Lab 3

Generated by Doxygen 1.9.3

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Algorithms Class Reference

Public Member Functions

- **Algorithms** (int equationNumber, int dimension, double lowerBound, double upperBound)
- double **randomWalk** (int iterations)
- double **localSearch** ()
- double **repeatedLocalSearch** (int iterations)
- double **differentialEvolution** (int populationSize, double crossoverFactor, double scalingFactor, double lambda, int generations, int strategy)
- double **particleSwarmOptimization** (int particles, double c1, double c2, int generations)

Public Attributes

- double * **bestFit**

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Algorithms()

```
Algorithms::Algorithms (
            int equationNumber,
            int dimension,
            double lowerBound,
            double upperBound )
```

Parameters

equationNumber	which equation will be considered for class methods
dimension	the dimension of the vectors to be used
lowerBound	the lower bound of the solution space
upperBound	the upper bound of the solution space

Exceptions

invalid_argument	when equationNumber $>$ 10 or $<$1 or dimension $<$ 1

3.1.2 Member Function Documentation

3.1.2.1 differentialEvolution()

```
double Algorithms::differentialEvolution (
            int populationSize,
            double crossoverFactor,
            double scalingFactor,
            double lambda,
            int generations,
            int strategy )
```

performs differential evolution

Parameters

populationSize	
crossoverFactor	
scalingFactor	
lambda	a second scaling factor used in some mutation strategy
generations	maximum generations
strategy	mutation strategy

Returns

best fitness found

3.1.2.2 localSearch()

```
double Algorithms::localSearch ( )
```

performs the local search algorithm stores best vector in the bestVector class variable

Returns

best fitness found

3.1.2.3 particleSwarmOptimization()

```
double Algorithms::particleSwarmOptimization (
            int particles,
            double c1,
            double c2,
            int generations )
```

performs particle swarm optimization

Parameters

particles	number of particles
c1	first learning factor
c2	second learning factor
generations	

Returns

best fitness found

3.1.2.4 randomWalk()

```
double Algorithms::randomWalk (
            int iterations )
```

performs the random walk algorithm stores best vector in the bestVector class variable

Parameters

iterations	how many vectors will by analyzed

Returns

best fitness found

Exceptions

invalid_argument	when iterations < 1

3.1.2.5 repeatedLocalSearch()

```
double Algorithms::repeatedLocalSearch (
            int iterations )
```

performs the repeated local search algorithm stores best vector in the bestVector class variable

Parameters

iterations	how many local searches will be performed

Returns

best fitness found

Exceptions

invalid_argument	when iterations < 1

The documentation for this class was generated from the following files:

- classes/CS471/Project 3/source/Algorithms.h
- classes/CS471/Project 3/source/Algorithms.cpp

3.2 Equations Class Reference

Public Member Functions

- double **schwefelsfunction** (double *vector, int length)
  evaluates Schwefel's function
- double **firstDeJongsfunction** (double *vector, int length)
  evaluates 1st De Jong's function
- double **rosenbrock** (double *vector, int length)
  evaluates Rosenbrock function
- double **rastrigin** (double *vector, int length)
  evaluates Rastrigin function
- double **griewangk** (double *vector, int length)
  evaluates Griewangk function
- double **sineEnvelopeSineWave** (double *vector, int length)
  evaluates Sine Envelope Sine Wave function
- double **stretchedVSineWave** (double *vector, int length)
  evaluates Stretched V Sine Wave function
- double **ackleysOne** (double *vector, int length)
  evaluates Ackley's One function
- double **ackleysTwo** (double *vector, int length)
  evaluates Ackley's Two function
- double **eggHolder** (double *vector, int length)
  evaluates Egg Holder function

3.2.1 Member Function Documentation

3.2.1.1 ackleysOne()

```
double Equations::ackleysOne (
            double * vector,
            int length )
```

evaluates Ackley's One function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.2 ackleysTwo()

```
double Equations::ackleysTwo (
            double * vector,
            int length )
```

evaluates Ackley's Two function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.3 eggHolder()

```
double Equations::eggHolder (
            double * vector,
            int length )
```

evaluates Egg Holder function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.4 firstDeJongsfunction()

```
double Equations::firstDeJongsfunction (
            double * vector,
            int length )
```

evaluates 1st De Jong's function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.5 griewangk()

```
double Equations::griewangk (
            double * vector,
            int length )
```

evaluates Griewangk function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.6 rastrigin()

```
double Equations::rastrigin (
            double * vector,
            int length )
```

evaluates Rastrigin function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.7 rosenbrock()

```
double Equations::rosenbrock (
            double * vector,
            int length )
```

evaluates Rosenbrock function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.8 schwefelsfunction()

```
double Equations::schwefelsfunction (
            double * vector,
            int length )
```

evaluates Schwefel's function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.9 sineEnvelopeSineWave()

```
double Equations::sineEnvelopeSineWave (
            double * vector,
            int length )
```

evaluates Sine Envelope Sine Wave function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

3.2.1.10 stretchedVSineWave()

```
double Equations::stretchedVSineWave (
            double * vector,
            int length )
```

evaluates Stretched V Sine Wave function

Parameters

vector	input vector
length	dimension of vector

Returns

fitness value

The documentation for this class was generated from the following files:

- classes/CS471/Project 3/source/Equations.h
- classes/CS471/Project 3/source/Equations.cpp

3.3 Population Class Reference

Public Member Functions

- **Population** (int experiments, int dimensionality, double lowerBound, double upperBound)
- void **runExperiment** (int type)

Public Attributes

- double ** **matrix**
- double * **fitness**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Population()

```
Population::Population (
            int experiments,
            int dimensionality,
            double lowerBound,
            double upperBound )
```

creates a **Population** (p. 14) object with a generated matrix as a class variable

Parameters

experiments	number of rows in generated population matrix
dimensionality	number of columns in generated population matrix
lowerBound	lower bound for values in population matrix
upperBound	upper bound bound for values in population matrix

3.3.2 Member Function Documentation

3.3.2.1 runExperiment()

```
void Population::runExperiment (
            int type )
```

runs vectors in population matrix through given equation and stores fitness vector in Population::fitness

Parameters

type	which equation will be used for experiment

The documentation for this class was generated from the following files:

- classes/CS471/Project 3/source/Population.h
- classes/CS471/Project 3/source/Population.cpp

Chapter 4

File Documentation

4.1 Algorithms.h

```
1 //
2 // Created by prest on 4/19/2022.
3 //
4
5 #ifndef SOURCE_ALGORITHMS_H
6 #define SOURCE_ALGORITHMS_H
7
8 #include "Population.h"
9 #include "Equations.h"
10 #include <functional>
11 #include <stdexcept>
12
13 class Algorithms {
14
15     int equationNum;
16     int dimension;
17     double lowerBound;
18     double upperBound;
19     Equations eq;
20     EquationsMemFn equation;
21
22     private:
23
24         void addVectors(double* result, double* arr1, double* arr2, int length);
25         void subtractVectors(double* result, double* arr1, double* arr2, int length);
26         void multiplyByScalar(double* result, double scalar, int length);
27         void mutuallyExclusiveParents(int parents[], int numToMake, int disallowedValue, std::mt19937*
      g1, std::uniform_int_distribution<>* d);
28         void forceInBounds(double* vector);
29
30         void deMutationStrategy1or6(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize,
31             double scalingFactor, double lambda, std::mt19937* g1);
32         void deMutationStrategy2or7(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize,
33             double scalingFactor, double lambda, std::mt19937* g1);
34         void deMutationStrategy3or8(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize,
35             double scalingFactor, double lambda, std::mt19937* g1);
36         void deMutationStrategy4or9(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize,
37             double scalingFactor, double lambda, std::mt19937* g1);
38         void deMutationStrategy5or10(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize,
39             double scalingFactor, double lambda, std::mt19937* g1);
40
41         void exponentialCrossover(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize, double crossoverFactor, std::mt19937* g1);
42         void binomialCrossover(Population* pop, double* popFitness, double** noisyVectors, int
      populationSize, double crossoverFactor, std::mt19937* g1);
43
44
45
46     public:
47
48         double* bestFit;
49
50         Algorithms(int equationNumber, int dimension, double lowerBound, double upperBound);
```

```
51          ~Algorithms();
52
53          double randomWalk(int iterations);
54          double localSearch();
55          double repeatedLocalSearch(int iterations);
56          double differentialEvolution(int populationSize, double crossoverFactor, double scalingFactor,
        double lambda, int generations, int strategy);
57          double particleSwarmOptimization(int particles, double c1, double c2, int generations);
58
59 };
60 typedef void (Algorithms::*mutationFunction)(Population* pop, double* popFitness, double** noisyVectors,
61          int populationSize, double scalingFactor, double lambda, std::mt19937* g1);
62 typedef void (Algorithms::*crossoverFunction)(Population* pop, double* popFitness, double** noisyVectors,
63          int populationSize, double crossoverFactor, std::mt19937* g1);
64
65 #endif //SOURCE_ALGORITHMS_H
```

4.2 Equations.h

```
1 #ifndef SOURCE_EQUATIONS_H
2 #define SOURCE_EQUATIONS_H
3
4 #include <cmath>
5 #define pi 4*atan(1)
6
7 class Equations{
8
9     public:
10          double schwefelsfunction(double* vector, int length);
11          double firstDeJongsfunction(double* vector, int length);
12          double rosenbrock(double* vector, int length);
13          double rastrigin(double* vector, int length);
14          double griewangk(double* vector, int length);
15          double sineEnvelopeSineWave(double* vector, int length);
16          double stretchedVSineWave(double* vector, int length);
17          double ackleysOne(double* vector, int length);
18          double ackleysTwo(double* vector, int length);
19          double eggHolder(double* vector, int length);
20
21 };
22 typedef double (Equations::*EquationsMemFn)(double* vector, int length);
23
24 #endif //SOURCE_EQUATIONS_H
```

4.3 Population.h

```
1 //
2 // Created by prest on 4/6/2022.
3 //
4
5 #ifndef SOURCE_POPULATION_H
6 #define SOURCE_POPULATION_H
7
8 #include "Equations.h"
9 #include <chrono>
10 #include <random>
11 #include <stdexcept>
12
13 class Population{
14
15     private:
16         int rows;
17         int columns;
18
19
20     public:
21
22         double ** matrix;
23         double* fitness;
24
25         Population(int experiments, int dimensionality, double lowerBound, double upperBound);
26         ~Population();
27         void runExperiment(int type);
28
29
30 };
31
32 #endif //SOURCE_POPULATION_H
```

Index