ParamILS/MO-ParamILS – Quick Start Guide

September 2016

1 Introduction

ParamILS [3, 2, 1] is a tool for Automatic Algorithm Configuration (AAC), whose workflow is recalled in Figure 1. Given a target algorithm, together with (i) a description of its parameters, (ii) some training instances, (iii) a dedicated wrapper outputing performance indicators, ParamILS will automatically perform evaluations of possible configurations on possible instances, to ultimately propose an optimised configuration of the target algorithm.

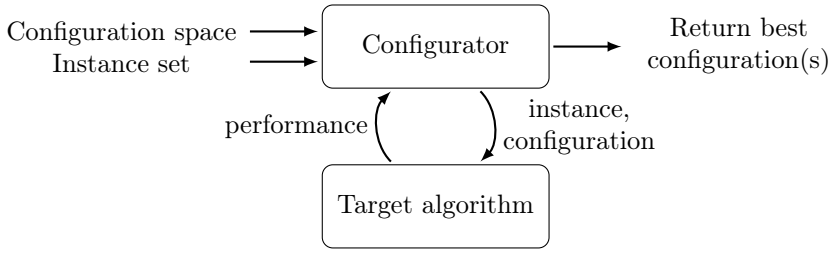

Figure 1: General workflow of Automatic Algorithm Configuration (AAC)

2 Getting ParamILS

ParamILS can be downloaded from its project website[1]. This quick start guide refers to the current latest version of ParamILS (3.0.0), which introduces the first version of MO-ParamILS.

Once the archive downloaded, extract it. You should find inside: `lib` folder, containing the necessary code to use ParamILS, four bash scripts (`paramils`, `moparamils`, `validate`, `movalidate`) used for starting ParamILS, and some example files for you to copy and modify.

[1] `http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/`

3 Scenario Files

The following presents a very simple use case.

3.1 Scenario File

The scenario file describes the automatic configuration scenario. It precises essential informations such as the target algorithm command line or the paths of the other configuration files.

my_scenario.txt

```
# algorithm path
execdir = .
# any output is put in the "output" folder
outdir = output

# algorithm command line
algo = ruby my_algo.rb
# parameters are detailled here
paramfile = my_params.txt

# instance files
instance_file = my_training_list.txt
test_instance_file = my_test_list.txt
# if the algorithm is not deterministic, seeding is used
# 1=true ; 0=false
deterministic = 1

# quality | time
run_obj = quality
# use arithmetic mean
overall_obj = mean
# target algorithms maximum runtime
cutoff_time = 1
# configurator stops after 10 seconds
tunerTimeout = 10
```

3.2 Target Algorithm Parameter File

The parameter file describes the possible configurations of the target algorithm. For each parameter, the possible values are indicated between curly braces and the default value between brackets. Conditional parameters and forbidden combinations are also precised here. Note that the default value must necessarily be one of the possible values of the parameter.

my_params.txt

```
# any list of arbitrary values
a {0,1,2,3,4,5}[0]
b {1.0, 1.1, 1.2, 1.3, 1.4, 1.5}[1.0]
c {foo, bar, baz}[bar]
d {none,10,25,50,all}[10]

# automatic expension
e {0,2..10}[6] # expends to {0,2,4,6,8,10}
f {1.0,1.1..1.5}[1.0] # same as above

# no restriction on parameter and values
my_parameter {my_first_value, 0, -1, 99.99}[0]

# conditional parameter
c | b in {1.0, 1.1} # c only exists if b is 1.0 or 1.1

# forbidden combinations
{a=0,b=1.0,c=foo} # will avoid matching configurations
```

3.3 Instance Files

Both training instance set and test instance set are precised in the scenario file by the parameters `instance_file` and `test_instance_file`, respectively, and follow the same format.

my_training_list.txt

```
an_instance
another_instance
path/to/third/instance
```

If the algorithm is deterministic, seeds are automatically associated to each instance as needed. To otherwise manually fix the seeds for each instance, use `instance_seed_file` and `test_instance_seed_file` instead in the scenario file and specify before each path the corresponding seed.

my_seeded_training_list.txt

```
0 an_instance
1 an_instance
41 another_instance
43 another_instance
```

Additionally, in both case, you can add instance-specific information after the path, which will be passed to the target algorithm wrapper.

3.4 Target Algorithm Wrapper

The partial target algorithm command line is precised in the scenario file by the parameter `algo`, here "`ruby my_algo.rb`". The full command line is computed by ParamILS as "cd <path> ; `<algo> <instance name> <instance-specific information> <cutoff time> <cutoff length> <seed> <params>`". Parameters are given as a list of parameters and values, parameters being prefixed by a dash and values between single quotes.

For example, the following is a full command such as computed by ParamILS: "`cd /your/absolute/path/.; ruby my_algo.rb my_dummy_instance.txt 0 1.0 2147483647 -1 -a '3' -b '1.0' -c 'foo' -d '25' -e '10' -f '1.4' -my_parameter '-1'`". The absolute path is here automatically computed by ParamILS using the current path and the relative path given in the scenario file.

This command line must result by a valid output, parsed by ParamILS. The expected format is "`Result: <status>, <runtime>, <quality>, <seed>`". The depreciated format using "`Result for ParamILS: <status>, <runtime>, <runlength>, <quality>, <seed>`" is still accepted among few others. The term "`<status>`" must be either "`SUCCESS`", "`TIMEOUT`", "`ABORT`" or "`CRASHED`". The term "`<quality>`" can either be a scalar value or a vector of comma-separated values between brackets. Note that even if multiple qualities are returned by the wrapper, you can still use ParamILS in single-objective mode, which will then only consider the first quality.

The wrapper can be written in the language of your choice. However, if your target algorithm has a really small runtime, take care to use a language that won't induce too much overhead (*e.g.*, perl is really fast and recommended, while both python and ruby interpreters are quite slow to start).

4 ParamILS Configuration

A list of basic ParamILS parameters can be obtained using the `--help` parameter. Additionnaly, the `--help-level` parameter can be set to either `BASIC`, `INTERMEDIATE` or `ADVANCED` to show the description of more or fewer parameters. The main ParamILS parameters are:

- `--MO` $\in$ {`TRUE`, `FALSE`}: to either use ParamILS (using the first quality if multiple are returned) or MO-ParamILS
- `--approach` $\in$ {`BASIC`, `FOCUSED`}: to either use BasicILS or FocusedILS
- `--R` $\in \mathbb{N}$: the number of initial random configurations
- `--max-run` $\in \mathbb{N}$: the maximum number of runs allowed for a single configuration
- `--min-run` $\in \mathbb{N}$: the minimum number of runs required for a single configuration

- `--random-restart` $\in [0, 1]$: the probability to restart from a random configuration

5 Using ParamILS

Once you have you scenario file, your parameter file, your instances files and a working wrapper for your target algorithm, you can launch ParamILS by using either the `paramils` or `moparamils` executable, by precising the scenario file using `--scenario-file` and the other ParamILS parameters. ParamILS store all outputs, results and data in the folder specified in the scenario file (see "`outdir`"), in a sub-folder named after the scenario file. The name of this subfolder can be changed using the `--rungroup` parameter.

Additional data stored by ParamILS include the final training and validation configurations, the trajectory of the run (*i.e.*, the best configurations at any moment of the run), and logs of the output.

Example: "`./moparamils --scenario-file my_scenario.txt --rungroup training --seed 1 --tunerTimeout 3600`" will run the `my_scenario.txt` scenario with MO-ParamILS for 1 hour, the seed being fixed to 1, and the output folder name changed to `training` instead of `my_scenario`.

References

[1] Aymeric Blot, Holger H. Hoos, Laetitia Jourdan, Marie-Éléonore Marmion, and Heike Trautmann. MO-ParamILS: A multi-objective automatic algorithm configuration framework. In *Proceedings of LION 10*, 2016.

[2] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *JAIR*, 36:267–306, 2009.

[3] Frank Hutter, Holger H. Hoos, and Thomas Stützle. Automatic algorithm configuration based on local search. In *Proceedings of AAAI 2007*, pages 1152–1157, 2007.