

**ASSIGMENT 3**

**BCS2313 ARTIFICIAL INTELLIGENCE**

**NAME**

**RUBASRLI BALAMURALI**

**ID**

**CD14024**

1. You have to find the **maximum** value for function 9x2 - 3x. The range of x is [-5,5], with the population size of 10 individuals/chromosomes. You can set the maximum generation to 100. Use Jenetics tool to find the maximum value with the **Roulette wheel selection**, with 0.1 **mutation** probability and 0.5 **single point crossover** probability.

**CODING**

**//RUBASRI\_CD`14024**

**package org.jenetics.example;**

**import static java.lang.Math.PI;**

**import static java.lang.Math.cos;**

**import static java.lang.Math.sin;**

**import static org.jenetics.engine.EvolutionResult.toBestPhenotype;**

**import static org.jenetics.engine.limit.bySteadyFitness;**

**import org.jenetics.DoubleGene;**

**import org.jenetics.MeanAlterer;**

**import org.jenetics.Mutator;**

**import org.jenetics.Optimize;**

**import org.jenetics.Phenotype;**

**import org.jenetics.engine.Engine;**

**import org.jenetics.engine.EvolutionStatistics;**

**import org.jenetics.engine.codecs;**

**import org.jenetics.util.DoubleRange;**

**import org.jenetics.SinglePointCrossover;**

**import org.jenetics.MultiPointCrossover;**

**import org.jenetics.RouletteWheelSelector;**

**public class SimpleFunction1 {**

**// The fitness function.**

**private static double fitness(final double x) {**

**return (9\*(Math.pow(x, 2))-(3\*x));**

**}**

**public static void main(final String[] args) {**

**final Engine<DoubleGene, Double> engine;**

**engine = Engine**

**// Create a new builder with the given fitness**

**// function and chromosome.**

**.builder(**

**SimpleFunction1::fitness,**

**codecs.ofScalar(DoubleRange.of(-5.0, 5.0)))**

**.populationSize(10)**

**.optimize(Optimize.MAXIMUM)**

**.offspringSelector(new RouletteWheelSelector<>())**

**.alterers(**

**new SinglePointCrossover<>(0.5),**

**//new MultiPointCrossover<>(0.7),**

**new Mutator<>(0.1))**

**//new MeanAlterer<>(0.6))**

**// Build an evolution engine with the**

**// defined parameters.**

**.build();**

**// Create evolution statistics consumer.**

**final EvolutionStatistics<Double, ?>**

**statistics = EvolutionStatistics.ofNumber();**

**final Phenotype<DoubleGene, Double> best = engine.stream()**

**// Truncate the evolution stream after 7 "steady"**

**// generations.**

**//.limit(bySteadyFitness(7))**

**// The evolution will stop after maximal 100**

**// generations.**

**.limit(100)**

**// Update the evaluation statistics after**

**// each generation**

**.peek(statistics)**

**// Collect (reduce) the evolution stream to**

**// its best phenotype.**

**.collect(toBestPhenotype());**

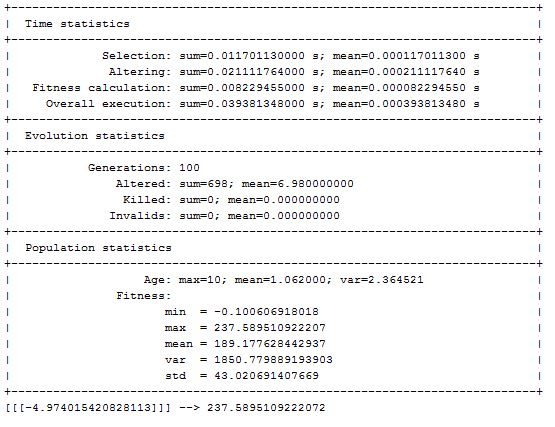
**System.out.println(statistics);**

**System.out.println(best);**

**}**

**}**

**SOLUTION**



1. You have to find the **minimum** value for function x3 + x2 + 5x. The range of x is [-10,10], with the population size of 10 individuals/chromosomes. You can set the maximum generation to 100. Use Jenetics tool to find the **minimum** value with the **Roulette wheel selection**, with 0.05 **mutation** probability and 0.6 **multi point crossover** probability

**CODING**

**//RUBASRI\_CD14024**

**package org.jenetics.example;**

**import static java.lang.Math.PI;**

**import static java.lang.Math.cos;**

**import static java.lang.Math.sin;**

**import static org.jenetics.engine.EvolutionResult.toBestPhenotype;**

**import static org.jenetics.engine.limit.bySteadyFitness;**

**import org.jenetics.DoubleGene;**

**import org.jenetics.MeanAlterer;**

**import org.jenetics.Mutator;**

**import org.jenetics.Optimize;**

**import org.jenetics.Phenotype;**

**import org.jenetics.engine.Engine;**

**import org.jenetics.engine.EvolutionStatistics;**

**import org.jenetics.engine.codecs;**

**import org.jenetics.util.DoubleRange;**

**import org.jenetics.SinglePointCrossover;**

**import org.jenetics.MultiPointCrossover;**

**import org.jenetics.RouletteWheelSelector;**

**public class SimpleFunction2 {**

**// The fitness function.**

**private static double fitness(final double x) {**

**return ((Math.pow(x, 3))+(Math.pow(x, 2))+(5\*x));**

**}**

**public static void main(final String[] args) {**

**final Engine<DoubleGene, Double> engine;**

**engine = Engine**

**// Create a new builder with the given fitness**

**// function and chromosome.**

**.builder(**

**SimpleFunction2::fitness,**

**codecs.ofScalar(DoubleRange.of(-10.0, 10.0)))**

**.populationSize(10)**

**.optimize(Optimize.MINIMUM)**

**.offspringSelector(new RouletteWheelSelector<>())**

**.alterers(**

**//new SinglePointCrossover<>(0.5),**

**new MultiPointCrossover<>(0.6),**

**new Mutator<>(0.05))**

**//new MeanAlterer<>(0.6))**

**// Build an evolution engine with the**

**// defined parameters.**

**.build();**

**// Create evolution statistics consumer.**

**final EvolutionStatistics<Double, ?>**

**statistics = EvolutionStatistics.ofNumber();**

**final Phenotype<DoubleGene, Double> best = engine.stream()**

**// Truncate the evolution stream after 7 "steady"**

**// generations.**

**//.limit(bySteadyFitness(7))**

**// The evolution will stop after maximal 100**

**// generations.**

**.limit(100)**

**// Update the evaluation statistics after**

**// each generation**

**.peek(statistics)**

**// Collect (reduce) the evolution stream to**

**// its best phenotype.**

**.collect(toBestPhenotype());**

**System.out.println(statistics);**

**System.out.println(best);**

**}**

**}**

**SOLUTION**

