

Gormanium-rush-pentlandite

Generated by Doxygen 1.9.1

1 Gormanium-rush-pentlandite	1
1.1 Authors	1
1.2 Version	1
1.3 Date	1
1.4 Dependencies	1
1.5 Installation	1
2 README_graph	3
2.1 Installation	3
2.2 Running the software	3
3 Namespace Index	5
3.1 Namespace List	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 graph Namespace Reference	11
6.1.1 Detailed Description	11
6.1.2 Function Documentation	11
6.1.2.1 loadDataset()	11
6.1.2.2 plot()	12
6.1.3 Variable Documentation	12
6.1.3.1 final	12
6.1.3.2 k	12
6.1.3.3 output1	12
6.1.3.4 verbose	12
6.2 run_tests Namespace Reference	13
6.2.1 Variable Documentation	13
6.2.1.1 DIR	13
6.2.1.2 files	13
6.2.1.3 glob_fail	13
6.2.1.4 output	13
6.2.1.5 test_failed	13
7 Class Documentation	15
7.1 CCircuit Class Reference	15
7.1.1 Detailed Description	17
7.1.2 Constructor & Destructor Documentation	17
7.1.2.1 CCircuit() [1/2]	17

7.1.2.2 CCircuit() [2/2]	18
7.1.2.3 ~CCircuit()	18
7.1.3 Member Function Documentation	18
7.1.3.1 calculate_monetary_value()	18
7.1.3.2 calculate_monetary_value_directly()	19
7.1.3.3 Check_Validity()	20
7.1.3.4 clear()	21
7.1.3.5 clear_marks()	21
7.1.3.6 generate_random()	21
7.1.3.7 generate_vector()	21
7.1.3.8 get_circuit_vector()	22
7.1.3.9 Guassian_elimination()	22
7.1.3.10 initialize_units()	23
7.1.3.11 initialize_vector()	23
7.1.3.12 mark_units()	24
7.1.3.13 matmul()	24
7.1.3.14 monetary_direct_solver()	25
7.1.3.15 save_flow()	26
7.1.3.16 set_initial_flow()	26
7.1.3.17 set_units()	26
7.1.3.18 setup_CUnits()	27
7.1.3.19 setup_initial_parameters()	28
7.1.4 Member Data Documentation	28
7.1.4.1 c_rate_gormanium	29
7.1.4.2 c_rate_waste	29
7.1.4.3 cells	29
7.1.4.4 circuit_vector	29
7.1.4.5 Conc	29
7.1.4.6 Conc_old	29
7.1.4.7 Feed	30
7.1.4.8 feed_mass_gormanium	30
7.1.4.9 feed_mass_waste	30
7.1.4.10 find_conc	30
7.1.4.11 find_tails	30
7.1.4.12 monetary_value	30
7.1.4.13 num_units	31
7.1.4.14 source	31
7.1.4.15 t_rate_gormanium	31
7.1.4.16 t_rate_waste	31
7.1.4.17 Tails	31
7.1.4.18 Tails_old	31
7.1.4.19 units	32

7.1.4.20 value_gormanium	32
7.1.4.21 value_waste	32
7.1.4.22 worst_case	32
7.2 CUnit Class Reference	32
7.2.1 Detailed Description	33
7.2.2 Member Function Documentation	33
7.2.2.1 calculate_outflow()	33
7.2.2.2 clearAll()	34
7.2.3 Member Data Documentation	34
7.2.3.1 c_rate_gormanium	34
7.2.3.2 c_rate_waste	34
7.2.3.3 conc	34
7.2.3.4 conc_num	35
7.2.3.5 connect_from_source	35
7.2.3.6 connect_to_conc	35
7.2.3.7 connect_to_tails	35
7.2.3.8 feed	35
7.2.3.9 feed_old	35
7.2.3.10 mark	36
7.2.3.11 t_rate_gormanium	36
7.2.3.12 t_rate_waste	36
7.2.3.13 tails	36
7.2.3.14 tails_num	36
7.3 Flow Class Reference	36
7.3.1 Detailed Description	37
7.3.2 Member Function Documentation	37
7.3.2.1 clear()	37
7.3.2.2 distance()	37
7.3.2.3 plus()	38
7.3.2.4 set() [1/2]	38
7.3.2.5 set() [2/2]	38
7.3.3 Member Data Documentation	38
7.3.3.1 Gormanium	38
7.3.3.2 waste	38
7.4 GeneticAlgorithm Class Reference	39
7.4.1 Detailed Description	40
7.4.2 Constructor & Destructor Documentation	40
7.4.2.1 GeneticAlgorithm() [1/2]	41
7.4.2.2 GeneticAlgorithm() [2/2]	41
7.4.2.3 ~GeneticAlgorithm()	41
7.4.3 Member Function Documentation	41
7.4.3.1 crossover()	41

7.4.3.2 find_max_index()	42
7.4.3.3 initialize_list()	43
7.4.3.4 main_process()	43
7.4.3.5 mutate()	44
7.4.3.6 print_next_generation()	44
7.4.3.7 print_parents()	44
7.4.3.8 select_parents()	45
7.4.3.9 setup_hyper_parameters()	45
7.4.4 Member Data Documentation	46
7.4.4.1 crossover_rate	46
7.4.4.2 iter_count	46
7.4.4.3 mutation_rate	46
7.4.4.4 next_generation	47
7.4.4.5 parent_count	47
7.4.4.6 parents	47
7.4.4.7 threshold	47
7.5 node Struct Reference	47
7.5.1 Detailed Description	48
7.5.2 Member Data Documentation	48
7.5.2.1 c_val	48
7.5.2.2 p_num	48
7.5.2.3 parent	48
7.5.2.4 t_val	48
7.5.2.5 unit	48
8 File Documentation	49
8.1 includes/CCircuit.h File Reference	49
8.1.1 Function Documentation	50
8.1.1.1 mark_units()	50
8.2 includes/CUnit.h File Reference	50
8.3 includes/Genetic_Algorithm.h File Reference	51
8.4 README.md File Reference	51
8.5 src/CCircuit.cpp File Reference	51
8.5.1 Typedef Documentation	52
8.5.1.1 node	52
8.5.2 Function Documentation	52
8.5.2.1 check_concentrate()	53
8.5.2.2 check_tailing()	53
8.6 src/CUnit.cpp File Reference	54
8.7 src/Genetic_Algorithm.cpp File Reference	54
8.8 src/main.cpp File Reference	55
8.8.1 Function Documentation	56

8.8.1.1 main()	56
8.9 tests/run_tests.py File Reference	56
8.10 tests/test1.cpp File Reference	57
8.10.1 Function Documentation	57
8.10.1.1 main()	57
8.11 tests/test2.cpp File Reference	58
8.11.1 Function Documentation	58
8.11.1.1 main()	59
8.12 visualization/graph.py File Reference	59
8.13 visualization/README_graph.md File Reference	59
Index	61

Chapter 1

Gormanium-rush-pentlandite

This software aims at using in minerals processing to find a optimal circuit for two products, concentration and tailings.

1.1 Authors

Hanxiao Zhang, Jiarun Yuan, Jin Yu, Junsen Li, Yao Jiang, Yihang Liao, Yuchen Stella Wang, Yuxin Qin, Weihua Gao

1.2 Version

V1.00

1.3 Date

2021-3-26

1.4 Dependencies

python 3.7 Conda make Graphviz Use the following command to install:
`conda install graphviz python-graphviz`

1.5 Installation

Use make to complie
`make all`
`make tests`
`make runtest`

Chapter 2

README_graph

This tool provides a basic visualization method for the the vector output of the genetic algorithm code. It used the graph visualization package Graphviz <https://graphviz.gitlab.io>.

2.1 Installation

Install the packages

```
conda env create -f environment.yml
conda activate acse-4-p3
```

or you can (conda) install the packages yourself

```
conda install graphviz python-graphviz
```

2.2 Running the software

```
conda activate acsse-4-p3
python graph.py
```

(You will need to in the work directory /visualization)

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

graph	11
run_tests	13

Chapter 4

Class Index

4.1 Class List

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

CCircuit	CCircuit A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the monetary_value calculated	15
CUnit	CUnit The class for a certain unit in the Circuit	32
Flow	Flow Representing the mass flow between units	36
GeneticAlgorithm	Genetic algorithm Optimise a system represented by a specification vector	39
node	47

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

includes/CCircuit.h	49
includes/CUnit.h	50
includes/Genetic_Algorithm.h	51
src/CCircuit.cpp	51
src/CUnit.cpp	54
src/Genetic_Algorithm.cpp	54
src/main.cpp	55
tests/run_tests.py	56
tests/test1.cpp	57
tests/test2.cpp	58
visualization/graph.py	59

Chapter 6

Namespace Documentation

6.1 graph Namespace Reference

Functions

- def `loadDataset` (file, k)
- def `plot` (output)

Variables

- string `output1` = 'output.txt'
- int `k` = 21
- list `final` = [y for x in `output1` for y in x]
- `verbose`

6.1.1 Detailed Description

```
@package docstring
File: This module is used to visualize the results of the output vector
```

6.1.2 Function Documentation

6.1.2.1 loadDataset()

```
def graph.loadDataset (
    file,
    k )
```

This function is used to load the previous output vector

Parameters:

file: A txt file with output vector saved
k: 21 for most cases

Return:

A python vector nested another python vector

```
>>> loadDataset('output.txt', 21)
[[3, 4, 5, 9, 3, 1, 11, 4, 7, 9, 8, 4, 6, 8, 2, 4, 0, 9, 1, 10, 4]]
```

Definition at line 8 of file graph.py.

6.1.2.2 plot()

```
def graph.plot (  
    output )
```

This function is to plot a flow chart.

Parameters:

Output: A python vector.

Return

If the graph is plot succussfully, you can see 'Plot successfully!' in the terminal.

```
>>> plot([3, 4, 5, 9, 3, 1, 11, 4, 7, 9, 8, 4, 6, 8, 2, 4, 0, 9, 1, 10, 4])  
Plot successfully!
```

Definition at line 32 of file graph.py.

6.1.3 Variable Documentation

6.1.3.1 final

```
list graph.final = [y for x in output1 for y in x]
```

Definition at line 80 of file graph.py.

6.1.3.2 k

```
int graph.k = 21
```

Definition at line 78 of file graph.py.

6.1.3.3 output1

```
def graph.output1 = 'output.txt'
```

Definition at line 77 of file graph.py.

6.1.3.4 verbose

```
graph.verbose
```

Definition at line 87 of file graph.py.

6.2 run_tests Namespace Reference

Variables

- string `DIR` = "tests/bin/*"
- `files` = `glob.glob(DIR)`
- bool `glob_fail` = False
- `output` = `subprocess.run([file], stdout=subprocess.PIPE, universal_newlines=True)`
- bool `test_failed` = False

6.2.1 Variable Documentation

6.2.1.1 DIR

```
string run_tests.DIR = "tests/bin/*"
```

Definition at line 8 of file run_tests.py.

6.2.1.2 files

```
run_tests.files = glob.glob(DIR)
```

Definition at line 10 of file run_tests.py.

6.2.1.3 glob_fail

```
bool run_tests.glob_fail = False
```

Definition at line 12 of file run_tests.py.

6.2.1.4 output

```
run_tests.output = subprocess.run([file], stdout=subprocess.PIPE, universal_newlines=True)
```

Definition at line 17 of file run_tests.py.

6.2.1.5 test_failed

```
bool run_tests.test_failed = False
```

Definition at line 18 of file run_tests.py.

Chapter 7

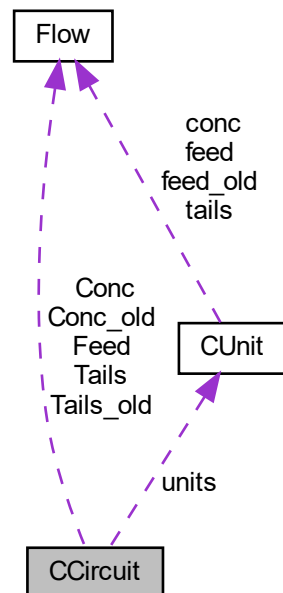
Class Documentation

7.1 CCircuit Class Reference

CCircuit A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the monetary_value calculated.

```
#include <CCircuit.h>
```

Collaboration diagram for CCircuit:



Public Member Functions

- [CCircuit](#) ()
- [CCircuit](#) (const [CCircuit](#) &to_copy)
- [~CCircuit](#) ()
- void [initialize_units](#) ()
initialize unit according to unit number
- void [set_initial_flow](#) (int unit)
- void [clear](#) ()
clear unit, concentration, tailing
- void [save_flow](#) ()
- void [clear_marks](#) ()
- void [calculate_monetary_value](#) (bool &diverge, double tolerance=1e-12, int max_iter=500)
Calculate monetary value, using iteration method.
- void [set_units](#) (int *vec)
set up units for a [CCircuit](#) class according to vector value
- void [initialize_vector](#) ()
initialize vector
- void [get_circuit_vector](#) ()
Extract the configuration vector from a [CCircuit](#) class.
- bool [Check_Validity](#) ()
this function is to check validity
- void [mark_units](#) (int unit_num)
- void [generate_random](#) ()
generate a random configuration for a [CCircuit](#) object, which is likely to be valid
- void [calculate_monetary_value_directly](#) ()
The direct linear solver to calculate the monetary value of the concentration for a [CCircuit](#) object, overwriting the monetary_value it stored.
- int * [generate_vector](#) ()
return a copy for the configuration vector for a [CCircuit](#) object.

Static Public Member Functions

- static void [setup_initial_parameters](#) (const int num_units, const double feed_mass_gormanium, const double feed_mass_waste, const double c_rate_gormanium, const double c_rate_waste, const double t_rate_gormanium, const double t_rate_waste, const double value_gormanium, const double value_waste)
set initial parameters
- static void [setup_CUnits](#) ()
set [CUnit](#) parameters according to [CCircuit](#)
- static double [monetary_direct_solver](#) (int *circuit_vector, int len)
- static void [Guassian_elimination](#) (double **a, double *b, int n)
Do Gaussian elimination, for solving the linear equation $Ax = b$.
- static void [matmul](#) (double **a, double *b, int n)
performing left Matrix-vector multiplication

Public Attributes

- CUnit * `units` = NULL
- Flow `Feed`
- Flow `Conc`
- Flow `Tails`
- Flow `Conc_old`
- Flow `Tails_old`
- int `source`
- double `monetary_value`
- bool `find_conc`
- bool `find_tails`
- int * `circuit_vector` = nullptr

Static Public Attributes

- static int `num_units`
- static int `cells`
- static double `feed_mass_gormanium`
- static double `feed_mass_waste`
- static double `c_rate_gormanium`
- static double `c_rate_waste`
- static double `t_rate_gormanium`
- static double `t_rate_waste`
- static double `value_gormanium`
- static double `value_waste`
- static double `worst_case`

7.1.1 Detailed Description

CCircuit A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the `monetary_value` calculated.

Definition at line 11 of file `CCircuit.h`.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 CCircuit() [1/2]

```
CCircuit::CCircuit ( ) [inline]
```

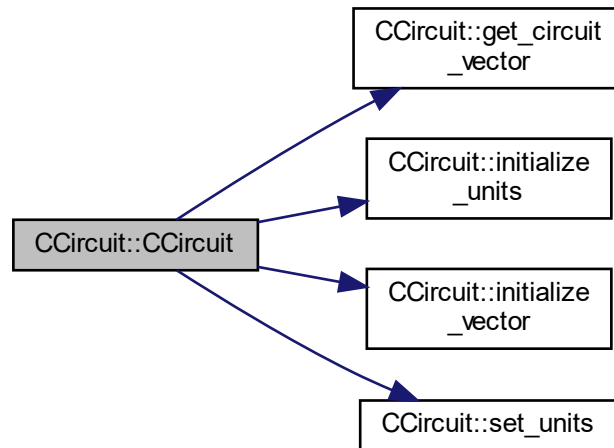
Definition at line 40 of file `CCircuit.h`.

7.1.2.2 CCircuit() [2/2]

```
CCircuit::CCircuit (
    const CCircuit & to_copy ) [inline]
```

Definition at line 41 of file CCircuit.h.

Here is the call graph for this function:



7.1.2.3 ~CCircuit()

```
CCircuit::~~CCircuit ( ) [inline]
```

Definition at line 51 of file CCircuit.h.

7.1.3 Member Function Documentation

7.1.3.1 calculate_monetary_value()

```
void CCircuit::calculate_monetary_value (
    bool & diverge,
    double tolerance = 1e-12,
    int max_iter = 500 )
```

Calculate monetary value, using iteration method.

Parameters

<i>diverge</i>	boolean indicating whether flow in the circuit is diverging or not.
<i>tolerance</i>	which is setting for the stopping criteria, if the change over time step is small, we achieve our balance state.
<i>max_iter</i>	we do not want our method taking too much time, if some case converge too slow and met max iteraton times we set, the method would quit.

Returns

void

Definition at line 143 of file CCircuit.cpp.

7.1.3.2 calculate_monetary_value_directly()

```
void CCircuit::calculate_monetary_value_directly ( )
```

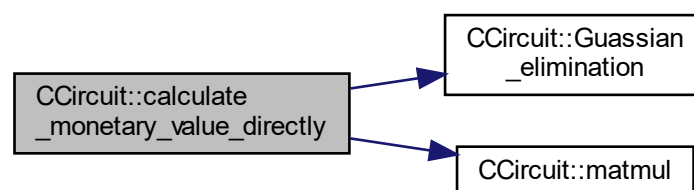
The direct linear solver to calculate the monetary value of the concentration for a [CCircuit](#) object, overwrtng the monetary_value it stored.

Returns

void

Definition at line 511 of file CCircuit.cpp.

Here is the call graph for this function:



7.1.3.3 Check_Validity()

```
bool CCircuit::Check_Validity ( )
```

this function is to check validity

Returns

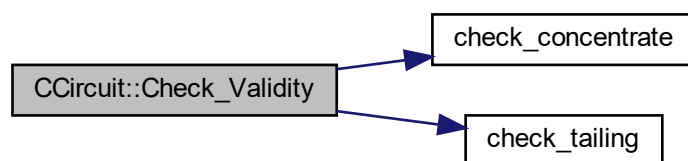
true if valid

Note

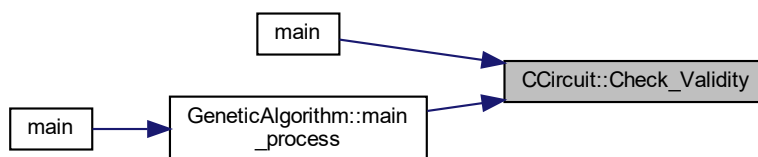
this function firstly construct the units array, if the unit is self connected, directly return false, then the function check whether all units are connected to tail and concentrate.

Definition at line 312 of file CCircuit.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.4 clear()

```
void CCircuit::clear ( )
```

clear unit, concentration, tailing

Returns

void

Definition at line 108 of file CCircuit.cpp.

7.1.3.5 clear_marks()

```
void CCircuit::clear_marks ( )
```

7.1.3.6 generate_random()

```
void CCircuit::generate_random ( )
```

generate a random configuration for a [CCircuit](#) object, which is likely to be valid

Returns

void

Definition at line 463 of file CCircuit.cpp.

7.1.3.7 generate_vector()

```
int * CCircuit::generate_vector ( )
```

return a copy for the configuration vector for a [CCircuit](#) object.

Returns

vector

Definition at line 492 of file CCircuit.cpp.

7.1.3.8 get_circuit_vector()

```
void CCircuit::get_circuit_vector ( )
```

Extract the configuration vector from a [CCircuit](#) class.

Returns

void

Definition at line 253 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.9 Guassian_elimination()

```
void CCircuit::Guassian_elimination (
    double ** a,
    double * b,
    int n ) [static]
```

Do Gaussian elimination, for solving the linear equation $Ax = b$.

Parameters

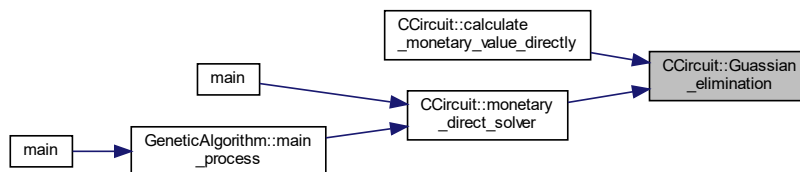
<i>a</i>	the matrix A on the left hand side
<i>b</i>	the RHS vector
<i>n</i>	the size of the problem

Returns

void

Definition at line 386 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.10 initialize_units()

```
void CCircuit::initialize_units ( )
```

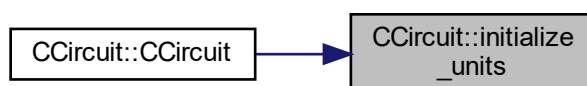
initialize unit according to unit number

Returns

void

Definition at line 90 of file `CCircuit.cpp`.

Here is the caller graph for this function:



7.1.3.11 initialize_vector()

```
void CCircuit::initialize_vector ( )
```

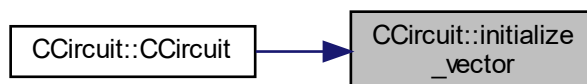
initialize vector

Returns

void

Definition at line 98 of file CCircuit.cpp.

Here is the caller graph for this function:

**7.1.3.12 mark_units()**

```
void CCircuit::mark_units (
    int unit_num )
```

7.1.3.13 matmul()

```
void CCircuit::matmul (
    double ** a,
    double * b,
    int n ) [static]
```

performing left Matrix-vector multiplication

Parameters

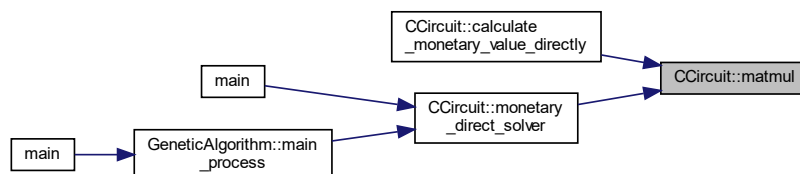
<i>a</i>	the matrix
<i>b</i>	the vector to be mutiplicated
<i>n</i>	the size of the problem

Returns

void

Definition at line 437 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.14 monetary_direct_solver()

```
double CCircuit::monetary_direct_solver (
    int * circuit_vector,
    int len ) [static]
```

Parameters

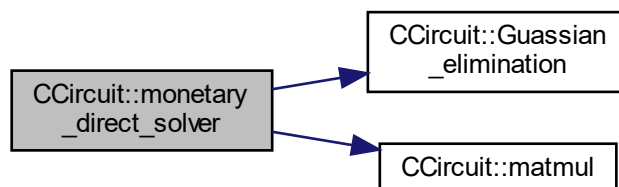
<i>circuit_vector</i>	The direct linear solver to calculate the monetary value of the concentration for a configuration vector, returning the monetary_value as a double.
<i>len</i>	The length of the Circuit configuration vector

Returns

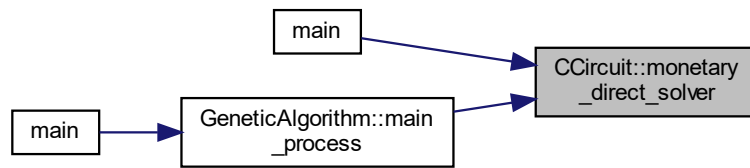
momentary value

Definition at line 561 of file CCircuit.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.15 `save_flow()`

```
void CCircuit::save_flow ( )
```

Definition at line 116 of file `CCircuit.cpp`.

7.1.3.16 `set_initial_flow()`

```
void CCircuit::set_initial_flow (
    int unit )
```

7.1.3.17 `set_units()`

```
void CCircuit::set_units (
    int * vec )
```

set up units for a [CCircuit](#) class according to vector value

Parameters

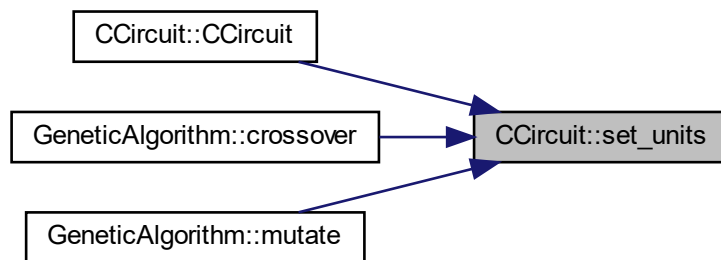
<code>vec</code>	the input vector, indicating a configuration.
------------------	---

Returns

`void`

Definition at line 237 of file `CCircuit.cpp`.

Here is the caller graph for this function:



7.1.3.18 `setup_CUnits()`

```
void CCircuit::setup_CUnits ( ) [static]
```

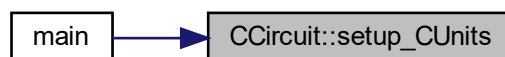
set [CUnit](#) parameters according to [CCircuit](#)

Returns

void

Definition at line 79 of file `CCircuit.cpp`.

Here is the caller graph for this function:



7.1.3.19 setup_initial_parameters()

```
void CCircuit::setup_initial_parameters (
    const int num_units,
    const double feed_mass_gormanium,
    const double feed_mass_waste,
    const double c_rate_gormanium,
    const double c_rate_waste,
    const double t_rate_gormanium,
    const double t_rate_waste,
    const double value_gormanium,
    const double value_waste ) [static]
```

set initial parameters

Parameters

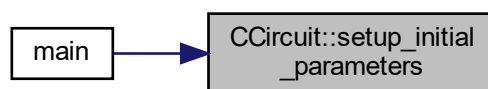
<i>num_units</i>	
<i>feed_mass_gormanium</i>	
<i>feed_mass_waste</i>	
<i>c_rate_gormanium</i>	
<i>c_rate_waste</i>	
<i>t_rate_gormanium</i>	
<i>t_rate_waste</i>	
<i>value_gormanium</i>	
<i>value_waste</i>	

Returns

void

Definition at line 58 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.4 Member Data Documentation

7.1.4.1 c_rate_gormanium

```
double CCircuit::c_rate_gormanium [static]
```

Definition at line 19 of file CCircuit.h.

7.1.4.2 c_rate_waste

```
double CCircuit::c_rate_waste [static]
```

Definition at line 20 of file CCircuit.h.

7.1.4.3 cells

```
int CCircuit::cells [static]
```

Definition at line 14 of file CCircuit.h.

7.1.4.4 circuit_vector

```
int* CCircuit::circuit_vector = nullptr
```

Definition at line 31 of file CCircuit.h.

7.1.4.5 Conc

```
Flow CCircuit::Conc
```

Definition at line 16 of file CCircuit.h.

7.1.4.6 Conc_old

```
Flow CCircuit::Conc_old
```

Definition at line 16 of file CCircuit.h.

7.1.4.7 Feed

`Flow CCircuit::Feed`

Definition at line 16 of file CCircuit.h.

7.1.4.8 feed_mass_gormanium

`double CCircuit::feed_mass_gormanium [static]`

Definition at line 17 of file CCircuit.h.

7.1.4.9 feed_mass_waste

`double CCircuit::feed_mass_waste [static]`

Definition at line 18 of file CCircuit.h.

7.1.4.10 find_conc

`bool CCircuit::find_conc`

Definition at line 29 of file CCircuit.h.

7.1.4.11 find_tails

`bool CCircuit::find_tails`

Definition at line 29 of file CCircuit.h.

7.1.4.12 monetary_value

`double CCircuit::monetary_value`

Definition at line 27 of file CCircuit.h.

7.1.4.13 num_units

```
int CCircuit::num_units [static]
```

Definition at line 13 of file CCircuit.h.

7.1.4.14 source

```
int CCircuit::source
```

Definition at line 26 of file CCircuit.h.

7.1.4.15 t_rate_gormanium

```
double CCircuit::t_rate_gormanium [static]
```

Definition at line 21 of file CCircuit.h.

7.1.4.16 t_rate_waste

```
double CCircuit::t_rate_waste [static]
```

Definition at line 22 of file CCircuit.h.

7.1.4.17 Tails

```
Flow CCircuit::Tails
```

Definition at line 16 of file CCircuit.h.

7.1.4.18 Tails_old

```
Flow CCircuit::Tails_old
```

Definition at line 16 of file CCircuit.h.

7.1.4.19 units

```
CUnit* CCircuit::units = NULL
```

Definition at line 15 of file CCircuit.h.

7.1.4.20 value_gormanium

```
double CCircuit::value_gormanium [static]
```

Definition at line 23 of file CCircuit.h.

7.1.4.21 value_waste

```
double CCircuit::value_waste [static]
```

Definition at line 24 of file CCircuit.h.

7.1.4.22 worst_case

```
double CCircuit::worst_case [static]
```

Definition at line 25 of file CCircuit.h.

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

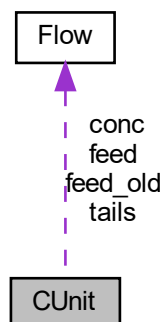
- includes/CCircuit.h
- src/CCircuit.cpp

7.2 CUnit Class Reference

[CUnit](#) The class for a certain unit in the Circuit.

```
#include <CUnit.h>
```

Collaboration diagram for CUnit:



Public Member Functions

- void [calculate_outflow](#) ()
Calculate the mass of Gormanium and waste for the concentration and tails outflows by a given feed.
- void [clearAll](#) ()
clear all flows stored in a certain unit.

Public Attributes

- int [conc_num](#)
- int [tails_num](#)
- bool [mark](#)
- [Flow](#) [feed](#)
- [Flow](#) [feed_old](#)
- [Flow](#) [conc](#)
- [Flow](#) [tails](#)
- bool [connect_to_conc](#)
- bool [connect_to_tails](#)
- bool [connect_from_source](#)

Static Public Attributes

- static double [c_rate_gormanium](#)
- static double [c_rate_waste](#)
- static double [t_rate_gormanium](#)
- static double [t_rate_waste](#)

7.2.1 Detailed Description

[CUnit](#) The class for a certain unit in the Circuit.

Definition at line 52 of file CUnit.h.

7.2.2 Member Function Documentation

7.2.2.1 [calculate_outflow\(\)](#)

```
void CUnit::calculate_outflow ( )
```

Calculate the mass of Gormanium and waste for the concentration and tails outflows by a given feed.

Returns

void

Definition at line 8 of file CUnit.cpp.

7.2.2.2 clearAll()

```
void CUnit::clearAll ( )
```

clear all flows stored in a certain unit.

Returns

void

Definition at line 20 of file CUnit.cpp.

Here is the call graph for this function:



7.2.3 Member Data Documentation

7.2.3.1 c_rate_gormanium

```
double CUnit::c_rate_gormanium [static]
```

Definition at line 61 of file CUnit.h.

7.2.3.2 c_rate_waste

```
double CUnit::c_rate_waste [static]
```

Definition at line 62 of file CUnit.h.

7.2.3.3 conc

```
Flow CUnit::conc
```

Definition at line 65 of file CUnit.h.

7.2.3.4 conc_num

```
int CUnit::conc_num
```

Definition at line 55 of file CUnit.h.

7.2.3.5 connect_from_source

```
bool CUnit::connect_from_source
```

Definition at line 73 of file CUnit.h.

7.2.3.6 connect_to_conc

```
bool CUnit::connect_to_conc
```

Definition at line 71 of file CUnit.h.

7.2.3.7 connect_to_tails

```
bool CUnit::connect_to_tails
```

Definition at line 72 of file CUnit.h.

7.2.3.8 feed

```
Flow CUnit::feed
```

Definition at line 65 of file CUnit.h.

7.2.3.9 feed_old

```
Flow CUnit::feed_old
```

Definition at line 65 of file CUnit.h.

7.2.3.10 mark

```
bool CUnit::mark
```

Definition at line 59 of file CUnit.h.

7.2.3.11 t_rate_gormanium

```
double CUnit::t_rate_gormanium [static]
```

Definition at line 63 of file CUnit.h.

7.2.3.12 t_rate_waste

```
double CUnit::t_rate_waste [static]
```

Definition at line 64 of file CUnit.h.

7.2.3.13 tails

```
Flow CUnit::tails
```

Definition at line 65 of file CUnit.h.

7.2.3.14 tails_num

```
int CUnit::tails_num
```

Definition at line 57 of file CUnit.h.

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

- [includes/CUnit.h](#)
- [src/CCircuit.cpp](#)
- [src/CUnit.cpp](#)

7.3 Flow Class Reference

[Flow](#) Representing the mass flow between units.

```
#include <CUnit.h>
```

Public Member Functions

- void [set](#) (double G, double W)
- void [set](#) ([Flow](#) flow)
- void [plus](#) ([Flow](#) feed)
- void [clear](#) ()
- double [distance](#) ([Flow](#) flow1)

Public Attributes

- double [Gormanium](#) = 0
- double [waste](#) = 0

7.3.1 Detailed Description

[Flow](#) Representing the mass flow between units.

Definition at line 9 of file CUnit.h.

7.3.2 Member Function Documentation

7.3.2.1 [clear\(\)](#)

```
void Flow::clear ( ) [inline]
```

Definition at line 34 of file CUnit.h.

Here is the caller graph for this function:



7.3.2.2 [distance\(\)](#)

```
double Flow::distance (
    Flow flow1 ) [inline]
```

Definition at line 40 of file CUnit.h.

7.3.2.3 plus()

```
void Flow::plus (
    Flow feed ) [inline]
```

Definition at line 28 of file CUnit.h.

7.3.2.4 set() [1/2]

```
void Flow::set (
    double G,
    double W ) [inline]
```

Definition at line 16 of file CUnit.h.

7.3.2.5 set() [2/2]

```
void Flow::set (
    Flow flow ) [inline]
```

Definition at line 22 of file CUnit.h.

7.3.3 Member Data Documentation

7.3.3.1 Gormanium

```
double Flow::Gormanium = 0
```

Definition at line 12 of file CUnit.h.

7.3.3.2 waste

```
double Flow::waste = 0
```

Definition at line 13 of file CUnit.h.

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

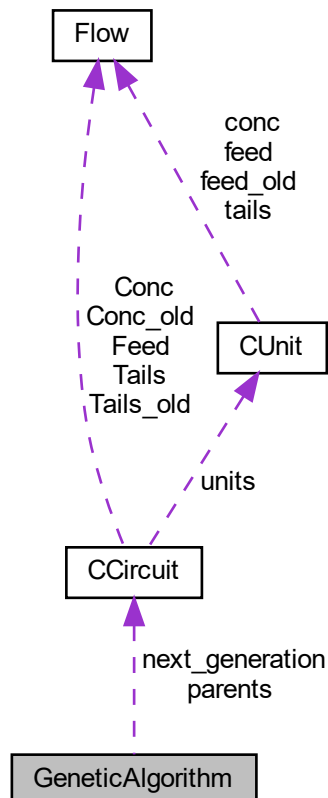
- includes/[CUnit.h](#)

7.4 GeneticAlgorithm Class Reference

Genetic algorithm Optimise a system represented by a specification vector.

```
#include <Genetic_Algorithm.h>
```

Collaboration diagram for GeneticAlgorithm:



Public Member Functions

- void `initialize_list` (int *vec)
Generate one random `CCircuit` vector.
- int `select_parents` (`CCircuit` *parents, int n)
Use tournament selection select parents.
- void `find_max_index` (`CCircuit` *parents, int &index, double &value)
Find the best vector.
- void `crossover` (`CCircuit` *mother, `CCircuit` *father)
@mother and @father are selected to cross, a random point in the vector is chosen and all of the values before that point are swapped with the corresponding points in the other vector.
- void `mutate` (`CCircuit` *parent)
Go over each of the numbers in the vector and decide whether to mutate them.

- void `main_process ()`
The main process of the basic genetic algorithm.
- void `print_parents ()`
Print value in parents.
- void `print_next_generation ()`
Print value in next generation.
- `GeneticAlgorithm ()`
constructor
- `GeneticAlgorithm (int iter_count, int threshold)`
constructor
- `~GeneticAlgorithm ()`
destructor

Static Public Member Functions

- static void `setup_hyper_parameters (int parent_count, double crossover_rate, double mutation_rate)`
set up hyper parameters

Public Attributes

- `CCircuit * parents = nullptr`
collection of valid circuits
- `CCircuit * next_generation = nullptr`
collection of valid circuits in next generation
- int `iter_count`
a set number of iterations
- double `threshold`
the minimum performance gap between two generation of best vector

Static Public Attributes

- static int `parent_count`
the number of offspring/parent n that are evaluated in each generation
- static double `crossover_rate`
the probability of crossing selected parents rather than passing them into the mutation step unchanged
- static double `mutation_rate`
the rate at which mutations are introduced

7.4.1 Detailed Description

Genetic algorithm Optimise a system represented by a specification vector.

Definition at line 6 of file Genetic_Algorithm.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 GeneticAlgorithm() [1/2]

```
GeneticAlgorithm::GeneticAlgorithm ( )
```

constructor

Definition at line 278 of file Genetic_Algorithm.cpp.

7.4.2.2 GeneticAlgorithm() [2/2]

```
GeneticAlgorithm::GeneticAlgorithm (
    int iter_count,
    int threshold )
```

constructor

Parameters

<i>iter_count</i>	max iteration
<i>threshold</i>	the minimum performance gap between two generation of best vector

Definition at line 289 of file Genetic_Algorithm.cpp.

7.4.2.3 ~GeneticAlgorithm()

```
GeneticAlgorithm::~~GeneticAlgorithm ( )
```

destructor

Definition at line 298 of file Genetic_Algorithm.cpp.

7.4.3 Member Function Documentation

7.4.3.1 crossover()

```
void GeneticAlgorithm::crossover (
    CCircuit * mother,
    CCircuit * father )
```

@mother and @father are selected to cross, a random point in the vector is chosen and all of the values before that point are swapped with the corresponding points in the other vector.

Parameters

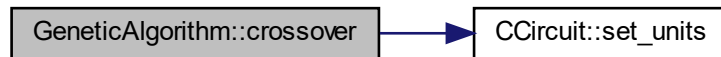
<i>mother</i>	selected parent
<i>father</i>	selected parent

Returns

void

Definition at line 100 of file Genetic_Algorithm.cpp.

Here is the call graph for this function:

**7.4.3.2 find_max_index()**

```
void GeneticAlgorithm::find_max_index (
    CCircuit * parents,
    int & index,
    double & value )
```

Find the best vector.

Parameters

<i>parents</i>	collection of valid circuits
<i>index</i>	index of best vector
<i>value</i>	value of best vector

Returns

void

Definition at line 83 of file Genetic_Algorithm.cpp.

7.4.3.3 initialize_list()

```
void GeneticAlgorithm::initialize_list (
    int * vec )
```

Generate one random [CCircuit](#) vector.

Parameters

<i>vec</i>	
------------	--

Definition at line 41 of file Genetic_Algorithm.cpp.

7.4.3.4 main_process()

```
void GeneticAlgorithm::main_process ( )
```

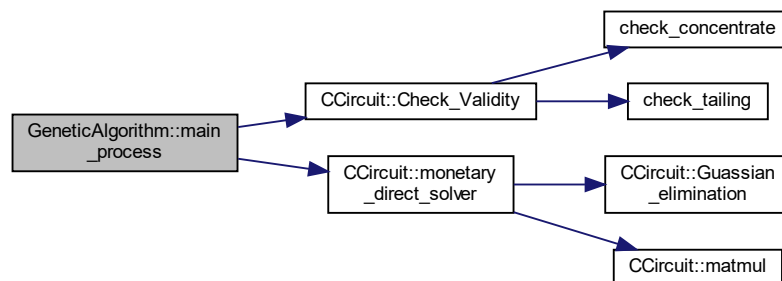
The main process of the basic genetic algorithm.

Returns

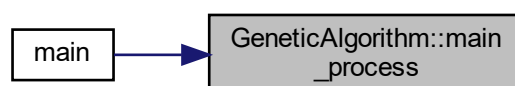
void

Definition at line 137 of file Genetic_Algorithm.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.5 mutate()

```
void GeneticAlgorithm::mutate (
    CCircuit * parent )
```

Go over each of the numbers in the vector and decide whether to mutate them.

Parameters

<i>parent</i>	
---------------	--

Returns

void

Definition at line 124 of file Genetic_Algorithm.cpp.

Here is the call graph for this function:



7.4.3.6 print_next_generation()

```
void GeneticAlgorithm::print_next_generation ( )
```

Print value in next generation.

Returns

void

Definition at line 263 of file Genetic_Algorithm.cpp.

7.4.3.7 print_parents()

```
void GeneticAlgorithm::print_parents ( )
```

Print value in parents.

Returns

void

Definition at line 247 of file Genetic_Algorithm.cpp.

7.4.3.8 select_parents()

```
int GeneticAlgorithm::select_parents (
    CCircuit * parents,
    int n )
```

Use tournament selection select parents.

Parameters

<i>parents</i>	collection of valid circuits
<i>n</i>	select n from parent then use largest one as parent

Returns

parent

Definition at line 61 of file Genetic_Algorithm.cpp.

7.4.3.9 setup_hyper_parameters()

```
void GeneticAlgorithm::setup_hyper_parameters (
    int parent_count,
    double crossover_rate,
    double mutation_rate ) [static]
```

set up hyper parameters

Parameters

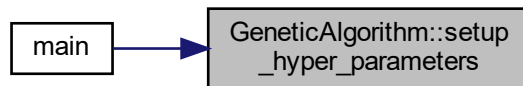
<i>parent_count</i>	the number of offspring n that are evaluated in each generation;
<i>crossover_rate</i>	the probability of crossing selected parents rather than passing them into the mutation step unchanged (a recommended range is between 0.8 and 1)
<i>mutation_rate</i>	the rate at which mutations are introduced (recommended probabilities of 1% or lower).

Returns

void

Definition at line 31 of file Genetic_Algorithm.cpp.

Here is the caller graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 crossover_rate

```
double GeneticAlgorithm::crossover_rate [static]
```

the probability of crossing selected parents rather than passing them into the mutation step unchanged

Definition at line 47 of file Genetic_Algorithm.h.

7.4.4.2 iter_count

```
int GeneticAlgorithm::iter_count
```

a set number of iterations

Definition at line 37 of file Genetic_Algorithm.h.

7.4.4.3 mutation_rate

```
double GeneticAlgorithm::mutation_rate [static]
```

the rate at which mutations are introduced

Definition at line 50 of file Genetic_Algorithm.h.

7.4.4.4 next_generation

```
CCircuit* GeneticAlgorithm::next_generation = nullptr
```

collection of valid circuits in next generation

Definition at line 33 of file Genetic_Algorithm.h.

7.4.4.5 parent_count

```
int GeneticAlgorithm::parent_count [static]
```

the number of offspring/parent n that are evaluated in each generation

Definition at line 43 of file Genetic_Algorithm.h.

7.4.4.6 parents

```
CCircuit* GeneticAlgorithm::parents = nullptr
```

collection of valid circuits

Definition at line 31 of file Genetic_Algorithm.h.

7.4.4.7 threshold

```
double GeneticAlgorithm::threshold
```

the minimum performance gap between two generation of best vector

Definition at line 40 of file Genetic_Algorithm.h.

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

- includes/[Genetic_Algorithm.h](#)
- src/[Genetic_Algorithm.cpp](#)

7.5 node Struct Reference

Public Attributes

- bool [c_val](#) = false
- bool [t_val](#) = false
- int [unit](#)
- int [p_num](#) = 0
- int * [parent](#)

7.5.1 Detailed Description

Definition at line 9 of file CCircuit.cpp.

7.5.2 Member Data Documentation

7.5.2.1 c_val

```
bool node::c_val = false
```

Definition at line 11 of file CCircuit.cpp.

7.5.2.2 p_num

```
int node::p_num = 0
```

Definition at line 14 of file CCircuit.cpp.

7.5.2.3 parent

```
int* node::parent
```

Definition at line 15 of file CCircuit.cpp.

7.5.2.4 t_val

```
bool node::t_val = false
```

Definition at line 12 of file CCircuit.cpp.

7.5.2.5 unit

```
int node::unit
```

Definition at line 13 of file CCircuit.cpp.

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

- [src/CCircuit.cpp](#)

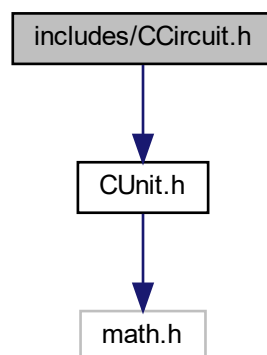
Chapter 8

File Documentation

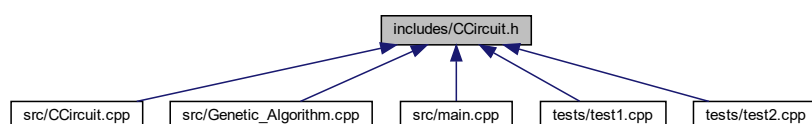
8.1 includes/CCircuit.h File Reference

```
#include "CUnit.h"
```

Include dependency graph for CCircuit.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CCircuit](#)

[CCircuit](#) A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the monetary_value calculated.

Functions

- void [mark_units](#) (int unit_num)

8.1.1 Function Documentation

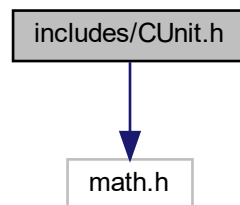
8.1.1.1 mark_units()

```
void mark_units (
    int unit_num )
```

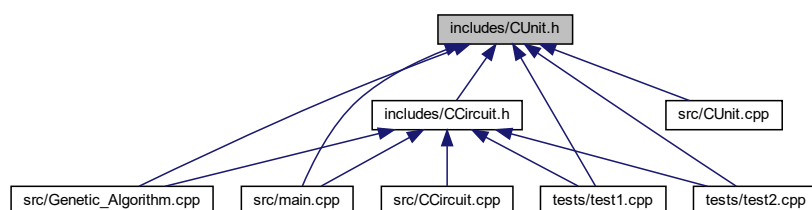
8.2 includes/CUnit.h File Reference

```
#include <math.h>
```

Include dependency graph for CUnit.h:



This graph shows which files directly or indirectly include this file:

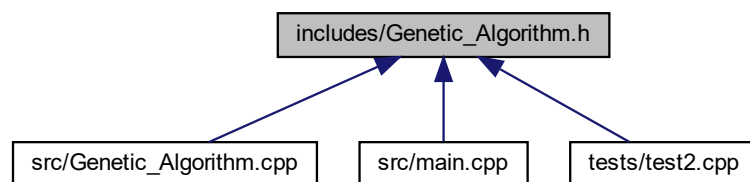


Classes

- class [Flow](#)
Flow Representing the mass flow between units.
- class [CUnit](#)
CUnit The class for a certain unit in the Circuit.

8.3 includes/Genetic_Algorithm.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [GeneticAlgorithm](#)
Genetic algorithm Optimise a system represented by a specification vector.

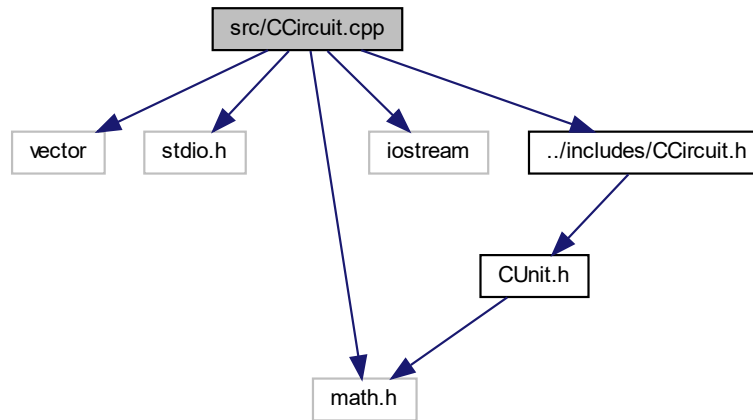
8.4 README.md File Reference

8.5 src/CCircuit.cpp File Reference

```
#include <vector>
#include <stdio.h>
#include <math.h>
#include <iostream>
```

```
#include "../includes/CCircuit.h"
```

Include dependency graph for CCircuit.cpp:



Classes

- struct [node](#)

Typedefs

- typedef struct [node](#) [node](#)

Functions

- void [check_tailing](#) (int tail, [node](#) *nodes)
this function is to check whether the units connect to the tailing
- void [check_concentrate](#) (int con, [node](#) *nodes)
this function is to check whether the units connect to the tailing

8.5.1 Typedef Documentation

8.5.1.1 node

```
typedef struct node node
```

8.5.2 Function Documentation

8.5.2.1 check_concentrate()

```
void check_concentrate (
    int con,
    node * nodes )
```

this function is to check whether the units connect to the tailing

Parameters

<i>concentrate</i>	concentrate number to locate the concentrate node
<i>nodes</i>	address of nodes array

Returns

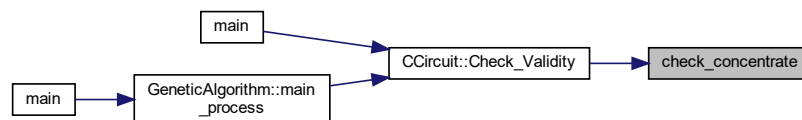
stop while backs to the concentrate

Note

this function backs recursively check the parents units, start from tail, and mark parent units true which indicates that unit has connected to concentrate

Definition at line 293 of file CCircuit.cpp.

Here is the caller graph for this function:



8.5.2.2 check_tailing()

```
void check_tailing (
    int tail,
    node * nodes )
```

this function is to check whether the units connect to the tailing

Parameters

<i>tail</i>	tail number to locate the tail node
<i>nodes</i>	address of nodes array

Returns

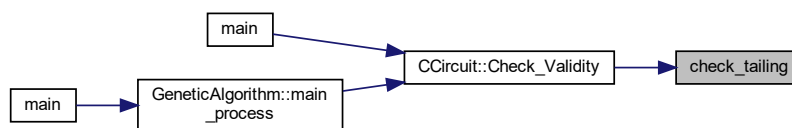
stop while backs to the tail

Note

this function backs recursively check the parents units, start from tail, and mark parent units true which indicates that unit has connected to tail

Definition at line 272 of file CCircuit.cpp.

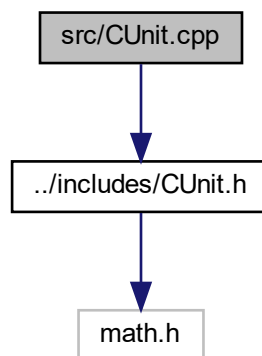
Here is the caller graph for this function:



8.6 src/CUnit.cpp File Reference

```
#include "../includes/CUnit.h"
```

Include dependency graph for CUnit.cpp:



8.7 src/Genetic_Algorithm.cpp File Reference

```
#include <cmath>
```

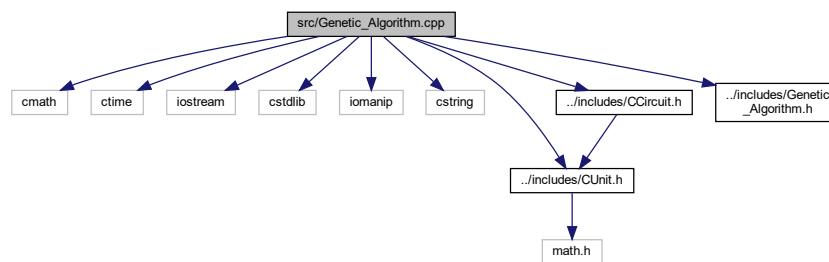
```
#include <ctime>
```

```

#include <iostream>
#include <cstdlib>
#include <iomanip>
#include <cstring>
#include "../includes/CUnit.h"
#include "../includes/CCircuit.h"
#include "../includes/Genetic_Algorithm.h"

```

Include dependency graph for Genetic_Algorithm.cpp:



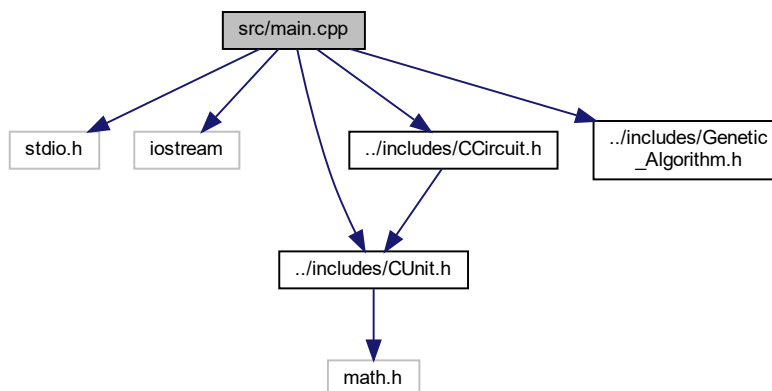
8.8 src/main.cpp File Reference

```

#include <stdio.h>
#include <iostream>
#include "../includes/CUnit.h"
#include "../includes/CCircuit.h"
#include "../includes/Genetic_Algorithm.h"

```

Include dependency graph for main.cpp:



Functions

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

8.8.1 Function Documentation

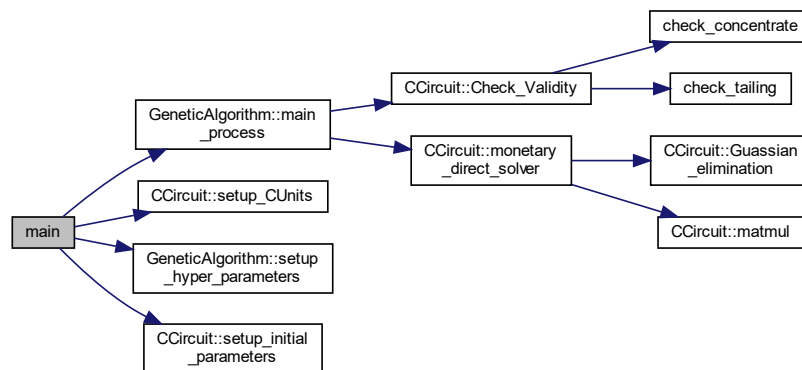
8.8.1.1 main()

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

the number of unit

Definition at line 13 of file main.cpp.

Here is the call graph for this function:



8.9 tests/run_tests.py File Reference

Namespaces

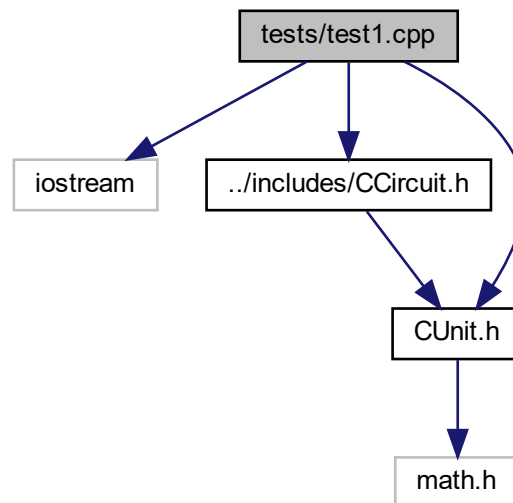
- [run_tests](#)

Variables

- string [run_tests.DIR](#) = "tests/bin/*"
- [run_tests.files](#) = glob.glob(DIR)
- bool [run_tests.glob_fail](#) = False
- [run_tests.output](#) = subprocess.run([file], stdout=subprocess.PIPE, universal_newlines=True)
- bool [run_tests.test_failed](#) = False

8.10 tests/test1.cpp File Reference

```
#include <iostream>
#include "../includes/CCircuit.h"
#include "../includes/CUnit.h"
Include dependency graph for test1.cpp:
```



Functions

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

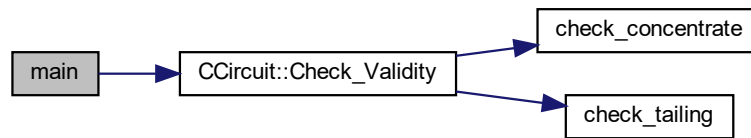
8.10.1 Function Documentation

8.10.1.1 main()

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

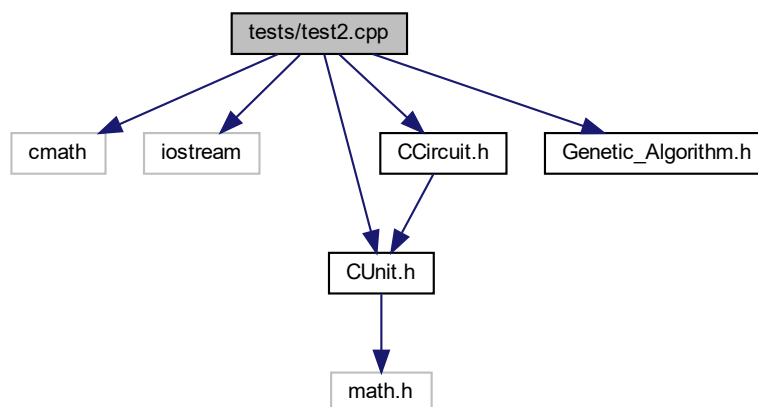
Definition at line 8 of file `test1.cpp`.

Here is the call graph for this function:



8.11 tests/test2.cpp File Reference

```
#include <cmath>
#include <iostream>
#include "CUnit.h"
#include "CCircuit.h"
#include "Genetic_Algorithm.h"
Include dependency graph for test2.cpp:
```



Functions

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

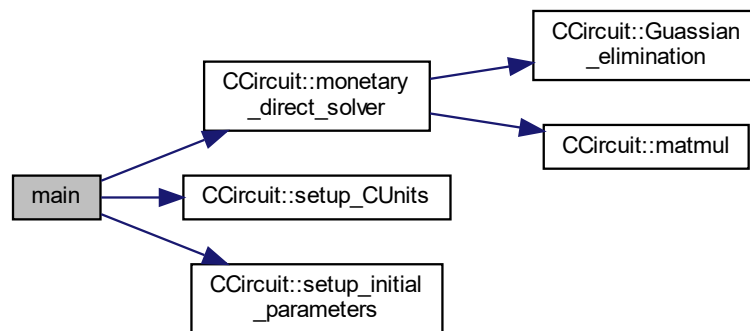
8.11.1 Function Documentation

8.11.1.1 main()

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

Definition at line 9 of file test2.cpp.

Here is the call graph for this function:



8.12 visualization/graph.py File Reference

Namespaces

- [graph](#)

Functions

- def [graph.loadDataset](#) (file, k)
- def [graph.plot](#) (output)

Variables

- string [graph.output1](#) = 'output.txt'
- int [graph.k](#) = 21
- list [graph.final](#) = [y for x in output1 for y in x]
- [graph.verbose](#)

8.13 visualization/README_graph.md File Reference

Index

- ~CCircuit
 - CCircuit, [18](#)
- ~GeneticAlgorithm
 - GeneticAlgorithm, [41](#)
- c_rate_gormanium
 - CCircuit, [28](#)
 - CUnit, [34](#)
- c_rate_waste
 - CCircuit, [29](#)
 - CUnit, [34](#)
- c_val
 - node, [48](#)
- calculate_monetary_value
 - CCircuit, [18](#)
- calculate_monetary_value_directly
 - CCircuit, [19](#)
- calculate_outflow
 - CUnit, [33](#)
- CCircuit, [15](#)
 - ~CCircuit, [18](#)
 - c_rate_gormanium, [28](#)
 - c_rate_waste, [29](#)
 - calculate_monetary_value, [18](#)
 - calculate_monetary_value_directly, [19](#)
 - CCircuit, [17](#)
 - cells, [29](#)
 - Check_Validity, [19](#)
 - circuit_vector, [29](#)
 - clear, [20](#)
 - clear_marks, [21](#)
 - Conc, [29](#)
 - Conc_old, [29](#)
 - Feed, [29](#)
 - feed_mass_gormanium, [30](#)
 - feed_mass_waste, [30](#)
 - find_conc, [30](#)
 - find_tails, [30](#)
 - generate_random, [21](#)
 - generate_vector, [21](#)
 - get_circuit_vector, [21](#)
 - Guassian_elimination, [22](#)
 - initialize_units, [23](#)
 - initialize_vector, [23](#)
 - mark_units, [24](#)
 - matmul, [24](#)
 - monetary_direct_solver, [25](#)
 - monetary_value, [30](#)
 - num_units, [30](#)
 - save_flow, [26](#)
 - set_initial_flow, [26](#)
 - set_units, [26](#)
 - setup_CUnits, [27](#)
 - setup_initial_parameters, [27](#)
 - source, [31](#)
 - t_rate_gormanium, [31](#)
 - t_rate_waste, [31](#)
 - Tails, [31](#)
 - Tails_old, [31](#)
 - units, [31](#)
 - value_gormanium, [32](#)
 - value_waste, [32](#)
 - worst_case, [32](#)
- CCircuit.cpp
 - check_concentrate, [52](#)
 - check_tailing, [53](#)
 - node, [52](#)
- CCircuit.h
 - mark_units, [50](#)
- cells
 - CCircuit, [29](#)
- check_concentrate
 - CCircuit.cpp, [52](#)
- check_tailing
 - CCircuit.cpp, [53](#)
- Check_Validity
 - CCircuit, [19](#)
- circuit_vector
 - CCircuit, [29](#)
- clear
 - CCircuit, [20](#)
 - Flow, [37](#)
- clear_marks
 - CCircuit, [21](#)
- clearAll
 - CUnit, [33](#)
- Conc
 - CCircuit, [29](#)
- conc
 - CUnit, [34](#)
- conc_num
 - CUnit, [34](#)
- Conc_old
 - CCircuit, [29](#)
- connect_from_source
 - CUnit, [35](#)
- connect_to_conc
 - CUnit, [35](#)
- connect_to_tails

- CUnit, 35
- crossover
 - GeneticAlgorithm, 41
- crossover_rate
 - GeneticAlgorithm, 46
- CUnit, 32
 - c_rate_gormanium, 34
 - c_rate_waste, 34
 - calculate_outflow, 33
 - clearAll, 33
 - conc, 34
 - conc_num, 34
 - connect_from_source, 35
 - connect_to_conc, 35
 - connect_to_tails, 35
 - feed, 35
 - feed_old, 35
 - mark, 35
 - t_rate_gormanium, 36
 - t_rate_waste, 36
 - tails, 36
 - tails_num, 36
- DIR
 - run_tests, 13
- distance
 - Flow, 37
- Feed
 - CCircuit, 29
- feed
 - CUnit, 35
- feed_mass_gormanium
 - CCircuit, 30
- feed_mass_waste
 - CCircuit, 30
- feed_old
 - CUnit, 35
- files
 - run_tests, 13
- final
 - graph, 12
- find_conc
 - CCircuit, 30
- find_max_index
 - GeneticAlgorithm, 42
- find_tails
 - CCircuit, 30
- Flow, 36
 - clear, 37
 - distance, 37
 - Gormanium, 38
 - plus, 37
 - set, 38
 - waste, 38
- generate_random
 - CCircuit, 21
- generate_vector
 - CCircuit, 21
- GeneticAlgorithm, 39
 - ~GeneticAlgorithm, 41
 - crossover, 41
 - crossover_rate, 46
 - find_max_index, 42
 - GeneticAlgorithm, 40, 41
 - initialize_list, 42
 - iter_count, 46
 - main_process, 43
 - mutate, 43
 - mutation_rate, 46
 - next_generation, 46
 - parent_count, 47
 - parents, 47
 - print_next_generation, 44
 - print_parents, 44
 - select_parents, 44
 - setup_hyper_parameters, 45
 - threshold, 47
- get_circuit_vector
 - CCircuit, 21
- glob_fail
 - run_tests, 13
- Gormanium
 - Flow, 38
- graph, 11
 - final, 12
 - k, 12
 - loadDataset, 11
 - output1, 12
 - plot, 11
 - verbose, 12
- Guassian_elimination
 - CCircuit, 22
- includes/CCircuit.h, 49
- includes/CUnit.h, 50
- includes/Genetic_Algorithm.h, 51
- initialize_list
 - GeneticAlgorithm, 42
- initialize_units
 - CCircuit, 23
- initialize_vector
 - CCircuit, 23
- iter_count
 - GeneticAlgorithm, 46
- k
 - graph, 12
- loadDataset
 - graph, 11
- main
 - main.cpp, 56
 - test1.cpp, 57
 - test2.cpp, 58
- main.cpp

- main, 56
- main_process
 - GeneticAlgorithm, 43
- mark
 - CUnit, 35
- mark_units
 - CCircuit, 24
 - CCircuit.h, 50
- matmul
 - CCircuit, 24
- monetary_direct_solver
 - CCircuit, 25
- monetary_value
 - CCircuit, 30
- mutate
 - GeneticAlgorithm, 43
- mutation_rate
 - GeneticAlgorithm, 46
- next_generation
 - GeneticAlgorithm, 46
- node, 47
 - c_val, 48
 - CCircuit.cpp, 52
 - p_num, 48
 - parent, 48
 - t_val, 48
 - unit, 48
- num_units
 - CCircuit, 30
- output
 - run_tests, 13
- output1
 - graph, 12
- p_num
 - node, 48
- parent
 - node, 48
- parent_count
 - GeneticAlgorithm, 47
- parents
 - GeneticAlgorithm, 47
- plot
 - graph, 11
- plus
 - Flow, 37
- print_next_generation
 - GeneticAlgorithm, 44
- print_parents
 - GeneticAlgorithm, 44
- README.md, 51
- run_tests, 13
 - DIR, 13
 - files, 13
 - glob_fail, 13
 - output, 13
- test_failed, 13
- save_flow
 - CCircuit, 26
- select_parents
 - GeneticAlgorithm, 44
- set
 - Flow, 38
- set_initial_flow
 - CCircuit, 26
- set_units
 - CCircuit, 26
- setup_CUnits
 - CCircuit, 27
- setup_hyper_parameters
 - GeneticAlgorithm, 45
- setup_initial_parameters
 - CCircuit, 27
- source
 - CCircuit, 31
- src/CCircuit.cpp, 51
- src/CUnit.cpp, 54
- src/Genetic_Algorithm.cpp, 54
- src/main.cpp, 55
- t_rate_gormanium
 - CCircuit, 31
 - CUnit, 36
- t_rate_waste
 - CCircuit, 31
 - CUnit, 36
- t_val
 - node, 48
- Tails
 - CCircuit, 31
- tails
 - CUnit, 36
- tails_num
 - CUnit, 36
- Tails_old
 - CCircuit, 31
- test1.cpp
 - main, 57
- test2.cpp
 - main, 58
- test_failed
 - run_tests, 13
- tests/run_tests.py, 56
- tests/test1.cpp, 57
- tests/test2.cpp, 58
- threshold
 - GeneticAlgorithm, 47
- unit
 - node, 48
- units
 - CCircuit, 31
- value_gormanium

- CCircuit, [32](#)
- value_waste
 - CCircuit, [32](#)
- verbose
 - graph, [12](#)
- visualization/graph.py, [59](#)
- visualization/README_graph.md, [59](#)
- waste
 - Flow, [38](#)
- worst_case
 - CCircuit, [32](#)