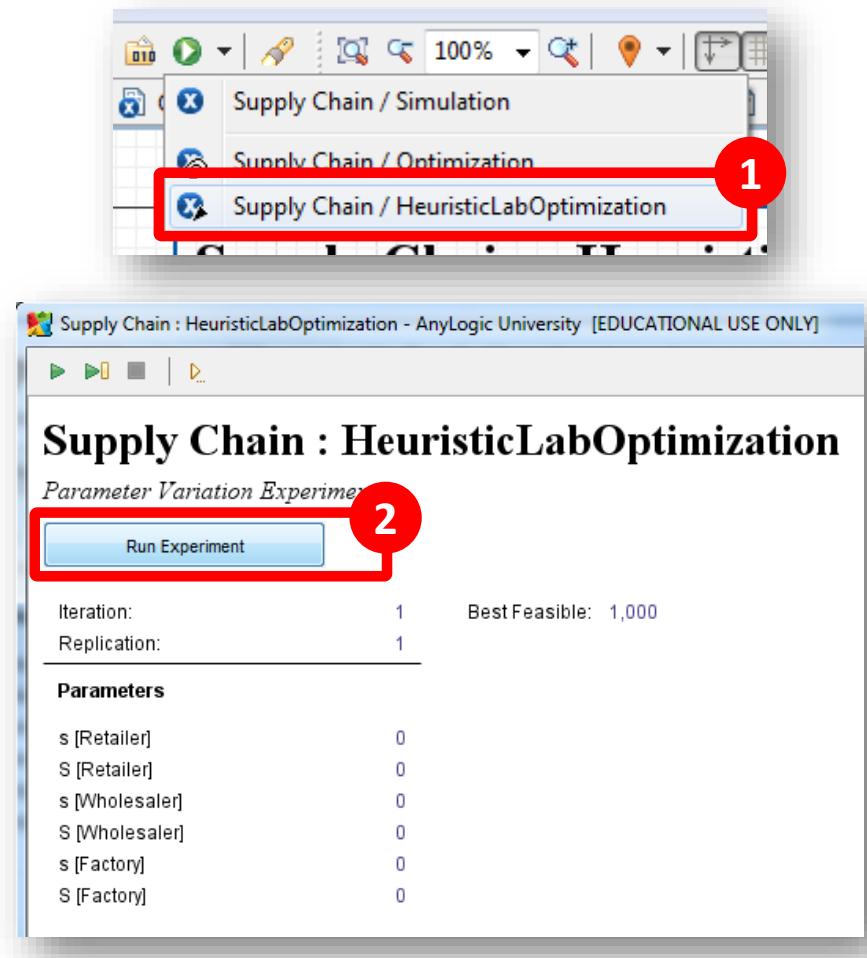
Supply Chain Simulation



- Create a new simulation experiment for evaluating parameters from HeuristicLab
- Create the problem definition in HeuristicLab
- Optimize

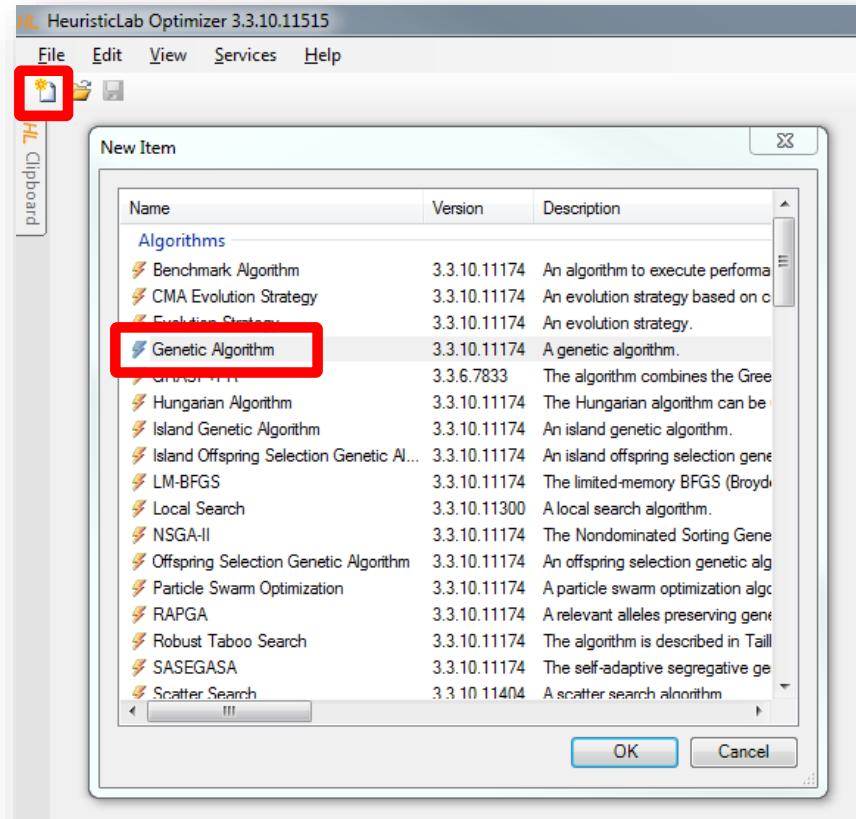
Supply Chain Simulation

- In AnyLogic click the arrow next to the green play button and select our newly created experiment
- Click to run the experiment at which point the model will wait to receive a solution message



Supply Chain Simulation

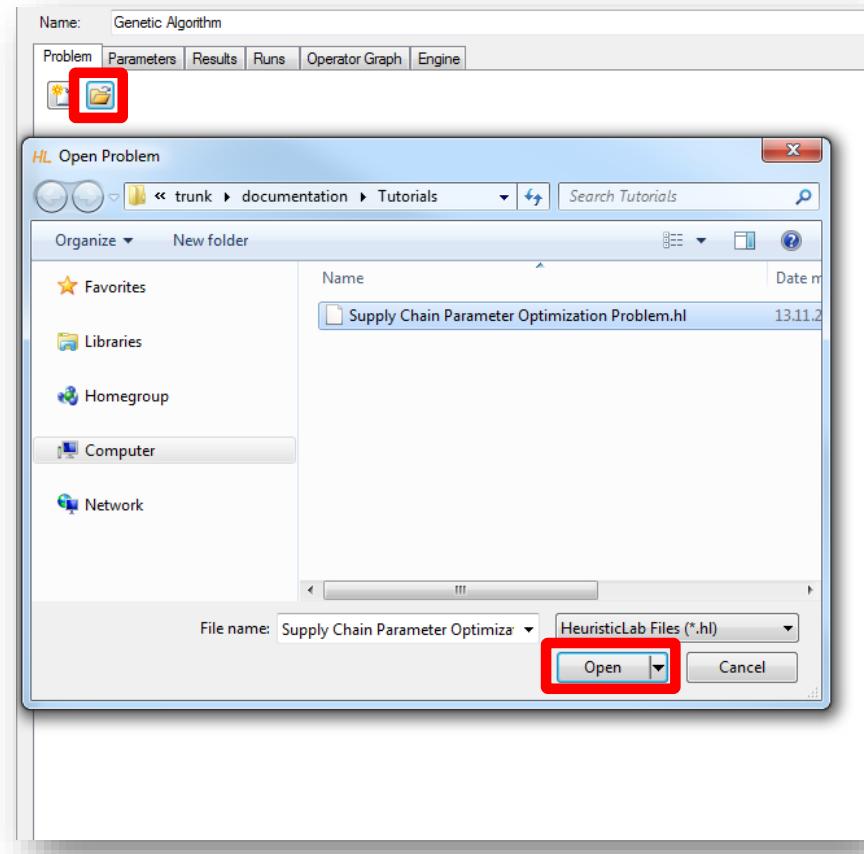
- In HeuristicLab click on the „New Item“ button to open the list of creatable items
- Select the Genetic Algorithm entry
- Click OK



Supply Chain Simulation



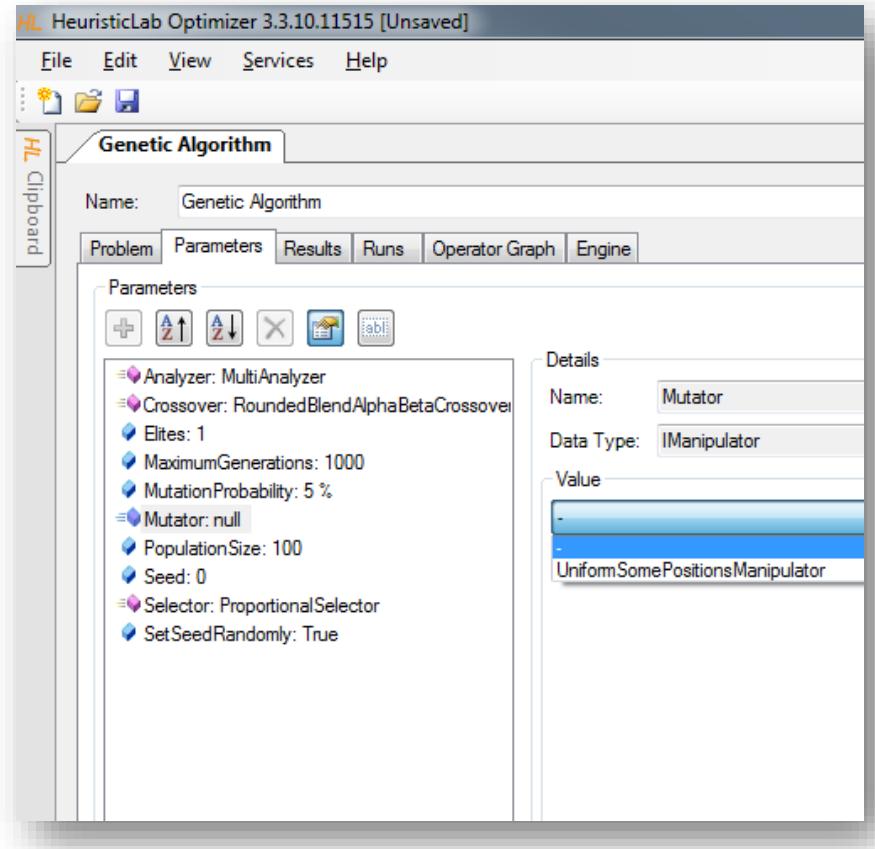
- Click on the Open button and select the previously saved problem definition
- Click Open in the dialog



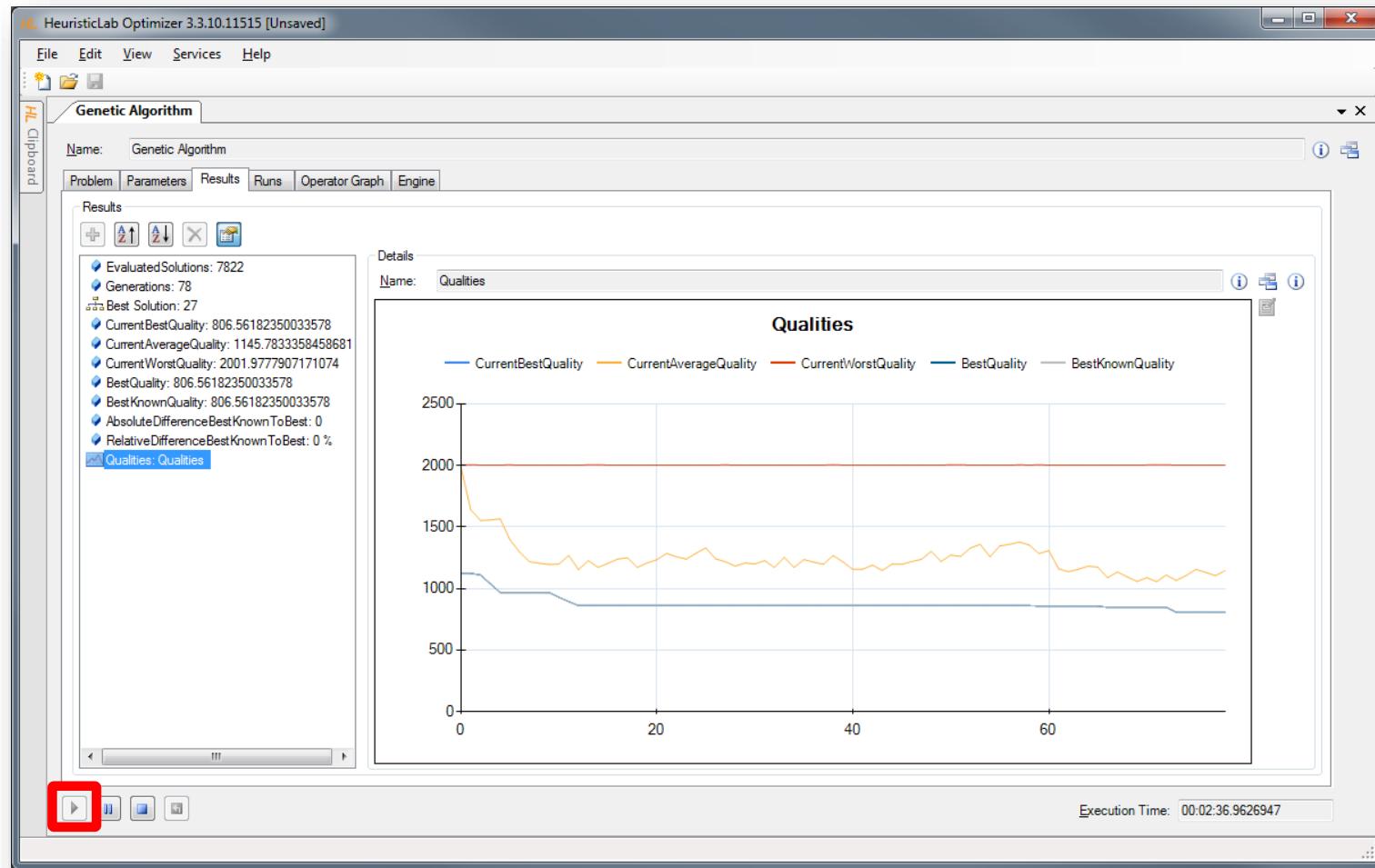
Supply Chain Simulation



- Click on the Mutator parameter
- Select the manipulator that we have added to the operators list in one of the previous steps
- Switch to the Results tab and hit the play button at the bottom



Supply Chain Simulation



Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- **Demonstration Part I: External Evaluation Problem**
- **Demonstration Part II: MATLAB and Scilab Parameter Optimization Problem**
- **Demonstration Part III: Programmable Problem**
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

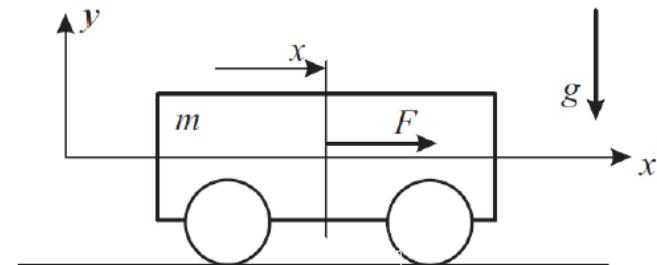
Demonstration Part II: Parameter Optimization Problem



- Parameter Optimization of Differential Equation Systems (Scilab)

Electric cart simulation

- Identify unknown parameter values of a simulation model
- Measure movements of an electric cart with known power
- Adapt parameters of an simulation model to match those measurements



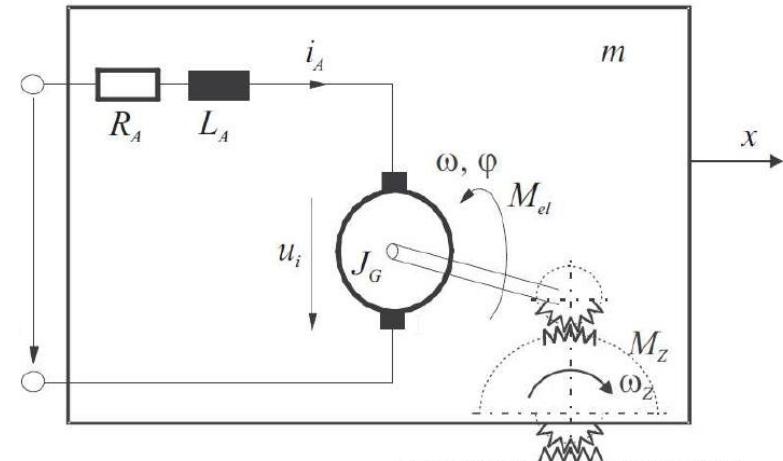
Electric cart simulation

- Identify friction coefficients d_1 , F_C and mass m
- Voltage u_A and initial values for position x , velocity v and amperage i_A are known
- Simulate changes according to differential equations

$$\dot{x} = x$$

$$\dot{v} = -\frac{d_1}{m} \cdot v - \frac{1}{m} F_C \cdot \text{sign}(v) + \frac{k_m \cdot n}{r \cdot m} \cdot i_A$$

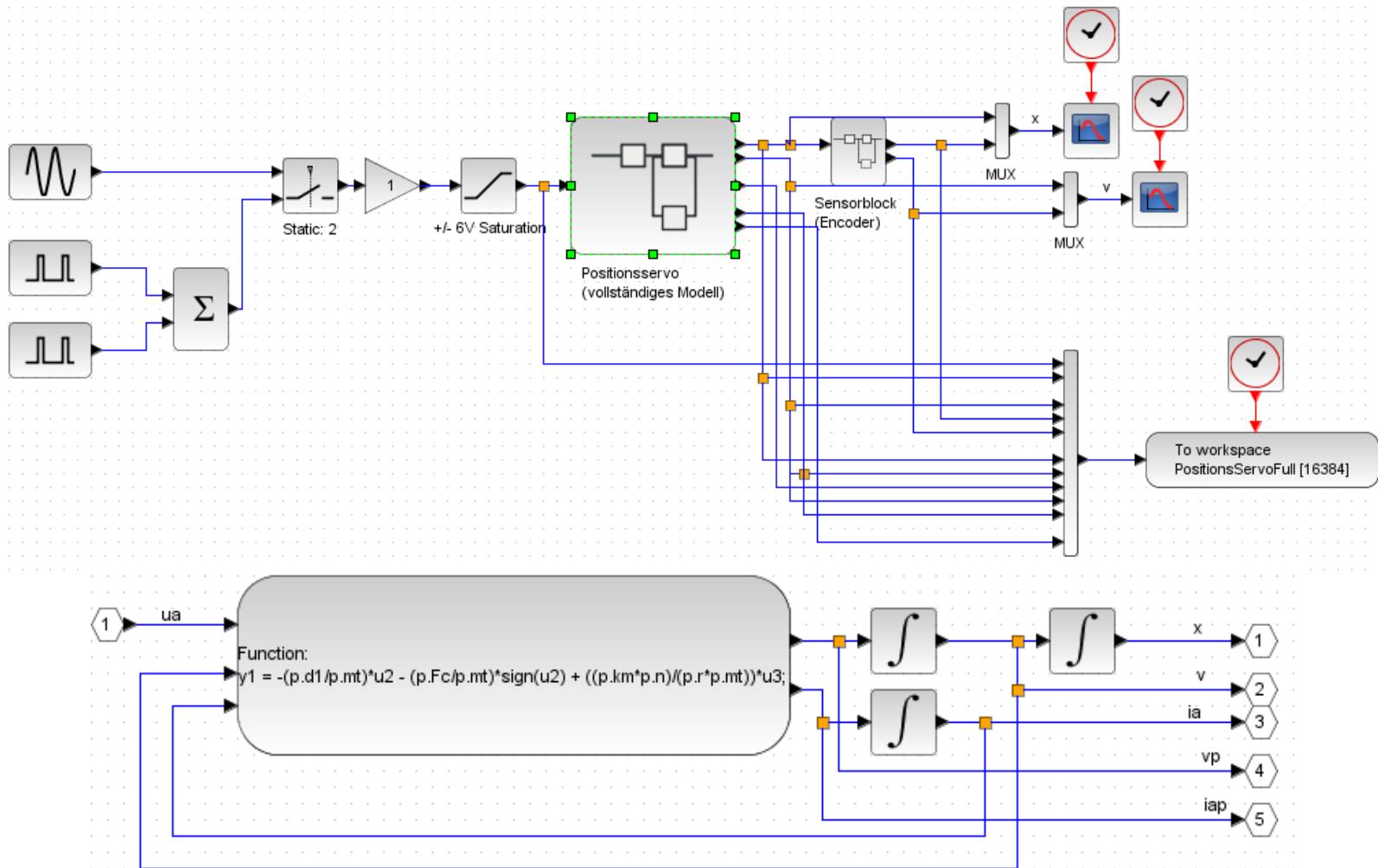
$$\dot{i}_A = -\frac{k_m \cdot n}{L_A \cdot r} \cdot v - \frac{R_A}{L_A} \cdot i_A + \frac{u_A}{L_A}$$



Measurements

Time	Input ua	x	v
1.808	0	1.38E-06	-4.60E-05
1.809	0	1.38E-06	-2.63E-05
1.81	0	1.38E-06	-5.23E-05
1.811	0	1.37E-06	2.10E-05
1.812	0	1.38E-06	-4.22E-05
1.813	0	1.37E-06	-2.54E-05
1.814	0	1.35E-06	4.60E-05
1.815	0	1.36E-06	2.63E-05
1.816	0	1.36E-06	5.23E-05
1.817	0	1.37E-06	-2.10E-05
1.818	0	1.36E-06	4.22E-05
1.819	0	1.37E-06	2.54E-05
1.82	0	1.38E-06	-4.60E-05

Simulation in Scilab

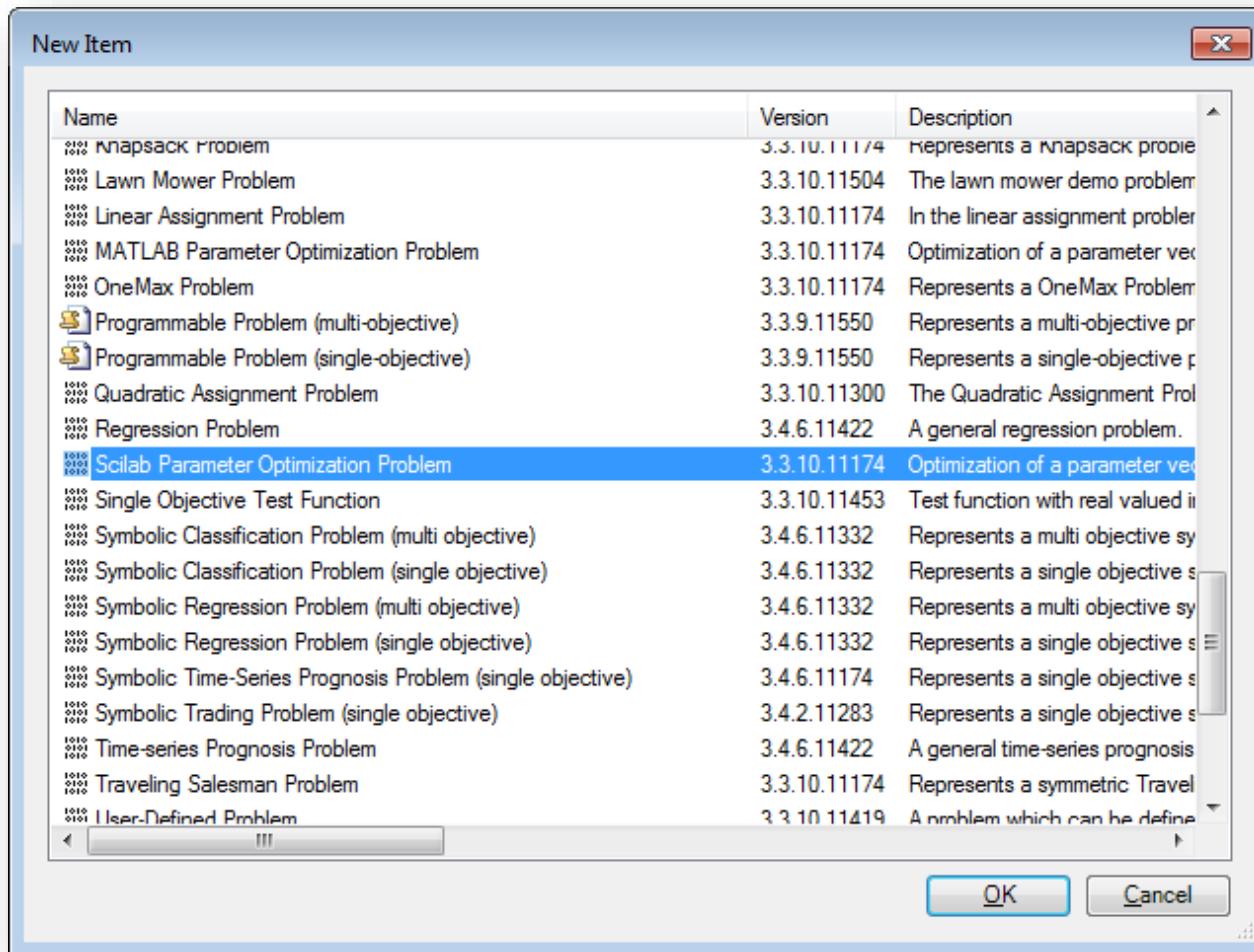


Parameter Optimization



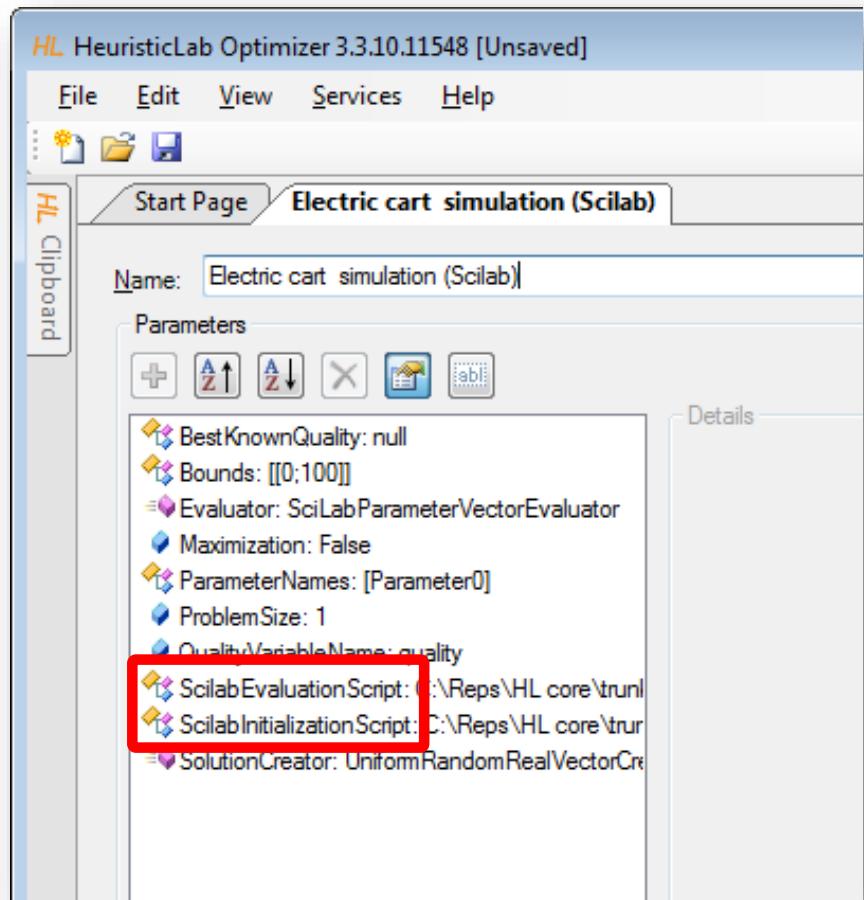
- Create a new Scilab parameter optimization problem in HeuristicLab
- Configure the problem with your scripts in HeuristicLab
- Optimize

Parameter Optimization



Parameter Optimization

- Specify path to Scilab scripts
- Initialization script
- Evaluation script



Initialization Script

- Sets constants and time intervals
- Loads the simulation model
- Reads the measured values from a csv file

```
cd "C:\Reps\HL core\trunk\documentation\Tutorials\Scilab Files"

T = 0.01;
Ts = 5;

// parameters of the dc-motor and the gear
p.Ra = 1.63;                      // terminal resistance
p.La = 270e-6;                     // terminal inductance
p.km = 37.7e-3;                    // torque constant
p.n  = 1;                          // gear transmission ratio
p.r  = 6e-3;                       // radian of the pinion
p.Ja = 41e-7;                      // moment of inertia (power train)

[result]=importXcosDiagram('SimulatePositionsServoReduced_NoScope.zcos')
original = servo.values(:,2); //2...x, 3...v, 4...x_encoder, 5...v_encoder
data = csvRead('PositionsServoReduced.csv','','','double',[],[],[],1);
original = data(:,3);
```

Evaluation Script

- Configures and runs the simulation with values from HeuristicLab
- Calculates quality as sum of absolute errors between simulated and measured values

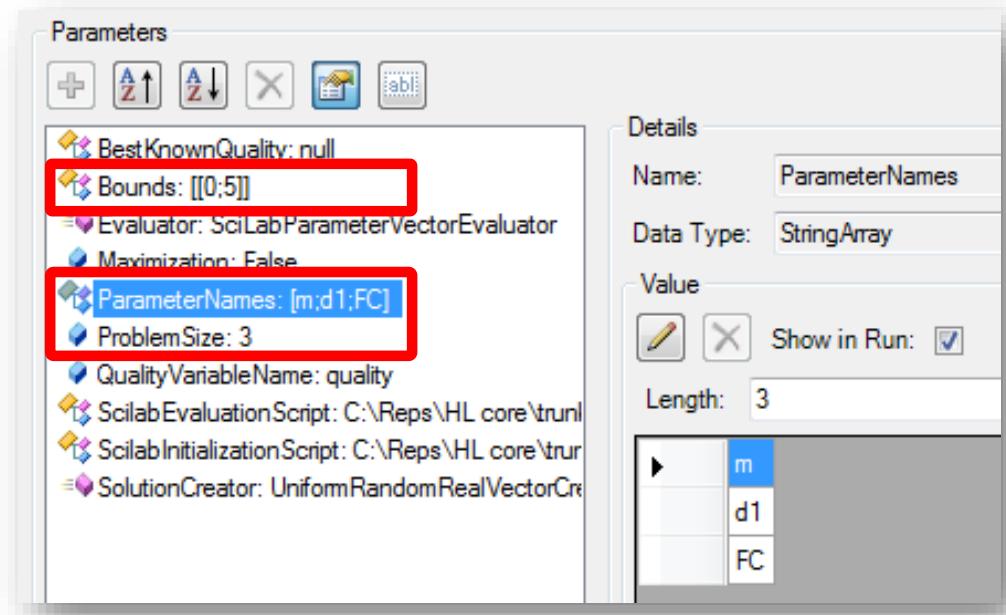
```
// parameters of the cart
p.m = m;                                // mass of the cart
p.d1 = d1;                                // viscous friction coefficient
p.Fc = FC;                                // Coulomb friction force

// equivalent parameters for the model
p.mr = p.Ja*(p.n/p.r)^2;                  // equivalent mass to moment of inertia
p.mt = p.m + p.mr;                         // equivalent mass (total)
p.del = (p.n/p.r*p.km)^2/p.Ra;             // linear damping from motor
p.dt = p.d1 + p.del;                        // linear damping coefficient
p.beta = p.n*p.km/p.r/p.Ra;                // input transformation

xcos_simulate(SCS_M, 4);
values = servo.values(:,2); // 2...x, 3...v, 4...x_encoder, 5...v_encoder
quality = sum(abs(values - original));
disp(quality);
```

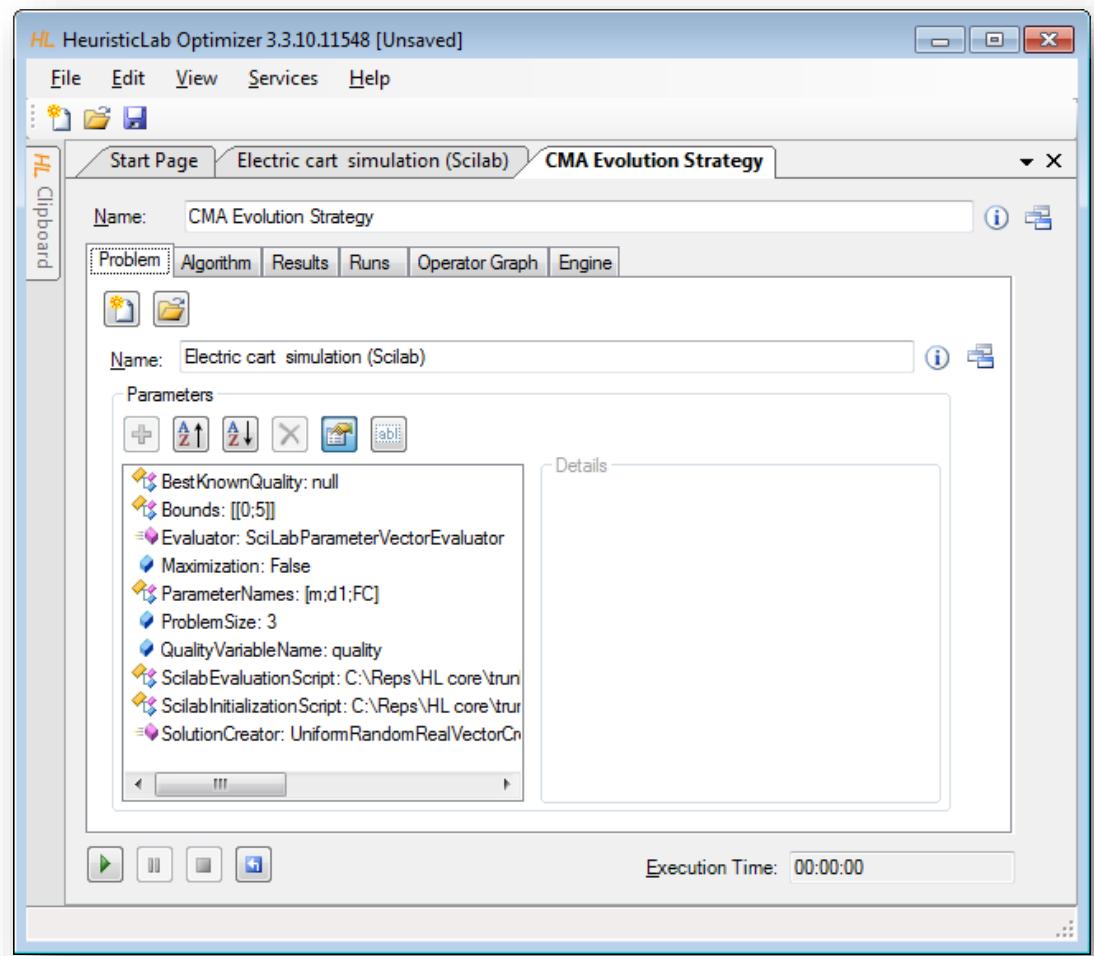
Parameter Optimization

- Configure problem size
- Configure parameter names
- Parameter names are created as variables in Scilab
- Adapt bounds



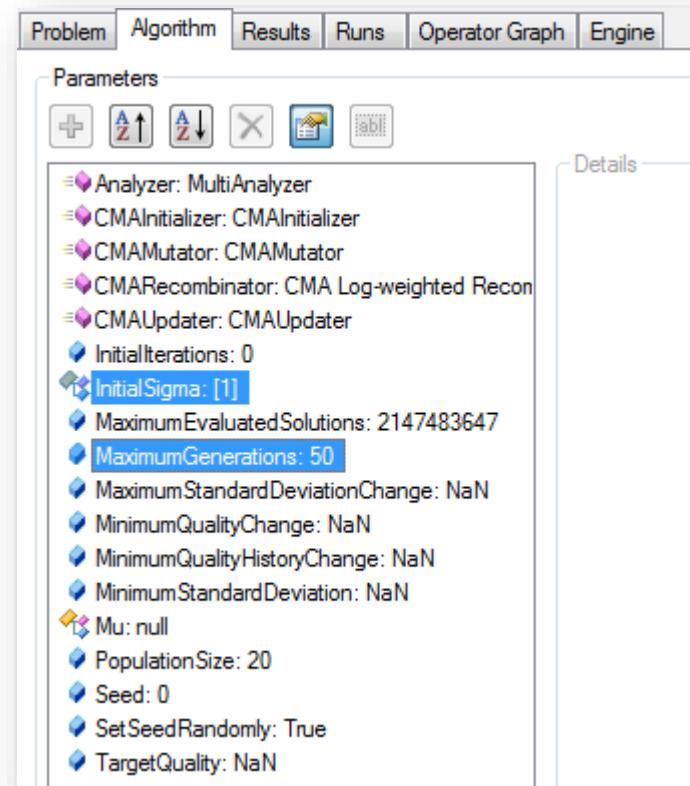
Parameter Optimization

- Create CMA-ES
- Drop problem on algorithm



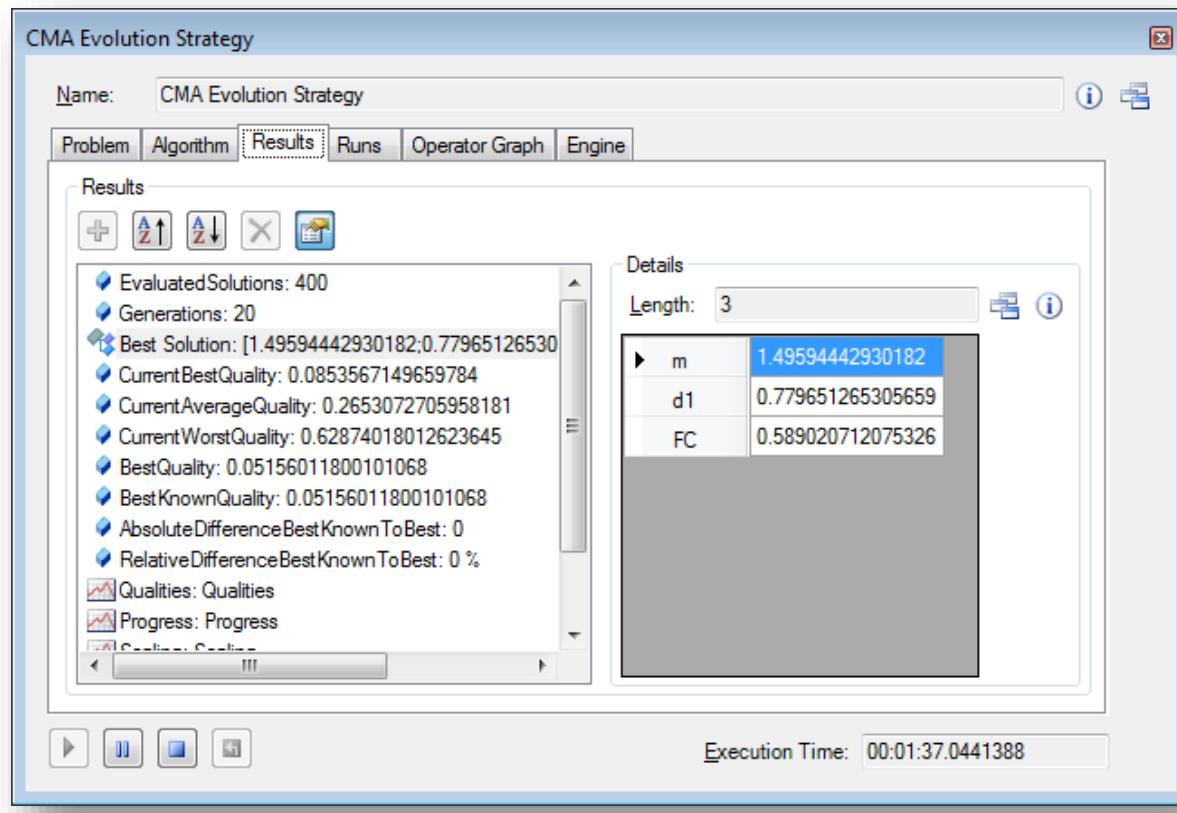
Parameter Optimization

- Configure algorithm
 - Initial Sigma 1.0
 - Maximum Generations 50
- Enable CMAAnalyzer

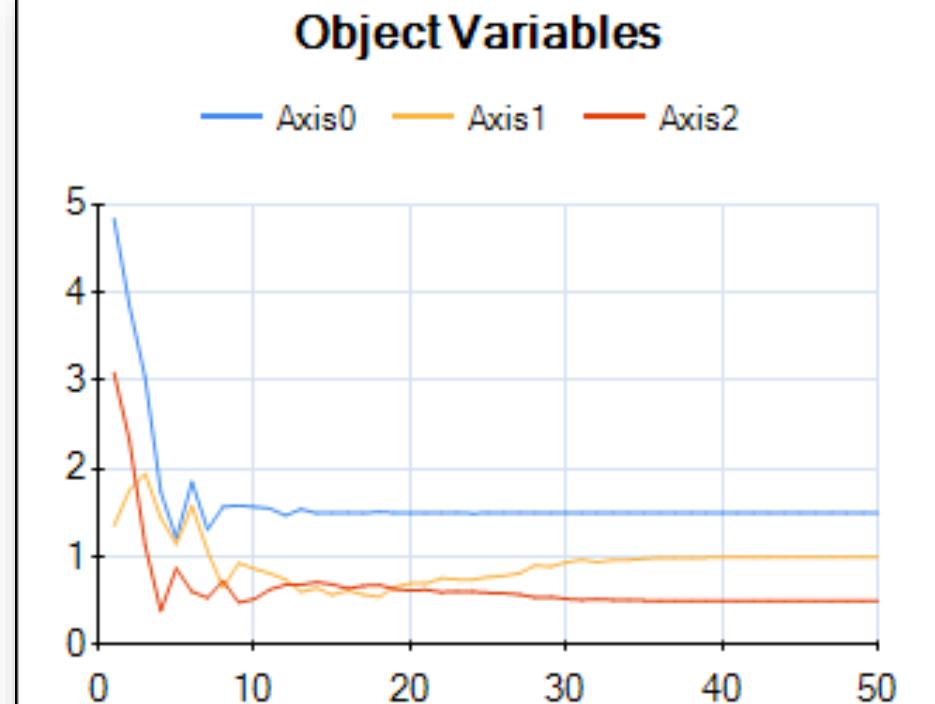
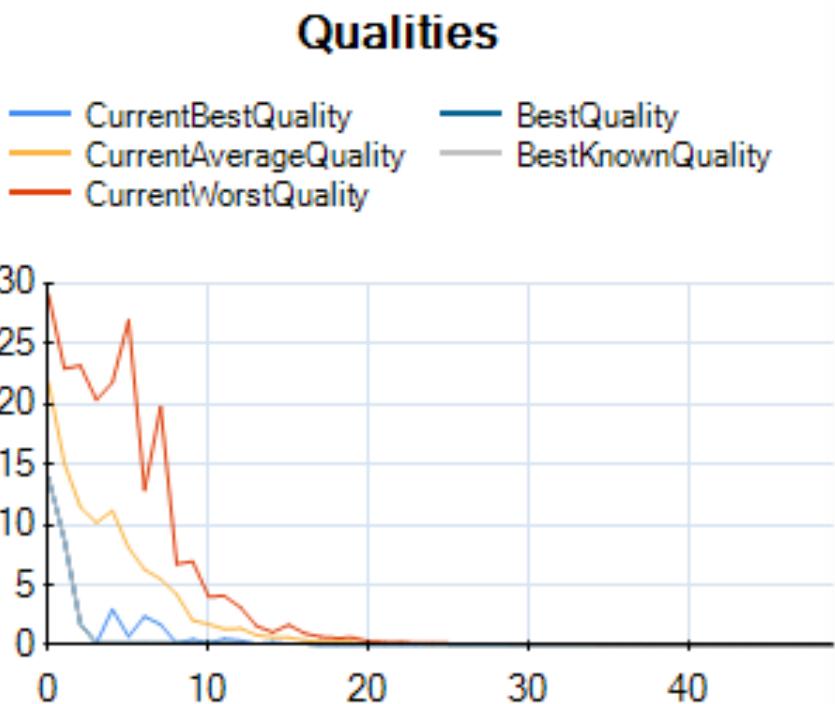


Parameter Optimization

- Optimal values ($m = 1.5$, $d1 = 1$, $FC = 0.5$)



Parameter Optimization



Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- **Demonstration Part I: External Evaluation Problem**
- **Demonstration Part II: MATLAB and Scilab Parameter Optimization Problem**
- **Demonstration Part III: Programmable Problem**
- Some Additional Features
- Planned Features
- Team
- Suggested Readings
- Bibliography
- Questions & Answers

Demonstration Part III: Programmable Problem

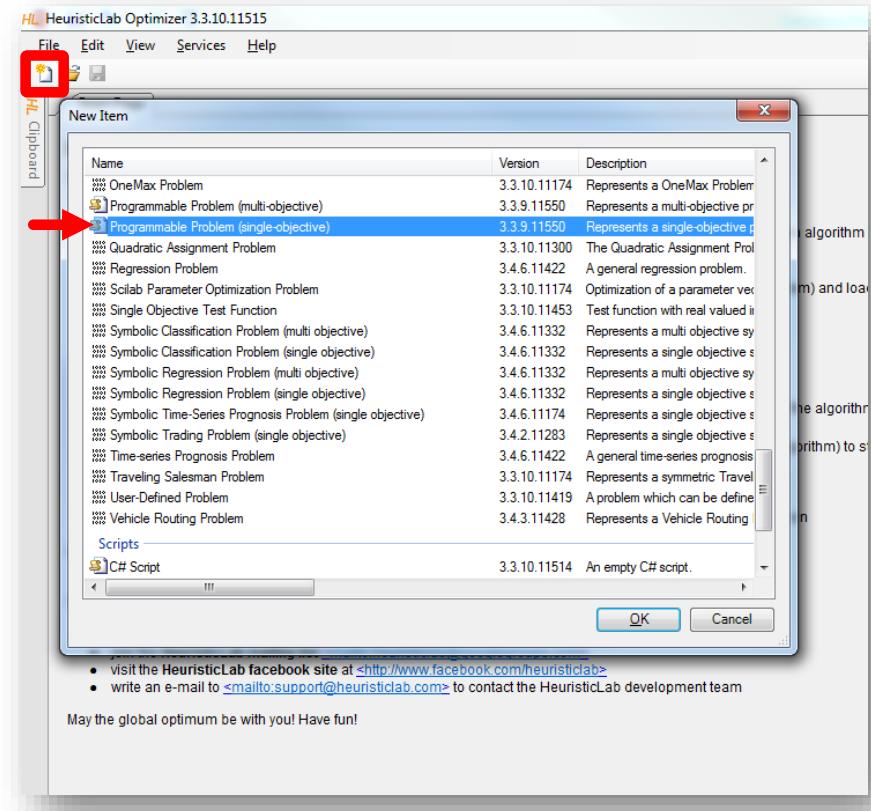


- Singleobjective Function Optimization
 - Solving the Styblinski-Tang function
- Multiobjective Function Optimization
 - Solving the Fonseca and Fleming function
- Non-linear Curve Fitting
 - Fitting a 2 dimensional Gaussian distribution to noisy data

Singleobjective Function Opt.



- Click on „New Item“ to get a list of creatables
- Create a new Programmable Problem (single-objective) by double clicking it



Singleobjective Function Opt.



The screenshot shows the HeuristicLab Optimizer interface. The main window title is "HeuristicLab Optimizer 3.3.10.11515 [Unsaved]". The menu bar includes File, Edit, View, Services, and Help. A toolbar on the left contains icons for Start Page, Recent, and Problem. The central workspace is titled "Programmable Problem (single...)" and shows a C# code editor with the following content:

```
using HeuristicLab.Optimization;
using HeuristicLab.Problems.Programmable;

public class CustomProblemDefinition : ProblemScriptBase, ISingleObjectiveProblemDefinition {
    public bool IsMaximizationProblem { get { return false; } }

    public CustomProblemDefinition() {
        // Define the solution encoding which can also consist of multiple vectors, examples below
        // Encoding = new BinaryEncoding("B", length: 5);
        // Encoding = new IntegerEncoding("i", length: 5, min: 2, max: 14, step: 4);
        // Encoding = new RealEncoding("r", length: 5, min: -1.0, max: 1.0);
        // Encoding = new PermutationEncoding("P", length: 5, type: PermutationTypes.Absolute);
        // Encoding = new MultiEncoding()
        //     .AddBinaryVector("B", length: 5)
        //     .AddIntegerVector("i", length: 5, min: 2, max: 14, step: 4)
        //     .AddRealVector("r", length: 5, min: -1.0, max: 1.0)
        //     .AddPermutation("P", length: 5, type: PermutationTypes.Absolute);
    }

    public void CreateVariables() {
        // when the definition is created here you can initialize variables in the variable store
    }

    public double Evaluate(IRandom random, Individual individual) {
        var quality = 0.0;
        // use vars.yourVariable to access variables in the variable store i.e. yourVariable
        // quality = individual.RealVector("r").Sum(x => x * x);
        return quality;
    }

    public void Analyze(Individual[] individuals, double[] qualities, ResultCollection results) {
        // write or update results given the range of vectors and resulting qualities
        // use e.g. vars.yourVariable to access variables in the variable store i.e. yourVariable
    }

    public override IEnumerable<Individual> GetNeighbors(IRandom random, Individual individual) {
        // Create new vectors, based on the given one that represent small changes
    }
}
```

A red rounded rectangle highlights the code editor area. Overlaid on this area is the text "C# Editor for Writing the Problem Definition" in large red font. To the right of the editor is a "Variables" panel with a table:

Name	Value	Type

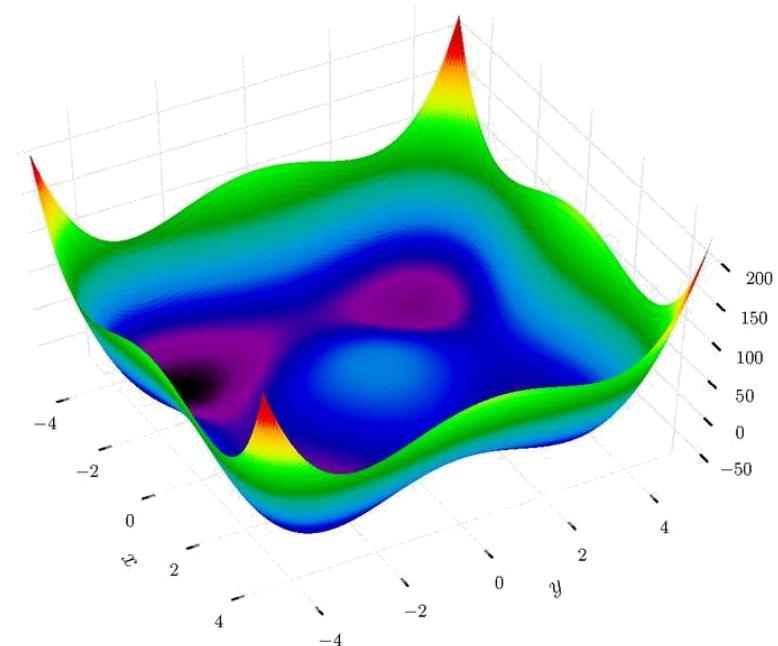
The bottom status bar shows "Ready" and "Powered by #develop".

Singleobjective Function Opt.

- Styblinski-Tang function:

$$f(x) = \frac{\sum_{i=1}^n x_i^4 - 16x_i^2 + 5x_i}{2}$$

with $-5 \leq x_i \leq 5$



Singleobjective Function Opt.



- Choose an appropriate solution encoding
 - 20-dimensional real-valued vector
 - Minimize the fitness value

```
public bool Maximization { get { return false; } }

public override void Initialize() {
    // Define the solution encoding which can also consist of multiple vectors, examples below
    // Encoding = new BinaryEncoding("b", length: 5);
    // Encoding = new IntegerEncoding("i", length: 5, min: 2, max: 14, step: 4);
    Encoding = new RealEncoding("vector", length: 20, min: -5.0, max: 5.0);
    // Encoding = new PermutationEncoding("P", length: 5, type: PermutationTypes.Absolute);
    // Encoding = new MultiEncoding()
    //     .AddBinaryVector("b", length: 5)
    //     .AddIntegerVector("i", length: 5, min: 2, max: 14, step: 4)
    //     .AddRealVector("r", length: 5, min: -1.0, max: 1.0)
    //     .AddPermutation("P", length: 5, type: PermutationTypes.Absolute)
    ;
}
```

Singleobjective Function Opt.

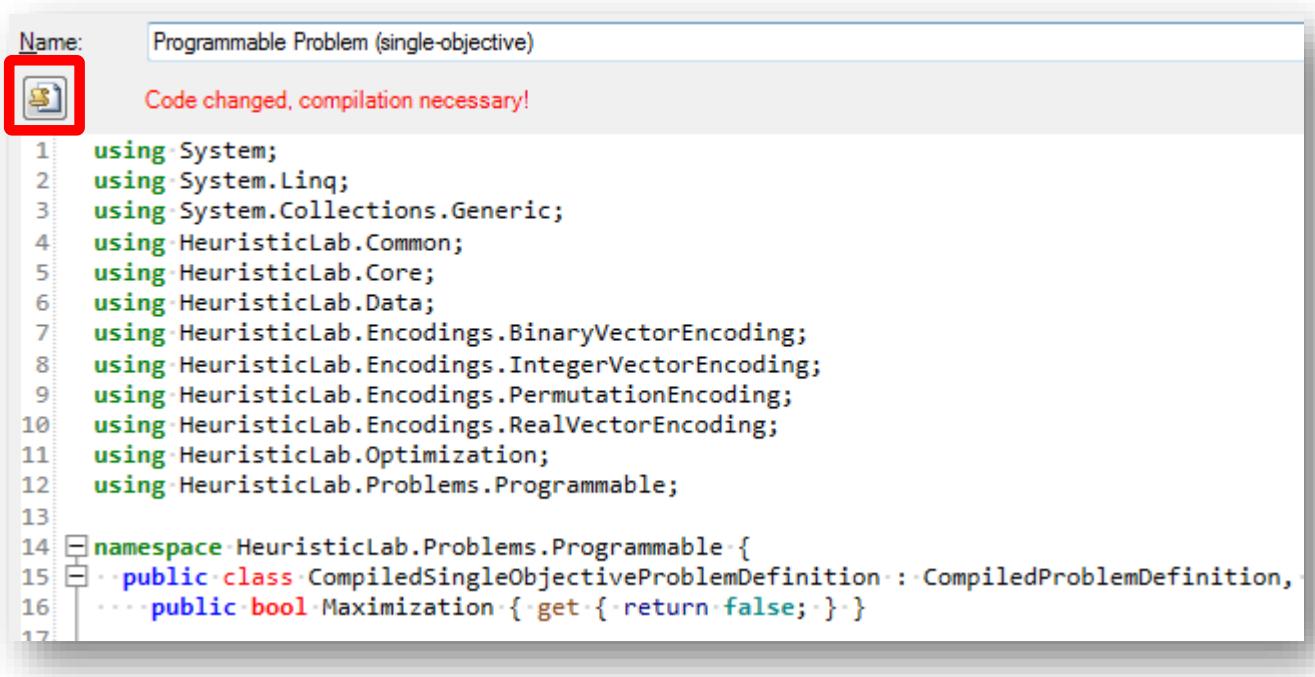


- Define the fitness function

```
public double Evaluate(Individual individual, IRandom random) {  
    var vector = individual.RealVector("vector");  
    return vector.Sum(x => x * x * x * x - 16.0 * x * x + 5 * x) / 2.0;  
}
```

Singleobjective Function Opt.

- Compile the problem defintion



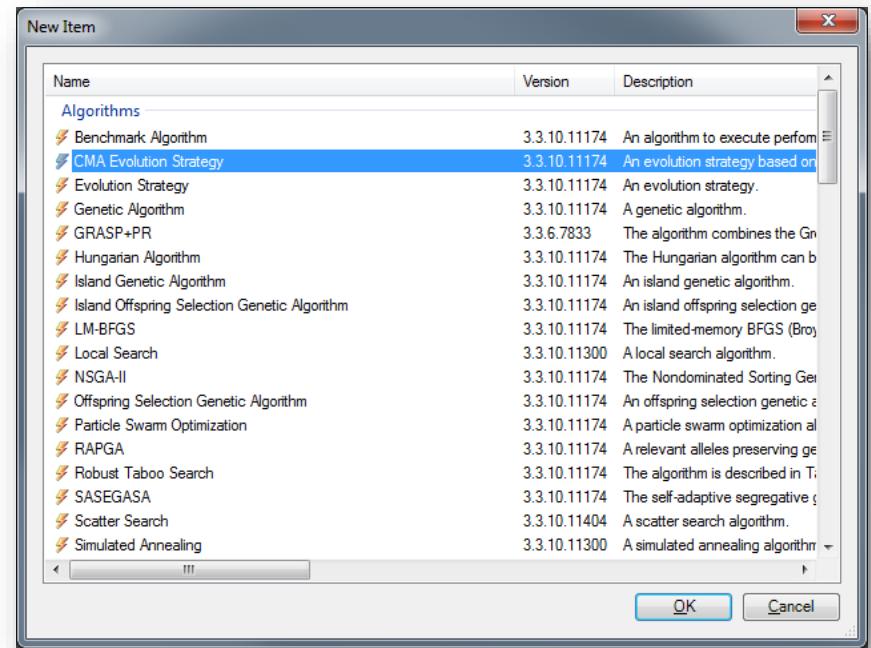
Name: Programmable Problem (single-objective)

Code changed, compilation necessary!

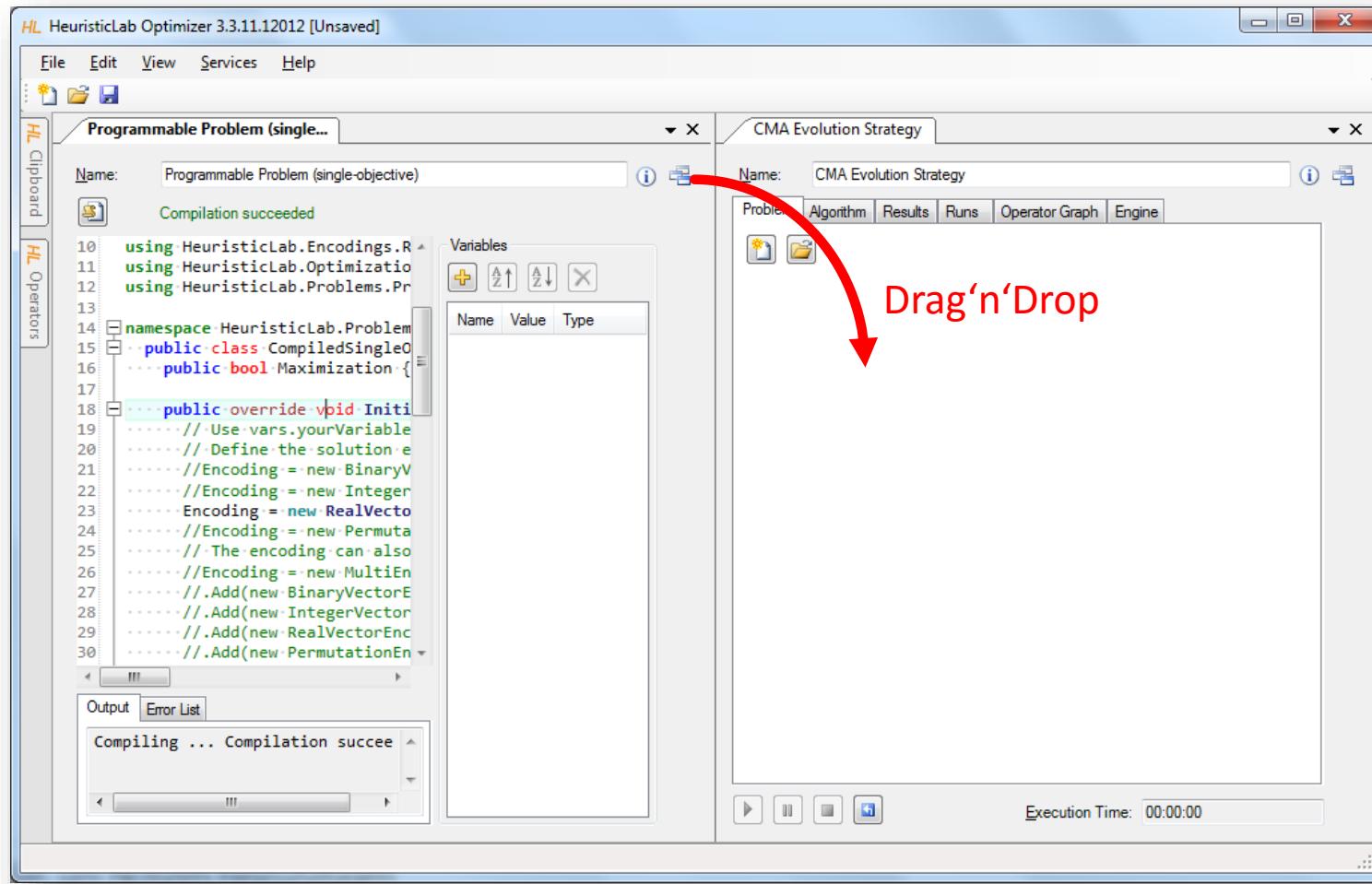
```
1  using System;
2  using System.Linq;
3  using System.Collections.Generic;
4  using HeuristicLab.Common;
5  using HeuristicLab.Core;
6  using HeuristicLab.Data;
7  using HeuristicLab.Encodings.BinaryVectorEncoding;
8  using HeuristicLab.Encodings.IntegerVectorEncoding;
9  using HeuristicLab.Encodings.PermutationEncoding;
10 using HeuristicLab.Encodings.RealVectorEncoding;
11 using HeuristicLab.Optimization;
12 using HeuristicLab.Problems.Programmable;
13
14 [namespace HeuristicLab.Problems.Programmable {
15   [public class CompiledSingleObjectiveProblemDefinition : CompiledProblemDefinition,
16    [public bool Maximization { get { return false; } }
```

Singleobjective Function Opt.

- Create a suitable optimization algorithm
- Select CMA Evolution Strategy
- Click OK

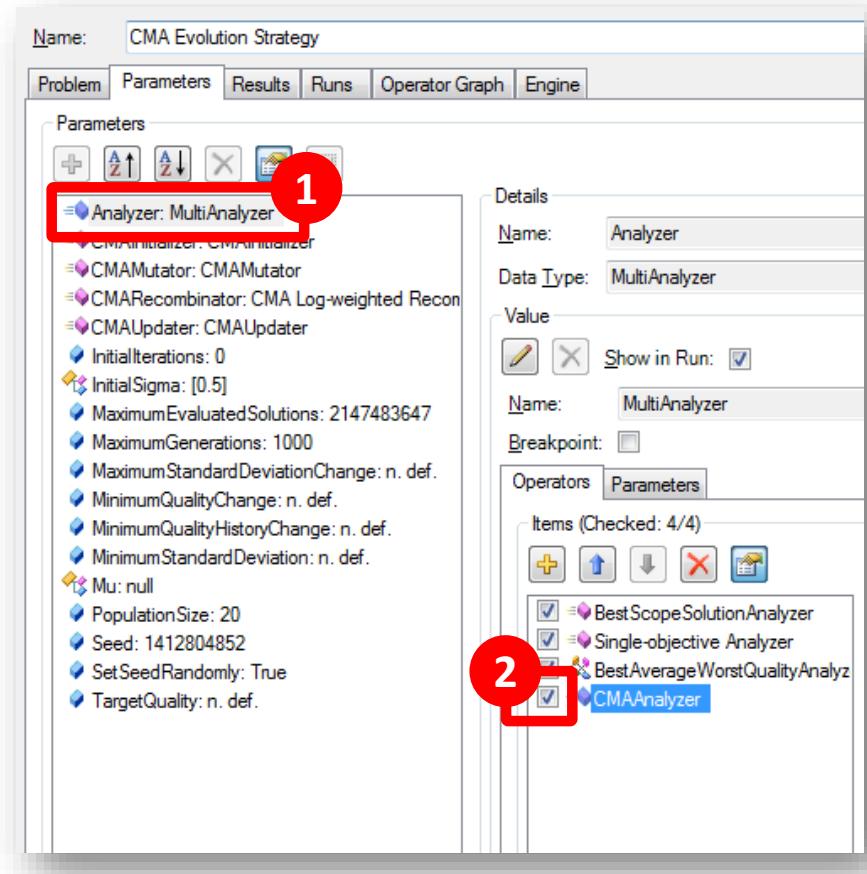


Singleobjective Function Opt.



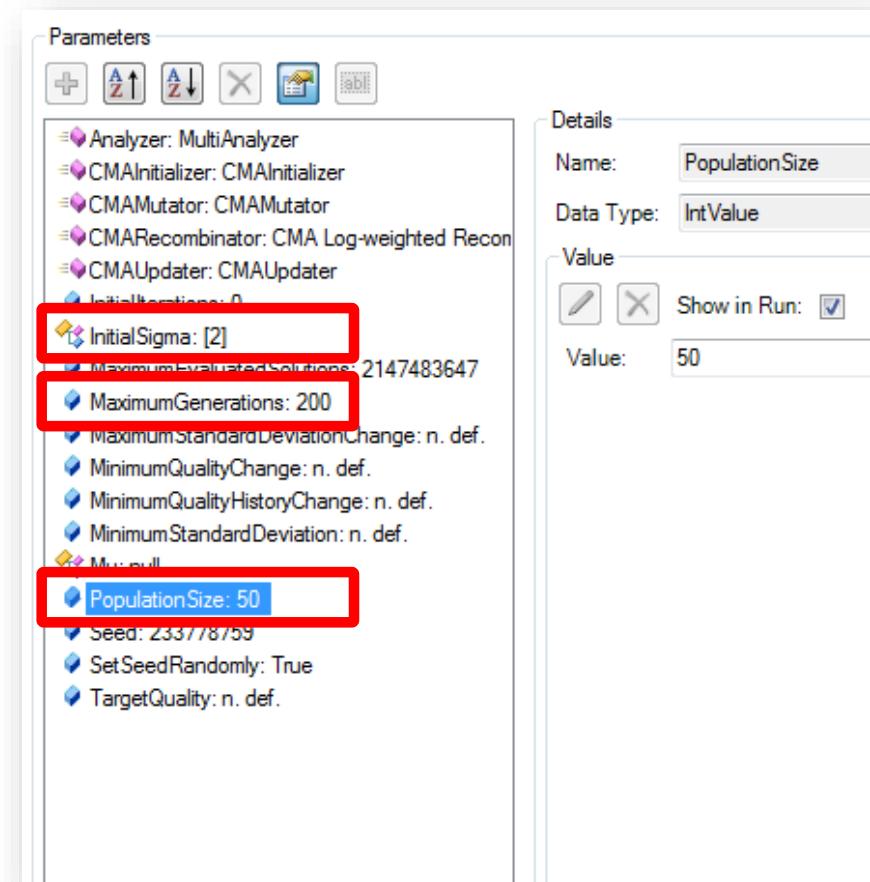
Singleobjective Function Opt.

- Switch to the Parameters tab
- Click on the Analyzer parameter
- Check CMAAnalyzer to get more detailed results

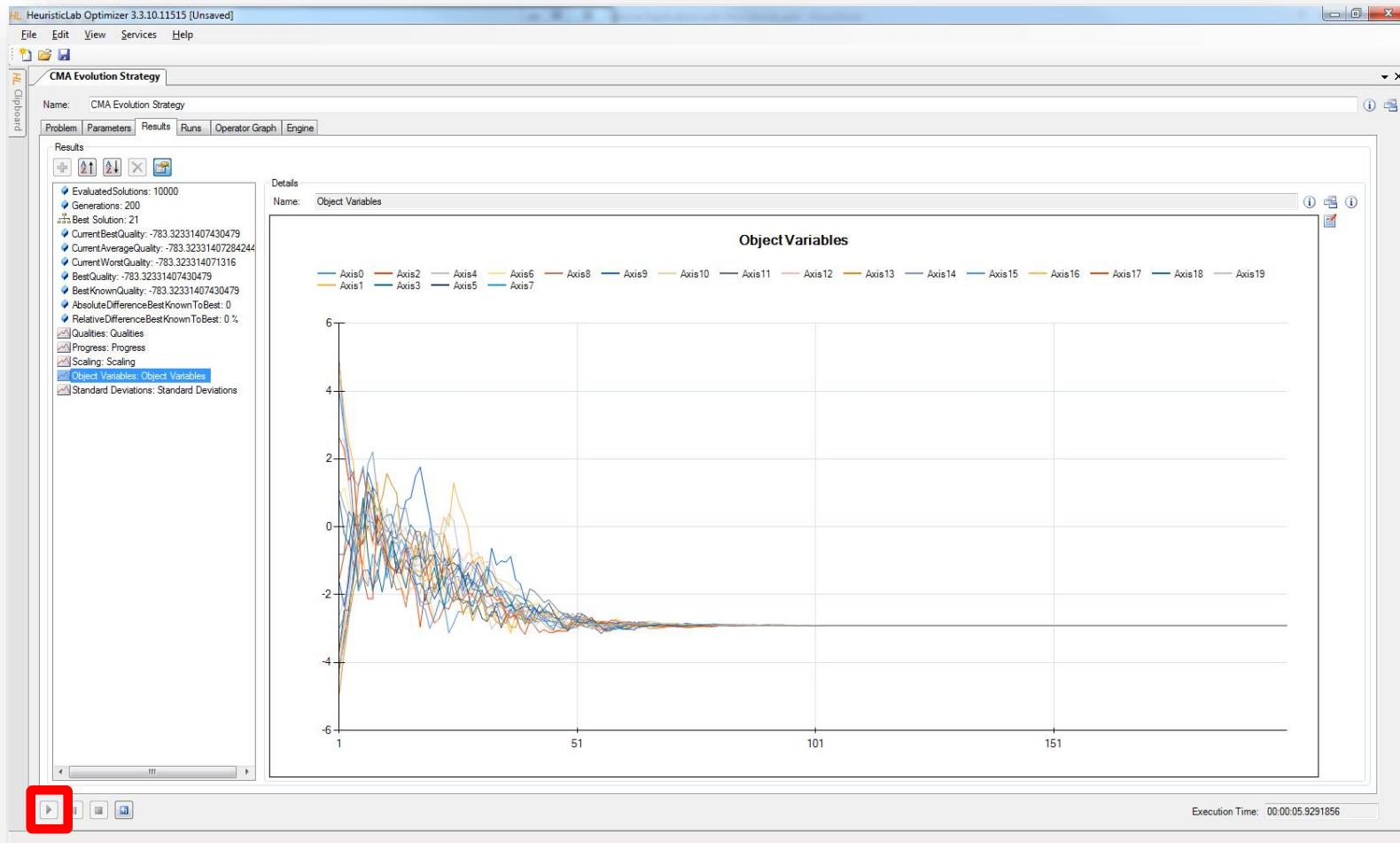


Singleobjective Function Opt.

- Choose suitable parameters
 - InitialSigma: 2
 - MaximumGenerations: 200
 - PopulationSize: 50



Singleobjective Function Opt.

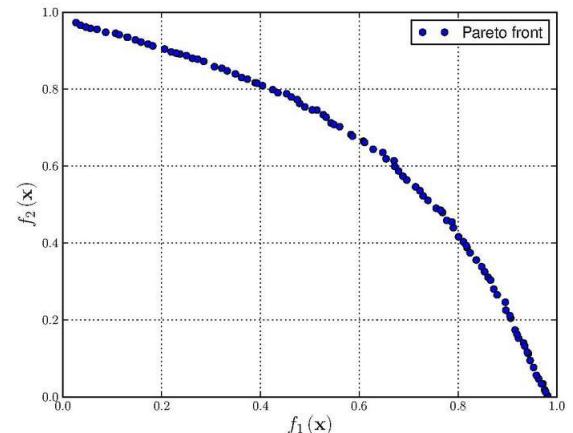


Multiobjective Function Opt.

- Fonseca and Fleming function:

$$\text{Minimize} = \begin{cases} f_1(x) = 1 - e^{-\sum_{i=1}^n (x_i - \frac{1}{\sqrt{n}})^2} \\ f_2(x) = 1 - e^{-\sum_{i=1}^n (x_i + \frac{1}{\sqrt{n}})^2} \end{cases}$$

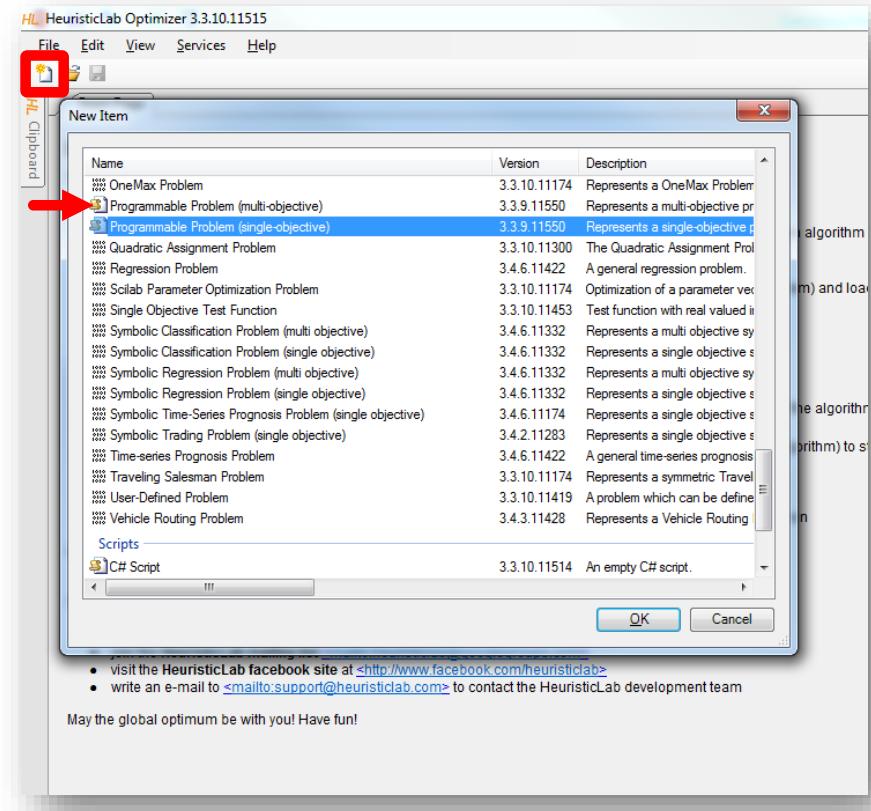
with $-4 \leq x \leq 4$



Multiobjective Function Opt.



- Click on „New Item“ to get a list of creatables
- Create a new Programmable Problem (multi-objective) by double clicking it



Multiobjective Function Opt.



- Choose an appropriate solution encoding
 - 10-dimensional real-valued vector
 - Minimize all fitness values

```
public bool[] Maximization { get { return new [] { false, false }; } }

public override void Initialize() {
    // Define the solution encoding which can also consist of multiple vectors, examples below
    // Encoding = new BinaryEncoding("b", length: 5);
    // Encoding = new IntegerEncoding("i", length: 5, min: 2, max: 14, step: 4);
    Encoding = new RealEncoding("vector", length: 10, min: -4.0, max: 4.0);
    // Encoding = new PermutationEncoding("P", length: 5, type: PermutationTypes.Absolute);
    // Encoding = new MultiEncoding()
    // .AddBinaryVector("b", length: 5)
    // .AddIntegerVector("i", length: 5, min: 2, max: 14, step: 4)
    // .AddRealVector("r", length: 5, min: -1.0, max: 1.0)
    // .AddPermutation("P", length: 5, type: PermutationTypes.Absolute)
}
;
```

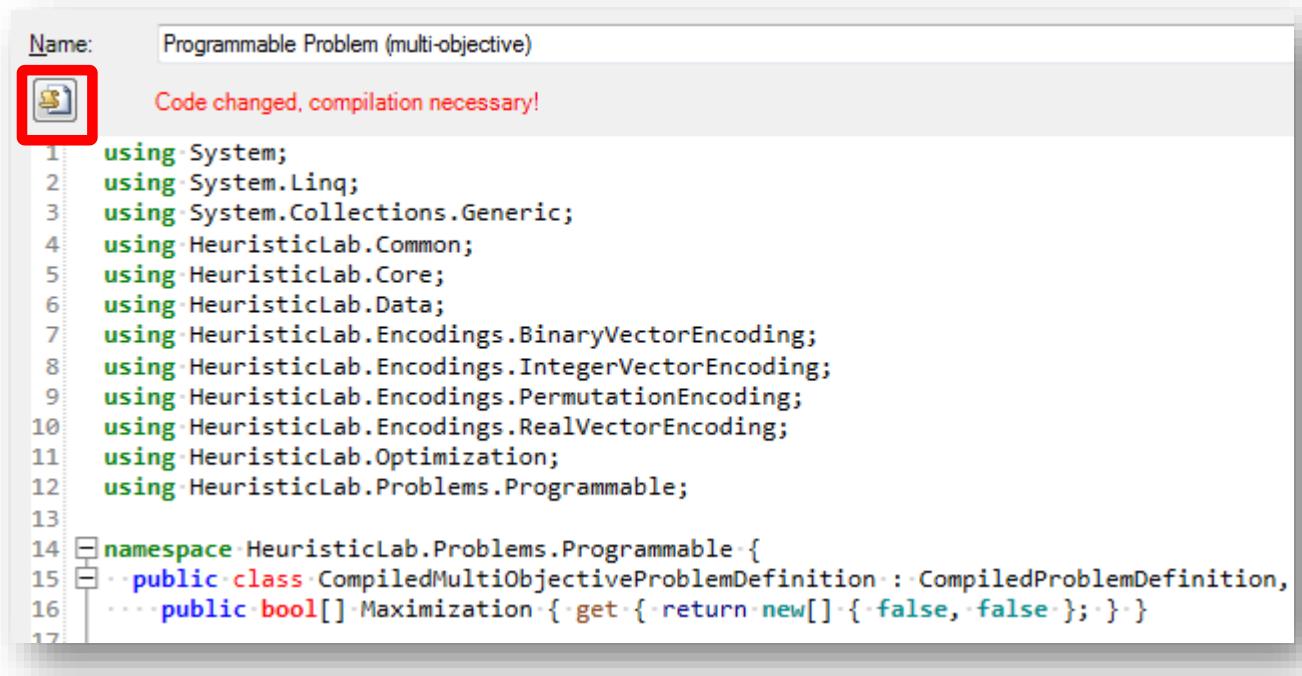
Multiobjective Function Opt.

- Define the fitness functions

```
public double[] Evaluate(Individual individual, IRandom random) {  
    var qualities = new double[2];  
    var vector = individual.RealVector("vector");  
    var n = vector.Length;  
    qualities[0] = 1.0 - Math.Exp(-vector.Sum(x => Math.Pow(x - 1.0/Math.Sqrt(n), 2)));  
    qualities[1] = 1.0 - Math.Exp(-vector.Sum(x => Math.Pow(x + 1.0/Math.Sqrt(n), 2)));  
    return qualities;  
}
```

Multiobjective Function Opt.

- Compile the problem definition



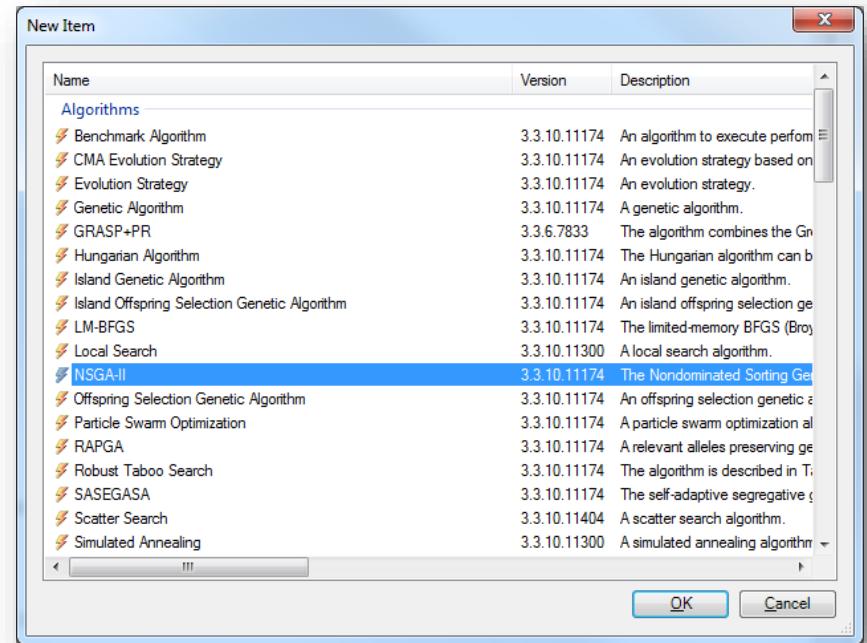
Name: Programmable Problem (multi-objective)

Code changed, compilation necessary!

```
1  using System;
2  using System.Linq;
3  using System.Collections.Generic;
4  using HeuristicLab.Common;
5  using HeuristicLab.Core;
6  using HeuristicLab.Data;
7  using HeuristicLab.Encodings.BinaryVectorEncoding;
8  using HeuristicLab.Encodings.IntegerVectorEncoding;
9  using HeuristicLab.Encodings.PermutationEncoding;
10 using HeuristicLab.Encodings.RealVectorEncoding;
11 using HeuristicLab.Optimization;
12 using HeuristicLab.Problems.Programmable;
13
14 namespace HeuristicLab.Problems.Programmable {
15     public class CompiledMultiObjectiveProblemDefinition : CompiledProblemDefinition,
16         public bool[] Maximization { get { return new[] { false, false }; } }
```

Multiobjective Function Opt.

- Create a suitable optimization algorithm
- Select NSGA-II
- Click OK



Multiobjective Function Opt.



The screenshot shows the HeuristicLab Optimizer interface. On the left, the 'Programmable Problem (multi-objective)' tab is active, displaying C# code for a multi-objective optimization problem. The code defines a class `CompiledMultiObjectiveProblemDefinition` that implements the `IProgrammableProblem` interface. It includes methods for initializing variables and evaluating individuals. A red arrow points from the 'Variables' section of the programmable problem tab to the 'NSGA-II' tab on the right, which is titled 'NSGA-II'. The 'NSGA-II' tab has tabs for 'Problem', 'Algorithm', 'Results', 'Runs', 'Operator Graph', and 'Engine'. The 'Algorithm' tab is selected. A red arrow labeled 'Drag'n'Drop' points from the 'NSGA-II' tab towards the 'Programmable Problem' tab, indicating the process of integrating the two.

```
using System;
using System.Linq;
using System.Collections.Generic;
using HeuristicLab.Common;
using HeuristicLab.Core;
using HeuristicLab.Data;
using HeuristicLab.Encodings.BinaryVectorEncoding;
using HeuristicLab.Encodings.IntegerVectorEncoding;
using HeuristicLab.Encodings.PermutationEncoding;
using HeuristicLab.Encodings.RealVectorEncoding;
using HeuristicLab.Optimization;
using HeuristicLab.Problems.Programmable;

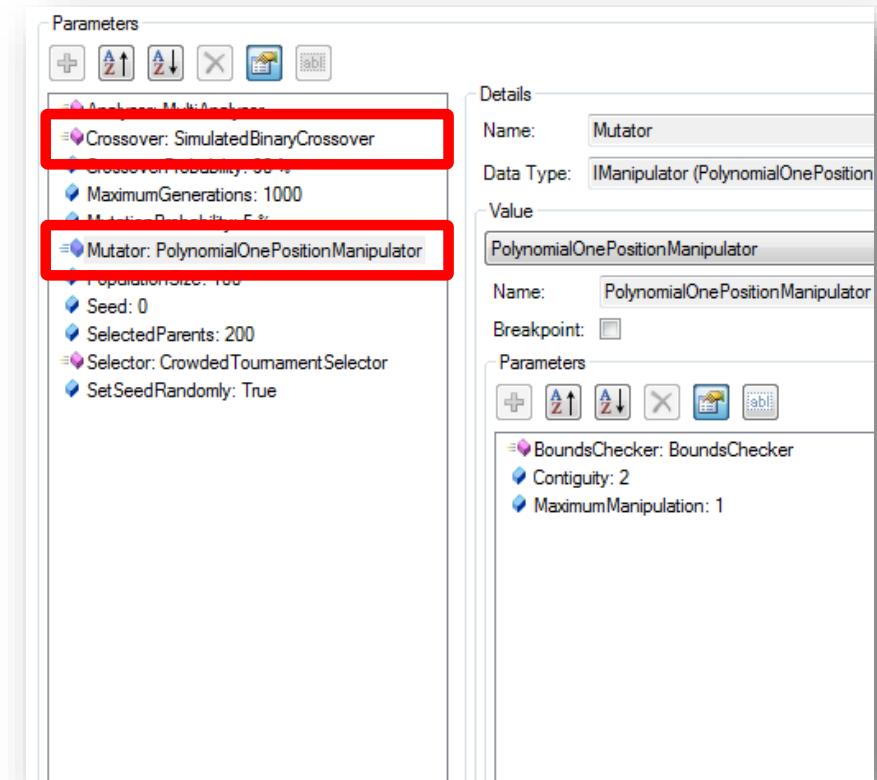
namespace HeuristicLab.Problems.Programmable {
    public class CompiledMultiObjectiveProblemDefinition : CompiledProblem
    {
        public bool[] Maximization { get { return new[] { false, false }; } }

        public override void Initialize()
        {
            // Use vars.yourVariable to access variables in the variable store
            // Define the solution encoding which can also consist of multiple encodings
            Encoding = new BinaryVectorEncoding("b", length: 5);
            //Encoding = new IntegerVectorEncoding("i", length: 5, min: 2, max: 14);
            //Encoding = new RealVectorEncoding("vector", length: 10, min: -4.0, max: 4.0);
            //Encoding = new PermutationEncoding("p", length: 5, type: PermutationType);
            // The encoding can also be a combination
            //Encoding = new MultiEncoding();
            //Add(new BinaryVectorEncoding("b", length: 5));
            //Add(new IntegerVectorEncoding("i", length: 5, min: 2, max: 14));
            //Add(new RealVectorEncoding("r", length: 5, min: -1.0, max: 1.0));
            //Add(new PermutationEncoding("p", length: 5, type: PermutationType));
        }

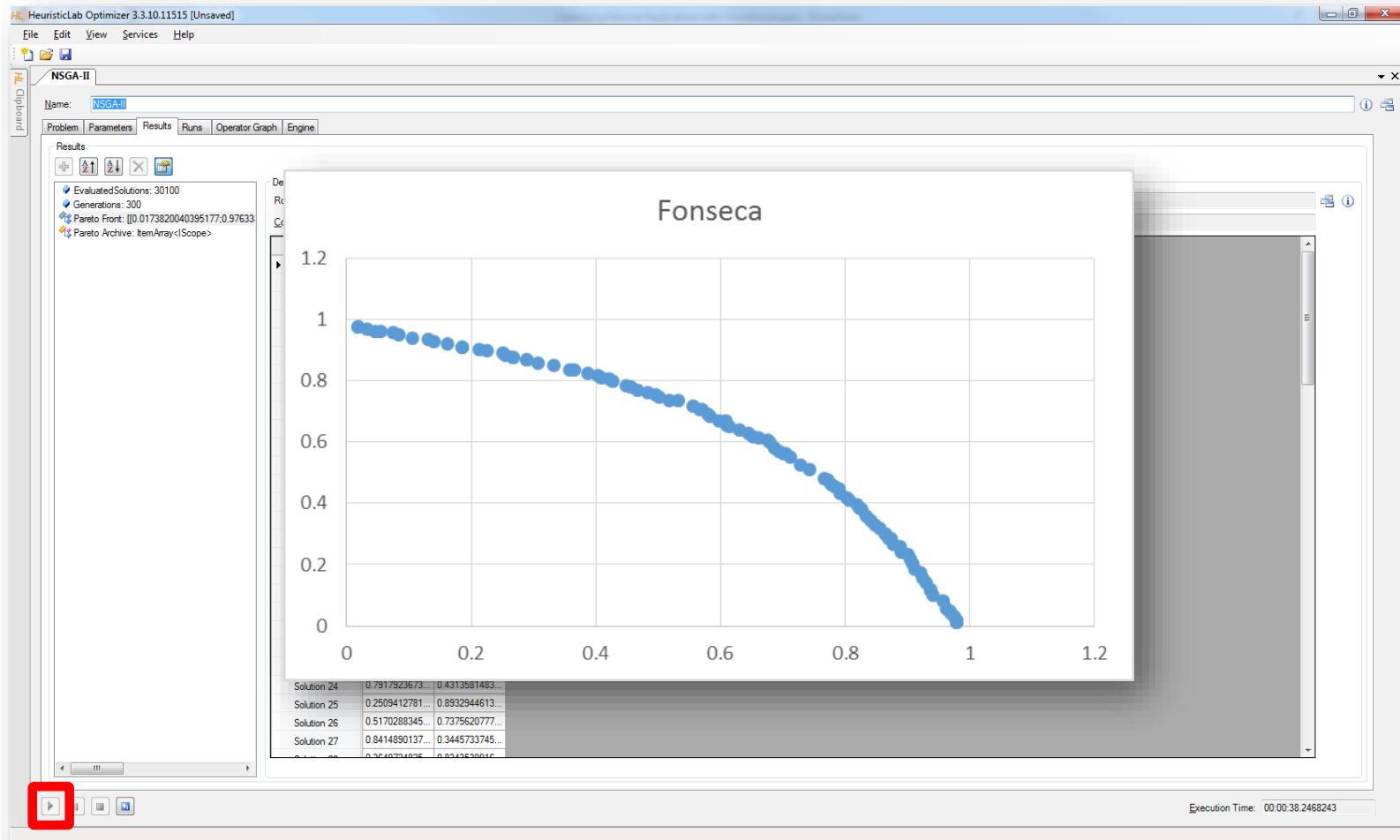
        public double[] Evaluate(Individual individual, IRandom random)
        {
            // Use vars.yourVariable to access variables in the variable store
            return null;
        }
    }
}
```

Multiobjective Function Opt.

- Adjust the parameters
- Set Crossover: SBX
- Set Mutator: Polynomial

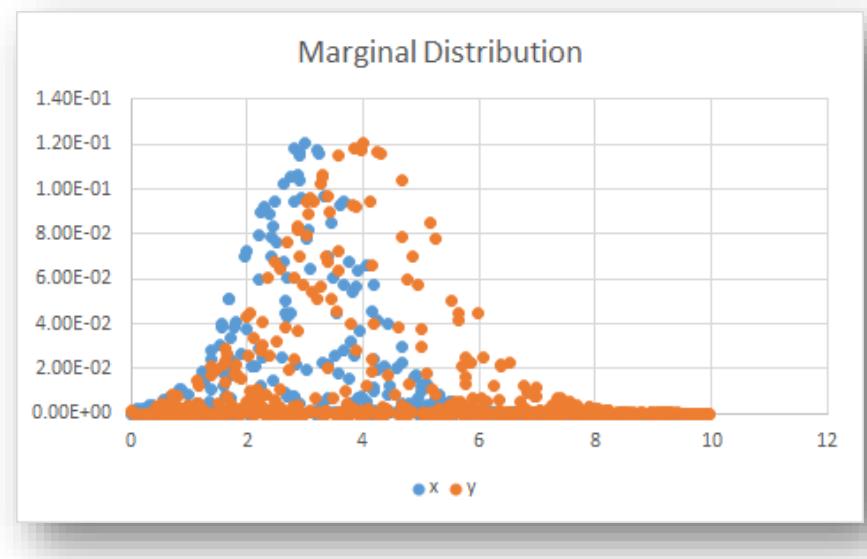


Multiobjective Function Opt.



Non-linear Curve Fitting

- In this example: Fitting a 2 dimensional Gaussian distribution to noisy data
- We will create a random dataset as instance
- We will then fit the parameters



Non-linear Curve Fitting

- The probability density function for a k dimensional Gaussian distribution is given

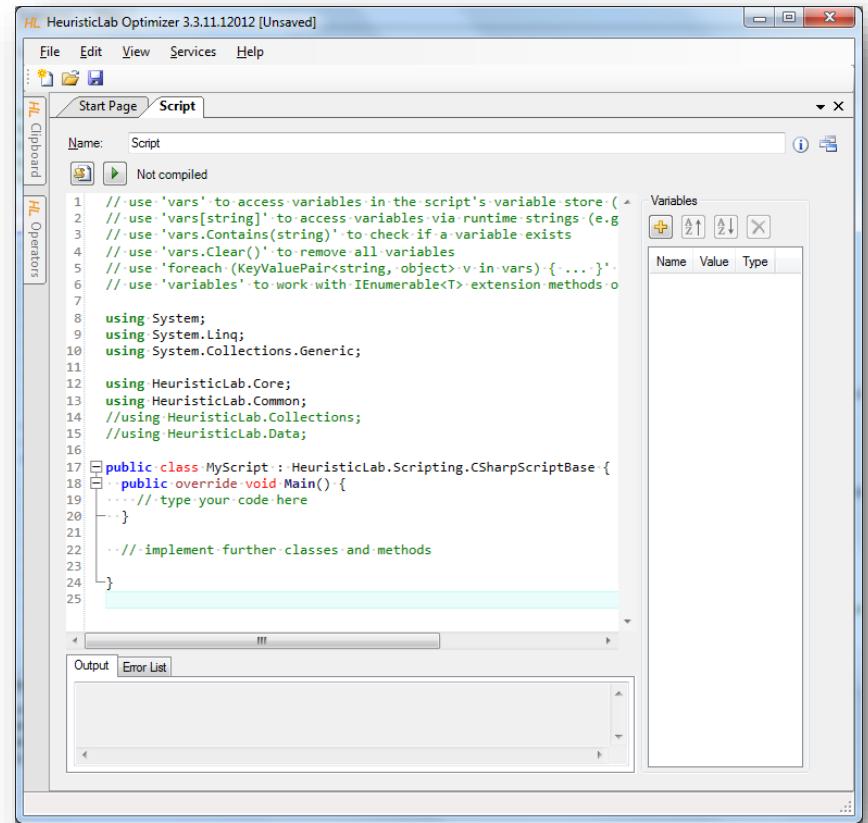
$$f_{\mathbf{x}}(x_1, \dots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

- For 2 dimensional Gaussian distribution we need to fit a total of 5 parameters
 - $\boldsymbol{\mu}$ (2 dimensions)
 - Σ (3 dimensions due to symmetry)

Source: http://en.wikipedia.org/wiki/Multivariate_normal_distribution

Non-linear Curve Fitting

- To generate the dataset, first create a new C# Script
- Then enter the code that generates the data on the next slide



The screenshot shows the HeuristicLab Optimizer 3.3.11.12012 application window. The title bar reads "HL HeuristicLab Optimizer 3.3.11.12012 [Unsaved]". The main area is a script editor titled "Script". The code in the editor is:

```

1 // use 'vars' to access variables in the script's variable store (e.g.
2 // use 'vars[string]' to access variables via runtime strings (e.g.
3 // use 'vars.Contains(string)' to check if a variable exists
4 // use 'vars.Clear()' to remove all variables
5 // use 'foreach (KeyValuePair<string, object> v in vars) { ... }' to
6 // use 'variables' to work with IEnumerable<T>-extension methods
7
8 using System;
9 using System.Linq;
10 using System.Collections.Generic;
11
12 using HeuristicLab.Core;
13 using HeuristicLab.Common;
14 //using HeuristicLab.Collections;
15 //using HeuristicLab.Data;
16
17 public class MyScript : HeuristicLab.Scripting.CSharpScriptBase {
18     public override void Main() {
19         ... // type your code here
20     }
21
22     ... // implement further classes and methods
23
24 }
  
```

On the right side of the interface, there is a "Variables" panel with a table:

Name	Value	Type

At the bottom of the interface, there are tabs for "Output" and "Error List".

Non-linear Curve Fitting

```

public class MyScript : HeuristicLab.Scripting.CSharpScriptBase {
  public override void Main() {
    var cov = new double[,] { { 1, 0.5 }, { 0.5, 2 } };
    var invCov = GetInverse(cov);
    var mu = new double[] { 3, 4 };

    var rand = new MersenneTwister();
    var nd = new NormalDistributedRandom(rand, 0, 0.01);

    var sampleSize = 500;
    var data = new DoubleMatrix(sampleSize, 3);
    for (var i = 0; i < sampleSize; i++) {
      data[i, 0] = rand.NextDouble() * 10;
      data[i, 1] = rand.NextDouble() * 10;
      data[i, 2] = TwoDGauss(data[i, 0], data[i, 1], mu, cov, invCov) + nd.NextDouble();
    }
    vars.data = data;
  }

  private double TwoDGauss(double x, double y, double[] mu, double[,] cov, double[,] invCov) {
    var a = x - mu[0];
    var b = y - mu[1];
    return 1.0 / (2.0 * Math.PI * Math.Sqrt(GetDeterminant(cov)))
      * Math.Exp(-0.5 * ((a * invCov[0, 0] + b * invCov[1, 0]) * a + (a * invCov[0, 1] + b * invCov[1, 1]) * b));
  }

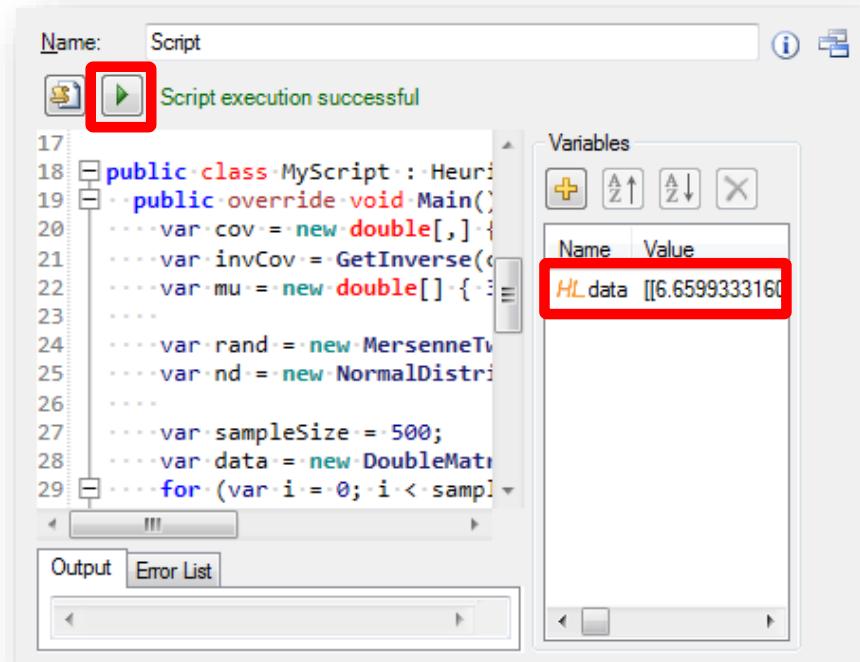
  private double[,] GetInverse(double[,] m) {
    var det = GetDeterminant(m);
    return new double[,] { { m[1, 1] / det, -m[0, 1] / det }, { -m[1, 0] / det, m[0, 0] / det } };
  }

  private double GetDeterminant(double[,] m) {
    return m[0, 0] * m[1, 1] - m[0, 1] * m[1, 0];
  }
}

```

Non-linear Curve Fitting

- Run the script by clicking the run button or by hitting F5
- The data will be generated as a matrix and appear in the VariableStore

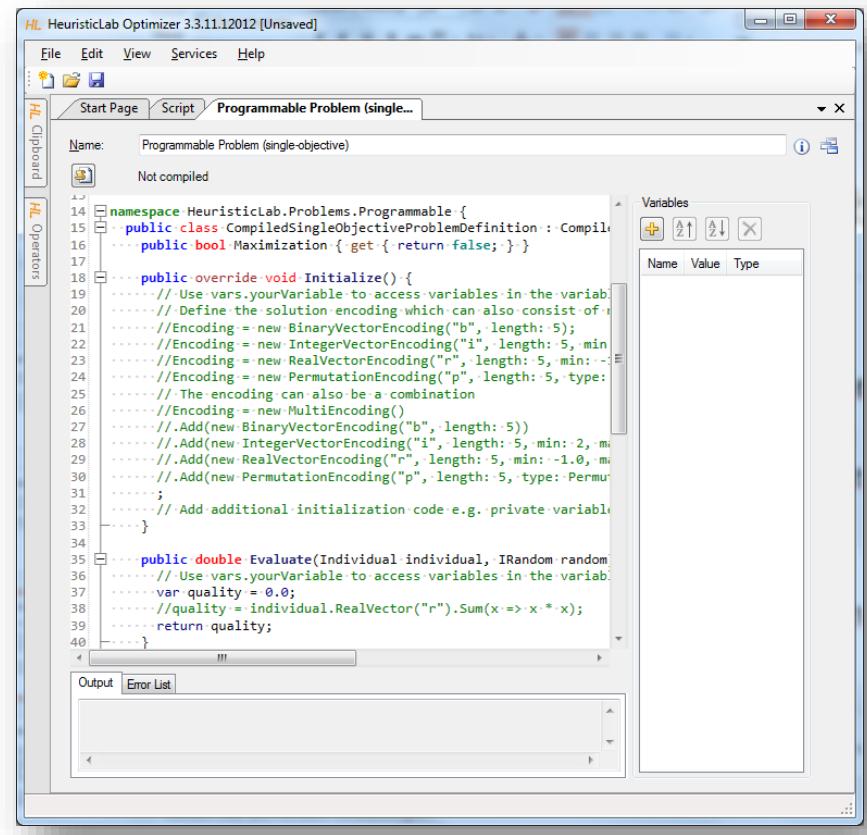


The screenshot shows the HeuristicLab Script Editor interface. The script window displays a Java-like code snippet for a class named MyScript. The run button (a green triangle icon) is highlighted with a red box. The status bar at the top right says "Script execution successful". In the Variables pane, there is a table with one row:

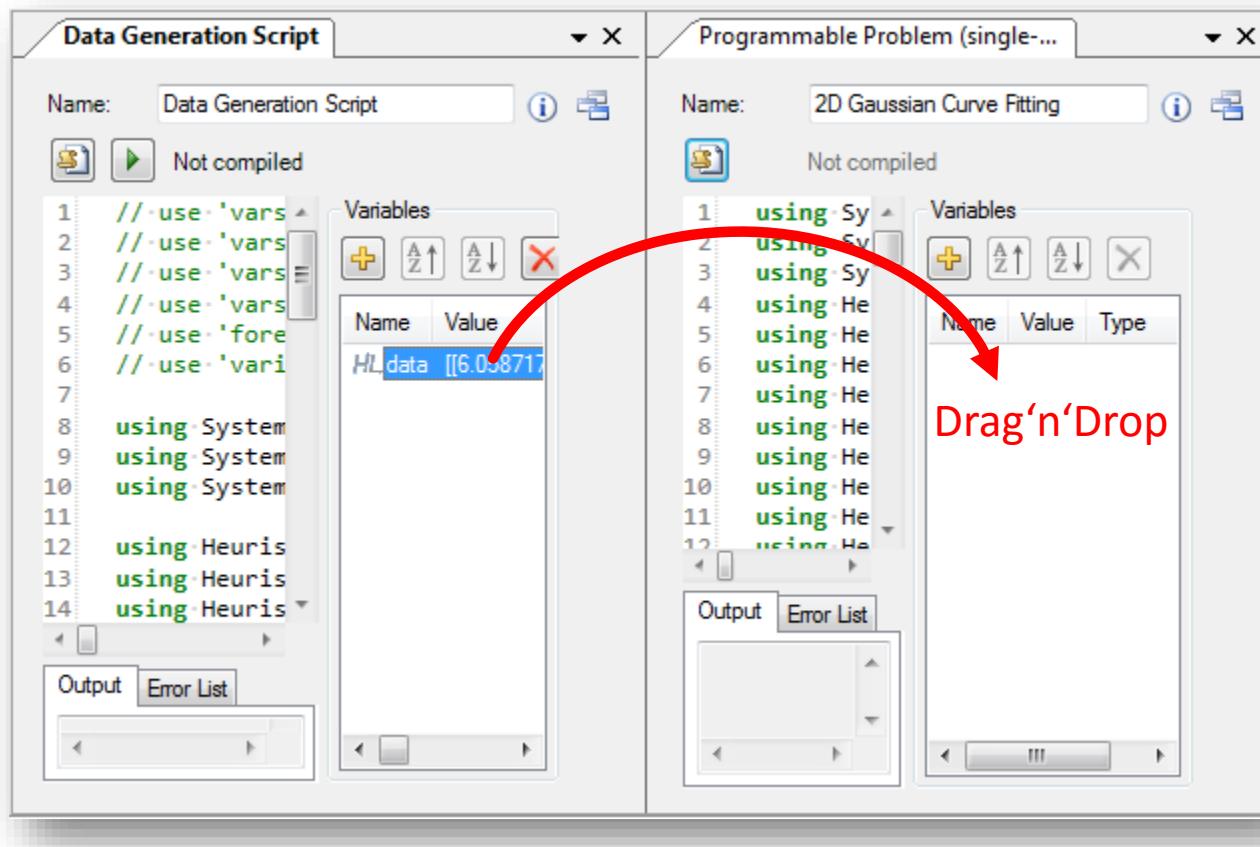
Name	Value
HL_data	[[6.6599333160]]

Non-linear Curve Fitting

- Create a new single-objective programmable problem
- Drag'n'drop the data from the script onto the problem's VariableStore
- Code the problem definition
- Compile the problem



Non-linear Curve Fitting



Non-linear Curve Fitting



```
public bool Maximization { get { return false; } }

public override void Initialize() {
    Encoding = new RealVectorEncoding("r", length: 5, min: 0.0, max: 10.0);
}

public double Evaluate(Individual individual, IRandom random) {
    var vec = individual.RealVector("r");
    var mu = new double[] { vec[0], vec[1] };
    var cov = new double[,] { { vec[2], vec[3] }, { vec[3], vec[4] } };
    var invCov = GetInverse(cov);

    var data = (DoubleMatrix)vars.data;
    var quality = 0.0;
    for (var i = 0; i < data.Rows; i++) {
        var estimated = TwoDGauss(data[i, 0], data[i, 1], mu, cov, invCov);
        if (double.IsNaN(estimated) || double.IsInfinity(estimated)) quality += 1000;
        else quality += (estimated - data[i, 2]) * (estimated - data[i, 2]);
    }
    return quality / data.Rows;
}

private double TwoDGauss(double x, double y, double[] mu, double[,] cov, double[,] invCov) {
    var a = x - mu[0];
    var b = y - mu[1];
    return 1.0 / (2.0 * Math.PI * Math.Sqrt(GetDeterminant(cov)))
        * Math.Exp(-0.5 * ((a * invCov[0, 0] + b * invCov[1, 0]) * a + (a * invCov[0, 1] + b * invCov[1, 1]) * b));
}

private double[,] GetInverse(double[,] m) {
    var det = GetDeterminant(m);
    return new double[,] { { m[1, 1] / det, -m[0, 1] / det }, { -m[1, 0] / det, m[0, 0] / det } };
}

private double GetDeterminant(double[,] m) {
    return m[0, 0] * m[1, 1] - m[0, 1] * m[1, 0];
}
```

Non-linear Curve Fitting

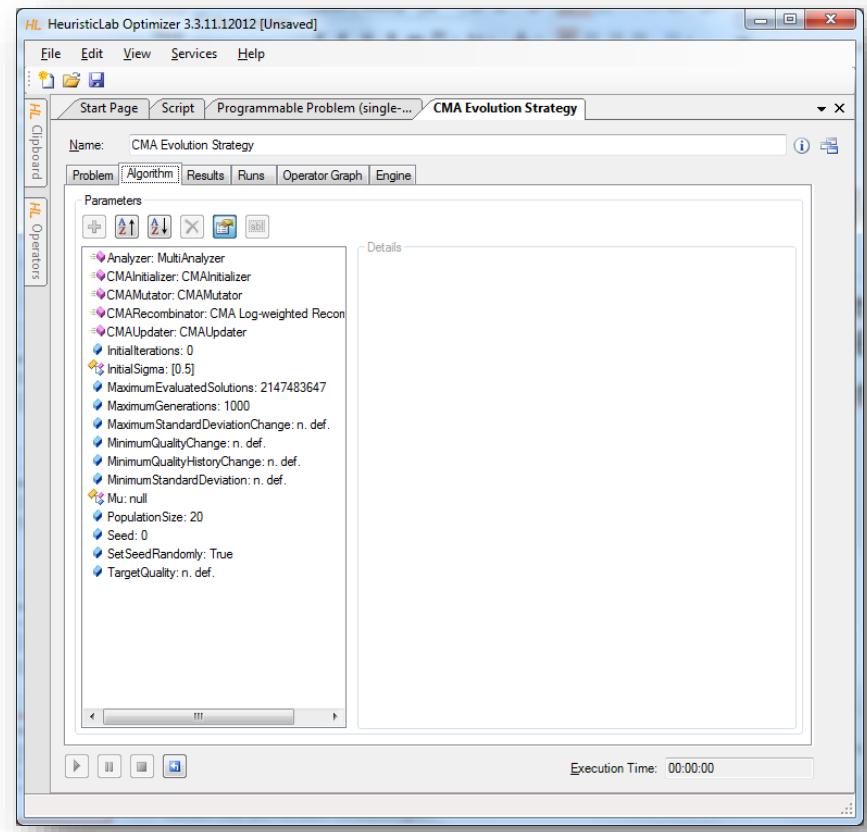


- Note that in the evaluation function we took care of degenerate cases
- If the estimation of the model is not a double number or infinity, an arbitrary, but high number is returned as a goodness of fit

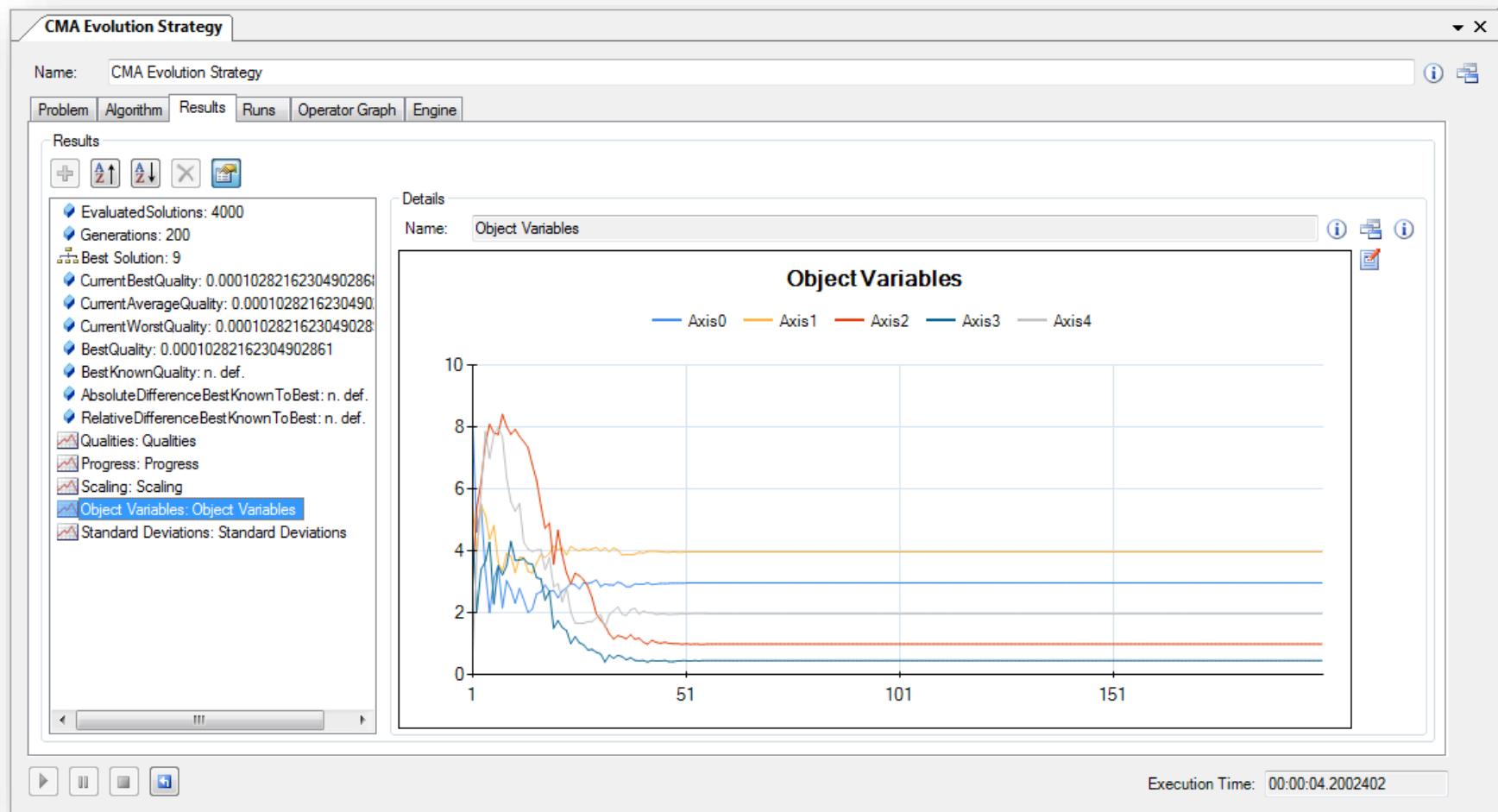
```
for (var i = 0; i < data.Rows; i++) {
    var estimated = TwoDGauss(data[i, 0], data[i, 1], mu, cov, invCov);
    if (double.IsNaN(estimated) || double.IsInfinity(estimated)) quality += 1000;
    else quality += (estimated - data[i, 2]) * (estimated - data[i, 2]);
}
```

Non-linear Curve Fitting

- Create a new suitable algorithm, e.g. CMA-ES
- Drop the problem onto the algorithm
- Set the algorithm parameters
 - MaximumGenerations: 200
 - InitialSigma: 5
- Run the algorithm



Non-linear Curve Fitting



Agenda

- Objectives of the Tutorial
- Introduction
- Where to get HeuristicLab?
- Plugin Infrastructure
- Graphical User Interface
- Available Algorithms & Problems
- **Demonstration Part I: External Evaluation Problem**
- **Demonstration Part II: MATLAB and Scilab Parameter Optimization Problem**
- **Demonstration Part III: Programmable Problem**
- 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
- 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
- Statistics
 - statistical tests and automated statistical analysis



Planned Features

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

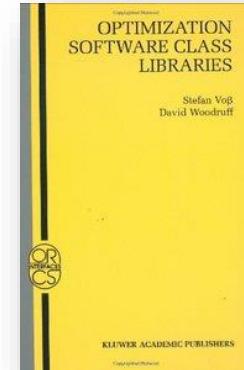
WWW: <http://heal.heuristiclab.com>



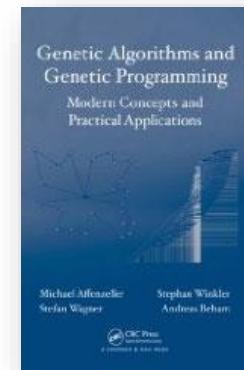
Suggested Readings



- 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>