Panobbgo Documentation

Release 0.0.1pre

Harald Schilly

CONTENTS

1	Intro	duction	1			
2	Main					
	2.1	Core				
	2.2		6			
	2.3	Heuristics				
	2.4	Configuration				
	2.5	User Interface	7			
3	Library					
	3.1	Classic Problems	9			
	3.2	Library Classes	11			
4	Examples 1					
5	Links					
Bi	bliogra	aphy	19			
Рy	Python Module Index					
In	index					

INTRODUCTION

Warning: It is currently work in progress and definitely not ready for any kind of usage.

Panobbgo is an open-source framework for parallel noisy black-box global optimization. The primary aim is to experiment with new ideas and algorithms. A couple of functional building blocks build the solver and exchange information via an EventBus among each other. This allows to rapidly prototype new modules and to combine them with existing parts. There are three basic types of parts that work together:

- the Strategy
- several Heuristics
- and Analyzers

Various tools for extracting statistical data and inspecting the optimization process are included (*planned*). Additionally, parallel evaluation of the objective black-box function can be archived as SMP or on a cluster via IPython [IP].

In the background, there are additional utility features for the configuration and dependency management (planned) available.

This software package is licensed under the Apache 2.0 License.

TWO

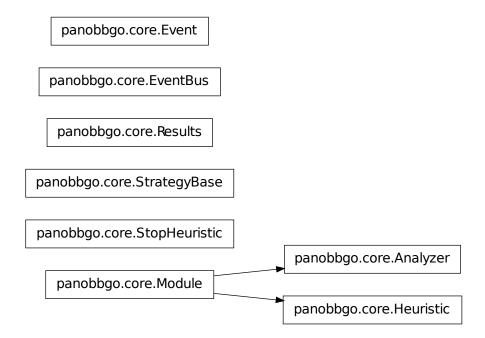
MAIN

This is the main part of Panobbgo.

2.1 Core

This is the core part. It contains the essential components and base-classes for the modules:

- Results: Database of all results, with some rudimentary queries and statistics.
- EventBus: This is the backbone for communicating between the strategy, the heuristics and the analyzers.
- "abstract" base-classes for the modules
 - heuristics
 - analyzers.
- ullet and most importantly, the <code>StrategyBase</code> which holds everything together and subclasses in <code>strategies</code> implement the actual strategies.



```
class panobbgo.core.Analyzer(name=None)
```

Bases: panobbgo.core.Module

Abstract parent class for all types of analyzers.

class panobbgo.core.Event (**kwargs)

Bases: object

This class holds the data for one single EventBus event.

class panobbgo.core.EventBus

Bases: object

This event bus is used to publish and send events. E.g. it is used to send information like "new best point" to all subscribing heuristics.

keys

List of all keys where you can send an Event to.

publish (key, event=None, terminate=False, **kwargs)

Publishes a new Event to all subscribers, who listen to the given key. It is either possible to send an existing event or to create an event object on the fly with the given **kwargs.

Args:

•event: if set, this given Event is sent (and not a new one created).

•terminate: if True, the associated thread will end. (use it for on_start and similar).

•**kwargs: any additional keyword arguments are stored inside the Event if event is None.

re = <module 're' from '/usr/lib/python2.7/re.pyc'>

```
register(target)
```

Registers a given target for this EventBus instance. It needs to have suitable on_<key> methods. For each of them, a Thread is spawn as a daemon.

subscribe (key, target)

Called by register().

Note: counterpart is unsubscribe().

unsubscribe (key, target)

Args:

•if key is None, the target is removed from all keys.

class panobbgo.core.Heuristic(name=None, cap=None)

Bases: panobbgo.core.Module

This is the "abstract" parent class for all types of point generating classes, which we call collectively "Heuristics".

Such a heuristic is capable of the following:

- 1. They can be parameterized by passing in optional arguments in the constructor. This should be reflected in the name!
- 2.The EventBus spawns a thread for each on_* method and calls them when a corresponding Event occurs.
- 3.Of course, they are capable of storing their state in the instance. This is also the way of how information is shared between those threads.
- 4. The *main purpose* of a heuristic is to emit new search points by calling either emit () or returning a list of points. The datatype must be numpy.ndarray of floats.
- 5. Additionally, the can get hold of other heuristics or analyzers via the strategy instance.

4 Chapter 2. Main

6.The EventBus inside this strategy instance allows them to publish their own events, too. This can be used to signal related heuristics something or to queue up tasks for itself.

active

This is queried by the strategy to determine, if it should still consider it. This is the case, iff there is still something in its output queue or if there is a chance that there will be something in the future (at least one thread is running).

```
clear_output()
```

emit (points)

This is used to send out new search points for evaluation. Args:

•points: Either a numpy.ndarray of float 64 or preferrably a list of them.

get_points(limit=None)

this drains the output Queue until limit elements are removed or the Queue is empty. For each actually emitted point, the performance value is discounted (i.e. "punishment" or "energy consumption")

class panobbgo.core.Module(name=None)

Bases: object

"Abstract" parent class for various panobbgo modules, e.g. Heuristic and Analyzer.

eventbus

name

The module's name.

Note: It should be unique, which is important for parameterized heuristics or analyzers!

problem

results

strategy

ui

class panobbgo.core.Results (strategy)

Bases: object

A very simple database of results with a notification for new results. The new results are fed directly by the StrategyBase, outside of the EventBus.

Note: Later on, maybe this will be a cool actual database which allows to persistenly store past evaluations for a given problem. This would allow resuming and further a-posteriory analysis.

```
add_results (new_results)
```

Add one single or a list of new @Result objects. Then, publish a new_result event.

info()

 $n_best(n)$

exception panobbgo.core.StopHeuristic(msg='stopped')

Bases: exceptions.Exception

Indicates the heuristic has finished and should be ignored/removed.

Args:

•msg: a custom message, will be visible in the log. (default: "stopped")

class panobbgo.core.StrategyBase(problem, heurs)

Bases: object

This abstract BaseStrategy is the parent class of all Strategies.

2.1. Core 5

Use it this way:

- 1. Subclass it, write your optional initializer, *afterwards* call the initializer of this class (it will start its the main loop).
- 2.Overwrite the execute(), which returns a list of new search points (by requesting them from the heuristics via the get_points() method) and might also emit Events.

This execute method will be called repeatedly as long as there are less than the given maximum number of search points evaluated.

```
PROBLEM_KEY = 'problem'
add_analyzer(a)
add_heuristic(h)
analyzer (who)
avg_time_per_task
best
execute()
    Overwrite this method when you extend this base strategy.
heuristic (who)
heuristics
info()
init_module (module)
    StrategyBase calls this method.
name
run()
time_cpu
    effective cpu time in seconds
time_start_str
time wall
    wall time in seconds
```

2.2 Strategies

This part outlines the coordination between the point-producing heuristics, the interaction with the cluster and the DB of evaluated points.

Basically, one or more threads produce points where to search, and another one consumes them and dispatches tasks. Subclass the StrategyBase class to implement a new strategy.

2.3 Heuristics

The main idea behind all heuristics is, ...

Each heuristic needs to listen to at least one stream of Events from the EventBus. Most likely, it is the *one-shot* event start, which is published by the StrategyBase.

6 Chapter 2. Main

2.4 Configuration

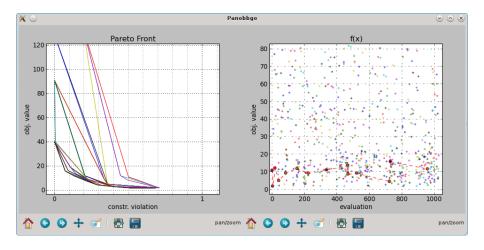
It's purpose is to parse a config file (create a default one if none is present) and replace values stored within it with those given via optional command-line arguments.

Note: This will also hold a class for configuring the Panobbgo framework in general. I.e. modules declare other modules as dependencies, etc...

```
class panobbgo.config.Config
    Bases: object
    debug
    get_logger(name, loglevel=None)
panobbgo.config.get_config()
```

2.5 User Interface

This draws a window and plots graphs.



```
class panobbgo.ui.UI
    Bases: panobbgo.core.Module, gtk.Window, threading.Thread
    UI
    add_notebook_page (label_text, frame)
    destroy (win)
    finish()
        called by base strategy in _cleanup for shutdown
    static mk_canvas()
        Creates a FigureCanvas, ready to be added to a gtk layout element
    redraw_canvas(c)
        If your canvas needs to be redrawn, pass it into this function.
    run()
    show()
```

2.4. Configuration 7

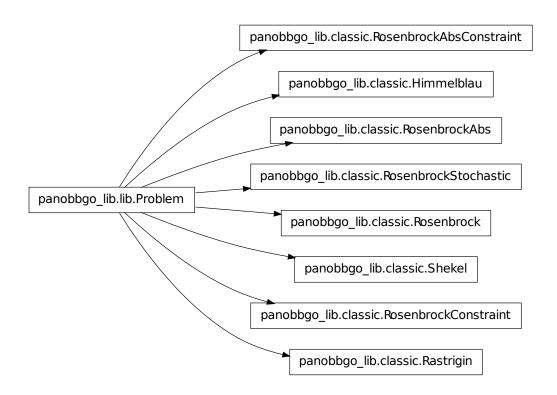
8 Chapter 2. Main

LIBRARY

This is the *library* of Panobbgo, used by the main part and the library. For example, the basic Problem class is defined here.

3.1 Classic Problems

This file contains the basic objects to build a problem and to do a single evaluation.



 ${\bf class} \; {\tt panobbgo_lib.classic.Himmelblau}$

Bases: panobbgo_lib.lib.Problem

Himmelblau [HB] testproblem.

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

eval(x)

class panobbgo_lib.classic.Rastrigin (dims, par1=10, offset=0)

Bases: panobbgo_lib.lib.Problem

Rastrigin

$$f(x) = par_1 \cdot n + \sum_{i} (x_i^2 - 10\cos(2\pi x_i))$$

eval(x)

class panobbgo_lib.classic.Rosenbrock (dims, par1=100)

Bases: panobbgo_lib.lib.Problem

Rosenbrock function with parameter par1.

$$f(x) = \sum_{i} (par_1(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$

eval(x)

class panobbgo_lib.classic.RosenbrockAbs (dims, par1=100)

Bases: panobbgo_lib.lib.Problem

Absolute Rosenbrock function.

$$f(x) = \sum_{i} par_1 ||x_{i+1} - ||x_i|| || + ||1 - x_i||$$

eval(x)

class panobbgo_lib.classic.RosenbrockAbsConstraint (dims, par1=100, par2=0.1)

 $Bases: \verb|panobbgo_lib.lib.Problem| \\$

Absolute Rosenbrock function.

$$\min f(x) = \sum_{i} par_{1} ||x_{i+1} - ||x_{i}|| || + ||1 - x_{i}||$$

$$s.t. ||x_{i+1} - x_{i}|| \ge par_{2} \forall i \in \{0, \dots, dim - 1\}$$

$$x_{i} \ge 0 \forall i$$

eval(x)

 $eval_constraints(x)$

class panobbgo_lib.classic.RosenbrockConstraint (dims, parl=100, par2=0.25)

Bases: panobbgo_lib.lib.Problem

Constraint Rosenbrock function with parameter par1 and par2.

$$\min f(x) = \sum_{i} par_1(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

$$s.t. \ (x_{i+1} - x_i)^2 \ge par_2 \ \forall i \in \{0, \dots, dim - 1\}$$

$$x_i \ge 0 \ \forall i$$

eval(x)

eval_constraints(x)

10 Chapter 3. Library

class panobbgo_lib.classic.RosenbrockStochastic (dims, parl=100, jitter=0.1)

Bases: panobbgo_lib.lib.Problem

Stochastic variant of Rosenbrock function.

$$f(x) = \sum_{i} (par_1 eps_i(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$

where eps_i is a uniformly random (n-1)-dimensional vector in $[0,1)^{n-1}$.

eval(x)

class panobbgo_lib.classic.Shekel (dims, m=10, a=None, c=None)

Bases: panobbgo_lib.lib.Problem

Shekel Function [SH].

For m minima in n dimensions:

f(

 $ec{x} = sum_{i = 1}^{m} frac{1}{c_{i} + sumlimits_{j = 1}^{n} (x_{j} - a_{ji})^{2}}$

eval(x)

3.2 Library Classes

This file contains the basic objects to build a problem and to do a single evaluation.

panobbgo lib.lib.Point

panobbgo_lib.lib.Result

panobbgo_lib.lib.Problem

Note: This is used by panobbgo and panobbgo_lib.

class panobbgo_lib.lib.Point (x, who)

Bases: object

This contains the x vector for a new point and a reference to who has generated it.

who

A string, which is the name of a heuristic.

To get the actual heuristic, use the strategie's heuristic method.

x

The vector x, a numpy . ndarray

```
class panobbgo_lib.lib.Problem(box)
     Bases: object
     this is used to store the objective function, information about the problem, etc.
     box must be a list of tuples, which specify the range of each variable.
     example: [(-1,1), (-100,0), (0,0.01)].
     box
           The bounding box for this problem, a (dim, 2)-array.
           Note: This might change to a more sophisticated Box object.
      dim
           The number of dimensions.
      eval(x)
           This is called to evaluate the given black-box function. The problem should be called directly
             _call__ special function wraps this) and the given problem should subclass this eval method.
      eval_constraints(x)
           This method is optionally overwritten by the problem to calculate the constraint violations. It has to
           return a numpy.ndarray of floats.
     project (point)
           projects given point into the search box. e.g. [-1.1, 1] with box [(-1, 1), (-1, 1)] gives [-1, 1]
      random point()
           generates a random point inside the given search box (ranges).
      ranges
           The ranges along each dimension, a numpy.ndarray.
class panobbgo_lib.lib.Result (point, fx, cv_vec=None, cv_norm=None, error=0.0)
     Bases: object
     This represents one result, wich is a mapping of a Point x \to f(x).
      Additionally, there is also
          •error: estimated or calculated \Delta f(x).
          •cv_vec: a possibly empty vector listing the constraint violation for each constraint.
          •cnt: An integer counter, starting at 0.
      Args:
          •cv: the constraint violation vector
          •cv_norm: the norm used to calculate cv. (see numpy.linalg.norm(), default None means
           2-norm)
      cnt
           Integer ID for this result.
      cv
           The chosen norm of cv_vec; see cv_norm in constructor.
           Note: Only the positive entries are used to calculate the norm!
      cv_vec
           Vector of constraint violations for each constraint, or None.
```

12 Chapter 3. Library

Note: Be aware, that entries could be negative. This is useful if you want to know how well a point is satisfied. The .cv property just looks at the positive entries, though.

error

Error margin of function evaluation, usually 0.0.

fx

The function value f(x) after evaluating it.

point

Returns the actual Point object.

рp

pareto point, i.e. array([cv, fx])

who

The name of the heuristic, who did generate this point (String).

x

Point x where this result has been evaluated.

14 Chapter 3. Library

CHAPTER

FOUR

EXAMPLES

- ex1
- ex2

CHAPTER

FIVE

LINKS

- source repository
- short introduction talk
- Indices and Tables
 - genindex
 - modindex
 - search

18 Chapter 5. Links

BIBLIOGRAPHY

- $[HB] \ \ http://en.wikipedia.org/wiki/Himmelblau\%27s_function$
- [SH] http://en.wikipedia.org/wiki/Shekel_function
- [IP] http://www.ipython.org

20 Bibliography

PYTHON MODULE INDEX

р

```
panobbgo,3
panobbgo.config,6
panobbgo.core,3
panobbgo.heuristics,6
panobbgo.strategies,6
panobbgo.ui,7
panobbgo_lib,9
panobbgo_lib.classic,9
panobbgo_lib.lib,11
```

INDEX

A	eval_constraints() (panob-
active (panobbgo.core.Heuristic attribute), 5	bgo_lib.classic.RosenbrockAbsConstraint
add_analyzer() (panobbgo.core.StrategyBase method),	method), 10
6	eval_constraints() (panob-
add_heuristic() (panobbgo.core.StrategyBase method), 6	bgo_lib.classic.RosenbrockConstraint method), 10
add_notebook_page() (panobbgo.ui.UI method), 7	eval_constraints() (panobbgo_lib.lib.Problem method),
add_results() (panobbgo.core.Results method), 5	Event (class in penalther core) 4
Analyzer (class in panobbgo.core), 3	Event (class in panobbgo.core), 4 EventBus (class in panobbgo.core), 4
analyzer() (panobbgo.core.StrategyBase method), 6	eventbus (panobbgo.core.Module attribute), 5
avg_time_per_task (panobbgo.core.StrategyBase attribute), 6	execute() (panobbgo.core.StrategyBase method), 6
В	F
	finish() (panobbgo.ui.UI method), 7
best (panobbgo.core.StrategyBase attribute), 6 box (panobbgo_lib.lib.Problem attribute), 12	fx (panobbgo_lib.lib.Result attribute), 13
С	G
clear_output() (panobbgo.core.Heuristic method), 5	get_config() (in module panobbgo.config), 7
cnt (panobbgo_lib.lib.Result attribute), 12	get_logger() (panobbgo.config.Config method), 7
Config (class in panobbgo.config), 7	get_points() (panobbgo.core.Heuristic method), 5
cv (panobbgo_lib.lib.Result attribute), 12	Н
cv_vec (panobbgo_lib.lib.Result attribute), 12	
D	Heuristic (class in panobbgo.core), 4 heuristic() (panobbgo.core.StrategyBase method), 6
	heuristics (panobbgo.core.StrategyBase attribute), 6
debug (panobbgo.config.Config attribute), 7 destroy() (panobbgo.ui.UI method), 7	Himmelblau (class in panobbgo_lib.classic), 9
dim (panobbgo_lib.lib.Problem attribute), 12	
	I
E	info() (panobbgo.core.Results method), 5
emit() (panobbgo.core.Heuristic method), 5	info() (panobbgo.core.StrategyBase method), 6
error (panobbgo_lib.lib.Result attribute), 13	init_module() (panobbgo.core.StrategyBase method), 6
eval() (panobbgo_lib.classic.Himmelblau method), 9 eval() (panobbgo_lib.classic.Rastrigin method), 10	K
eval() (panobbgo_lib.classic.Rosenbrock method), 10	keys (panobbgo.core.EventBus attribute), 4
eval() (panobbgo_lib.classic.RosenbrockAbs method),	
10	M
$eval () \ (panobbgo_lib.classic.RosenbrockAbsConstraint$	mk_canvas() (panobbgo.ui.UI static method), 7
method), 10	Module (class in panobbgo.core), 5
eval() (panobbgo_lib.classic.RosenbrockConstraint method), 10	N
eval() (panobbgo_lib.classic.RosenbrockStochastic	
method), 11	n_best() (panobbgo.core.Results method), 5 name (panobbgo.core.Module attribute), 5
eval() (panobbgo_lib.classic.Shekel method), 11	name (panobbgo.core.StrategyBase attribute), 6
eval() (panobbgo_lib.lib.Problem method), 12	Q

```
P
                                                        ui (panobbgo.core.Module attribute), 5
                                                        unsubscribe() (panobbgo.core.EventBus method), 4
panobbgo (module), 3
panobbgo.config (module), 6
                                                        W
panobbgo.core (module), 3
                                                        who (panobbgo lib.lib.Point attribute), 11
panobbgo.heuristics (module), 6
                                                        who (panobbgo lib.lib.Result attribute), 13
panobbgo.strategies (module), 6
panobbgo.ui (module), 7
                                                        X
panobbgo lib (module), 9
panobbgo lib.classic (module), 9
                                                        x (panobbgo_lib.lib.Point attribute), 11
panobbgo_lib.lib (module), 11
                                                        x (panobbgo_lib.lib.Result attribute), 13
Point (class in panobbgo_lib.lib), 11
point (panobbgo lib.lib.Result attribute), 13
pp (panobbgo lib.lib.Result attribute), 13
Problem (class in panobbgo lib.lib), 11
problem (panobbgo.core.Module attribute), 5
PROBLEM KEY
                         (panobbgo.core.StrategyBase
         attribute), 6
project() (panobbgo_lib.lib.Problem method), 12
publish() (panobbgo.core.EventBus method), 4
R
random_point() (panobbgo_lib.lib.Problem method), 12
ranges (panobbgo_lib.lib.Problem attribute), 12
Rastrigin (class in panobbgo lib.classic), 10
re (panobbgo.core.EventBus attribute), 4
redraw_canvas() (panobbgo.ui.UI method), 7
register() (panobbgo.core.EventBus method), 4
Result (class in panobbgo lib.lib), 12
Results (class in panobbgo.core), 5
results (panobbgo.core.Module attribute), 5
Rosenbrock (class in panobbgo_lib.classic), 10
RosenbrockAbs (class in panobbgo lib.classic), 10
RosenbrockAbsConstraint
                              (class
                                               panob-
         bgo_lib.classic), 10
RosenbrockConstraint (class in panobbgo_lib.classic),
RosenbrockStochastic (class in panobbgo_lib.classic),
run() (panobbgo.core.StrategyBase method), 6
run() (panobbgo.ui.UI method), 7
S
Shekel (class in panobbgo_lib.classic), 11
show() (panobbgo.ui.UI method), 7
StopHeuristic, 5
strategy (panobbgo.core.Module attribute), 5
StrategyBase (class in panobbgo.core), 5
subscribe() (panobbgo.core.EventBus method), 4
Т
time_cpu (panobbgo.core.StrategyBase attribute), 6
time start str (panobbgo.core.StrategyBase attribute),
time_wall (panobbgo.core.StrategyBase attribute), 6
U
```

24 Index

UI (class in panobbgo.ui), 7