

My Project

Generated by Doxygen 1.16.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Population Class Reference	5
3.1.1 Constructor & Destructor Documentation	5
3.1.1.1 Population()	5
4 File Documentation	7
4.1 Blind.h	7
4.2 Population.h	7
4.3 Problem.h	7
4.4 RepeatedLocalSearch.h	8
Index	9

Chapter 1

Class Index

1.1 Class List

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

Population	5
--------------------------------------	---

Chapter 2

File Index

2.1 File List

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

Blind.h	7
Population.h	7
Problem.h	7
RepeatedLocalSearch.h	8

Chapter 3

Class Documentation

3.1 Population Class Reference

Public Member Functions

- [Population](#) (size_t *pop_size*, size_t *dimensions*)

Public Attributes

- vector< vector< float > > **population**
- vector< float > **fitness**

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Population()

```
Population::Population (
    size_t pop_size,
    size_t dimensions)
```

Constructor for [Population](#) initializes population matrix to be of size *pop_size* x *dimensions* and fitness vector to size *pop_size*

Parameters

in	<i>pop_size, dimensions</i>	
----	-----------------------------	--

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

- Population.h
- Population.cpp

Chapter 4

File Documentation

4.1 Blind.h

```
00001 // Name: Harrison Ingram-Bate
00002 #ifndef BLIND
00003 #define BLIND
00004
00005 #include <random>
00006 #include <memory>
00007 #include <chrono>
00008 #include "Population.h"
00009
00015 Population Blind(std::unique_ptr<vector<uniform_real_distribution<float>> distribution_vec,
00016           float (*fitness)(const vector<float>&),
00017           size_t pop_size);
00018
00019 #endif
```

4.2 Population.h

```
00001 // Name: Harrison Ingram-Bate
00002 #ifndef POPULATION
00003 #define POPULATION
00004 #include <vector>
00005 #include <random>
00006 using namespace std;
00007
00008 class Population {
00009 public:
00015     Population(size_t pop_size, size_t dimensions);
00016
00017     vector<vector<float>> population;
00018     vector<float> fitness;
00019
00020 };
00021
00022 #endif
```

4.3 Problem.h

```
00001 // Name: Harrison Ingram-Bate
00002 #ifndef PROBLEM
00003 #define PROBLEM
00004 #include <vector>
00005 using namespace std;
00006
00012 float Schwefel(const vector<float>& vec);
00013
00019 float FirstDeJong(const vector<float>& vec);
00020
00026 float Rosenbrock(const vector<float>& vec);
```

```
00027
00033 float Rastrigin(const vector<float>& vec);
00034
00040 float Griewangk(const vector<float>& vec);
00041
00047 float SineEnvelope(const vector<float>& vec);
00048
00054 float StretchedV(const vector<float>& vec);
00055
00061 float AckleyOne(const vector<float>& vec);
00062
00068 float AckleyTwo(const vector<float>& vec);
00069
00075 float EggHolder(const vector<float>& vec);
00076 #endif
```

4.4 RepeatedLocalSearch.h

```
00001 // Name: Harrison Ingram-Bate
00002 #ifndef REPEATED_LOCAL_SEARCH
00003 #define REPEATED_LOCAL_SEARCH
00004
00005 #include <memory>
00006 #include <random>
00007 #include <chrono>
00008 #include "Population.h"
00009
00010 typedef std::unique_ptr<vector<uniform_real_distribution<float>> Distributions;
00011
00017 Population RepeatedLocalSearch(Distributions distribution_vec,
00018                                     float (*fitness)(const vector<float>&),
00019                                     size_t pop_size,
00020                                     float step_size);
00021
00028 Distributions SetBestVecFromRandPop(Distributions distribution_vec,
00029                                         float (*fitness)(const vector<float>&),
00030                                         mt19937& rand_gen,
00031                                         vector<float>& dest_vec,
00032                                         size_t pop_size);
00033
00039 Distributions LocalSearch(Distributions distribution_vec,
00040                               float (*fitness)(const vector<float>&),
00041                               vector<float>& best_vec,
00042                               float step_size);
00043 #endif
```

Index

Population, [5](#)
Population, [5](#)