Sparkle User Guide

Release 0.2

ADA Research Group, LIACS

Jun 30, 2022

Sparkle User Guide

1 Quick start **2**

2 Installing Sparkle **3**
- 2.1 Get a copy of Sparkle . . . 3
- 2.2 Install dependencies . . . 3

3 Algorithm Configuration **4**
- 3.1 Creating a wrapper for your algorithm . . . 4
- 3.2 Parameter configuration space (PCS) file . . . 4

4 Algorithm Selection **5**
- 4.1 Creating a wrapper for your algorithm . . . 5

5 Executing commands **5**

6 File structure **5**
- 6.1 A typical instance directory . . . 5
- 6.2 A typical solver directory (configuration) . . . 6
- 6.3 A typical solver directory (selection) . . . 6

7 Wrappers **7**
- 7.1 `sparkle_run_default_wrapper.py` . . . 7

8 Commands **7**
- 8.1 about.py . . . 8
- 8.2 add_feature_extractor.py . . . 8
- 8.3 add_instances.py . . . 9
- 8.4 add_solver.py . . . 9
- 8.5 cleanup_current_sparkle_platform.py . . . 10
- 8.6 cleanup_temporary_files.py . . . 10
- 8.7 compute_features.py . . . 10
- 8.8 compute_features_parallel.py . . . 11
- 8.9 compute_marginal_contribution.py . . . 11
- 8.10 configure_solver.py . . . 11
- 8.11 construct_sparkle_portfolio_selector.py . . . 12
- 8.12 generate_report.py . . . 13
- 8.13 initialise.py . . . 13

8.14 load_record.py . . . 14
8.15 remove_feature_extractor.py . . . 14
8.16 remove_instances.py . . . 14
8.17 remove_record.py . . . 14
8.18 remove_solver.py . . . 15
8.19 run_ablation.py . . . 15
8.20 run_configured_solver.py . . . 16
8.21 run_solvers.py . . . 16
8.22 run_sparkle_portfolio_selector.py . . . 17
8.23 run_status.py . . . 17
8.24 save_record.py . . . 17
8.25 sparkle_wait.py . . . 18
8.26 system_status.py . . . 18
8.27 validate_configured_vs_default.py . . . 18

9 Sparkle settings **19**
9.1 Example `sparkle_settings.ini` . . . 19
9.2 Names and possible values . . . 20
9.3 Priorities . . . 21
9.4 Slurm (focused on Grace) . . . 22

10 Required packages **23**
10.1 Sparkle on Grace . . . 23

11 Installation and compilation of examples **23**
11.1 Solvers . . . 23

Index **25**

1 Quick start

Note: Sparkle currently relies on Slurm[3], and does not work without it.

Follow these steps:

1. *Install Sparkle*
2. Prepare an *algorithm configuration* or an *algorithm selection*.
3. *Execute commands*

[3] https://slurm.schedmd.com/

2 Installing Sparkle

Note: The installation process use the `conda` command available in Anaconda[4] or Miniconda[5] to manage some dependencies.

2.1 Get a copy of Sparkle

To get a copy of Sparkle you can clone the repository.

If `git` is available on your system, this will clone the Sparkle repository and create a subdirectory named `sparkle` :

```
$ git clone https://bitbucket.org/sparkle-ai/sparkle.git
```

You can also download the stable version here: https://bitbucket.org/sparkle-ai/sparkle/get/main.zip

2.2 Install dependencies

Sparkle depends on Python 3.9+, swig 3.0, gnuplot, LaTeX, and multiple Python packages. An easy way to install most of what is needed is to use the `conda` package manager (https://docs.conda.io/en/latest/miniconda.html).

Note: LaTeX is used to create the reports and the documentation and must be installed manually on the system. If you don't plan to use the reports or recreate the documentations, you can skip it.

You can install the base requirements with

```
$ conda env create -f environment.yml
```

This will create an environment named `sparkle` that contains everything needed to run Sparkle, except LaTeX that needs to be installed manually.

To activate the environment in the current shell, execute:

```
$ conda activate sparkle
```

Note: You will need to reactivate the environment every time you log in, before using Sparkle.

The file `environment.yml` contains a tested list of Python packages with fixed versions required to execute Sparkle. We recommended using it.

The file `environment-dev.txt` contains unpinned packages and the dependencies are not resolved. It is used for development and may cause problems.

The two environments can be created in parallel since one is named `sparkle` and the other `sparkle-dev`. If you want to update an environment it is better to do a clean installation by removing and recreating it. For example:

[4] https://www.anaconda.com/
[5] https://docs.conda.io/en/latest/miniconda.html

```
$ conda deactivate
$ conda env remove -n sparkle
$ conda env create -f environment.yml
$ conda activate sparkle
```

This should be fast as both `conda` and `pip` use local cache for the packages.

3 Algorithm Configuration

Configuring an algorithm has the following minimal requirements for the algorithm (for an example of a solver directory see Section 6.2):

- A working solver executable
- An algorithm wrapper called `sprakle_smac_wrapper.py`
- A PCS (parameter configuration space) file
- The runsolver binary (e.g. from `Examples/Resources/Solvers/PbO-CCSAT-Generic/`)

Further, training and testing instance sets are needed (for an example of an instances directory see Section 6.1). For the purpose of testing whether your configuration setup works with Sparkle, it is advised to primarily use instances that are solved (relatively) quickly even with the default parameters.

Note: See the example page for a walk-through on how to perform configuration with Sparkle.

3.1 Creating a wrapper for your algorithm

A template for the wrapper that connects your algorithm with Sparkle is available at `Examples/Resources/Solvers/template/sparkle_smac_wrapper.py`. Within this template a number of `TODO`s are indicated where you are likely to need to make changes for your specific algorithm. You can also compare the different example solvers to get an idea for what kind of changes are needed.

3.2 Parameter configuration space (PCS) file

The PCS (parameter configuration space) format[1] is used to pass the possible parameter ranges of an algorithm to Sparkle in a `.pcs` file. For an example see e.g. `Examples/Resources/Solvers/PbO-CCSAT-Generic/PbO-CCSAT-params_test.pcs`.

In this file you should enter all configurable parameters of your algorithm. Note that parameters such as the random seed used by the algorithm should not be configured and therefore should also not be included in the PCS file.

[1] See: http://aclib.net/cssc2014/pcs-format.pdf

4 Algorithm Selection

Creating a portfolio selector requires multiple algorithms with the following minimal requirements (for an example of a solver directory see Section 6.3):

- A working solver executable
- An algorithm wrapper called `sprakle_run_default_wrapper.py`

Further, training and testing instance sets are needed (for an example of an instances directory see Section 6.1). For the purpose of testing whether your selection setup works with Sparkle, it is advised to primarily use instances that are solved (relatively) quickly.

Note: See the example page for a walk-through on how to perform selection with Sparkle.

4.1 Creating a wrapper for your algorithm

A template for the wrapper that connects your algorithm with Sparkle is available at `Examples/Resources/Solvers/template/sparkle_run_default_wrapper.py`. Within this template a number of `TODO`s are indicated where you are likely to need to make changes for your specific algorithm. You can also compare the different example solvers to get an idea for what kind of changes are needed.

5 Executing commands

Executing commands in Sparkle is as simple as running them in the top directory of Sparkle, for example:

```
Commands/initialise.py
```

Do note that when running on a cluster additional arguments may be needed, for instance under the Slurm workload manager the above command would change to something like:

```
srun -N1 -n1 -c1 Commands/initialise.py
```

In the `Examples/` directory a number of common command sequences are given. For instance, for configuration with specified training and testing sets see e.g. `Examples/configuration.md` for an example of a sequence of commands to execute. Note that some command run in the background and need time to complete before the next command is executed. To see whether a command is still running the Slurm command `squeue` can be used.

In the `Output/` directory paths to generated scripts and logs are gathered per executed command.

6 File structure

6.1 A typical instance directory

An instance directory should look something like this:

```
Instances/
  Example_Instance_Set/
    instance_a.cnf
```

(continues on next page)

(continued from previous page)

```
    instance_b.cnf
    ...        ...
    instance_z.cnf
```

This directory simply contains a collection of instances, as example here SAT instances in the CNF format are given.

For instances consisting of multiple files one additional file called `sparkle_instance_list.txt` should be included in the `Example_Instance_Set` directory, describing which files together form an instance. The format is a single instance per line with each file separated by a space, as shown below.

```
instance_a_part_one.abc instance_a_part_two.xyz
instance_b_part_one.abc instance_b_part_two.xyz
...                     ...
instance_z_part_one.abc instance_z_part_two.xyz
```

6.2 A typical solver directory (configuration)

A solver directory should look something like this:

```
Solver/
  Example_Solver/
    solver
    sparkle_smac_wrapper.py
    parameters.pcs
    runsolver
```

Here `solver` is a binary executable of the solver that is to be configured. The `sprakle_smac_wrapper.py` is a wrapper that Sparkle should call to run the solver with specific settings, and then returns a result for the configurator. In `parameters.pcs` the configurable parameters are described in the PCS format. Finally, `runsolver` is a binary executable of the runsolver tool. This allows Sparkle to make fair time measurements for all configuration experiments.

Note: Currently the runsolver binary has to be in every solver directory, it can be found in the `Examples/Resources/Solvers/PbO-CCSAT-Generic/` directory.

6.3 A typical solver directory (selection)

A solver directory should look something like this:

```
Solver/
  Example_Solver/
    solver
    sparkle_run_default_wrapper.py
```

Here `solver` is a binary executable of a solver that is to be included in a portfolio selector. The `sprakle_run_default_wrapper.py` is a wrapper that Sparkle should call to run the solver on a specific instance.

7 Wrappers

7.1 sparkle_run_default_wrapper.py

The `sparkle_run_default_wrapper.py` has two functions that need to be implemented for each algorithm:

- `print_command(instance_file, seed_str: str, cutoff_time_str: str)`
- `print_output(terminal_output_file: str)`

`print_command(...)` should print a command line call that Sparkle can use to run the algorithm on a given instance file. Ideally, for reproducibility purposes, the seed provided by Sparkle should also be passed to the algorithm. If the algorithm requires this, the cutoff time can also be passed to the algorithm. However, in this case the cutoff time should be made very large. For instance by multiplying by ten with: `cutoff_time_str = str(int(cutoff_time_str) * 10)`. This is necessary to ensure Sparkle stops the algorithm after the cutoff time, rather than the algorithm itself. By doing this it is ensured runtime measurements are always done by Sparkle, and thus consistent between algorithms that might measure time differently.

`print_output(...)` should process the algorithm output. If the performance measure is `RUNTIME`, this function only needs to output the algorithm status. For all `QUALITY` performance measures both the algorithm status and the solution quality have to be given. Sparkle internally measures `RUNTIME`, while it can be overwritten by the user if desired, for consistent runtime measurements between solvers this is not recommended. The output should be printed and formatted as in the example below.

```
quality 8734
status SUCCESS
```

Status can hold the following values `{SUCCESS, TIMEOUT, CRASHED}`. If the status is not known, reporting `SUCCESS` will allow Sparkle to continue, but may mean that Sparkle does not know when the algorithm crashed, and continues with faulty results.

8 Commands

Currently the commands below are available in Sparkle (listed alphabetically). Every command can be called with the `-help` option to get a description of the required arguments and other options.

- *about.py*
- *add_feature_extractor.py*
- *add_instances.py*
- *add_solver.py*
- *cleanup_current_sparkle_platform.py*
- *cleanup_temporary_files.py*
- *compute_features.py*
- *compute_features_parallel.py*
- *compute_marginal_contribution.py*
- *configure_solver.py*
- *construct_sparkle_portfolio_selector.py*
- *generate_report.py*

- *initialise.py*
- *load_record.py*
- *remove_feature_extractor.py*
- *remove_instances.py*
- *remove_record.py*
- *remove_solver.py*
- *run_ablation.py*
- *run_configured_solver.py*
- *run_solvers.py*
- *run_sparkle_portfolio_selector.py*
- *run_status.py*
- *save_record.py*
- *sparkle_wait.py*
- *system_status.py*
- *validate_configured_vs_default.py*

Note: Arguments in [square brackets] are optional, arguments without brackets are mandatory. Input in <chevrons> indicate required text input, {curly brackets} indicate a set of inputs to choose from.

8.1 about.py

```
usage: about.py [-h]
```

-h, --help
show this help message and exit

8.2 add_feature_extractor.py

```
usage: add_feature_extractor.py [-h] [--run-extractor-now | --run-extractor-later] [--
↪nickname NICKNAME]
                                [--parallel]
                                extractor-path
```

extractor-path
path to the feature extractor

-h, --help
show this help message and exit

--run-extractor-now
immediately run the newly added feature extractor on the existing instances

--run-extractor-later
: do not immediately run the newly added feature extractor on the existing instances (default)

--nickname <nickname>
: set a nickname for the feature extractor

--parallel
: run the feature extractor on multiple instances in parallel

8.3 add_instances.py

```
usage: add_instances.py [-h] [--run-extractor-now | --run-extractor-later] [--run-solver-
↪now | --run-solver-later]
                        [--nickname NICKNAME] [--parallel]
                        instances-path
```

instances-path
: path to the instance set

-h, --help
: show this help message and exit

--run-extractor-now
: immediately run the feature extractor(s) on the newly added instances

--run-extractor-later
: do not immediately run the feature extractor(s) on the newly added instances (default)

--run-solver-now
: immediately run the solver(s) on the newly added instances

--run-solver-later
: do not immediately run the solver(s) on the newly added instances (default)

--nickname <nickname>
: set a nickname for the instance set

--parallel
: run the solvers and feature extractor on multiple instances in parallel

8.4 add_solver.py

Add a solver to the Sparkle platform.

```
usage: add_solver.py [-h] --deterministic {0,1} [--run-solver-now | --run-solver-later]␣
↪[--nickname NICKNAME]
                     [--parallel] [--solver-variations SOLVER_VARIATIONS]
                     solver-path
```

solver-path
: path to the solver

-h, --help
: show this help message and exit

--deterministic {0,1}

indicate whether the solver is deterministic or not

--run-solver-now

immediately run the newly added solver on all instances

--run-solver-later

do not immediately run the newly added solver on all instances (default)

--nickname <nickname>

set a nickname for the solver

--parallel

run the solver on multiple instances in parallel

--solver-variations <solver_variations>

Use this option to add multiple variations of the solver by using a different random seed for each varation.

8.5 cleanup_current_sparkle_platform.py

```
usage: cleanup_current_sparkle_platform.py [-h]
```

-h, --help

show this help message and exit

8.6 cleanup_temporary_files.py

```
usage: cleanup_temporary_files.py [-h]
```

-h, --help

show this help message and exit

8.7 compute_features.py

```
usage: compute_features.py [-h] [--recompute] [--parallel] [--settings-file SETTINGS_
↪FILE]
```

-h, --help

show this help message and exit

--recompute

re-run feature extractor for instances with previously computed features

--parallel

run the feature extractor on multiple instances in parallel

--settings-file

specify the settings file to use in case you want to use one other than the default

8.8 compute_features_parallel.py

```
usage: compute_features_parallel.py [-h] [--recompute]
```

-h, --help
show this help message and exit

--recompute
re-run feature extractor for instances with previously computed features

8.9 compute_marginal_contribution.py

```
usage: compute_marginal_contribution.py [-h] (--perfect | --actual) [--recompute]
                                        [--performance-measure {RUNTIME,QUALITY_ABSOLUTE,
↪QUALITY}]
                                        [--settings-file SETTINGS_FILE]
```

-h, --help
show this help message and exit

--perfect
compute the marginal contribution for the perfect selector

--actual
compute the marginal contribution for the actual selector

--recompute
force marginal contribution to be recomputed even when it already exists in file for for the current selector

--performance-measure
the performance measure, e.g. runtime

--settings-file
specify the settings file to use in case you want to use one other than the default

8.10 configure_solver.py

Configure a solver in the Sparkle platform.

```
usage: configure_solver.py [-h] [--validate] [--ablation] --solver SOLVER --instance-set-
↪train INSTANCE_SET_TRAIN
                           [--instance-set-test INSTANCE_SET_TEST]
                           [--performance-measure {RUNTIME,QUALITY_ABSOLUTE,QUALITY}]
                           [--target-cutoff-time TARGET_CUTOFF_TIME] [--budget-per-run␣
↪BUDGET_PER_RUN]
                           [--number-of-runs NUMBER_OF_RUNS] [--settings-file SETTINGS_
↪FILE] [--use-features]
```

-h, --help
show this help message and exit

--validate
validate after configuration

--ablation

run ablation after configuration

--solver <solver>

path to solver

--instance-set-train <instance_set_train>

path to training instance set

--instance-set-test <instance_set_test>

path to testing instance set (only for validating)

--performance-measure

the performance measure, e.g. runtime

--target-cutoff-time

cutoff time per target algorithm run in seconds

--budget-per-run

configuration budget per configurator run in seconds

--number-of-runs

number of configuration runs to execute

--settings-file

specify the settings file to use instead of the default

--use-features

use the training set's features for configuration

Note that the test instance set is only used if the `--ablation` or ``--validation`` flags are given

8.11 construct_sparkle_portfolio_selector.py

```
usage: construct_sparkle_portfolio_selector.py [-h] [--recompute-portfolio-selector]
                                               [--recompute-marginal-contribution]
                                               [--performance-measure {RUNTIME,QUALITY_
↪ABSOLUTE,QUALITY}]
```

-h, --help

show this help message and exit

--recompute-portfolio-selector

force the construction of a new portfolio selector even when it already exists for the current feature and performance data. NOTE: This will also result in the computation of the marginal contributions of solvers to the new portfolio selector.

--recompute-marginal-contribution

force marginal contribution to be recomputed even when it already exists in file for the current selector

--performance-measure

the performance measure, e.g. runtime

8.12 generate_report.py

Without any arguments a report for the most recent algorithm selection or algorithm configuration procedure is generated.

```
usage: generate_report.py [-h] [--solver SOLVER] [--instance-set-train INSTANCE_SET_
↪TRAIN]
                          [--instance-set-test INSTANCE_SET_TEST] [--no-ablation [FLAG_
↪ABLATION]] [--selection]
                          [--test-case-directory TEST_CASE_DIRECTORY]
                          [--performance-measure {RUNTIME,QUALITY_ABSOLUTE,QUALITY}]
                          [--settings-file SETTINGS_FILE]
```

-h, --help
show this help message and exit

--solver <solver>
path to solver for an algorithm configuration report

--instance-set-train <instance_set_train>
path to training instance set included in Sparkle for an algorithm configuration report

--instance-set-test <instance_set_test>
path to testing instance set included in Sparkle for an algorithm configuration report

--no-ablation <flag_ablation>
turn off reporting on ablation for an algorithm configuration report

--selection
set to generate a normal selection report

--test-case-directory <test_case_directory>
Path to test case directory of an instance set for a selection report

--performance-measure
the performance measure, e.g. runtime

--settings-file
specify the settings file to use in case you want to use one other than the default

Note that if a test instance set is given, the training instance set must also be given.

8.13 initialise.py

Initialise the Sparkle platform, this command does not have any arguments.

```
usage: initialise.py [-h]
```

-h, --help
show this help message and exit

8.14 load_record.py

```
usage: load_record.py [-h] record-file-path
```

record-file-path

path to the record file

-h, --help

show this help message and exit

8.15 remove_feature_extractor.py

```
usage: remove_feature_extractor.py [-h] [--nickname] extractor-path
```

extractor-path

path to or nickname of the feature extractor

-h, --help

show this help message and exit

--nickname

if set to True extractor_path is used as a nickname for the feature extractor

8.16 remove_instances.py

```
usage: remove_instances.py [-h] [--nickname] instances-path
```

instances-path

path to or nickname of the instance set

-h, --help

show this help message and exit

--nickname

if given instances_path is used as a nickname for the instance set

8.17 remove_record.py

```
usage: remove_record.py [-h] record-file-path
```

record-file-path

path to the record file

-h, --help

show this help message and exit

8.18 remove_solver.py

```
usage: remove_solver.py [-h] [--nickname] solver-path
```

solver-path

path to or nickname of the solver

-h, --help

show this help message and exit

--nickname

if set to True solver_path is used as a nickname for the solver

8.19 run_ablation.py

Runs parameter importance between the default and configured parameters with ablation. This command requires a finished configuration for the solver instance pair.

```
usage: run_ablation.py [-h] [--solver SOLVER] [--instance-set-train INSTANCE_SET_TRAIN]
                       [--instance-set-test INSTANCE_SET_TEST] [--ablation-settings-help]
                       [--performance-measure {RUNTIME,QUALITY_ABSOLUTE,QUALITY}]
                       [--target-cutoff-time TARGET_CUTOFF_TIME] [--budget-per-run␣
↪BUDGET_PER_RUN]
                       [--number-of-runs NUMBER_OF_RUNS] [--racing RACING] [--settings-
↪file SETTINGS_FILE]
```

-h, --help

show this help message and exit

--solver <solver>

path to solver

--instance-set-train <instance_set_train>

path to training instance set

--instance-set-test <instance_set_test>

path to testing instance set

--ablation-settings-help

prints a list of setting that can be used for the ablation analysis

--performance-measure

the performance measure, e.g. runtime

--target-cutoff-time

cutoff time per target algorithm run in seconds

--budget-per-run

configuration budget per configurator run in seconds

--number-of-runs

number of configuration runs to execute

--racing

performs abaltion analysis with racing

--settings-file

specify the settings file to use in case you want to use one other than the default

Note that if no test instance set is given, the validation is performed on the training set.

8.20 run_configured_solver.py

```
usage: run_configured_solver.py [-h] [--settings-file SETTINGS_FILE]
                                [--performance-measure {RUNTIME,QUALITY_ABSOLUTE,QUALITY}
↪] [--parallel]
                                instance_path [instance_path ...]
```

instance_path

Path(s) to instance file(s) (when multiple files are given, it is assumed this is a multi-file instance) or instance directory.

-h, --help

show this help message and exit

--settings-file <settings_file>

settings file to use instead of the default (default: Settings/sparkle_settings.ini)

--performance-measure {RUNTIME,QUALITY_ABSOLUTE,QUALITY}

the performance measure, e.g. runtime (default: RUNTIME)

--parallel

run the solver on multiple instances in parallel

8.21 run_solvers.py

```
usage: run_solvers.py [-h] [--recompute] [--parallel] [--performance-measure {RUNTIME,
↪QUALITY_ABSOLUTE,QUALITY}]
                      [--target-cutoff-time TARGET_CUTOFF_TIME] [--also-construct-
↪selector-and-report]
                      [--verifier {NONE,SAT}] [--settings-file SETTINGS_FILE]
```

-h, --help

show this help message and exit

--recompute

recompute the performance of all solvers on all instances

--parallel

run the solver on multiple instances in parallel

--performance-measure

the performance measure, e.g. runtime

--target-cutoff-time

cutoff time per target algorithm run in seconds

--also-construct-selector-and-report

after running the solvers also construct the selector and generate the report

--verifier

problem specific verifier that should be used to verify solutions found by a target algorithm

--settings-file

specify the settings file to use in case you want to use one other than the default

8.22 run_sparkle_portfolio_selector.py

```
usage: run_sparkle_portfolio_selector.py [-h] [--settings-file SETTINGS_FILE]
                                         [--performance-measure {RUNTIME,QUALITY_
↪ABSOLUTE,QUALITY}]
                                         instance_path [instance_path ...]
```

instance_path

Path to instance or instance directory

-h, --help

show this help message and exit

--settings-file

settings file to use instead of the default

--performance-measure

the performance measure, e.g. runtime

8.23 run_status.py

```
usage: run_status.py [-h] [--verbose]
```

-h, --help

show this help message and exit

--verbose, -v

output run status in verbose mode

8.24 save_record.py

```
usage: save_record.py [-h]
```

-h, --help

show this help message and exit

8.25 sparkle_wait.py

```
usage: sparkle_wait.py [-h]
                       [--job-id JOB_ID | --command {ABOUT,ADD_FEATURE_EXTRACTOR,ADD_
↪INSTANCES,ADD_SOLVER,CLEANUP_CURRENT_SPARKLE_PLATFORM,CLEANUP_TEMPORARY_FILES,COMPUTE_
↪FEATURES,COMPUTE_MARGINAL_CONTRIBUTION,CONFIGURE_SOLVER,CONSTRUCT_SPARKLE_PORTFOLIO_
↪SELECTOR,GENERATE_REPORT,INITIALISE,LOAD_RECORD,REMOVE_FEATURE_EXTRACTOR,REMOVE_
↪INSTANCES,REMOVE_RECORD,REMOVE_SOLVER,RUN_ABLATION,RUN_SOLVERS,RUN_SPARKLE_PORTFOLIO_
↪SELECTOR,RUN_STATUS,SAVE_RECORD,SPARKLE_WAIT,SYSTEM_STATUS,VALIDATE_CONFIGURED_VS_
↪DEFAULT,RUN_CONFIGURED_SOLVER,CONSTRUCT_SPARKLE_PARALLEL_PORTFOLIO,RUN_SPARKLE_
↪PARALLEL_PORTFOLIO}]
```

-h, --help
: show this help message and exit

--job-id <job_id>
: job ID to wait for

--command {ABOUT,ADD_FEATURE_EXTRACTOR,ADD_INSTANCES,ADD_SOLVER,
CLEANUP_CURRENT_SPARKLE_PLATFORM,CLEANUP_TEMPORARY_FILES,COMPUTE_FEATURES,
COMPUTE_MARGINAL_CONTRIBUTION,CONFIGURE_SOLVER,CONSTRUCT_SPARKLE_PORTFOLIO_SELECTOR,
GENERATE_REPORT,INITIALISE,LOAD_RECORD,REMOVE_FEATURE_EXTRACTOR,REMOVE_INSTANCES,
REMOVE_RECORD,REMOVE_SOLVER,RUN_ABLATION,RUN_SOLVERS,RUN_SPARKLE_PORTFOLIO_SELECTOR,
RUN_STATUS,SAVE_RECORD,SPARKLE_WAIT,SYSTEM_STATUS,VALIDATE_CONFIGURED_VS_DEFAULT,
RUN_CONFIGURED_SOLVER,CONSTRUCT_SPARKLE_PARALLEL_PORTFOLIO,
RUN_SPARKLE_PARALLEL_PORTFOLIO}
: command you want to run. Sparkle will wait for the dependencies of this command to be completed

8.26 system_status.py

```
usage: system_status.py [-h] [--verbose]
```

-h, --help
: show this help message and exit

--verbose, -v
: output system status in verbose mode

8.27 validate_configured_vs_default.py

Test the performance of the configured solver and the default solver by doing validation experiments on the training and test sets.

```
usage: validate_configured_vs_default.py [-h] --solver SOLVER --instance-set-train␣
↪INSTANCE_SET_TRAIN
                                         [--instance-set-test INSTANCE_SET_TEST]
                                         [--performance-measure {RUNTIME,QUALITY_
↪ABSOLUTE,QUALITY}]
                                         [--target-cutoff-time TARGET_CUTOFF_TIME] [--
↪settings-file SETTINGS_FILE]
```

-h, --help
: show this help message and exit

--solver <solver>
: path to solver

--instance-set-train <instance_set_train>
: path to training instance set

--instance-set-test <instance_set_test>
: path to testing instance set

--performance-measure
: the performance measure, e.g. runtime

--target-cutoff-time
: cutoff time per target algorithm run in seconds

--settings-file
: specify the settings file to use instead of the default

9 Sparkle settings

Most settings can be controlled through `Settings/sparkle_settings.ini`. Possible settings are summarised per category in Section 9.2. For any settings that are not provided the defaults will be used. Meaning, in the extreme case, that if the settings file is empty (and nothing is set through the command line) everything will run with default values.

For convenience after every command `Settings/latest.ini` is written with the used settings. This can, for instance, be used to provide the same settings to the next command in a chain. E.g. for `validate_configured_vs_default` after `configure_solver`. The used settings are also recorded in the relevant `Output/` subdirectory. Note that when writing settings Sparkle always uses the name, and not an alias.

9.1 Example `sparkle_settings.ini`

This is a short example to show the format, see the settings file in `Settings/sparkle_settings.ini` for more.

```
[general]
performance_measure = RUNTIME
target_cutoff_time = 60

[configuration]
number_of_runs = 25

[slurm]
number_of_runs_in_parallel = 25
```

9.2 Names and possible values

[general]

`performance_measure`

aliases: `smac_run_obj`
values: `{RUNTIME, QUALITY_ABSOLUTE` (also: `QUALITY)}`
description: The type of performance measure that sparkle uses. `RUNTIME` focuses on runtime the solver requires, `QUALITY_ABSOLUTE` focuses on the average absolute improvements on the instances and `QUALITY` does the same a the former.

`target_cutoff_time`

aliases: `smac_each_run_cutoff_time`, `cutoff_time_each_performance_computation`
values: integer
description: The time a solver is allowed to run before it is terminated.

`extractor_cutoff_time`

aliases: `cutoff_time_each_feature_computation`
values: integer
description: The time a feature extractor is allowed to run before it is terminated. In case of multiple feature extractors this budget is divided equally.

`penalty_multiplier`

aliases: `penalty_number`
values: integer
description: In case of not solving an instance within the cutoff time the runtime is set to be the `penalty_multiplier * cutoff_time`.

`solution_verifier`

aliases: N/A
values: `{NONE, SAT}`
note: Only available for SAT solving.

[configuration]

`budget_per_run`

aliases: `smac_whole_time_budget`
values: integer
description: The wallclock time one configuration run is allowed to use for finding configurations.

`number_of_runs`

aliases: `num_of_smac_runs`
values: integer
description: The number of separate configurations runs.

[smac]

`target_cutoff_length`

aliases: `smac_each_run_cutoff_length`
values: `{max}` (other values: whatever is allowed by SMAC)

[ablation]

`racing`

aliases: `ablation_racing`
values: boolean
description: Use racing when performing the ablation analysis between the default and configured parameters

[slurm]

`number_of_runs_in_parallel`

aliases: `smac_run_obj`
values: integer
description: The number of configuration runs that can run in parallel.

`clis_per_node`

aliases: N/A
values: integer
note: Not really a Slurm option, will likely be moved to another section.
description: The number of parallel processes that can be run on one compute node. In case a node has 32 cores and each solver uses 2 cores, the `cli_per_node` is at most 16.

9.3 Priorities

Sparkle has a large flexibility with passing along settings. Settings provided through different channels have different priorities as follows:

- Default –- Default values will be overwritten if a value is given through any other mechanism;
- File –- Settings form the `Settings/sparkle_settings.ini` overwrite default values, but are overwritten by settings given through the command line;
- Command line file -– Settings files provided through the command line, overwrite default values and other settings files.
- Command line –- Settings given through the command line overwrite all other settings, including settings files provided through the command line.

9.4 Slurm (focused on Grace)

Slurm settings can be specified in the `Settings/sparkle_slurm_settings.txt` file. Currently these settings are inserted *as is* in any `srun` or `sbatch` calls done by Sparkle. This means that any options exclusive to one or the other currently should not be used (see Section 9.4).

Tested options

Below a list of tested Slurm options for `srun` and `sbatch` is included. Most other options for these commands should also be safe to use (given they are valid), but have not been explicitly tested. Note that any options related to commands other than `srun` and `sbatch` should not be used with Sparkle, and should not be included in `Settings/sparkle_slurm_settings.txt`.

- `--partition / -p`
- `--exclude`
- `--nodelist`

Disallowed options

The options below are exclusive to `sbatch` and are thus disallowed:

- `--array`
- `--clusters`
- `--wrap`

The options below are exclusive to `srun` and are thus disallowed:

- `--label`

Nested `srun` calls

A number of Sparkle commands internally call the `srun` command, and for those commands the provided settings need to match the restrictions of your call to a Sparkle command. Take for instance the following command:

```
srun -N1 -n1 -p graceTST Commands/configure_solver.py --solver Solvers/Pb0-CCSAT-Generic␣
↪--instances-train Instances/PTN/
```

This call restricts itself to the `graceTST` partition (the `graceTST` partition only consists of node 22). So if the settings file contains the setting `-exclude=ethnode22`, all available nodes are excluded, and the command cannot execute any internal `srun` commands it may have.

Finally, Slurm ignores nested partition settings for `srun`, but not for `sbatch`. This means that if you specify the `graceTST` partition (as above) in your command, but the `graceADA` partition in the settings file, Slurm will still execute any nested `srun` commands on the `graceTST` partition only.

10 Required packages

10.1 Sparkle on Grace

Grace is the computing cluster of the ADA group[2] at LIACS, Leiden University. Since not all packages required by Sparkle are installed on the system, some have to be installed local to the user.

Making your algorithm run on Grace

Shell and Python scripts should work as is. If a compiled binary does not work, you may have to compile it on Grace and manually install packages on Grace that are needed by your algorithm.

General requirements

Other software used by Sparkle:

- `pdflatex`
- `latex`
- `bibtex`
- `gnuplot`
- `gnuplot-x11`

To manually install `gnuplot` see for instance the instructions on their website http://www.gnuplot.info/development/

11 Installation and compilation of examples

11.1 Solvers

CSCCSat

CSCCSat can be recompiled as follows in the `Examples/Resources/Solvers/CSCCSat/` directory:

```
unzip src.zip
cd src/CSCCSat_source_codes/
make
cp CSCCSat ../../
```

[2] http://ada.liacs.nl/

MiniSAT

MiniSAT can be recompiled as follows in the `Examples/Resources/Solvers/MiniSAT/` directory:

```
unzip src.zip
cd minisat-master/
make
cp build/release/bin/minisat ../
```

PbO-CCSAT

PbO-CCSAT can be recompiled as follows in the `Examples/Resources/Solvers/PbO-CCSAT-Generic/` directory:

```
unzip src.zip
cd PbO-CCSAT-master/PbO-CCSAT_process_oriented_version_source_code/
make
cp PbO-CCSAT ../../
```

TCA and FastCA

The TCA and FastCA solvers, require `GLIBCXX_3.4.21`. This library comes with `GCC 5.1.0` (or greater). Following installation you may have to update environment variables such as `LD_LIBRARY_PATH, LD_RUN_PATH, CPATH` to point to your installation directory.

TCA can be recompiled as follows in the `Examples/Resources/CCAG/Solvers/TCA/` directory:

```
unzip src.zip
cd TCA-master/
make clean
make
cp TCA ../
```

FastCA can be recompiled as follows in the `Examples/Resources/CCAG/Solvers/FastCA/` directory:

```
unzip src.zip
cd fastca-master/fastCA/
make clean
make
cp FastCA ../../
```

VRP_SISRs

VRP_SISRs can be recompiled as follows in the `Examples/Resources/CVRP/Solvers/VRP_SISRs/` directory:

```
unzip src.zip
cd src/
make
cp VRP_SISRs ../
```

Index

Symbols

--ablation
    configure_solver.py command line option, 11
--ablation-settings-help
    run_ablation.py command line option, 15
--actual
    compute_marginal_contribution.py command line option, 11
--also-construct-selector-and-report
    run_solvers.py command line option, 16
--budget-per-run
    configure_solver.py command line option, 12
    run_ablation.py command line option, 15
--command
    sparkle_wait.py command line option, 18
--deterministic
    add_solver.py command line option, 9
--help
    about.py command line option, 8
    add_feature_extractor.py command line option, 8
    add_instances.py command line option, 9
    add_solver.py command line option, 9
    cleanup_current_sparkle_platform.py command line option, 10
    cleanup_temporary_files.py command line option, 10
    compute_features.py command line option, 10
    compute_features_parallel.py command line option, 11
    compute_marginal_contribution.py command line option, 11
    configure_solver.py command line option, 11
    construct_sparkle_portfolio_selector.py command line option, 12
    generate_report.py command line option, 13
    initialise.py command line option, 13
    load_record.py command line option, 14
    remove_feature_extractor.py command line option, 14
    remove_instances.py command line option, 14
    remove_record.py command line option, 14
    remove_solver.py command line option, 15
    run_ablation.py command line option, 15
    run_configured_solver.py command line option, 16
    run_solvers.py command line option, 16
    run_sparkle_portfolio_selector.py command line option, 17
    run_status.py command line option, 17
    save_record.py command line option, 17
    sparkle_wait.py command line option, 18
    system_status.py command line option, 18
    validate_configured_vs_default.py command line option, 18
--instance-set-test
    configure_solver.py command line option, 12
    generate_report.py command line option, 13
    run_ablation.py command line option, 15
    validate_configured_vs_default.py command line option, 19
--instance-set-train
    configure_solver.py command line option, 12
    generate_report.py command line option, 13
    run_ablation.py command line option, 15
    validate_configured_vs_default.py command line option, 19
--job-id
    sparkle_wait.py command line option, 18
--nickname
    add_feature_extractor.py command line option, 9
    add_instances.py command line option, 9
    add_solver.py command line option, 10
    remove_feature_extractor.py command line option, 14
    remove_instances.py command line option, 14
    remove_solver.py command line option, 15
--no-ablation
    generate_report.py command line option, 13
--number-of-runs
    configure_solver.py command line option, 12
    run_ablation.py command line option, 15
--parallel
    add_feature_extractor.py command line option, 9
    add_instances.py command line option, 9

add_solver.py command line option, 10
compute_features.py command line option, 10
run_configured_solver.py command line option, 16
run_solvers.py command line option, 16
--perfect
compute_marginal_contribution.py command line option, 11
--performance-measure
compute_marginal_contribution.py command line option, 11
configure_solver.py command line option, 12
construct_sparkle_portfolio_selector.py command line option, 12
generate_report.py command line option, 13
run_ablation.py command line option, 15
run_configured_solver.py command line option, 16
run_solvers.py command line option, 16
run_sparkle_portfolio_selector.py command line option, 17
validate_configured_vs_default.py command line option, 19
--racing
run_ablation.py command line option, 15
--recompute
compute_features.py command line option, 10
compute_features_parallel.py command line option, 11
compute_marginal_contribution.py command line option, 11
run_solvers.py command line option, 16
--recompute-marginal-contribution
construct_sparkle_portfolio_selector.py command line option, 12
--recompute-portfolio-selector
construct_sparkle_portfolio_selector.py command line option, 12
--run-extractor-later
add_feature_extractor.py command line option, 8
add_instances.py command line option, 9
--run-extractor-now
add_feature_extractor.py command line option, 8
add_instances.py command line option, 9
--run-solver-later
add_instances.py command line option, 9
add_solver.py command line option, 10
--run-solver-now
add_instances.py command line option, 9
add_solver.py command line option, 10
--selection
generate_report.py command line option, 13
--settings-file
compute_features.py command line option, 10
compute_marginal_contribution.py command line option, 11
configure_solver.py command line option, 12
generate_report.py command line option, 13
run_ablation.py command line option, 15
run_configured_solver.py command line option, 16
run_solvers.py command line option, 17
run_sparkle_portfolio_selector.py command line option, 17
validate_configured_vs_default.py command line option, 19
--solver
configure_solver.py command line option, 12
generate_report.py command line option, 13
run_ablation.py command line option, 15
validate_configured_vs_default.py command line option, 19
--solver-variations
add_solver.py command line option, 10
--target-cutoff-time
configure_solver.py command line option, 12
run_ablation.py command line option, 15
run_solvers.py command line option, 16
validate_configured_vs_default.py command line option, 19
--test-case-directory
generate_report.py command line option, 13
--use-features
configure_solver.py command line option, 12
--validate
configure_solver.py command line option, 11
--verbose
run_status.py command line option, 17
system_status.py command line option, 18
--verifier
run_solvers.py command line option, 16
-h

about.py command line option, 8
add_feature_extractor.py command line option, 8
add_instances.py command line option, 9
add_solver.py command line option, 9
cleanup_current_sparkle_platform.py command line option, 10
cleanup_temporary_files.py command line option, 10
compute_features.py command line option, 10
compute_features_parallel.py command line option, 11
compute_marginal_contribution.py command line option, 11
configure_solver.py command line option, 11
construct_sparkle_portfolio_selector.py command line option, 12
generate_report.py command line option, 13
initialise.py command line option, 13
load_record.py command line option, 14
remove_feature_extractor.py command line option, 14
remove_instances.py command line option, 14
remove_record.py command line option, 14
remove_solver.py command line option, 15
run_ablation.py command line option, 15
run_configured_solver.py command line option, 16
run_solvers.py command line option, 16
run_sparkle_portfolio_selector.py command line option, 17
run_status.py command line option, 17
save_record.py command line option, 17
sparkle_wait.py command line option, 18
system_status.py command line option, 18
validate_configured_vs_default.py command line option, 18
-v
run_status.py command line option, 17
system_status.py command line option, 18

A

about.py command line option
--help, 8
-h, 8
add_feature_extractor.py command line option
--help, 8
--nickname, 9
--parallel, 9
--run-extractor-later, 8
--run-extractor-now, 8
-h, 8
extractor-path, 8
add_instances.py command line option
--help, 9
--nickname, 9
--parallel, 9
--run-extractor-later, 9
--run-extractor-now, 9
--run-solver-later, 9
--run-solver-now, 9
-h, 9
instances-path, 9
add_solver.py command line option
--deterministic, 9
--help, 9
--nickname, 10
--parallel, 10
--run-solver-later, 10
--run-solver-now, 10
--solver-variations, 10
-h, 9
solver-path, 9

C

cleanup_current_sparkle_platform.py command line option
--help, 10
-h, 10
cleanup_temporary_files.py command line option
--help, 10
-h, 10
compute_features.py command line option
--help, 10
--parallel, 10
--recompute, 10
--settings-file, 10
-h, 10
compute_features_parallel.py command line option
--help, 11
--recompute, 11
-h, 11
compute_marginal_contribution.py command line option
--actual, 11
--help, 11
--perfect, 11
--performance-measure, 11
--recompute, 11
--settings-file, 11
-h, 11

configure_solver.py command line option
    --ablation, 11
    --budget-per-run, 12
    --help, 11
    --instance-set-test, 12
    --instance-set-train, 12
    --number-of-runs, 12
    --performance-measure, 12
    --settings-file, 12
    --solver, 12
    --target-cutoff-time, 12
    --use-features, 12
    --validate, 11
    -h, 11
construct_sparkle_portfolio_selector.py command line option
    --help, 12
    --performance-measure, 12
    --recompute-marginal-contribution, 12
    --recompute-portfolio-selector, 12
    -h, 12

E

extractor-path
    add_feature_extractor.py command line option, 8
    remove_feature_extractor.py command line option, 14

G

generate_report.py command line option
    --help, 13
    --instance-set-test, 13
    --instance-set-train, 13
    --no-ablation, 13
    --performance-measure, 13
    --selection, 13
    --settings-file, 13
    --solver, 13
    --test-case-directory, 13
    -h, 13

I

initialise.py command line option
    --help, 13
    -h, 13
instance_path
    run_configured_solver.py command line option, 16
    run_sparkle_portfolio_selector.py command line option, 17
instances-path
    add_instances.py command line option, 9
    remove_instances.py command line option, 14

L

load_record.py command line option
    --help, 14
    -h, 14
    record-file-path, 14

R

record-file-path
    load_record.py command line option, 14
    remove_record.py command line option, 14
remove_feature_extractor.py command line option
    --help, 14
    --nickname, 14
    -h, 14
    extractor-path, 14
remove_instances.py command line option
    --help, 14
    --nickname, 14
    -h, 14
    instances-path, 14
remove_record.py command line option
    --help, 14
    -h, 14
    record-file-path, 14
remove_solver.py command line option
    --help, 15
    --nickname, 15
    -h, 15
    solver-path, 15
run_ablation.py command line option
    --ablation-settings-help, 15
    --budget-per-run, 15
    --help, 15
    --instance-set-test, 15
    --instance-set-train, 15
    --number-of-runs, 15
    --performance-measure, 15
    --racing, 15
    --settings-file, 15
    --solver, 15
    --target-cutoff-time, 15
    -h, 15
run_configured_solver.py command line option
    --help, 16
    --parallel, 16
    --performance-measure, 16
    --settings-file, 16
    -h, 16
    instance_path, 16

run_solvers.py command line option
    --also-construct-selector-and-report, 16
    --help, 16
    --parallel, 16
    --performance-measure, 16
    --recompute, 16
    --settings-file, 17
    --target-cutoff-time, 16
    --verifier, 16
    -h, 16
run_sparkle_portfolio_selector.py command
        line option
    --help, 17
    --performance-measure, 17
    --settings-file, 17
    -h, 17
    instance_path, 17
run_status.py command line option
    --help, 17
    --verbose, 17
    -h, 17
    -v, 17

S

save_record.py command line option
    --help, 17
    -h, 17
solver-path
    add_solver.py command line option, 9
    remove_solver.py command line option, 15
sparkle_wait.py command line option
    --command, 18
    --help, 18
    --job-id, 18
    -h, 18
system_status.py command line option
    --help, 18
    --verbose, 18
    -h, 18
    -v, 18

V

validate_configured_vs_default.py command
        line option
    --help, 18
    --instance-set-test, 19
    --instance-set-train, 19
    --performance-measure, 19
    --settings-file, 19
    --solver, 19
    --target-cutoff-time, 19
    -h, 18