My Project

Generated by Doxygen 1.8.15

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 Package optimazation.pkg1

Classes

- class Algorithm
- class CreateCSV

  CreateCSV class.
- class CreateMatrix

  CreateMatrix class.
- class Firefly
- class Functioning

  Functioning class.
- class Functions
- class HS
- class Main

  Optimization1 class.
- class MTRandom
- class PSO

4.1.1 Detailed Description

Author

JuneYeob Lee(24629603)

-

Date

5/17/2019 Contact:Leej @cwu.edu Created on: 5/17/2019

Chapter 5

Class Documentation

5.1 optimazation.pkg1.Algorithm Class Reference

Public Member Functions

- Algorithm (int rows, int colums, int functNum)

5.1.1 Detailed Description

Algorithm class has 11 methods for Genetic Algorithm and Differential Evolution Algorithm.

-

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Algorithm()

```
optimazation.pkg1.Algorithm.Algorithm (
            int rows,
            int colums,
            int functNum )
```

- The constructor for Genetic Algorithm and Differential Algorithm. Create initialize populations. and calculate by function which has been chosen.

Parameters

rows	number of populations
colums	size of Dimension
functNum	implement function number

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

- Algorithm.java

5.2 optimazation.pkg1.CreateCSV Class Reference

CreateCSV class.

Public Member Functions

- void CreateCsv0 (double[ ][ ] a, int colums, String C) throws IOException
- void CreateCsv1 (double[ ][ ]a, String C) throws IOException

5.2.1 Detailed Description

CreateCSV class.

Author

JuneYeob Lee(2462 9603)

This class is for creating CSV file about results

Date

4/5/2019 !\Contact:Leej @cwu.edu !\Created on: 3/28/2019

5.2.2 Member Function Documentation

5.2.2.1 CreateCsv0()

```
void optimazation.pkg1.CreateCSV.CreateCsv0 (
            double a[][],
            int colums,
            String C ) throws IOException
```

CreateCsv0 method for creating the csv file of the best Vectors

Parameters

a	(a for the best solution);
colums	(the number of colums)
C	

Exceptions

IOException	

5.2.2.2 CreateCsv1()

```
void optimazation.pkg1.CreateCSV.CreateCsv1 (
            doublea [][],
            String C ) throws IOException
```

CreateCsv method to create the csv file for best solutions

Parameters

a(the	best solutions 2d array length should be 18 size should be the number of iteration)
C	(File name)

Exceptions

IOException	

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

- CreateCSV.java

5.3 optimazation.pkg1.CreateMatrix Class Reference

CreateMatrix class.

Public Member Functions

- CreateMatrix (int a, int b)
  A constructor.
- void create (int length, int size)
- void add (double min, double max)

5.3.1 Detailed Description

CreateMatrix class.

Author

JuneYeob Lee(24629603)

-

Date

4/5/2019 \Contact:Leej @cwu.edu \Created on: 3/28/2019

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CreateMatrix()

```
optimazation.pkg1.CreateMatrix.CreateMatrix (
            int a,
            int b )
```

A constructor.

Parameters

int	a the first argument(rows)
int	b the second argument(colums)

5.3.3 Member Function Documentation

5.3.3.1 add()

```
void optimazation.pkg1.CreateMatrix.add (
            double min,
            double max )
```

A add method !this method will fill out input with random numbers(this method will create 7 files which

Parameters

min	lower bound for input
max	upper bound for input

5.3.3.2 create()

```
void optimazation.pkg1.CreateMatrix.create (
            int length,
            int size )
```

A create method /∗! this method will create input 2d arrays with all 0.0

Parameters

length	length of input matrix
size	size of input matrix

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

- CreateMatrix.java

5.4 optimazation.pkg1.Firefly Class Reference

Public Member Functions

- Firefly (int functionNum, int NP, int DIM)
- void caldistance (int i, int j)
- void calattract (int i, int j)
- void equation4 (int i, int j)
- void evaluate ()
- void calc ()

5.4.1 Detailed Description

This class is FireFly algorithm class. This class to implement FireFlies algorithm

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Firefly()

```
optimazation.pkg1.Firefly.Firefly (
            int functionNum,
            int NP,
            int DIM )
```

constructor which parameters are function number and number of population and size of vector.

Parameters

functionNum	implement function number
NP	number of fireflies
DIM	size of input dimension

5.4.3 Member Function Documentation

5.4.3.1 calattract()

```
void optimazation.pkg1.Firefly.calattract (
            int i,
            int j )
```

Calattract method to calculate attractivness

Parameters

i	index number of row
j	index number of colum

5.4.3.2 calc()

```
void optimazation.pkg1.Firefly.calc ( )
```

calc method to calculate the updated fireflies and update new gbest value.

5.4.3.3 caldistance()

```
void optimazation.pkg1.Firefly.caldistance (
            int i,
            int j )
```

caldistance method is to calculate the distance between two fireflies.

Parameters

i	
j	

5.4.3.4 equation4()

```
void optimazation.pkg1.Firefly.equation4 (
            int i,
            int j )
```

equation4 to implement equation4

Parameters

i	index number of row
j	index number of colum

5.4.3.5 evaluate()

```
void optimazation.pkg1.Firefly.evaluate ( )
```

evalute method. evalute and update the worst firefly in populations

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

- Firefly.java

5.5 optimazation.pkg1.Functioning Class Reference

Functioning class.

Public Member Functions

- double [ ] cal (int functionNum, int length, int size, double[ ][ ] a)

5.5.1 Detailed Description

Functioning class.

Author

JuneYeob Lee(24629603) \This Functioning class has made to implement all 18 functions by ranges.

-

Date

4/5/2019 Contact:Leej @cwu.edu Created on: 3/28/2019

5.5.2 Member Function Documentation

5.5.2.1 cal()

```
double [] optimazation.pkg1.Functioning.cal (
            int functionNum,
            int length,
            int size,
            double a[][] )
```

Cal method for calculate 18 functions about 2D Matrix

Parameters

functionNum	Function number
length	number of populations
size	size of dimension
a	input 2d Matrix

Returns

Return result array for specific function.

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

- Functioning.java

5.6 optimazation.pkg1.Functions Class Reference

Public Member Functions

- **Functions** (int populations, int size)
- double [ ] Schwefel (double[ ][ ] a)
- double [ ] DeJong (double[ ][ ] a)
- double [ ] Rosenbrock (double[ ][ ] a)
- double [ ] Rastrigin (double[ ][ ] a)
- double [ ] Griewangk (double[ ][ ] a)
- double [ ] SineEnvelope (double[ ][ ] a)
- double [ ] StretchedV (double[ ][ ] a)
- double [ ] Ackley1 (double[ ][ ] a)
- double [ ] Ackely2 (double[ ][ ] a)
- double [ ] EggHolder (double[ ][ ] a)
- double [ ] Rana (double[ ][ ] a)
- double [ ] Pathological (double[ ][ ] a)
- double [ ] Michalewicz (double[ ][ ] a)
- double [ ] Masters (double[ ][ ] a)
- double [ ] Quartic (double[ ][ ] a)
- double [ ] Levy (double[ ][ ] a)
- double [ ] Step (double[ ][ ] a)
- double [ ] Alpine (double[ ][ ] a)

5.6.1 Detailed Description

Author

JuneYeob Lee(24629603) Functions class is composed with 18 functions

-

Date

4/5/2019 Contact:Leej @cwu.edu Created on: 3/28/2019

5.6.2 Member Function Documentation

5.6.2.1 Ackely2()

```
double [] optimazation.pkg1.Functions.Ackely2 (
            double a[][] )
```

Ackley Two function Range[-32,32] expected global minimum = 0

Parameters

a	(input matrix)

Returns

result

5.6.2.2 Ackley1()

```
double [] optimazation.pkg1.Functions.Ackley1 (
            double a[][] )
```

Ackley One Function

- Range[-32,32] expected global minimum = -7.54276-2.91867(n-3)

Parameters

a	(input matrix)

Returns

result

5.6.2.3 Alpine()

```
double [] optimazation.pkg1.Functions.Alpine (
            double a[][] )
```

Alpine function Range[-100,100] expected global minimum = 0

Parameters

a	(input matrix)

Returns

result

5.6.2.4 DeJong()

```
double [] optimazation.pkg1.Functions.DeJong (
            double a[][] )
```

DeJong 1 function Range [-100,100] expected minimum =0

Parameters

a	(input matrix)

Returns

result

5.6.2.5 EggHolder()

```
double [] optimazation.pkg1.Functions.EggHolder (
            double a[][] )
```

EggHolder Range[-500,500] expected global minimum none

Parameters

a	(input matrix)

Returns

result

5.6.2.6 Griewangk()

```
double [] optimazation.pkg1.Functions.Griewangk (
            double a[][] )
```

Griewangk function Range[-500,500] expected global minimum =0

Parameters

a	(input matrix)

Returns

result

5.6.2.7 Levy()

```
double [] optimazation.pkg1.Functions.Levy (
            double a[][] )
```

Levy function Range[-10,10] expected global minimum = 0

Parameters

a	(input matrix)

Returns

result

5.6.2.8 Masters()

```
double [] optimazation.pkg1.Functions.Masters (
            double a[][] )
```

Masters'Cosine Wave function Range[-30,30] expected global minimum = 1-n when n is size of input matrix

Parameters

a	(input matrix)

Returns

result

5.6.2.9 Michalewicz()

```
double [] optimazation.pkg1.Functions.Michalewicz (
            double a[][] )
```

Michalewicz function Range[0,Pi] expected global minimum = 0.996n when n is size of input matrix

Parameters

a	(input matrix)

Returns

result

5.6.2.10 Pathological()

```
double [] optimazation.pkg1.Functions.Pathological (
            double a[][] )
```

Pathological function Range[-100,100] expected global minimum = none

Parameters

a	(input matrix)

Returns

result

5.6.2.11 Quartic()

```
double [] optimazation.pkg1.Functions.Quartic (
            double a[][] )
```

Quartic function Range[-100,100] expected global minimum = 0

Parameters

a	(input matrix)

Returns

result

5.6.2.12 Rana()

```
double [] optimazation.pkg1.Functions.Rana (
            double a[][] )
```

Rana function Range[-500,500] expected global minimum =none

Parameters

a	(input matrix)

5.6.2.13 Rastrigin()

```
double [] optimazation.pkg1.Functions.Rastrigin (
            double a[][] )
```

Rastrigin function Range[-30,30] expectedglobal minimum =0

Parameters

a	(input matrix)

Returns

result

5.6.2.14 Rosenbrock()

```
double [] optimazation.pkg1.Functions.Rosenbrock (
            double a[][] )
```

Rosenbrock's Saddle function Range[-100,100] expected global minimum =0

Parameters

a	(input matrix)

Returns

result

5.6.2.15 Schwefel()

```
double [] optimazation.pkg1.Functions.Schwefel (
            double a[][] )
```

Schwefel method input range[-512,512] expect global minimum = 0

Parameters

a	(input matrix)

Returns

result

5.6.2.16 SineEnvelope()

```
double [] optimazation.pkg1.Functions.SineEnvelope (
            double a[][] )
```

Sine Envelope Sine Wave Range[-30,30] expected global minimum =-1.4915(n-1) when n is size of input matrix

Parameters

a	(input matrix)

5.6.2.17 Step()

```
double [] optimazation.pkg1.Functions.Step (
            double a[][] )
```

Step function Range[-100,100] expected global minimum = 0

Parameters

a	(input matrix)

Returns

result

5.6.2.18 StretchedV()

```
double [] optimazation.pkg1.Functions.StretchedV (
            double a[][] )
```

StretchedV Range[-30,30] expected global minimum = 0

Parameters

a	(input matrix)

Returns

result

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

- Functions.java

5.7 optimazation.pkg1.HS Class Reference

Public Member Functions

- HS (int functionNum, int NP, int DIM)
- void sortHarmony ()
- void improvise ()
- void convert ()
- void last ()

5.7.1 Detailed Description

Harmony Search Algorithm. this class to implement HS algorithm

5.7.2 Constructor & Destructor Documentation

5.7.2.1 HS()

```
optimazation.pkg1.HS.HS (
            int functionNum,
            int NP,
            int DIM )
```

HS constructor

Parameters

functionNum	Function number
NP	number of harmonies
DIM	size of 2D array

5.7.3 Member Function Documentation

5.7.3.1 convert()

```
void optimazation.pkg1.HS.convert ( )
```

convert method. check new harmony's solution is better than worst solution and if it is better convert.

5.7.3.2 improvise()

```
void optimazation.pkg1.HS.improvise ( )
```

improvise method. in this method adjust pitch to get new harmonics

5.7.3.3 last()

```
void optimazation.pkg1.HS.last ( )
```

last method set the best harmony value and worst harmony value.

5.7.3.4 sortHarmony()

```
void optimazation.pkg1.HS.sortHarmony ( )
```

sortHarmony method sort harmony in ascending order.

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

- HS.java

5.8 optimazation.pkg1.Main Class Reference

Optimization1 class.

Static Public Member Functions

- static void main (String[ ] args) throws FileNotFoundException, IOException
- static void PSO () throws IOException
- static void Ff () throws IOException
- static void HS () throws IOException

5.8.1 Detailed Description

Optimization1 class.

5.8.2 Member Function Documentation

5.8.2.1 Ff()

```
static void optimazation.pkg1.Main.Ff ( ) throws IOException  [static]
```

Ff method implement Fireflies Algorithm

Exceptions

IOException	

5.8.2.2 HS()

```
static void optimazation.pkg1.Main.HS ( ) throws IOException  [static]
```

HS method implement Harmony search Algorithm

Exceptions

IOException	

5.8.2.3 main()

```
static void optimazation.pkg1.Main.main (
            String [] args ) throws FileNotFoundException, IOException  [static]
```

Main function will

Parameters

args	the command line arguments

Exceptions

java.io.FileNotFoundException	
java.io.UnsupportedEncodingException	

5.8.2.4 PSO()

```
static void optimazation.pkg1.Main.PSO ( ) throws IOException  [static]
```

PSO algorithm implement Particle Swarm Optimization Algorithm

Exceptions

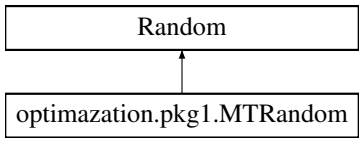
IOException	

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

- Main.java

5.9 optimazation.pkg1.MTRandom Class Reference

Inheritance diagram for optimazation.pkg1.MTRandom:



Public Member Functions

- MTRandom ()
- MTRandom (boolean compatible)
- MTRandom (long seed)
- MTRandom (byte[ ] buf)
- MTRandom (int[ ] buf)
- final synchronized void setSeed (long seed)
- final void setSeed (byte[ ] buf)
- final synchronized void setSeed (int[ ] buf)

Static Public Member Functions

- static int [ ] pack (byte[ ] buf)

Protected Member Functions

- final synchronized int next (int bits)

5.9.1 Detailed Description

Version

1.0

Author

David Beaumont, Copyright 2005

A Java implementation of the MT19937 (Mersenne Twister) pseudo random number generator algorithm based upon the original C code by Makoto Matsumoto and Takuji Nishimura (see http://www.math.sci.↩ hiroshima-u.ac.jp/~m-mat/MT/emt.html for more information.

As a subclass of java.util.Random this class provides a single canonical method next() for generating bits in the pseudo random number sequence. Anyone using this class should invoke the public inherited methods (next↩ Int(), nextFloat etc.) to obtain values as normal. This class should provide a drop-in replacement for the standard implementation of java.util.Random with the additional advantage of having a far longer period and the ability to use a far larger seed value.

This is **not** a cryptographically strong source of randomness and should **not** be used for cryptographic systems or in any other situation where true random numbers are required.

This software is licensed under the CC-GNU LGPL.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 MTRandom() [1/5]

```
optimazation.pkg1.MTRandom.MTRandom ( )
```

The default constructor for an instance of MTRandom. This invokes the no-argument constructor for java.util.↩ Random which will result in the class being initialised with a seed value obtained by calling System.currentTime↩ Millis().

5.9.2.2 MTRandom() [2/5]

```
optimazation.pkg1.MTRandom.MTRandom (
            boolean compatible )
```

This version of the constructor can be used to implement identical behaviour to the original C code version of this algorithm including exactly replicating the case where the seed value had not been set prior to calling genrand_int32.

If the compatibility flag is set to true, then the algorithm will be seeded with the same default value as was used in the original C code. Furthermore the setSeed() method, which must take a 64 bit long value, will be limited to using only the lower 32 bits of the seed to facilitate seamless migration of existing C code into Java where identical behaviour is required.

Whilst useful for ensuring backwards compatibility, it is advised that this feature not be used unless specifically required, due to the reduction in strength of the seed value.

Parameters

compatible	Compatibility flag for replicating original behaviour.

5.9.2.3 MTRandom() [3/5]

```
optimazation.pkg1.MTRandom.MTRandom (
            long seed )
```

This version of the constructor simply initialises the class with the given 64 bit seed value. For a better random number sequence this seed value should contain as much entropy as possible.

Parameters

seed	The seed value with which to initialise this class.

5.9.2.4 MTRandom() [4/5]

```
optimazation.pkg1.MTRandom.MTRandom (
            byte [] buf )
```

This version of the constructor initialises the class with the given byte array. All the data will be used to initialise this instance.

Parameters

buf	The non-empty byte array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

5.9.2.5 MTRandom() [5/5]

```
optimazation.pkg1.MTRandom.MTRandom (
            int [] buf )
```

This version of the constructor initialises the class with the given integer array. All the data will be used to initialise this instance.

Parameters

buf	The non-empty integer array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

5.9.3 Member Function Documentation

5.9.3.1 next()

```
final synchronized int optimazation.pkg1.MTRandom.next (
            int bits )  [protected]
```

This method forms the basis for generating a pseudo random number sequence from this class. If given a value of 32, this method behaves identically to the genrand_int32 function in the original C code and ensures that using the standard nextInt() function (inherited from Random) we are able to replicate behaviour exactly.

Note that where the number of bits requested is not equal to 32 then bits will simply be masked out from the top of the returned integer value. That is to say that:

```
mt.setSeed(12345);
int foo = mt.nextInt(16) + (mt.nextInt(16) << 16);
```

will not give the same result as

```
mt.setSeed(12345);
int foo = mt.nextInt(32);
```

Parameters

bits	The number of significant bits desired in the output.

Returns

The next value in the pseudo random sequence with the specified number of bits in the lower part of the integer.

5.9.3.2 pack()

```
static int [] optimazation.pkg1.MTRandom.pack (
            byte [] buf )  [static]
```

This simply utility method can be used in cases where a byte array of seed data is to be used to repeatedly re-seed the random number sequence. By packing the byte array into an integer array first, using this method, and then invoking setSeed() with that; it removes the need to re-pack the byte array each time setSeed() is called.

If the length of the byte array is not a multiple of 4 then it is implicitly padded with zeros as necessary. For example:

```
byte[] { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 }
```

becomes

```
int[]  { 0x04030201, 0x00000605 }
```

Note that this method will not complain if the given byte array is empty and will produce an empty integer array, but the setSeed() method will throw an exception if the empty integer array is passed to it.

Parameters

buf	The non-null byte array to be packed.

Returns

A non-null integer array of the packed bytes.

Exceptions

NullPointerException	if the given byte array is null.

5.9.3.3 setSeed() [1/3]

```
final synchronized void optimazation.pkg1.MTRandom.setSeed (
            long seed )
```

This method resets the state of this instance using the 64 bits of seed data provided. Note that if the same seed data is passed to two different instances of MTRandom (both of which share the same compatibility state) then the sequence of numbers generated by both instances will be identical.

If this instance was initialised in 'compatibility' mode then this method will only use the lower 32 bits of any seed value passed in and will match the behaviour of the original C code exactly with respect to state initialisation.

Parameters

seed	The 64 bit value used to initialise the random number generator state.

5.9.3.4 setSeed() [2/3]

```
final void optimazation.pkg1.MTRandom.setSeed (
            byte [] buf )
```

This method resets the state of this instance using the byte array of seed data provided. Note that calling this method is equivalent to calling "setSeed(pack(buf))" and in particular will result in a new integer array being generated during the call. If you wish to retain this seed data to allow the pseudo random sequence to be restarted then it would be more efficient to use the "pack()" method to convert it into an integer array first and then use that to re-seed the instance. The behaviour of the class will be the same in both cases but it will be more efficient.

Parameters

buf	The non-empty byte array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

5.9.3.5 setSeed() [3/3]

```
final synchronized void optimazation.pkg1.MTRandom.setSeed (
            int [] buf )
```

This method resets the state of this instance using the integer array of seed data provided. This is the canonical way of resetting the pseudo random number sequence.

Parameters

buf	The non-empty integer array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

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

- MTRandom.java

5.10 optimazation.pkg1.PSO Class Reference

Public Member Functions

- PSO (int functionNum, int NP, int DIM)
- void InitialgBest ()
- void updateVelocity ()
- void updateParticles ()
- double [ ] calculate ()
- void compare (double[ ]a)
- double gBest (double[ ]a)

5.10.1 Detailed Description

Particle Swarm Optimization class file. This class for implementing PSO algorithm.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 PSO()

```
optimazation.pkg1.PSO.PSO (
            int functionNum,
            int NP,
            int DIM )
```

POS constructor

Parameters

functionNum	function number 0∼17
NP	number of vectors
DIM	size of dimension

5.10.3 Member Function Documentation

5.10.3.1 calculate()

```
double [] optimazation.pkg1.PSO.calculate ( )
```

calculate method to calculate updated particles

Returns

result array for specific function

5.10.3.2 compare()

```
void optimazation.pkg1.PSO.compare (
            double [] a )
```

compare method to compare pbest and new fitness of updated particles

Parameters

a	result of updated particles

5.10.3.3 gBest()

```
double optimazation.pkg1.PSO.gBest (
            double [] a )
```

gBest method to calculate new global best value.

Parameters

a	new pBest

Returns

global best value

5.10.3.4 InitialgBest()

```
void optimazation.pkg1.PSO.InitialgBest ( )
```

initial gBest method for initialize the global best value.

5.10.3.5 updateParticles()

```
void optimazation.pkg1.PSO.updateParticles ( )
```

updateParticles method to update particles and check swarm boundary.

5.10.3.6 updateVelocity()

```
void optimazation.pkg1.PSO.updateVelocity ( )
```

updateVelocity method for update Velocity.

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

- PSO.java

Index