

## Project 3 Evolutionary Algorithms

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>Class Documentation</b>	<b>3</b>
2.1	benchmarkfunctions Struct Reference . . . . .	3
2.2	DiffAlgorithm Class Reference . . . . .	3
2.2.1	Member Function Documentation . . . . .	4
2.2.1.1	bincrossover() . . . . .	4
2.2.1.2	crossover() . . . . .	4
2.2.1.3	fileReader() . . . . .	4
2.2.1.4	getBestSolution() . . . . .	5
2.2.1.5	getChild1() . . . . .	6
2.2.1.6	getChild2() . . . . .	6
2.2.1.7	getParent1() . . . . .	6
2.2.1.8	getParent2() . . . . .	7
2.2.1.9	mutate() . . . . .	7
2.2.1.10	runDiffAlgorithm() . . . . .	7
2.2.1.11	select() . . . . .	8
2.2.1.12	setChild1() . . . . .	8
2.2.1.13	setChild2() . . . . .	8
2.2.1.14	setParent1() . . . . .	8
2.2.1.15	setParent2() . . . . .	9
2.3	GeneticAlgorithm Class Reference . . . . .	9
2.3.1	Member Function Documentation . . . . .	10

2.3.1.1	<a href="#">crossover()</a>	10
2.3.1.2	<a href="#">fileReader()</a>	10
2.3.1.3	<a href="#">getBestSolution()</a>	10
2.3.1.4	<a href="#">getChild1()</a>	11
2.3.1.5	<a href="#">getChild2()</a>	11
2.3.1.6	<a href="#">getFitness()</a>	11
2.3.1.7	<a href="#">getParent1()</a>	11
2.3.1.8	<a href="#">getParent2()</a>	12
2.3.1.9	<a href="#">mutate()</a>	12
2.3.1.10	<a href="#">reduce()</a>	12
2.3.1.11	<a href="#">runGeneticAlgorithm()</a>	13
2.3.1.12	<a href="#">select()</a>	13
2.3.1.13	<a href="#">selectParent()</a>	13
2.3.1.14	<a href="#">setChild1()</a>	14
2.3.1.15	<a href="#">setChild2()</a>	14
2.3.1.16	<a href="#">setParent1()</a>	14
2.3.1.17	<a href="#">setParent2()</a>	15
2.3.1.18	<a href="#">sortByCost()</a>	15
2.4	<a href="#">Population Class Reference</a>	15
2.4.1	<a href="#">Member Function Documentation</a>	16
2.4.1.1	<a href="#">createArray()</a>	16
2.4.1.2	<a href="#">generatePopulation()</a>	17
2.4.1.3	<a href="#">getBestFitness()</a>	17
2.4.1.4	<a href="#">getBestValue()</a>	18
2.4.1.5	<a href="#">getCost()</a>	18
2.4.1.6	<a href="#">getCrossoverRate()</a>	18
2.4.1.7	<a href="#">getDimensions()</a>	18
2.4.1.8	<a href="#">getEliteRate()</a>	19
2.4.1.9	<a href="#">getFitness()</a>	19
2.4.1.10	<a href="#">getIterations()</a>	19

2.4.1.11	<a href="#">getMax()</a>	19
2.4.1.12	<a href="#">getMin()</a>	20
2.4.1.13	<a href="#">getMutationRange()</a>	20
2.4.1.14	<a href="#">getMutationRate()</a>	20
2.4.1.15	<a href="#">getMutPrec()</a>	20
2.4.1.16	<a href="#">getNewPopulation()</a>	21
2.4.1.17	<a href="#">getPopSize()</a>	21
2.4.1.18	<a href="#">getPopulation()</a>	21
2.4.1.19	<a href="#">getProb()</a>	21
2.4.1.20	<a href="#">getScaleFactor()</a>	22
2.4.1.21	<a href="#">getTotalFitness()</a>	22
2.4.1.22	<a href="#">normalize()</a>	22
2.4.1.23	<a href="#">setbestFitness()</a>	23
2.4.1.24	<a href="#">setbestValue()</a>	23
2.4.1.25	<a href="#">setCost()</a>	23
2.4.1.26	<a href="#">setCrossoverRate()</a>	23
2.4.1.27	<a href="#">setDimensions()</a>	24
2.4.1.28	<a href="#">setEliteRate()</a>	24
2.4.1.29	<a href="#">setFitness()</a>	24
2.4.1.30	<a href="#">setIterations()</a>	25
2.4.1.31	<a href="#">setMax()</a>	25
2.4.1.32	<a href="#">setMin()</a>	25
2.4.1.33	<a href="#">setMutationRate()</a>	25
2.4.1.34	<a href="#">setMutationRange()</a>	26
2.4.1.35	<a href="#">setMutPrec()</a>	26
2.4.1.36	<a href="#">setNewPopulation()</a>	26
2.4.1.37	<a href="#">setPopSize()</a>	27
2.4.1.38	<a href="#">setPopulation()</a>	27
2.4.1.39	<a href="#">setProb()</a>	27
2.4.1.40	<a href="#">setScaleFactor()</a>	27
2.4.1.41	<a href="#">setTotalFitness()</a>	28
2.4.1.42	<a href="#">solveFitness()</a>	28
2.5	<a href="#">run Class Reference</a>	29
2.5.1	<a href="#">Member Function Documentation</a>	29
2.5.1.1	<a href="#">runProject3()</a>	29
2.6	<a href="#">strs Struct Reference</a>	29



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">benchmarkfunctions</a>	3
<a href="#">DiffAlgorithm</a>	3
<a href="#">GeneticAlgorithm</a>	9
<a href="#">Population</a>	15
<a href="#">run</a>	29
<a href="#">strs</a>	29





## 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 `binrossover()`

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

Performs binomial crossover between the parent and the noisy vector

#### Parameters

<i>CR</i>	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

<i>index</i>	an integer value of the parents index in the population
<i>CR</i>	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**

<i>population</i>	a 2D double array representing current population
<i>foo</i>	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

<i>index</i>	a integer value representing working index
<i>strategy</i>	a integer value representing current mutation strategy being ran
<i>population</i>	a 2D double array representing current population
<i>min</i>	a double representing the minimum of the range
<i>max</i>	a double representing the maximum of the range
<i>foo</i>	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

<i>strategy</i>	a integer value representing current mutation strategy being ran
<i>min</i>	a double representing the minimum of the range
<i>max</i>	a double representing the maximum of the range
<i>name</i>	a string representing name of the current fitness function
<i>foo</i>	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

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

#### 2.2.1.12 setChild1()

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

Sets the first child

##### Parameters

<i>childone</i>	an array of type double
-----------------	-------------------------

#### 2.2.1.13 setChild2()

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

Sets the second child

##### Parameters

<i>childtwo</i>	an array of type double
-----------------	-------------------------

#### 2.2.1.14 setParent1()

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

```
double * parentone )
```

Sets the the first parent

#### Parameters

<i>parentone</i>	an array of type double
------------------	-------------------------

#### 2.2.1.15 setParent2()

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

Sets the second parent

#### Parameters

<i>parenttwo</i>	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 myArray[]))
- 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 )
```

performs crossover for the two parents resulting in two children

#### Parameters

<i>parent1</i>	a double array representing the first parent
<i>parent2</i>	a double array representing the second parent
<i>CR</i>	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

<i>population</i>	a 2D double array representing current population
<i>foo</i>	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 array for a given population

##### Parameters

<i>population</i>	a 2D double array representing current population
<i>foo</i>	a pointer to the current fitness function

##### Returns

an array containing the fitness 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**

<i>child1</i>	a double array representing the first child
<i>child2</i>	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**

<i>population</i>	a 2D double array representing old population
<i>newpopulation</i>	a 2D double array representing new population after mutation and crossover
<i>EliteSN</i>	a double that represents the number of offspring to move to the next generation
<i>foo</i>	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

<i>min</i>	a double representing the minimum of the range
<i>max</i>	a double representing the maximum of the range
<i>name</i>	a string value representing the name of the fitness function being ran
<i>foo</i>	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

<i>population</i>	a 2D double array representing current population
<i>foo</i>	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**

<i>population</i>	a 2D double array representing current population
<i>foo</i>	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**

<i>childone</i>	an array of type double
-----------------	-------------------------

**2.3.1.15 setChild2()**

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

Sets the second child

**Parameters**

<i>childtwo</i>	an array of type double
-----------------	-------------------------

**2.3.1.16 setParent1()**

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

Sets the the first parent

**Parameters**

<i>parentone</i>	an array of type double
------------------	-------------------------

### 2.3.1.17 setParent2()

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

Sets the second parent

#### Parameters

<i>parenttwo</i>	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

<i>population</i>	a 2D double array representing current population
<i>foo</i>	a pointer to the current fitness function

#### Returns

an array containing the sorted indices

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**

<i>dim</i>	an integer representing the number of dimensions
<i>min</i>	a double representing the minimum of the range
<i>max</i>	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**

<i>popSize</i>	an integer representing the size of the population
<i>dim</i>	an integer representing the number of dimensions
<i>min</i>	a double representing the minimum of the range
<i>max</i>	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**

<i>dim</i>	an integer representing the number of dimensions
<i>population</i>	2D array of type double containing the current population
<i>popSize</i>	an integer representing the size of the population
<i>min</i>	a double representing the minimum of the range
<i>max</i>	a double representing the maximum of the range
<i>foo</i>	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

<i>bestFit</i>	a value of type double
----------------	------------------------

#### 2.4.1.24 setbestValue()

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

Sets the Best Value

##### Parameters

<i>bestVal</i>	an array of type double
----------------	-------------------------

#### 2.4.1.25 setCost()

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

Sets the Cost Array

##### Parameters

<i>costtemp</i>	an array of type double
-----------------	-------------------------

#### 2.4.1.26 setCrossoverRate()

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

Sets the Crossover Rate

## Parameters

<i>crossRate</i>	a value of type double
------------------	------------------------

## 2.4.1.27 setDimensions()

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

Sets the number of dimensions

## Parameters

<i>dimensions1</i>	a value of type integer
--------------------	-------------------------

## 2.4.1.28 setEliteRate()

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

Sets the Elitism Rate

## Parameters

<i>ER</i>	a value of type double
-----------	------------------------

## 2.4.1.29 setFitness()

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

Sets the fitness array

## Parameters

<i>fit</i>	an array of type double
------------	-------------------------

#### 2.4.1.30 setIterations()

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

Sets the number of iterations

##### Parameters

<i>iter</i>	a value of type integer
-------------	-------------------------

#### 2.4.1.31 setMax()

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

Sets the maximum of the current fitness function

##### Parameters

<i>maximum</i>	a value of type double
----------------	------------------------

#### 2.4.1.32 setMin()

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

Sets the minimum of the current fitness function

##### Parameters

<i>minimum</i>	a value of type double
----------------	------------------------

#### 2.4.1.33 setMutationRate()

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

Sets the Mutation Rate

## Parameters

<i>mutRate</i>	a value of type double
----------------	------------------------

**2.4.1.34 setMutationRange()**

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

Sets the Mutation Range

## Parameters

<i>mutRange</i>	a value of type double
-----------------	------------------------

**2.4.1.35 setMutPrec()**

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

Sets the Mutation Precision

## Parameters

<i>mutationPrec1</i>	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

<i>population</i>	a 2D array of type double
-------------------	---------------------------



#### 2.4.1.37 setPopSize()

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

Sets the [Population](#) Size

##### Parameters

<i>popsiz</i>	a value of type integer
---------------	-------------------------

#### 2.4.1.38 setPopulation()

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

Sets the [Population](#)

##### Parameters

<i>population</i>	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

<i>Prob</i>	an array of type double
-------------	-------------------------

#### 2.4.1.40 setScaleFactor()

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

Sets the Scaleing Factor

## Parameters

<i>scale</i>	a value of type double
--------------	------------------------

2.4.1.41 `setTotalFitness()`

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

Sets the total fitness

## Parameters

<i>TotalFit</i>	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

<i>dim</i>	an integer representing the number of dimensions
<i>population</i>	an 2D double array containing the current population
<i>popSize</i>	an integer representing the size of the population
<i>min</i>	a double representing the minimum of the range
<i>max</i>	a double representing the maximum of the range
<i>foo</i>	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

- benchmarkfunctions, [3](#)
- bincrossover
  - DiffAlgorithm, [4](#)
- createArray
  - Population, [16](#)
- crossover
  - DiffAlgorithm, [4](#)
  - GeneticAlgorithm, [10](#)
- DiffAlgorithm, [3](#)
  - bincrossover, [4](#)
  - crossover, [4](#)
  - fileReader, [4](#)
  - getBestSolution, [4](#)
  - getChild1, [6](#)
  - getChild2, [6](#)
  - getParent1, [6](#)
  - getParent2, [6](#)
  - mutate, [7](#)
  - runDiffAlgorithm, [7](#)
  - select, [8](#)
  - setChild1, [8](#)
  - setChild2, [8](#)
  - setParent1, [8](#)
  - setParent2, [9](#)
- fileReader
  - DiffAlgorithm, [4](#)
  - GeneticAlgorithm, [10](#)
- generatePopulation
  - Population, [17](#)
- GeneticAlgorithm, [9](#)
  - crossover, [10](#)
  - fileReader, [10](#)
  - getBestSolution, [10](#)
  - getChild1, [10](#)
  - getChild2, [11](#)
  - getFitness, [11](#)
  - getParent1, [11](#)
  - getParent2, [12](#)
  - mutate, [12](#)
  - reduce, [12](#)
  - runGeneticAlgorithm, [13](#)
  - select, [13](#)
  - selectParent, [13](#)
  - setChild1, [14](#)
  - setChild2, [14](#)
  - setParent1, [14](#)
  - setParent2, [15](#)
  - sortByCost, [15](#)
- getBestFitness
  - Population, [17](#)
- getBestSolution
  - DiffAlgorithm, [4](#)
  - GeneticAlgorithm, [10](#)
- getBestValue
  - Population, [17](#)
- getChild1
  - DiffAlgorithm, [6](#)
  - GeneticAlgorithm, [10](#)
- getChild2
  - DiffAlgorithm, [6](#)
  - GeneticAlgorithm, [11](#)
- getCost
  - Population, [18](#)
- getCrossoverRate
  - Population, [18](#)
- getDimensions
  - Population, [18](#)
- getEliteRate
  - Population, [18](#)
- getFitness
  - GeneticAlgorithm, [11](#)
  - Population, [19](#)
- getIterations
  - Population, [19](#)
- getMax
  - Population, [19](#)
- getMin
  - Population, [19](#)
- getMutPrec
  - Population, [20](#)
- getMutationRange
  - Population, [20](#)
- getMutationRate
  - Population, [20](#)
- getNewPopulation
  - Population, [20](#)
- getParent1
  - DiffAlgorithm, [6](#)
  - GeneticAlgorithm, [11](#)
- getParent2
  - DiffAlgorithm, [6](#)
  - GeneticAlgorithm, [12](#)
- getPopSize
  - Population, [21](#)
- getPopulation

- Population, 21
- getProb
  - Population, 21
- getScaleFactor
  - Population, 21
- getTotalFitness
  - Population, 22
- mutate
  - DiffAlgorithm, 7
  - GeneticAlgorithm, 12
- normalize
  - Population, 22
- Population, 15
  - createArray, 16
  - generatePopulation, 17
  - getBestFitness, 17
  - getBestValue, 17
  - getCost, 18
  - getCrossoverRate, 18
  - getDimensions, 18
  - getEliteRate, 18
  - getFitness, 19
  - getIterations, 19
  - getMax, 19
  - getMin, 19
  - getMutPrec, 20
  - getMutationRange, 20
  - getMutationRate, 20
  - getNewPopulation, 20
  - getPopSize, 21
  - getPopulation, 21
  - getProb, 21
  - getScaleFactor, 21
  - getTotalFitness, 22
  - normalize, 22
  - setCost, 23
  - setCrossoverRate, 23
  - setDimensions, 24
  - setEliteRate, 24
  - setFitness, 24
  - setIterations, 24
  - setMax, 25
  - setMin, 25
  - setMutPrec, 26
  - setMutationRate, 25
  - setMutationRange, 26
  - setNewPopulation, 26
  - setPopSize, 26
  - setPopulation, 27
  - setProb, 27
  - setScaleFactor, 27
  - setTotalFitness, 28
  - setbestFitness, 22
  - setbestValue, 23
  - solveFitness, 28
- reduce
  - GeneticAlgorithm, 12
- run, 29
  - runProject3, 29
- runDiffAlgorithm
  - DiffAlgorithm, 7
- runGeneticAlgorithm
  - GeneticAlgorithm, 13
- runProject3
  - run, 29
- select
  - DiffAlgorithm, 8
  - GeneticAlgorithm, 13
- selectParent
  - GeneticAlgorithm, 13
- setChild1
  - DiffAlgorithm, 8
  - GeneticAlgorithm, 14
- setChild2
  - DiffAlgorithm, 8
  - GeneticAlgorithm, 14
- setCost
  - Population, 23
- setCrossoverRate
  - Population, 23
- setDimensions
  - Population, 24
- setEliteRate
  - Population, 24
- setFitness
  - Population, 24
- setIterations
  - Population, 24
- setMax
  - Population, 25
- setMin
  - Population, 25
- setMutPrec
  - Population, 26
- setMutationRate
  - Population, 25
- setMutationRange
  - Population, 26
- setNewPopulation
  - Population, 26
- setParent1
  - DiffAlgorithm, 8
  - GeneticAlgorithm, 14
- setParent2
  - DiffAlgorithm, 9
  - GeneticAlgorithm, 15
- setPopSize
  - Population, 26
- setPopulation
  - Population, 27
- setProb
  - Population, 27
- setScaleFactor

---

- Population, [27](#)
- setTotalFitness
  - Population, [28](#)
- setbestFitness
  - Population, [22](#)
- setbestValue
  - Population, [23](#)
- solveFitness
  - Population, [28](#)
- sortByCost
  - GeneticAlgorithm, [15](#)
- strs, [29](#)