CSMOn

Generated by Doxygen 1.8.11

Contents

Chapter 1

Convergence Stabilization Modeling operating in Online Mode

CSMOn (formely called of C'MOn!) is an automated method to estimate the best moment to stop swarm iterations based on the analysis of the convergence behavior presented during optimization, aiming to provide an effective balance between saving fitness evaluations and keeping the optimization quality. The convergence analysis is performed through a sequence of linear regressions using exponential and log-like curves.

Date

04/Mar/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Data Structure Documentation

5.1 _Param Struct Reference

Contains a parameter received from / sent to the Python caller.

Data Fields

- char $*$ **name**
- char **c**
- int **i**
- long **l**
- float **f**
- double **d**

5.1.1 Detailed Description

Contains a parameter received from / sent to the Python caller.

Definition at line 49 of file CSMOn_wrapper.cpp.

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

- python/CSMOn_wrapper.cpp

5.2 _point Struct Reference

A point representing the number of evaluations and the respective fitness value.

`#include <CSMOn.hpp>`

Public Member Functions

- **_point** (int _x, double _y)

Data Fields

- int **x**
- double **y**

5.2.1 Detailed Description

A point representing the number of evaluations and the respective fitness value.

Date

04/Mar/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

Definition at line 61 of file CSMOn.hpp.

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

- cpp/CSMOn.hpp

5.3 CSMOn Class Reference

Convergence Stabilization Modeling operating in Online Mode.

`#include <CSMOn.hpp>`

Public Member Functions

- CSMOn (ISearch ∗search, int M, double R, double minEstimatedFit)

  Class for CSMOn.
- void run ()

  Call this method to execute the search.
- void **getBest** (int nBest)
- int **adjustExp** (double r)
- int **adjustLog** (double r, int pT)
- int getNEvals ()

  Get the actual number of evaluations executed.
- double getFitness ()

  Get the final fitness value.
- int getBestPos (double ∗x)

  Get the final optimized result (position).

5.3.1 Detailed Description

Convergence Stabilization Modeling operating in Online Mode.

Date

04/Mar/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

Definition at line 75 of file CSMOn.hpp.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CSMOn::CSMOn (ISearch ∗ *search*, int *M*, double *R*, double *minEstimatedFit*)

Class for CSMOn.

Always call this class directly (instead of the search method).

Parameters

search	The instance for the search method.
M	The maximum number of fitness function evaluations allocated for this run.
R	The relaxation to be used for the limit calculations, in the interval]0,1[. For decreasing relaxation, provide it in negative values.
minEstimatedFit	The lowest fitness value expected.

Definition at line 43 of file CSMOn.cpp.

5.3.3 Member Function Documentation

5.3.3.1 int CSMOn::getBestPos (double ∗ *x*)

Get the final optimized result (position).

Returns

The final position.

Definition at line 198 of file CSMOn.cpp.

5.3.3.2 double CSMOn::getFitness ( )

Get the final fitness value.

Returns

The final fitness value.

Definition at line 178 of file CSMOn.cpp.

5.3.3.3 int CSMOn::getNEvals ( )

Get the actual number of evaluations executed.

Returns

The actual number of evaluations executed.

Definition at line 188 of file CSMOn.cpp.

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

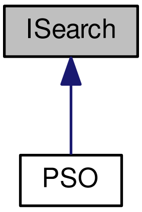
- cpp/CSMOn.hpp
- cpp/CSMOn.cpp

5.4 ISearch Class Reference

The interface that the search methods must to implement.

`#include <ISearch.hpp>`

Inheritance diagram for ISearch:



Public Member Functions

- virtual void startup ()=0

  Startup the search method.
- virtual void next (int M)=0

  Obtain the next improvement.
- virtual int getBestPos (double ∗_x)=0

  Get the best result obtained up to the moment.
- virtual int getNEvals ()=0

  Get the number of fitness function evaluations performed up to the moment.
- virtual double getFitness ()=0

  Get the best fitness value found up to the moment.

5.4.1 Detailed Description

The interface that the search methods must to implement.

Date

04/Mar/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

Definition at line 40 of file ISearch.hpp.

5.4.2 Member Function Documentation

5.4.2.1 virtual int ISearch::getBestPos (double ∗ *_x*) `[pure virtual]`

Get the best result obtained up to the moment.

Parameters

↩ _↩ *x*	A pointer to store the positions of the best result.

Returns

An ID for the best result (implementation specific).

Implemented in PSO.

5.4.2.2 virtual double ISearch::getFitness ( ) `[pure virtual]`

Get the best fitness value found up to the moment.

Returns

The best fitness value found.

Implemented in PSO.

5.4.2.3 virtual int ISearch::getNEvals ( ) `[pure virtual]`

Get the number of fitness function evaluations performed up to the moment.

Returns

The number of evaluations performed.

Implemented in PSO.

5.4.2.4 virtual void ISearch::next (int *M*) `[pure virtual]`

Obtain the next improvement.

Parameters

M	The maximum number of evaluations allowed.

Implemented in PSO.

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

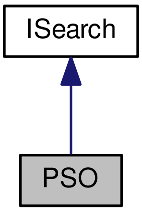
- cpp/ISearch.hpp
- cpp/CSMOn.cpp

5.5 PSO Class Reference

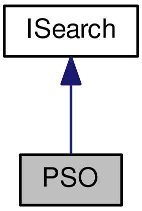
Particle Swarm Optimization.

`#include <PSO.hpp>`

Inheritance diagram for PSO:



Collaboration diagram for PSO:



Public Member Functions

- PSO (callback_t fitnessFunction, double s1, double s2, int p, int n, double w, double c1, double c2)

  A standard implementation of PSO.
- void startup ()

  Startup the PSO.
- void next (int M)

  Obtain the next improvement.
- int getBestPos (double ∗_x)

  Get the best result obtained up to the moment (global best).
- int getNEvals ()

  Get the number of fitness function evaluations performed up to the moment.
- double getFitness ()

  Get the best fitness value found up to the moment.
- unsigned int getRandomSeed ()

  Get a random number to be used as seed for the random number generator.

5.5.1 Detailed Description

Particle Swarm Optimization.

Date

04/Mar/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

Definition at line 53 of file PSO.hpp.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 PSO::PSO (callback_t *fitnessFunction,* double *s1,* double *s2,* int *p,* int *n,* double *w,* double *c1,* double *c2*)

A standard implementation of PSO.

Parameters

fitnessFunction	The callback reference to the fitness function.
s1	The lower bound of the search space.
s2	The upper bound of the search space.
p	The number of particles.
n	The number of dimensions of the problem.
w	The acceleration coefficient. For linear decreasing weight, provide it in negative values
c1	The cognitive knowledge rate.
c2	The social knowledge rate.

Definition at line 45 of file PSO.cpp.

5.5.3 Member Function Documentation

5.5.3.1 int PSO::getBestPos (double ∗ *_x*) `[virtual]`

Get the best result obtained up to the moment (global best).

Parameters

↩ _↩ *x*	A pointer to store the positions of the global best.

Returns

The index of the particle that found the global best position.

Implements ISearch.

Definition at line 160 of file PSO.cpp.

5.5.3.2 double PSO::getFitness ( ) `[virtual]`

Get the best fitness value found up to the moment.

Returns

The best fitness value found.

Implements ISearch.

Definition at line 180 of file PSO.cpp.

5.5.3.3 int PSO::getNEvals ( ) [virtual]

Get the number of fitness function evaluations performed up to the moment.

Returns

The number of evaluations performed.

Implements ISearch.

Definition at line 170 of file PSO.cpp.

5.5.3.4 unsigned int PSO::getRandomSeed ( )

Get a random number to be used as seed for the random number generator.

This implementation can be adapted/changed as necessary.

Returns

A random seed.

Definition at line 191 of file PSO.cpp.

5.5.3.5 void PSO::next (int *M*) [virtual]

Obtain the next improvement.

Parameters

M	The maximum number of evaluations allowed.

Implements ISearch.

Definition at line 120 of file PSO.cpp.

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

- cpp/PSO.hpp
- cpp/PSO.cpp

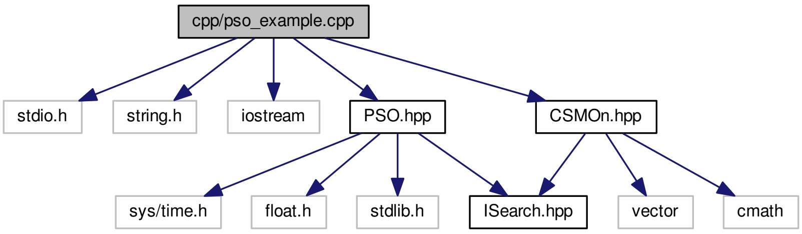
Chapter 6

File Documentation

6.1 cpp/pso_example.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <iostream>
#include "PSO.hpp"
#include "CSMOn.hpp"
```

Include dependency graph for pso_example.cpp:



Functions

- double fitnessFunction (double ∗x, int n)

  Fitness function implementation.
- int **main** (int argc, char ∗argv[ ])

6.1.1 Detailed Description

This file provides an implementation example to call CSMOn.

Date

02/Jul/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

6.1.2 Function Documentation

6.1.2.1 double fitnessFunction (double ∗ *x*, int *n*)

Fitness function implementation.

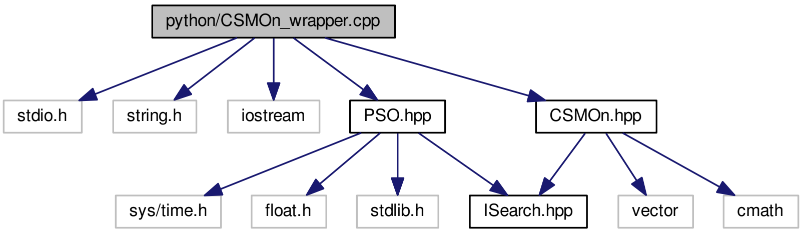
Put your fitness function here.

Definition at line 112 of file pso_example.cpp.

6.2 python/CSMOn_wrapper.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <iostream>
#include "PSO.hpp"
#include "CSMOn.hpp"
```

Include dependency graph for CSMOn_wrapper.cpp:



Data Structures

- struct _Param

  Contains a parameter received from / sent to the Python caller.

Typedefs

- typedef struct _Param Param

  Contains a parameter received from / sent to the Python caller.

Functions

- void search (char ∗method, Param ∗inParam, Param ∗outParam, double ∗outPos, callback_t fitnessFunction)

  The wrapper function for Python calls.

6.2.1 Detailed Description

This file contains the wrapper code for Python calls.

Date

02/Jul/2017

Author

Peter Frank Perroni (`pfperroni@gmail.com`)

6.2.2 Function Documentation

6.2.2.1 void search (char ∗ *method*, Param ∗ *inParam*, Param ∗ *outParam*, double ∗ *outPos*, callback_t *fitnessFunction*)

The wrapper function for Python calls.

Call this method from Python using ctypes interface.

Parameters

method	A string speficying the search method to be used.
inParam	The parameters required to call the search method specified.
outParam	The parameters returned by the search method called. This will be returned back to the Python caller.
outPos	The final result containing the optimized position. This will be returned back to the Python caller.
fitnessFunction	The callback Python function containing the code for the fitness function evaluation.

Definition at line 69 of file CSMOn_wrapper.cpp.

Index