Project 3 Evolutionary Algorithms

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 benchmarkfunctions Struct Reference

Public Attributes

- string **name**
- double **min**
- double **max**
- double(∗ **foo**)(int dim, double myArray[ ])

The documentation for this struct was generated from the following file:

- FunctionStructs.h

2.2 DiffAlgorithm Class Reference

Public Member Functions

- double runDiffAlgorithm (int strategy, double min, double max, std::string name, double(∗foo)(int dim, double myArray[ ]))
- void fileReader ()
- void select (int index, double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- void mutate (int index, int strategy, double ∗∗population, double min, double max, double(∗foo)(int dim, double myArray[ ]))
- void crossover (int index, double CR)
- void bincrossover (double CR)
- void getBestSolution (double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- void setParent1 (double ∗parentone)
- void setParent2 (double ∗parenttwo)
- double ∗ getParent1 ()
- double ∗ getParent2 ()
- void setChild1 (double ∗childone)
- void setChild2 (double ∗childtwo)
- double ∗ getChild1 ()
- double ∗ getChild2 ()

Public Attributes

- Population * **pop** = new Population()

2.2.1 Member Function Documentation

2.2.1.1 bincrossover()

```
void DiffAlgorithm::bincrossover (
            double CR )
```

Performs binomial crossover between the parent and the noisy vector

Parameters

CR	a double representing the crossover rate

2.2.1.2 crossover()

```
void DiffAlgorithm::crossover (
            int index,
            double CR )
```

Performs crossover between the parent and the noisy vector

Parameters

index	an integer value of the parents index in the population
CR	a double representing the crossover rate

2.2.1.3 fileReader()

```
void DiffAlgorithm::fileReader ( )
```

Reads from the input file to initialize and set the private feilds of the population class to the values for Differential Evolution Algorithm

2.2.1.4 getBestSolution()

```
void DiffAlgorithm::getBestSolution (
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Calculates the best solution

Parameters

population	a 2D double array representing current population
foo	a pointer to the current fitness function

2.2.1.5 getChild1()

```
double * DiffAlgorithm::getChild1 ( )
```

Returns the first child

Returns

The first child

2.2.1.6 getChild2()

```
double * DiffAlgorithm::getChild2 ( )
```

Returns the second child

Returns

the second child

2.2.1.7 getParent1()

```
double * DiffAlgorithm::getParent1 ( )
```

Returns the first parent

Returns

the first parent

2.2.1.8 getParent2()

```
double * DiffAlgorithm::getParent2 ( )
```

Returns the second parent

Returns

the second parent

2.2.1.9 mutate()

```
void DiffAlgorithm::mutate (
            int index,
            int strategy,
            double ** population,
            double min,
            double max,
            double(*)(int dim, double myArray[]) foo )
```

Mutates several genes based on different strategies for the algorithm

Parameters

index	a integer value representing working index
strategy	a integer value representing current mutation strategy being ran
population	a 2D double array representing current population
min	a double representing the minimum of the range
max	a double representing the maximum of the range
foo	a pointer to the current fitness function

2.2.1.10 runDiffAlgorithm()

```
double DiffAlgorithm::runDiffAlgorithm (
            int strategy,
            double min,
            double max,
            std::string name,
            double(*)(int dim, double myArray[]) foo )
```

Runs the instance of the Differential evolutionary algorithm

Parameters

strategy	a integer value representing current mutation strategy being ran
min	a double representing the minimum of the range
max	a double representing the maximum of the range
name	a string representing name of the current fitness function
foo	a pointer to the current fitness function

2.2.1.11 select()

```
void DiffAlgorithm::select (
            int index,
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Confirms that the two selected parents are not the same parent

Parameters

index	a integer value representing working index
population	a 2D double array representing current population
foo	a pointer to the current fitness function

2.2.1.12 setChild1()

```
void DiffAlgorithm::setChild1 (
            double * childone )
```

Sets the first child

Parameters

childone	an array of type double

2.2.1.13 setChild2()

```
void DiffAlgorithm::setChild2 (
            double * childtwo )
```

Sets the second child

Parameters

childtwo	an array of type double

2.2.1.14 setParent1()

```
void DiffAlgorithm::setParent1 (
```

```
            double * parentone )
```

Sets the the first parent

Parameters

parentone	an array of type double

2.2.1.15 setParent2()

```
void DiffAlgorithm::setParent2 (
            double * parenttwo )
```

Sets the second parent

Parameters

parenttwo	an array of type double

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

- DiffAlgorithm.h
- DiffAlgorithm.cpp

2.3 GeneticAlgorithm Class Reference

Public Member Functions

- double runGeneticAlgorithm (double min, double max, std::string name, double(∗foo)(int dim, double my↩ Array[ ]))
- void reduce (double ∗∗population, double ∗∗newpopulation, double EliteSN, double(∗foo)(int dim, double myArray[ ]))
- void select (double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- void mutate (double ∗child1, double ∗child2)
- void crossover (double ∗parent1, double ∗parent2, double CR)
- void fileReader ()
- int ∗ sortbyCost (double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- int selectParent (double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- void getFitness (double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- void getBestSolution (double ∗∗population, double(∗foo)(int dim, double myArray[ ]))
- double ∗ getParent1 ()
- double ∗ getParent2 ()
- double ∗ getChild1 ()
- double ∗ getChild2 ()
- void setParent1 (double parentone[ ])
- void setParent2 (double parenttwo[ ])
- void setChild1 (double childone[ ])
- void setChild2 (double childtwo[ ])

Public Attributes

- Population * **pop** = new Population()

2.3.1 Member Function Documentation

2.3.1.1 crossover()

```
void GeneticAlgorithm::crossover (
            double * parent1,
            double * parent2,
            double CR )
```

performes crossover for the two parents resulting in two children

Parameters

parent1	a double array representing the first parent
parent2	a double array representing the second parent
CR	a double representing the crossover rate

2.3.1.2 fileReader()

```
void GeneticAlgorithm::fileReader ( )
```

Reads from the input file to initialize and set the private feilds of the population class to the values for Genetic Algorithm

2.3.1.3 getBestSolution()

```
void GeneticAlgorithm::getBestSolution (
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Calculates and sets the fittest member of the population

Parameters

population	a 2D double array representing current population
foo	a pointer to the current fitness function

2.3.1.4 getChild1()

```
double * GeneticAlgorithm::getChild1 ( )
```

Returns the first child

Returns

The first child

2.3.1.5 getChild2()

```
double * GeneticAlgorithm::getChild2 ( )
```

Returns the second child

Returns

the second child

2.3.1.6 getFitness()

```
void GeneticAlgorithm::getFitness (
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Calculates the fitness aray for a given population

Parameters

population	a 2D double array representing current population
foo	a pointer to the current fitness function

Returns

an array containing the fitess of each member of the population

2.3.1.7 getParent1()

```
double * GeneticAlgorithm::getParent1 ( )
```

Returns the first parent

Returns

the first parent

2.3.1.8 getParent2()

```
double * GeneticAlgorithm::getParent2 ( )
```

Returns the second parent

Returns

the second parent

2.3.1.9 mutate()

```
void GeneticAlgorithm::mutate (
            double * child1,
            double * child2 )
```

Mutates several genes of each child

Parameters

child1	a double array representing the first child
child2	a double array representing the second child

2.3.1.10 reduce()

```
void GeneticAlgorithm::reduce (
            double ** population,
            double ** newpopulation,
            double EliteSN,
            double(*)(int dim, double myArray[]) foo )
```

Combines and reduces the old population with the new one.

Parameters

population	a 2D double array representing old population
newpopulation	a 2D double array representing new population after mutation and crossover
EliteSN	a double that represents the number of offspring to move to the next generation
foo	a pointer to the current fitness function

2.3.1.11 runGeneticAlgorithm()

```
double GeneticAlgorithm::runGeneticAlgorithm (
            double min,
            double max,
            std::string name,
            double(*)(int dim, double myArray[]) foo )
```

Runs the instance of the Genetic Algorithm class

Parameters

min	a double representing the minimum of the range
max	a double representing the maximum of the range
name	a string value representing the name of the fitness function being ran
foo	a pointer to the current fitness function

Returns

the best fitness found

2.3.1.12 select()

```
void GeneticAlgorithm::select (
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Confirms that the two selected parents are not the same parent

Parameters

population	a 2D double array representing current population
foo	a pointer to the current fitness function

2.3.1.13 selectParent()

```
int GeneticAlgorithm::selectParent (
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Selects the two parrents for crossover

Parameters

population	a 2D double array representing current population
foo	a pointer to the current fitness function

Returns

an integer corisponding to the index of the parent

2.3.1.14 setChild1()

```
void GeneticAlgorithm::setChild1 (
            double childone[] )
```

Sets the first child

Parameters

childone	an array of type double

2.3.1.15 setChild2()

```
void GeneticAlgorithm::setChild2 (
            double childtwo[] )
```

Sets the second child

Parameters

childtwo	an array of type double

2.3.1.16 setParent1()

```
void GeneticAlgorithm::setParent1 (
            double parentone[] )
```

Sets the the first parent

Parameters

parentone	an array of type double

2.3.1.17 setParent2()

```
void GeneticAlgorithm::setParent2 (
            double parenttwo[] )
```

Sets the second parent

Parameters

parenttwo	an array of type double

2.3.1.18 sortbyCost()

```
int * GeneticAlgorithm::sortbyCost (
            double ** population,
            double(*)(int dim, double myArray[]) foo )
```

Combines and reduces the old population with the new one.

Parameters

population	a 2D double array representing current population
foo	a pointer to the current fitness function

Returns

an array containing the sorted indicies

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

- GeneticAlgorithm.h
- GeneticAlgorithm.cpp

2.4 Population Class Reference

Public Member Functions

- void generatePopulation (int popSize, int dim, double min, double max)
- double * solveFitness (int dim, double **Population, int popSize, double min, double max, double(*foo)(int dim, double myArray[]))
- double * normalize (int dim, double **Population, int popSize, double min, double max, double(*foo)(int dim, double myArray[]))
- double * createArray (int dim, double min, double max)

- double ∗∗ getPopulation ()
- double ∗∗ getNewPopulation ()
- double ∗ getFitness ()
- double ∗ getBestValue ()
- double ∗ getCost ()
- double ∗ getProb ()
- double getMin ()
- double getMax ()
- double getTotalFitness ()
- double getBestFitness ()
- double getMutationRate ()
- double getMutationRange ()
- double getCrossoverRate ()
- double getMutPrec ()
- double getEliteRate ()
- double getScaleFactor ()
- int getPopSize ()
- int getDimensions ()
- int getIterations ()
- void setNewPopulation (double ∗∗population)
- void setCrossoverRate (double crossRate)
- void setPopulation (double ∗∗population)
- void setMutationRange (double mutRange)
- void setTotalFitness (double TotalFit)
- void setMutPrec (double mutationPrec)
- void setDimensions (int dimensions1)
- void setbestFitness (double bestFit)
- void setMutaionRate (double mutRate)
- void setbestValue (double ∗bestVal)
- void setScaleFactor (double scale)
- void setCost (double ∗costtemp)
- void setPopSize (int popsize)
- void setEliteRate (double ER)
- void setIterations (int iter)
- void setMin (double minimum)
- void setMax (double maximum)
- void setFitness (double ∗fit)
- void setProb (double ∗Prob)

2.4.1 Member Function Documentation

2.4.1.1 createArray()

```
double * Population::createArray (
            int dim,
            double min,
            double max )
```

Returns a Array of type double containing dim random values randomly generated between min and max by the createArray function.

Parameters

dim	an integer representing the number of dimensions
min	a double representing the minimum of the range
max	a double representing the maximum of the range

Returns

an array of dim doubles between min and max

2.4.1.2 generatePopulation()

```
void Population::generatePopulation (
            int popSize,
            int dim,
            double min,
            double max )
```

initializer functions- initializes the various parts of the population

Returns a two dimensional array of type double containing popsize arrays of dim elements, between the range of min to max.

Parameters

popSize	an integer representing the size of the population
dim	an integer representing the number of dimensions
min	a double representing the minimum of the range
max	a double representing the maximum of the range

Returns

an 2D array of doubles of size dim by popSize representing a new population

2.4.1.3 getBestFitness()

```
double Population::getBestFitness ( )
```

Returns the best fitness

Returns

The current best fitness

2.4.1.4 getBestValue()

```
double * Population::getBestValue ( )
```

Returns the Best Value

Returns

The current best value

2.4.1.5 getCost()

```
double * Population::getCost ( )
```

Returns the cost array

Returns

The current cost array

2.4.1.6 getCrossoverRate()

```
double Population::getCrossoverRate ( )
```

Returns the Crossover Rate

Returns

The current CrossoverRate

2.4.1.7 getDimensions()

```
int Population::getDimensions ( )
```

Returns the number of dimensions

Returns

The current number of dimensions

2.4.1.8 getEliteRate()

```
double Population::getEliteRate ( )
```

Returns the Elitism Rate

Returns

The current Elitism Rate

2.4.1.9 getFitness()

```
double * Population::getFitness ( )
```

Returns the fitness array

Returns

The current fitness array

2.4.1.10 getIterations()

```
int Population::getIterations ( )
```

Returns the number of iterations

Returns

The current number of iterations

2.4.1.11 getMax()

```
double Population::getMax ( )
```

Returns the maximum

Returns

The current maximum

2.4.1.12 getMin()

```
double Population::getMin ( )
```

Returns the minimum

Returns

The current minimum

2.4.1.13 getMutationRange()

```
double Population::getMutationRange ( )
```

Returns the Mutation Range

Returns

The current Mutation Range

2.4.1.14 getMutationRate()

```
double Population::getMutationRate ( )
```

Returns the Mutation Rate

Returns

The current Mutation Rate

2.4.1.15 getMutPrec()

```
double Population::getMutPrec ( )
```

Returns the Mutation Precision

Returns

The current Mutation Precision

2.4.1.16 getNewPopulation()

```
double ** Population::getNewPopulation ( )
```

Returns the new population

Returns

The new population

2.4.1.17 getPopSize()

```
int Population::getPopSize ( )
```

Returns the Population Size

Returns

The current population size

2.4.1.18 getPopulation()

```
double ** Population::getPopulation ( )
```

Get Methods - gets values of private feilds

Returns the Population

Returns

The current population

2.4.1.19 getProb()

```
double * Population::getProb ( )
```

Returns the Probability array

Returns

The current Probability array

2.4.1.20 getScaleFactor()

```
double Population::getScaleFactor ( )
```

Returns the Scaleing Factor

Returns

The current Scaleing Factor

2.4.1.21 getTotalFitness()

```
double Population::getTotalFitness ( )
```

Returns the total fitness

Returns

The current total fitness

2.4.1.22 normalize()

```
double * Population::normalize (
            int dim,
            double ** population,
            int popSize,
            double min,
            double max,
            double(*)(int dim, double myArray[]) foo )
```

Returns a array of type double containing the normalized values of the fitnesses of each chromosome in the population.

Parameters

dim	an integer representing the number of dimensions
population	2D array of type double containing the current population
popSize	an integer representing the size of the population
min	a double representing the minimum of the range
max	a double representing the maximum of the range
foo	a pointer to the current fitness function

Returns

an array of type double containing normalized fitness values

2.4.1.23 setbestFitness()

```
void Population::setbestFitness (
            double bestFit )
```

Sets the best fitness

Parameters

bestFit	a value of type double

2.4.1.24 setbestValue()

```
void Population::setbestValue (
            double * bestVal )
```

Sets the Best Value

Parameters

bestVal	an array of type double

2.4.1.25 setCost()

```
void Population::setCost (
            double * costtemp )
```

Sets the Cost Array

Parameters

costtemp	an array of type double

2.4.1.26 setCrossoverRate()

```
void Population::setCrossoverRate (
            double crossRate )
```

Sets the Crossover Rate

Parameters

crossRate	a value of type double

2.4.1.27 setDimensions()

```
void Population::setDimensions (
            int dimensions1 )
```

Sets the number of dimensions

Parameters

dimensions1	a value of type integer

2.4.1.28 setEliteRate()

```
void Population::setEliteRate (
            double ER )
```

Sets the Elitism Rate

Parameters

ER	a value of type double

2.4.1.29 setFitness()

```
void Population::setFitness (
            double * fit )
```

Sets the fitness array

Parameters

fit	an array of type double

2.4.1.30 setIterations()

```
void Population::setIterations (
            int iter )
```

Sets the number of iterations

Parameters

iter	a value of type integer

2.4.1.31 setMax()

```
void Population::setMax (
            double maximum )
```

Sets the maximum of the current fitness function

Parameters

maximum	a value of type double

2.4.1.32 setMin()

```
void Population::setMin (
            double minimum )
```

Sets the minimum of the current fitness function

Parameters

minimum	a value of type double

2.4.1.33 setMutaionRate()

```
void Population::setMutaionRate (
            double mutRate )
```

Sets the Mutation Rate

Parameters

mutRate	a value of type double

2.4.1.34 setMutationRange()

```
void Population::setMutationRange (
            double mutRange )
```

Sets the Mutation Range

Parameters

mutRange	a value of type double

2.4.1.35 setMutPrec()

```
void Population::setMutPrec (
            double mutationPrec1 )
```

Sets the Mutation Precision

Parameters

mutationPrec1	a value of type double

2.4.1.36 setNewPopulation()

```
void Population::setNewPopulation (
            double ** population )
```

Set Methods - sets values of private feilds

Sets the New Population

Parameters

population	a 2D array of type double

2.4.1.37 setPopSize()

```
void Population::setPopSize (
            int popsize )
```

Sets the Population Size

Parameters

popsize	a value of type integer

2.4.1.38 setPopulation()

```
void Population::setPopulation (
            double ** population )
```

Sets the Population

Parameters

population	a 2D array of type double

2.4.1.39 setProb()

```
void Population::setProb (
            double * Prob )
```

Sets the Probability of each member of the population

Parameters

Prob	an array of type double

2.4.1.40 setScaleFactor()

```
void Population::setScaleFactor (
            double scale )
```

Sets the Scaleing Factor

Parameters

scale	a value of type double

2.4.1.41 setTotalFitness()

```
void Population::setTotalFitness (
            double TotalFit )
```

Sets the total fitness

Parameters

TotalFit	a value of type double

2.4.1.42 solveFitness()

```
double * Population::solveFitness (
            int dim,
            double ** Population,
            int popSize,
            double min,
            double max,
            double(*)(int dim, double myArray[]) foo )
```

Returns a array of type double containing dim elements representing the fitnesses of each member of the population.

Parameters

dim	an integer representing the number of dimensions
population	an 2D double array containing the current population
popSize	an integer representing the size of the population
min	a double representing the minimum of the range
max	a double representing the maximum of the range
foo	a pointer to the current fitness function

Returns

an 2D array of doubles of size dim by popSize representing a new population

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

- Population.h
- Population.cpp

2.5 run Class Reference

Public Member Functions

- void runProject3 ()

2.5.1 Member Function Documentation

2.5.1.1 runProject3()

```
void run::runProject3 ( )
```

Runs instances of the Genetic and differential Evolution Algorithms. Runs each algorithms for all 15 fitness functions.

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

- run.h
- run.cpp

2.6 strs Struct Reference

Public Attributes

- double **value**
- int **index**

The documentation for this struct was generated from the following file:

- GeneticAlgorithm.h

Index