

Project 3

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 algorithmOutput Struct Reference	5
3.2 Population::bestResult Struct Reference	5
3.3 Bounds Struct Reference	5
3.4 Population Class Reference	6
4 File Documentation	7
4.1 BlindSearch.h	7
4.2 DE.h	7
4.3 Functions.h	8
4.4 LocalSearch.h	8
4.5 Population.h	8
4.6 PSO.h	9
4.7 RepeatedLocalSearch.h	9
Index	11

Chapter 1

Class Index

1.1 Class List

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

algorithmOutput	5
Population::bestResult	5
Bounds	5
Population	6

Chapter 2

File Index

2.1 File List

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

BlindSearch.h	7
DE.h	7
Functions.h	8
LocalSearch.h	8
Population.h	8
PSO.h	9
RepeatedLocalSearch.h	9

Chapter 3

Class Documentation

3.1 algorithmOutput Struct Reference

Public Attributes

- float **bestFitness**
- float **runtimeMs**

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

- BlindSearch.h

3.2 Population::bestResult Struct Reference

Public Attributes

- std::vector< float > **x**
- float **fitness**
- int **index**

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

- Population.h

3.3 Bounds Struct Reference

Public Attributes

- float **min**
- float **max**

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

- ProjectMain.cpp

3.4 Population Class Reference

Classes

- struct [bestResult](#)

Public Member Functions

- **Population** (int n, int m)
- void **generate** (float lower, float upper)
- void **evaluate** (int problemType)
- void **writeFitnessCSV** (const std::string &filename, bool append, float runtime_ms) const
- [bestResult best](#) () const
- void **writeBestCSV** (const std::string &filename, bool append, const [bestResult](#) &best, float runtime_ms) const
- float **evaluateVector** (const std::vector< float > &x, int problemType) const
- const std::vector< float > & **getRow** (int i) const
- void **setRow** (int i, const std::vector< float > &x)
- int **size** () const

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

- Population.h
- Population.cpp

Chapter 4

File Documentation

4.1 BlindSearch.h

```
00001 // Zack Lee
00002 //BlindSearch.h
00003
00004 #ifndef BLINDSEARCH_H
00005 #define BLINDSEARCH_H
00006
00007 #include <string>
00008
00009 // Struct to hold algorithm output
00010 struct algorithmOutput {
00011     float bestFitness;
00012     float runtimeMs;
00013 };
00014
00015 // Function declaration for blind search
00016 algorithmOutput blindSearch(int n, int m, float lower, float upper, int problemType, const
00017 std::string& csvFile, bool append);
00018 #endif
```

4.2 DE.h

```
00001 // Zack Lee
00002 // DE.h
00003 // 02/18/2026
00004
00005 #ifndef DE_H
00006 #define DE_H
00007
00008 #include "Population.h"
00009 #include "BlindSearch.h"
00010
00011 // DE function declarations
00012 algorithmOutput differentialEvolution( int NP, int D, float lower, float upper, int problemType, int
00013 strategy, float F, float CR, float lambda, int generations, const std::string& csvFile, bool append);
00014
00015 // Returns mutant vector v for a given strategy.
00016 // pop[k] is x_k, best is x_best, xi is x_i.
00017 std::vector<float> makeMutant(
00018     int strategy,
00019     const std::vector<std::vector<float>>& pop,
00020     const std::vector<float>& xi,
00021     const std::vector<float>& best,
00022     int r1,int r2,int r3,int r4,int r5,
00023     float F, float lambda
00024 );
00025
00026 #endif
```

4.3 Functions.h

```

00001 // Zackary Lee
00002 // Functions.h
00003 // This file contains declarations of various utility functions.
00004 // Date: 02/18/2026
00005
00006 #ifndef FUNCTIONS_H
00007 #define FUNCTIONS_H
00008
00009 #include <cmath>
00010 #include <vector>
00011 using namespace std;
00012
00013 float schwefelFunction(const vector<float>& x);
00014 float dejongFunction(const vector<float>& x);
00015 float rosenbrockFunction(const vector<float>& x);
00016 float rastriginFunction(const vector<float>& x);
00017 float griewankFunction(const vector<float>& x);
00018 float sineEnvelopeSineWaveFunction(const vector<float>& x);
00019 float stretchedVSineWaveFunction(const vector<float>& x);
00020 float ackleyOneFunction(const vector<float>& x);
00021 float ackleyTwoFunction(const vector<float>& x);
00022 float eggHolderFunction(const vector<float>& x);
00023
00024 #endif

```

4.4 LocalSearch.h

```

00001 // Zack Lee
00002 // LocalSearch.h
00003
00004 #ifndef LOCALSEARCH_H
00005 #define LOCALSEARCH_H
00006 #include "Population.h"
00007 #include "BlindSearch.h"
00008
00009 // Forward declaration
00010 class Population;
00011
00012 // Perform one local step to generate new candidate solution
00013 std::vector<float> localStep(const Population& pop, const std::vector<float>& x_i, int problemType,
    float alpha, float lowerBound, float upperBound);
00014
00015 // Improve once
00016 std::vector<float> improveOnce(const Population& pop, const std::vector<float>& x_t, int problemType,
    float alpha, float lowerBound, float upperBound);
00017
00018 // Perform local search algorithm
00019 algorithmOutput localSearch(int n, int m, float lower, float upper, int problemType, float alpha,
    const std::string& csvFile, bool append);
00020
00021 #endif

```

4.5 Population.h

```

00001 // Zack Lee
00002 // Population.h
00003 // 02/18/2026
00004
00005 #ifndef POPULATION_H
00006 #define POPULATION_H
00007
00008 #include <vector>
00009 #include <string>
00010 #include <random>
00011 #include <fstream>
00012 #include <stdexcept>
00013 #include <chrono>
00014
00015 class Population {
00016 public:
00017
00018     // n = 30, m = 10, 20, or 30
00019     Population(int n, int m);
00020
00021     // Generate population values uniformly in [min, max]

```

```

00022     void generate(float lower, float upper);
00023
00024     // Evaluate ONLY the chosen problem (1..10) for all 30 rows
00025     void evaluate(int problemType);
00026
00027     // Write fitness values (one per row) to CSV
00028     void writeFitnessCSV(const std::string &filename, bool append, float runtime_ms) const;
00029
00030     // struct to help store vectors and the fitness to find the best one
00031     struct bestResult {
00032         std::vector<float> x;
00033         float fitness;
00034         int index;
00035     };
00036
00037     bestResult best() const;
00038
00039     // Write only the BEST fitness to the CSV out of the matrix
00040     void writeBestCSV(const std::string & filename, bool append, const bestResult & best, float
00041     runtime_ms) const;
00042
00043     // Evaluate a single vector using the selected problem
00044     float evaluateVector(const std::vector<float>& x, int problemType) const;
00045
00046     // Accessors
00047     const std::vector<float>& getRow(int i) const;
00048     void setRow(int i, const std::vector<float>& x);
00049     int size() const;
00050
00051
00052 private:
00053     int n_;
00054     int m_;
00055
00056     // Population matrix (n x m)
00057     std::vector<std::vector<float>> pop_;
00058     std::vector<float> fitness_;
00059
00060     // Evaluate a single vector using the selected problem
00061     float evalOne(const std::vector<float>& x, int problemType) const;
00062 };
00063
00064 #endif

```

4.6 PSO.h

```

00001 // Zack Lee
00002 // PSO.h
00003 // 02/18/2026
00004
00005 #ifndef PSO_H
00006 #define PSO_H
00007
00008 #include <string>
00009 #include "Population.h"
00010 #include "BlindSearch.h"
00011
00012 // PSO function declaration
00013 algorithmOutput particleSwarmOptimization(int particles, int m, float lower, float upper, int
00014     problemType, int generations, float c1, float c2, const std::string & csvFile, bool append);
00015
00016 #endif // PSO_H

```

4.7 RepeatedLocalSearch.h

```

00001 // Zack Lee
00002 // RepeatedLocalSearch.h
00003
00004 #ifndef REPEATEDLOCALSEARCH_H
00005 #define REPEATEDLOCALSEARCH_H
00006 #include "Population.h"
00007 #include "LocalSearch.h"
00008 #include "BlindSearch.h"
00009
00010 // Function declaration for repeated local search
00011 algorithmOutput repeatedLocalSearch(int n, int m, float lower, float upper, int problemType, float
00012     alpha, int repeats, const std::string & csvFile, bool append);
00013
00014 #endif

```


Index

algorithmOutput, [5](#)

Bounds, [5](#)

Population, [6](#)

Population::bestResult, [5](#)