

CS471 Project 2

Generated by Doxygen 1.16.1

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 Run_Functions Namespace Reference	7
4.1.1 Function Documentation	7
4.1.1.1 main()	7
4.1.2 Variable Documentation	7
4.1.2.1 EXECUTABLE	7
4.1.2.2 NUM_RUNS	7
4.1.2.3 OUTPUT_CSV	7
5 Class Documentation	9
5.1 algorithmOutput Struct Reference	9
5.1.1 Member Data Documentation	9
5.1.1.1 bestFitness	9
5.1.1.2 runtimeMs	9
5.2 Population::bestResult Struct Reference	9
5.2.1 Member Data Documentation	10
5.2.1.1 fitness	10
5.2.1.2 index	10
5.2.1.3 x	10
5.3 Bounds Struct Reference	10
5.3.1 Member Data Documentation	10
5.3.1.1 max	10
5.3.1.2 min	10
5.4 Population Class Reference	11
5.4.1 Constructor & Destructor Documentation	11
5.4.1.1 Population()	11
5.4.2 Member Function Documentation	11
5.4.2.1 best()	11
5.4.2.2 evaluate()	11
5.4.2.3 evaluateVector()	11
5.4.2.4 generate()	12
5.4.2.5 getRow()	12
5.4.2.6 setRow()	12
5.4.2.7 size()	12
5.4.2.8 writeBestCSV()	12

5.4.2.9 writeFitnessCSV()	12
6 File Documentation	13
6.1 BlindSearch.cpp File Reference	13
6.1.1 Function Documentation	13
6.1.1.1 blindSearch()	13
6.2 BlindSearch.h File Reference	13
6.2.1 Function Documentation	14
6.2.1.1 blindSearch()	14
6.3 BlindSearch.h	14
6.4 Functions.cpp File Reference	14
6.4.1 Function Documentation	15
6.4.1.1 ackleyOneFunction()	15
6.4.1.2 ackleyTwoFunction()	15
6.4.1.3 dejongFunction()	15
6.4.1.4 eggHolderFunction()	15
6.4.1.5 griewankFunction()	15
6.4.1.6 rastriginFunction()	15
6.4.1.7 rosenbrockFunction()	16
6.4.1.8 schwefelFunction()	16
6.4.1.9 sineEnvelopeSineWaveFunction()	16
6.4.1.10 stretchedVSineWaveFunction()	16
6.5 Functions.h File Reference	16
6.5.1 Function Documentation	16
6.5.1.1 ackleyOneFunction()	16
6.5.1.2 ackleyTwoFunction()	17
6.5.1.3 dejongFunction()	17
6.5.1.4 eggHolderFunction()	17
6.5.1.5 griewankFunction()	17
6.5.1.6 rastriginFunction()	17
6.5.1.7 rosenbrockFunction()	17
6.5.1.8 schwefelFunction()	17
6.5.1.9 sineEnvelopeSineWaveFunction()	17
6.5.1.10 stretchedVSineWaveFunction()	17
6.6 Functions.h	18
6.7 LocalSearch.cpp File Reference	18
6.7.1 Function Documentation	18
6.7.1.1 improveOnce()	18
6.7.1.2 localSearch()	19
6.7.1.3 localStep()	19
6.8 LocalSearch.h File Reference	19
6.8.1 Function Documentation	19

6.8.1.1 improveOnce()	19
6.8.1.2 localSearch()	20
6.8.1.3 localStep()	20
6.9 LocalSearch.h	20
6.10 Population.cpp File Reference	20
6.11 Population.h File Reference	21
6.12 Population.h	21
6.13 ProjectMain.cpp File Reference	22
6.13.1 Function Documentation	22
6.13.1.1 main()	22
6.14 RepeatedLocalSearch.cpp File Reference	22
6.14.1 Function Documentation	23
6.14.1.1 repeatedLocalSearch()	23
6.15 RepeatedLocalSearch.h File Reference	23
6.15.1 Function Documentation	23
6.15.1.1 repeatedLocalSearch()	23
6.16 RepeatedLocalSearch.h	24
6.17 Run_Functions.py File Reference	24
Index	25

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Run_Functions	7
----------------------	-------	---

Chapter 2

Class Index

2.1 Class List

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

algorithmOutput	9
Population::bestResult	9
Bounds	10
Population	11

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

BlindSearch.cpp	13
BlindSearch.h	13
Functions.cpp	14
Functions.h	16
LocalSearch.cpp	18
LocalSearch.h	19
Population.cpp	20
Population.h	21
ProjectMain.cpp	22
RepeatedLocalSearch.cpp	22
RepeatedLocalSearch.h	23
Run_Functions.py	24

Chapter 4

Namespace Documentation

4.1 Run_Functions Namespace Reference

Functions

- `main ()`

Variables

- str `EXECUTABLE` = "./ProjectMain"
- str `OUTPUT_CSV` = "results.csv"
- int `NUM_RUNS` = 30

4.1.1 Function Documentation

4.1.1.1 `main()`

```
Run_Functions.main ()
```

4.1.2 Variable Documentation

4.1.2.1 `EXECUTABLE`

```
str Run_Functions.EXECUTABLE = "./ProjectMain"
```

4.1.2.2 `NUM_RUNS`

```
int Run_Functions.NUM_RUNS = 30
```

4.1.2.3 `OUTPUT_CSV`

```
str Run_Functions.OUTPUT_CSV = "results.csv"
```


Chapter 5

Class Documentation

5.1 algorithmOutput Struct Reference

```
#include <BlindSearch.h>
```

Public Attributes

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

5.1.1 Member Data Documentation

5.1.1.1 bestFitness

```
float algorithmOutput::bestFitness
```

5.1.1.2 runtimeMs

```
float algorithmOutput::runtimeMs
```

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

- **BlindSearch.h**

5.2 Population::bestResult Struct Reference

```
#include <Population.h>
```

Public Attributes

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

5.2.1 Member Data Documentation

5.2.1.1 **fitness**

```
float Population::bestResult::fitness
```

5.2.1.2 **index**

```
int Population::bestResult::index
```

5.2.1.3 **x**

```
std::vector<float> Population::bestResult::x
```

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

- **Population.h**

5.3 Bounds Struct Reference

Public Attributes

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

5.3.1 Member Data Documentation

5.3.1.1 **max**

```
float Bounds::max
```

5.3.1.2 **min**

```
float Bounds::min
```

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

- **ProjectMain.cpp**

5.4 Population Class Reference

```
#include <Population.h>
```

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

5.4.1 Constructor & Destructor Documentation

5.4.1.1 Population()

```
Population::Population (
    int n,
    int m)
```

5.4.2 Member Function Documentation

5.4.2.1 best()

```
Population::bestResult Population::best () const
```

5.4.2.2 evaluate()

```
void Population::evaluate (
    int problemType)
```

5.4.2.3 evaluateVector()

```
float Population::evaluateVector (
    const std::vector< float > &x,
    int problemType) const
```

5.4.2.4 `generate()`

```
void Population::generate (
    float lower,
    float upper)
```

5.4.2.5 `getRow()`

```
const std::vector< float > & Population::getRow (
    int i) const
```

5.4.2.6 `setRow()`

```
void Population::setRow (
    int i,
    const std::vector< float > & x)
```

5.4.2.7 `size()`

```
int Population::size () const
```

5.4.2.8 `writeBestCSV()`

```
void Population::writeBestCSV (
    const std::string & filename,
    bool append,
    const bestResult & best,
    float runtime_ms) const
```

5.4.2.9 `writeFitnessCSV()`

```
void Population::writeFitnessCSV (
    const std::string & filename,
    bool append,
    float runtime_ms) const
```

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

- **Population.h**
- **Population.cpp**

Chapter 6

File Documentation

6.1 BlindSearch.cpp File Reference

```
#include "BlindSearch.h"
#include "Population.h"
#include <chrono>
```

Functions

- **algorithmOutput blindSearch** (int n, int m, float lower, float upper, int problemType, const std::string &csvFile, bool append)

6.1.1 Function Documentation

6.1.1.1 blindSearch()

```
algorithmOutput blindSearch (
    int n,
    int m,
    float lower,
    float upper,
    int problemType,
    const std::string & csvFile,
    bool append)
```

6.2 BlindSearch.h File Reference

```
#include <string>
```

Classes

- struct **algorithmOutput**

Functions

- **algorithmOutput blindSearch** (int n, int m, float lower, float upper, int problemType, const std::string &csvFile, bool append)

6.2.1 Function Documentation

6.2.1.1 blindSearch()

```
algorithmOutput blindSearch (
    int n,
    int m,
    float lower,
    float upper,
    int problemType,
    const std::string & csvFile,
    bool append)
```

6.3 BlindSearch.h

Go to the documentation of this file.

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

6.4 Functions.cpp File Reference

```
#include "Functions.h"
```

Functions

- float **schwefelFunction** (const vector< float > &x)
- float **dejongFunction** (const vector< float > &x)
- float **rosenbrockFunction** (const vector< float > &x)
- float **rastriginFunction** (const vector< float > &x)
- float **griewankFunction** (const vector< float > &x)
- float **sineEnvelopeSineWaveFunction** (const vector< float > &x)
- float **stretchedVSineWaveFunction** (const vector< float > &x)
- float **ackleyOneFunction** (const vector< float > &x)
- float **ackleyTwoFunction** (const vector< float > &x)
- float **eggHolderFunction** (const vector< float > &x)

6.4.1 Function Documentation

6.4.1.1 ackleyOneFunction()

```
float ackleyOneFunction (
    const vector< float > & x)
```

6.4.1.2 ackleyTwoFunction()

```
float ackleyTwoFunction (
    const vector< float > & x)
```

6.4.1.3 dejongFunction()

```
float dejongFunction (
    const vector< float > & x)
```

6.4.1.4 eggHolderFunction()

```
float eggHolderFunction (
    const vector< float > & x)
```

6.4.1.5 griewankFunction()

```
float griewankFunction (
    const vector< float > & x)
```

6.4.1.6 rastriginFunction()

```
float rastriginFunction (
    const vector< float > & x)
```

6.4.1.7 rosenbrockFunction()

```
float rosenbrockFunction (
    const vector< float > & x)
```

6.4.1.8 schwefelFunction()

```
float schwefelFunction (
    const vector< float > & x)
```

6.4.1.9 sineEnvelopeSineWaveFunction()

```
float sineEnvelopeSineWaveFunction (
    const vector< float > & x)
```

6.4.1.10 stretchedVSineWaveFunction()

```
float stretchedVSineWaveFunction (
    const vector< float > & x)
```

6.5 Functions.h File Reference

```
#include <cmath>
#include <vector>
```

Functions

- float **schwefelFunction** (const vector< float > &x)
- float **dejongFunction** (const vector< float > &x)
- float **rosenbrockFunction** (const vector< float > &x)
- float **rastriginFunction** (const vector< float > &x)
- float **griewankFunction** (const vector< float > &x)
- float **sineEnvelopeSineWaveFunction** (const vector< float > &x)
- float **stretchedVSineWaveFunction** (const vector< float > &x)
- float **ackleyOneFunction** (const vector< float > &x)
- float **ackleyTwoFunction** (const vector< float > &x)
- float **eggHolderFunction** (const vector< float > &x)

6.5.1 Function Documentation

6.5.1.1 ackleyOneFunction()

```
float ackleyOneFunction (
    const vector< float > & x)
```

6.5.1.2 ackleyTwoFunction()

```
float ackleyTwoFunction (
    const vector< float > & x)
```

6.5.1.3 dejongFunction()

```
float dejongFunction (
    const vector< float > & x)
```

6.5.1.4 eggHolderFunction()

```
float eggHolderFunction (
    const vector< float > & x)
```

6.5.1.5 griewankFunction()

```
float griewankFunction (
    const vector< float > & x)
```

6.5.1.6 rastriginFunction()

```
float rastriginFunction (
    const vector< float > & x)
```

6.5.1.7 rosenbrockFunction()

```
float rosenbrockFunction (
    const vector< float > & x)
```

6.5.1.8 schwefelFunction()

```
float schwefelFunction (
    const vector< float > & x)
```

6.5.1.9 sineEnvelopeSineWaveFunction()

```
float sineEnvelopeSineWaveFunction (
    const vector< float > & x)
```

6.5.1.10 stretchedVSineWaveFunction()

```
float stretchedVSineWaveFunction (
    const vector< float > & x)
```

6.6 Functions.h

[Go to the documentation of this file.](#)

```
00001 // Zackary Lee
00002 // Functions.h
00003 // This file contains declarations of various utility functions.
00004 // Date: 02/03/2026
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
```

6.7 LocalSearch.cpp File Reference

```
#include "LocalSearch.h"
#include "Population.h"
#include "BlindSearch.h"
```

Functions

- **algorithmOutput localSearch** (int n, int m, float lower, float upper, int problemType, float alpha, const std::string &csvFile, bool append)
- std::vector< float > **localStep** (const **Population** &pop, const std::vector< float > &x_t, int problemType, float alpha, float lowerBound, float upperBound)
- std::vector< float > **improveOnce** (const **Population** &pop, const std::vector< float > &x_t, int problemType, float alpha, float lowerBound, float upperBound)

6.7.1 Function Documentation

6.7.1.1 **improveOnce()**

```
std::vector< float > improveOnce (
    const Population & pop,
    const std::vector< float > & x_t,
    int problemType,
    float alpha,
    float lowerBound,
    float upperBound)
```

6.7.1.2 localSearch()

```
algorithmOutput localSearch (
    int n,
    int m,
    float lower,
    float upper,
    int problemType,
    float alpha,
    const std::string & csvFile,
    bool append)
```

6.7.1.3 localStep()

```
std::vector< float > localStep (
    const Population & pop,
    const std::vector< float > & x_t,
    int problemType,
    float alpha,
    float lowerBound,
    float upperBound)
```

6.8 LocalSearch.h File Reference

```
#include "Population.h"
#include "BlindSearch.h"
```

Functions

- std::vector< float > **localStep** (const **Population** &pop, const std::vector< float > &x_i, int problemType, float alpha, float lowerBound, float upperBound)
- std::vector< float > **improveOnce** (const **Population** &pop, const std::vector< float > &x_t, int problemType, float alpha, float lowerBound, float upperBound)
- **algorithmOutput** **localSearch** (int n, int m, float lower, float upper, int problemType, float alpha, const std::string &csvFile, bool append)

6.8.1 Function Documentation

6.8.1.1 improveOnce()

```
std::vector< float > improveOnce (
    const Population & pop,
    const std::vector< float > & x_t,
    int problemType,
    float alpha,
    float lowerBound,
    float upperBound)
```

6.8.1.2 localSearch()

```
algorithmOutput localSearch (
    int n,
    int m,
    float lower,
    float upper,
    int problemType,
    float alpha,
    const std::string & csvFile,
    bool append)
```

6.8.1.3 localStep()

```
std::vector< float > localStep (
    const Population & pop,
    const std::vector< float > & x_i,
    int problemType,
    float alpha,
    float lowerBound,
    float upperBound)
```

6.9 LocalSearch.h

[Go to the documentation of this file.](#)

```
00001 // LocalSearch.h
00002
00003 #ifndef LOCALSEARCH_H
00004 #define LOCALSEARCH_H
00005 #include "Population.h"
00006 #include "BlindSearch.h"
00007
00008 // Forward declaration
00009 class Population;
00010
00011 // Perform one local step to generate new candidate solution
00012 std::vector<float> localStep(const Population& pop, const std::vector<float>& x_i, int problemType,
00013     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,
00017     float alpha, float lowerBound, float upperBound);
00018
00019 // Perform local search algorithm
00020 algorithmOutput localSearch(int n, int m, float lower, float upper, int problemType, float alpha,
00021     const std::string& csvFile, bool append);
00022 #endif
```

6.10 Population.cpp File Reference

```
#include "Population.h"
#include "Functions.h"
```

6.11 Population.h File Reference

```
#include <vector>
#include <string>
#include <random>
#include <fstream>
#include <stdexcept>
#include <chrono>
```

Classes

- class **Population**
- struct **Population::bestResult**

6.12 Population.h

[Go to the documentation of this file.](#)

```
00001 // Zack Lee
00002 // Population.h
00003 // 02/03/2026
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
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

```

6.13 ProjectMain.cpp File Reference

```

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <unordered_map>
#include <stdexcept>
#include <chrono>
#include <cctype>
#include <iomanip>
#include "Population.h"
#include "BlindSearch.h"
#include "LocalSearch.h"
#include "RepeatedLocalSearch.h"

```

Classes

- struct **Bounds**

Functions

- int **main** (int argc, char *argv[])

6.13.1 Function Documentation

6.13.1.1 main()

```

int main (
    int argc,
    char * argv[])

```

6.14 RepeatedLocalSearch.cpp File Reference

```

#include "RepeatedLocalSearch.h"
#include "Population.h"
#include <chrono>

```

Functions

- **algorithmOutput repeatedLocalSearch** (int n, int m, float lower, float upper, int problemType, float alpha, int repeats, const std::string &csvFile, bool append)

6.14.1 Function Documentation

6.14.1.1 repeatedLocalSearch()

```
algorithmOutput repeatedLocalSearch (
    int n,
    int m,
    float lower,
    float upper,
    int problemType,
    float alpha,
    int repeats,
    const std::string & csvFile,
    bool append)
```

6.15 RepeatedLocalSearch.h File Reference

```
#include "Population.h"
#include "LocalSearch.h"
#include "BlindSearch.h"
```

Functions

- **algorithmOutput repeatedLocalSearch** (int n, int m, float lower, float upper, int problemType, float alpha, int repeats, const std::string &csvFile, bool append)

6.15.1 Function Documentation

6.15.1.1 repeatedLocalSearch()

```
algorithmOutput repeatedLocalSearch (
    int n,
    int m,
    float lower,
    float upper,
    int problemType,
    float alpha,
    int repeats,
    const std::string & csvFile,
    bool append)
```

6.16 RepeatedLocalSearch.h

[Go to the documentation of this file.](#)

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

6.17 Run_Functions.py File Reference

Namespaces

- namespace **Run_Functions**

Functions

- **Run_Functions.main ()**

Variables

- str **Run_Functions.EXECUTABLE** = "./ProjectMain"
- str **Run_Functions.OUTPUT_CSV** = "results.csv"
- int **Run_Functions.NUM_RUNS** = 30

Index

ackleyOneFunction
 Functions.cpp, 15
 Functions.h, 16
ackleyTwoFunction
 Functions.cpp, 15
 Functions.h, 16
algorithmOutput, 9
 bestFitness, 9
 runtimeMs, 9

best
 Population, 11
bestFitness
 algorithmOutput, 9
blindSearch
 BlindSearch.cpp, 13
 BlindSearch.h, 14
BlindSearch.cpp, 13
 blindSearch, 13
BlindSearch.h, 13
 blindSearch, 14
Bounds, 10
 max, 10
 min, 10

dejongFunction
 Functions.cpp, 15
 Functions.h, 17

eggHolderFunction
 Functions.cpp, 15
 Functions.h, 17

evaluate
 Population, 11
evaluateVector
 Population, 11
EXECUTABLE
 Run_Functions, 7

fitness
 Population::bestResult, 10
Functions.cpp, 14
 ackleyOneFunction, 15
 ackleyTwoFunction, 15
 dejongFunction, 15
 eggHolderFunction, 15
 griewankFunction, 15
 rastriginFunction, 15
 rosenbrockFunction, 15
 schwefelFunction, 16

sineEnvelopeSineWaveFunction, 16
stretchedVSineWaveFunction, 16
Functions.h, 16
 ackleyOneFunction, 16
 ackleyTwoFunction, 16
 dejongFunction, 17
 eggHolderFunction, 17
 griewankFunction, 17
 rastriginFunction, 17
 rosenbrockFunction, 17
 schwefelFunction, 17
 sineEnvelopeSineWaveFunction, 17
 stretchedVSineWaveFunction, 17

generate
 Population, 11
getRow
 Population, 12
griewankFunction
 Functions.cpp, 15
 Functions.h, 17

improveOnce
 LocalSearch.cpp, 18
 LocalSearch.h, 19
index
 Population::bestResult, 10

localSearch
 LocalSearch.cpp, 18
 LocalSearch.h, 19
LocalSearch.cpp, 18
 improveOnce, 18
 localSearch, 18
 localStep, 19
LocalSearch.h, 19
 improveOnce, 19
 localSearch, 19
 localStep, 20
localStep
 LocalSearch.cpp, 19
 LocalSearch.h, 20

main
 ProjectMain.cpp, 22
 Run_Functions, 7
max
 Bounds, 10
min
 Bounds, 10

NUM_RUNS
Run_Functions, 7

OUTPUT_CSV
Run_Functions, 7

Population, 11
best, 11
evaluate, 11
evaluateVector, 11
generate, 11
getRow, 12
Population, 11
setRow, 12
size, 12
writeBestCSV, 12
writeFitnessCSV, 12

Population.cpp, 20
Population.h, 21
Population::bestResult, 9
fitness, 10
index, 10
x, 10

ProjectMain.cpp, 22
main, 22

rastriginFunction
Functions.cpp, 15
Functions.h, 17

repeatedLocalSearch
RepeatedLocalSearch.cpp, 23
RepeatedLocalSearch.h, 23

RepeatedLocalSearch.cpp, 22
repeatedLocalSearch, 23

RepeatedLocalSearch.h, 23
repeatedLocalSearch, 23

rosenbrockFunction
Functions.cpp, 15
Functions.h, 17

Run_Functions, 7
EXECUTABLE, 7
main, 7
NUM_RUNS, 7
OUTPUT_CSV, 7

Run_Functions.py, 24

runtimeMs
algorithmOutput, 9

schwefelFunction
Functions.cpp, 16
Functions.h, 17

setRow
Population, 12

sineEnvelopeSineWaveFunction
Functions.cpp, 16
Functions.h, 17

size
Population, 12

stretchedVSineWaveFunction