perprof-py Documentation

Release 1.0.0

Raniere

August 22, 2015

Contents

perprof-py is a free/open source Python module for performance profiling (as described by Dolan and Moré) with TikZ and matplotlib output.

Contents:

CHAPTER 1

Install

Important: This package requires Python3 and isn't compatible with Python2.

Below you will find instructions to install perprof-py for any operation system that run Python and after that more detail instructions for some popular operation system.

1.1 General

Install perprof-py in a Python environment with pip is easy:

```
# pip install -r REQUIREMENTS
# python setup.py install
```

Note: You can install it locally if you want but to take advantage of bash completion we recomend you to install it globablly.

To see a demo:

```
$ perprof --mp -o demo -f --demo
```

1.2 GNU/Linux Distribution

The general instructions probably work for you if you already using Python3 as default. Instructions for distributions that still using Python2 are found below.

1.2.1 Debian

```
# apt-get install python3 pip3
```

And follow the general instructions replacing `python` with `python3` and `pip` with `pip3`.

1.2.2 Ubuntu

```
# apt-get install python3 pip3
```

And follow the general instructions replacing `python` with `python3` and `pip` with `pip3`.

1.2.3 Fedora

```
# yum install python3 pip3
```

And follow the general instructions replacing `python` with `python3` and `pip` with `pip3`.

1.2.4 Arch

```
# pacman -S pip
```

And follow the general instructions.

1.2.5 Gentoo

```
# emerge pip
```

And follow the general instructions.

1.3 Mac OS X

Note: Comming soon.

1.4 Windows

Note: Comming soon.

CHAPTER 2

Input File Format

The current format start with a optional YAML header for metadata follow by the data as show in the template below:

```
---
<Metadata 01>: <Value 01>
<Metadata 02>: <Value 02>
---
<Problem Name 01> <Exit Flag 01> <Cost 01>
<Problem Name 02> <Exit Flag 02> <Cost 02>
<Problem Name 03> <Exit Flag 03> <Cost 03>
...
```

where

<Metadata XX> is the name of metadata field

<Vallue XX> is the value of the metadata field

<Problem Name XX> is the name of the problem

<Exit Flag XX> is `c` or `d`, meaning converged and diverged, respectively

<Cost XX> is the "cost" (e.g. time spent) to be used for the performance profile until solve the problem or give up.

Some examples of input file are provided at *perprof/examples*. To see the examples already

```
$ cd perprof/examples
$ ./make-examples.sh
```

This will generate 8 simple examples in the folder *perprof/examples/plots*.

2.1 YAML header

The YAML header store some useful metadata information and optionally some configurations.

2.1.1 Metadata

algname The name of the algorithmic/solver to be used in the plot.

2.1.2 Configuration

subset The name of the file to be used for the subset.

success List of strings to mark success.

mintime The minimum time that a algorithmic/solver need to run.

maxtime The maximum time that a algorithmic/solver can run.

free_format Only check for mark of success.

CHAPTER 3

Old Input File Format

Important: This is keep to backward compatibility.

The old format follow the template below:

```
#Name <Solver Name>
<Problem Name 01> <Exit Flag 01> <Cost 01>
<Problem Name 02> <Exit Flag 02> <Cost 02>
<Problem Name 03> <Exit Flag 03> <Cost 03>
...
```

where

<Solver Name> is the name of the solver to be used in the plot

<Problem Name XX> is the name of the problem

<Exit Flag XX> is `c` or `d`, meaning converged and diverged, respectively

<Cost XX> is the "cost" (e.g. time spent) to be used for the performance profile until solve the problem or give up.

CHAPTER 4

Command line flags

perprof-py have a lot of options. To see a list with all of then:

```
$ perprof --help
```

4.1 Store flags in a file

To fully customize your needs, you may need to add a few flags for *perprof*. The best way to do this is to create a file with a flag in each line and calling *perprof* with that file as argument, with a @ preceding the file name:

```
$ perprof @filename [more flags] FILE1 FILE2 [FILE3 ...]
```

For a example you can look at *test/pdf.args*. To use it, enter

```
$ perprof @test/pdf.args -f -o tmp test/*.long
```

Please note that the arguments in the file and in the command line are treated equally, so you can't add conflicting options.

CHAPTER 5

API for Developers

Here you will find the API provide for developers.

5.1 Entry point

This is the main file for perprof

class perprof.main.**PerProfSetup**(*args*)
This is a class to store the files to be used.

perprof.main.**main**()
This is the entry point when calling perprof.

perprof.main.**set_arguments**(*args*)
Set all the arguments of perprof

5.2 File parse

Functions to parse the files

The files must be in the following format:

```
---
<Metadata 01>: <Value 01>
<Metadata 02>: <Value 02>
---
<Problem Name 01> <Exit Flag 01> <Cost 01>
<Problem Name 02> <Exit Flag 02> <Cost 02>
<Problem Name 03> <Exit Flag 03> <Cost 03>
...
```

perprof.parse.**parse_file**(*filename*, *subset=None*, *success='c'*, *mintime=0*, *maxtime=inf*, *free_format=False*)
Parse one file.

Parameters

- **filename** (*str*) – name of the file to be parser
- **subset** (*list*) – list with the name of the problems to use
- **success** (*list*) – list with strings to mark sucess

- **mintime** (*int*) – minimum time running the solver
- **maxtime** (*int*) – maximum time running the solver
- **free_format** (*bool*) – if False request that fail be mark with `d`

Returns performance profile data and name of the solver

5.3 Profile interface

The functions related with the perform (not the output).

class `perprof.prof.`**Pdata**(*setup*)
Store data for performance profile.

get_set_problems()
Get the set of problems to use.

Returns list of problems

get_set_solvers()
Get the set of solvers to use.

Returns list of solvers

plot()
This should be implemented by a child of this class.

scale()
Scale time.

set_percent_problems_solved_by_time()
Set the percent of problems solved by time.

`perprof.prof.`**load_data**(*setup*)
Load the data.

Parameters **setup** (main.PerProfSetup) – the setup configurations

5.4 Profiler using TikZ

This handle the plot using tikz.

class `perprof.tikz.`**Profiler**(*setup*)
The profiler using TikZ.

plot()
Create the performance profile using matplotlib.

5.5 Profiler using matplotlib

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

p

G

L

M

P

S