A4M33BIA: Exercise #3

Simple Genetic Algorithm Replacement Strategy

Jiří Kubalík
kubalik@ciirc.cvut.cz

Jan Drchal
drchajan@fel.cvut.cz

Task

- Download sources from https://cw.felk.cvut.cz/wiki/courses/a4m33bia/labs
- Implement methods
  - evolvePopGenerational()
  - evolvePopSteadyState()

  in class SGA.java
- Experiment with defined fitness functions
  - try different replacement strategies
  - try different setting of the Tournament selection parameter
  - …

Generational Strategy

```
1   initialize(oldPopulation)
2   evaluate(oldPopulation)
3   while(not termination condition)
4       newPopulation ← bestOf(oldPopulation)        // elitism
5       while(newPopulation not full)
6           parents ← select(oldPopulation)
7           offspring ← crossover(parents)
8           mutate(offspring)        // optional
9           evaluate(offspring)
10          newPopulation ← offspring
11      swap(oldPopulation, newPopulation)
12  return bestOf(oldPopulation)
```

Steady-State Strategy

```
1  initialize(population)
2  evaluate(population)
3  while(not termination condition)
4        parents ← select(population)
5        offspring ← crossover(parents)
6        mutate(offspring)        // optional
7        evaluate(offspring)
8        population ← offspring        // replacement rule
9  return bestOf(population)
```

class ContinuousFunction.java

Methods

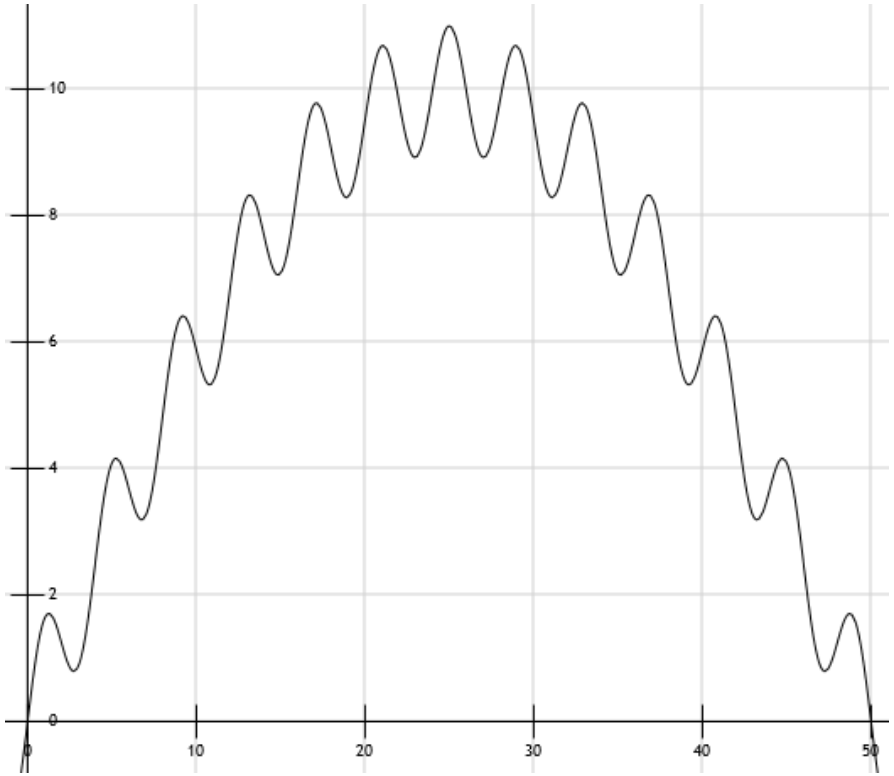
```
// calculates a fitness value in single-objective opt. problem
double f(double[] x)

// calculates objectives in multi-objective opt. problem
double[] o(int[] x)

// returns a dimension of the problem
int getDimension()

// returns true if the optimization function is to be maximized, otherwise
// returns false
boolean maximize()
```

Example function

class ExampleFunction.java

Maximize: $f(x) = \sin\left(\pi \cdot \frac{x}{2}\right) + 10 \cdot \sin\left(\pi \ \frac{x}{50}\right)$

Optimum: $f(25) = 11$

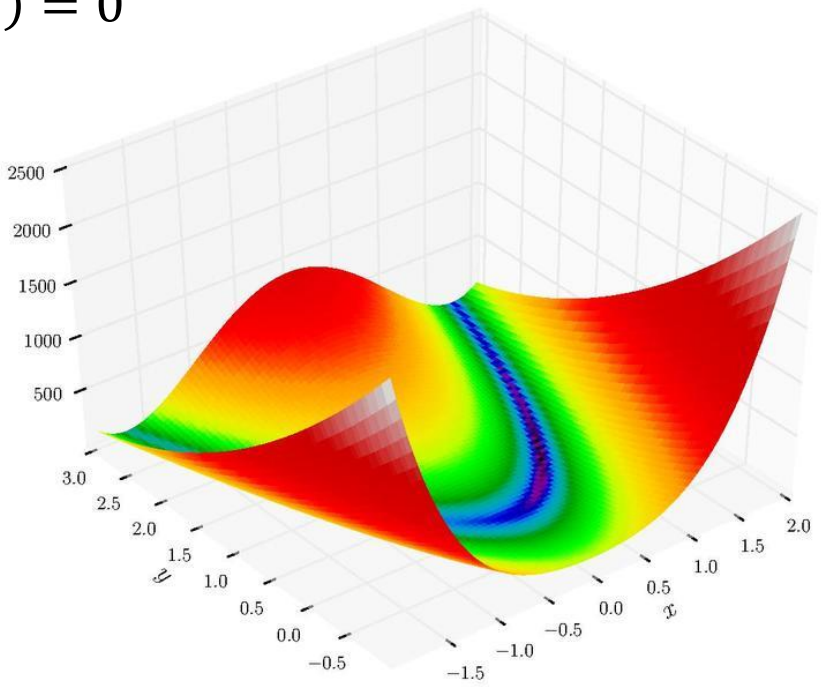


Rosenbrock's Function

class RosenbrockFunction.java

Minimize: $f(x_1, x_2) = (1 - x_1)^2 + 100 \cdot (x_2 - {x_1}^2)$

Optimum: $f(1, 1) = 0$



You can play with other functions as well:
http://en.wikipedia.org/wiki/Test_functions_for_optimization

class Individual.java

Attributes:

- int[] genes
- double fitness
- SGAParameters parameters

Methods

- int getNumberOfGenes()
- double[] decodeGenes()
- double[] normalizeX(double[] x)
- void getFitness()
- Individual[] crossover(Individual mate)
- void mutation()
- boolean betterThan(Individual ind)

class SGA.java

Attributes:

- Individual[] population // population of candidate solutions to be // evolved
- SGAParameters parameters

Methods

// evolves the population using a generational replacement strategy
void **evolvePopGenerational**(Individual[] oldPop, int totEvals)

// evolves the population using a steady-state replacement strategy
void **evolvePopSteadyState**(Individual[] population, int totEvals)

// returns an index of the worst individual in the current population
int worstInPopulation(Individual[] population)

// returns an index of one selected individual
int selection(Individual[] population)

class SGAParameters.java

Attributes

- String functionName; // fitness function class name
- ContinuousFunction function = null; // fitness function
- int genesPerValue = 16; // nb. of bits per variable
- double minX; // variable domain range
- double maxX;
- double rangeX;
- int popSize; // population size
- int evaluations; // maximum number of fitness evaluations
- int crossType; // crossover type: 0 ... uniform, 1 ... 1-point, 2 ... 2-point
- double pC; // crossover probability
- double pM; // mutation probability
- int selectionType; // selection type: 0 ... tournament selection
- int tournamentSize; // tournament size parameter
- int replacementType; // repl. strategy: 0 ... generational, 1 ... steady-state