sparkle Documentation

Release 1.2

ADA Research Group, LIACS

Apr 16, 2021

CONTENTS:

CHAPTER

ONE

INSTALLATION

Before starting Sparkle, please install the following packages with the specific versions:

1. Install Python 3.9 – other 3.x versions may work, but were not tested
   - with anaconda:

     ```
     conda create -n <env_name> python=3.9
     ```

     ```
     conda activate <env_name>
     ```
2. Install swig version 3.0
   - with anaconda:

     ```
     conda install swig=3.0
     ```
3. Install first set of required python packages:
   - `pip install -r requirements_first.txt`
   - or with anaconda:

     ```
     /home/<username>/<anaconda_dir>/envs/<env_name>/bin/pip install -r
     requirements_first.txt
     ```
4. Install second set of required python packages:
   - `pip install -r requirements_second.txt`
   - or with anaconda:

     ```
     /home/<username>/<anaconda_dir>/envs/<env_name>/bin/pip install -r
     requirements_second.txt
     ```
5. Install other requirements if they are not on your system yet:
   - `epstopdf`
   - `LaTeX`
   - `BibTeX`
   - `gnuplot`

CHAPTER

TWO

SPARKLE USER GUIDE

2.1 Quick start

Note: Sparkle currently relies on Slurm (`https://slurm.schedmd.com/`), and does not work without it.

Follow these steps:

- Install Sparkle
- Prepare your configuration environment; or *1.3* Prepare your selection environment
- Execute commands

2.1.1 Installing Sparkle

1. Copy the Sparkle files to your desired directory
2. Install Python 3.9 (other 3.x versions may work, but were not tested with the packages included in the `requirements_first.txt` and `requirements_second.txt` files.
   With Anaconda:
   `conda create -n <env_name> python=3.9`
3. Install Swig 3.0
   With Anaconda:
   `conda install swig=3.0`
4. Navigate into the Sparkle directory
5. Install `requirements_first.txt`:
   `pip install -r requirements_first.txt`
   With Anaconda:
   `/home/<username>/<anaconda_dir>/envs/<env_name>/bin/pip install -r requirements_first.txt`
6. Install `requirements_second.txt`:
   `pip install -r requirements_second.txt`
   With Anaconda:
   `/home/<username>/<anaconda_dir>/envs/<env_name>/bin/pip install -r requirements_second.txt`
7. Install `epstopdf`
   (if manually, for instance on a cluster, as described in Section *6.1.2*)
8. Install other requirements if they are not on your system yet:

```
LaTeX
BibTeX
gnuplot
```

2.1.2 Algorithm Configuration

Configuring an algorithm has the following minimal requirements for the algorithm (for an example of a solver directory see Section *2.2*):

- 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 *2.1*). 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.

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.

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.

2.1.3 Algorithm Selection

Creating a portfolio selector requires multiple algorithms with the following minimal requirements (for an example of a solver directory see Section *2.3*):

- 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 *2.1*). For the purpose of testing whether your selection setup works with Sparkle, it is advised to primarily use instances that are solved (relatively) quickly.

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

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.

2.1.4 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 Slurm the above command would change to something like:

```
srun -N1 -n1 -p graceTST 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.

2.2 File structure

2.2.1 A typical instance directory

An instance directory should look something like this:

```
Instances/
  Example_Instance_Set/
    instance_a.cnf
    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 should be included in the `Example_Instance_Set` directory, describing which files together form an instance. The format is a signle 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
```

2.2.2 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.

2.2.3 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.

2.3 Wrappers

2.3.1 `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.

2.4 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_parallel.py`
- `compute_features.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_solvers.py`
- `run_sparkle_portfolio_selector.py`
- `run_status.py`
- `save_record.py`
- `system_status.py`
- `validate_configured_vs_default.py`

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.

2.4.1 add_solver.py

Add a solver to the Sparkle platform.

Arguments:

- [-run-solver-later]
- [-parallel]
- -deterministic {0, 1}
- <solver_source_directory>

2.4.2 configure_solver.py

Configure a solver in the Sparkle platform.

Arguments:

- -solver <solver>
- -instance-set-train <instance-set-train>
- [-instance-set-test <instance-set-test>]
- -validate
- -ablation

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

2.4.3 generate_report.py

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

Generate a configuration report

Generate a report describing the configuration results for a solver and specific instance sets in the Sparkle platform.

Arguments:

- -solver <solver>
- [-instance-set-train <instance-set-train>]
- [-instance-set-test <instance-set-test>]

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

2.4.4 initialise.py

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

2.4.5 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.

Arguments:

- `-solver <solver>`
- `[-instance-set-train <instance-set-train>]`
- `[-instance-set-test <instance-set-test>]`

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

2.4.6 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.

Arguments:

- `-solver <solver>`
- `-instance-set-train <instance-set-train>`
- `[-instance-set-test <instance-set-test>]`

2.5 Settings

2.5.1 Sparkle settings

Most settings can be controlled through `Settings/sparkle_settings.ini`. Possible settings are summarised per category in Sect. *5.1.2*. 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.

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
```

Names and possible values

- **[general]**
- `performance_measure`

  aliases: `smac_run_obj`

  values: `{RUNTIME, QUALITY_ABSOLUTE` (also: `QUALITY)}`

- `target_cutoff_time`

  aliases: `smac_each_run_cutoff_time, cutoff_time_each_performance_computation`

  values: integer

- `extractor_cutoff_time`

  aliases: `cutoff_time_each_feature_computation`

  values: integer

- `penalty_multiplier`

  aliases: `penalty_number`

  values: integer

- `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

- `number_of_runs`

  aliases: `num_of_smac_runs`

values: integer

- **[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

- **[slurm]**
- `number_of_runs_in_parallel`

  aliases: `smac_run_obj`

  values: integer

- `clis_per_node`

  aliases: N/A

  values: integer

  note: Not really a Slurm option, will likely be moved to another section.

2.5.2 Priorities

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.

2.5.3 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 *5.3.2*).

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/PbO-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.

2.6 Required packages

2.6.1 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.

epstopdf

The `epstopdf` package (or a package containing it) is required for Sparkle's reporting component to work (e.g. `generate_report, generate_report_for_configuration`), it can be installed in your user directory as follows:

1. Download `epstopdf`

   ```
   wget http://mirrors.ctan.org/support/epstopdf.zip
   ```

2. Unzip the package (ideally somewhere static, rather than a `/Downloads/` directory)

   ```
   unzip epstopdf.zip
   ```

3. Rename `epstopdf.pl` (inside the directory you just unzipped)

   ```
   mv epstopdf.pl epstopdf
   ```

4. Add this line to your `.bashrc` (open with e.g. `vim ~/.bashrc`)

   ```
   export PATH="/<directory>/epstopdf:$PATH"
   ```

   (replace "`<directory>`" with the path to the `epstopdf` directory, e.g.: `home/blomkvander/bin`)

5. Reload `.bashrc` to make sure everything is updated

   ```
   source ~/.bashrc
   ```

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/

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

2.7 Installation and compilation of examples

2.7.1 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 ../../
```

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
```

(continues on next page)

(continued from previous page)

```
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 ../
```

CHAPTER

THREE

EXAMPLES

3.1 Use Sparkle for algorithm configuration

3.1.1 Initialise the Sparkle platform

```
Commands/initialise.py
```

3.1.2 Add instances

Add train, and optionally test, instances (in this case in CNF format) in a given directory, without running solvers or feature extractors yet

```
Commands/add_instances.py --run-solver-later --run-extractor-later Examples/
Resources/Instances/PTN/          Commands/add_instances.py --run-solver-later
--run-extractor-later Examples/Resources/Instances/PTN2/
```

3.1.3 Add a configurable solver

Add a configurable solver (here for SAT solving) with a wrapper containing the executable name of the solver and a string of command line parameters, without running the solver yet

The solver directory should contain the solver executable, the `sparkle_smac_wrapper.py` wrapper, and a `.pcs` file describing the configurable parameters

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/Solvers/PbO-CCSAT-Generic/
```

If needed solvers can also include additional files or scripts in their directory, but keeping additional files to a minimum speeds up copying.

3.1.4 Configure the solver

Perform configuration on the solver to obtain a target configuration

```
Commands/configure_solver.py --solver Solvers/PbO-CCSAT-Generic/
--instance-set-train Instances/PTN/
```

3.1.5 Validate the configuration

To make sure configuration is completed before running validation you can use the `sparkle_wait` command

```
Commands/sparkle_wait.py
```

Validate the performance of the best found parameter configuration. The test set is optional.

```
Commands/validate_configured_vs_default.py --solver Solvers/PbO-CCSAT-Generic/
--instance-set-train Instances/PTN/ --instance-set-test Instances/PTN2/
```

3.1.6 Generate a report

Wait for validation to be completed

```
Commands/sparkle_wait.py
```

Generate a report detailing the results on the training (and optionally testing) set. This includes the experimental procedure and performance information; this will be located in a `Configuration_Reports/` subdirectory for the solver, training set, and optionally test set like `PbO-CCSAT-Generic_PTN/Sparkle-latex-generator-for-configuration/`

```
Commands/generate_report.py
```

By default the `generate_report` command will create a report for the most recent solver and instance set(s). To generate a report for older solver-instance set combinations, the desired solver can be specified with `--solver Solvers/PbO-CCSAT-Generic/`, the training instance set with `--instance-set-train Instances/PTN/`, and the testing instance set with `--instance-set-test Instances/PTN2/`.

3.1.7 Run ablation to determine parameter importance based on default (from the `.pcs` file) and configured parameters

Run ablation

Run ablation using the training instances and validate the parameter importance with the test set

```
Commands/run_ablation.py --solver Solvers/PbO-CCSAT-Generic/
--instance-set-train Instances/PTN/ --instance-set-test Instances/PTN2/
```

Generate a report

Wait for ablation to be completed

```
Commands/sparkle_wait.py
```

Generate a report including ablation, and as before the results on the train (and optionally test) set, the experimental procedure and performance information; this will be located in a `Configuration_Reports/` subdirectory for the solver, training set, and optionally test set like `PbO-CCSAT-Generic_PTN/Sparkle-latex-generator-for-configuration/`

```
Commands/generate_report.py
```

The ablation section can be suppressed with `--no-ablation`

3.1.8 Immediate ablation and validation after configuration

By adding `--ablation` and/or `--validate` to the `configure_solver.py` command, ablation and respectively validation will run directly after the configuration is finished.

There is no need to execute `run_ablation.py` and/or `validate_configured_vs_default.py` when these flags are given with the `configure_solver.py` command

Training set only

```
Commands/configure_solver.py --solver Solvers/PbO-CCSAT-Generic/
--instance-set-train Instances/PTN/ --ablation --validate
```

Training and testing sets

Wait for the previous example to be completed

```
Commands/sparkle_wait.py
```

```
Commands/configure_solver.py --solver Solvers/PbO-CCSAT-Generic/
--instance-set-train Instances/PTN/ --instance-set-test Instances/PTN2/
--ablation --validate
```

3.2 Use Sparkle for algorithm configuration

3.2.1 Initialise the Sparkle platform

```
Commands/initialise.py
```

3.2.2 Add instances

Add train, and optionally test, instances (in this case for the VRP) in a given directory, without running solvers or feature extractors yet

```
Commands/add_instances.py --run-solver-later --run-extractor-later Examples/
Resources/Instances/X-1-10/       Commands/add_instances.py --run-solver-later
--run-extractor-later Examples/Resources/Instances/X-11-20/
```

3.2.3 Add a configurable solver

Add a configurable solver (here for vehicle routing) with a wrapper containing the executable name of the solver and a string of command line parameters, without running the solver yet

The solver directory should contain the solver executable, the `sparkle_smac_wrapper.py` wrapper, and a `.pcs` file describing the configurable parameters

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/Solvers/VRP_SISRs/
```

If needed solvers can also include additional files or scripts in their directory, but keeping additional files to a minimum speeds up copying.

3.2.4 Configure the solver

Perform configuration on the solver to obtain a target configuration. For the VRP we measure the absolute quality performance by setting the `--performance-measure` option, to avoid needing this for every command it can also be set in `Settings/sparkle_settings.ini`.

```
Commands/configure_solver.py --solver Solvers/VRP_SISRs/ --instance-set-train
Instances/X-1-10/ --performance-measure QUALITY_ABSOLUTE
```

3.2.5 Validate the configuration

To make sure configuration is completed before running validation you can use the `sparkle_wait` command

```
Commands/sparkle_wait.py
```

Validate the performance of the best found parameter configuration. The test set is optional. We again set the performance measure to absolute quality.

```
Commands/validate_configured_vs_default.py --solver Solvers/VRP_SISRs/
--instance-set-train Instances/X-1-10/ --instance-set-test Instances/X-11-20/
--performance-measure QUALITY_ABSOLUTE
```

3.2.6 Generate a report

Wait for validation to be completed

```
Commands/sparkle_wait.py
```

Generate a report detailing the results on the training (and optionally testing) set. This includes the experimental procedure and performance information; this will be located in a `Configuration_Reports/` subdirectory for the solver, training set, and optionally test set like `VRP_SISRs_X-1-10/Sparkle-latex-generator-for-configuration/`. We again set the performance measure to absolute quality.

```
Commands/generate_report.py --performance-measure QUALITY_ABSOLUTE
```

By default the `generate_report` command will create a report for the most recent solver and instance set(s). To generate a report for older solver-instance set combinations, the desired solver can be specified with `--solver Solvers/VRP_SISRs/`, the training instance set with `--instance-set-train Instances/X-1-10/`, and the testing instance set with `--instance-set-test Instances/X-11-20/`.

3.3 Use Sparkle for algorithm selection

3.3.1 Initialise the Sparkle platform

```
Commands/initialise.py
```

3.3.2 Add instances

Add instance files (in this case in CNF format) in a given directory, without running solvers or feature extractors yet

```
Commands/add_instances.py --run-solver-later --run-extractor-later Examples/
Resources/Instances/PTN/
```

3.3.3 Add solvers

Add solvers (here for SAT solving) with a wrapper containing the executable name of the solver and a string of command line parameters, without running the solvers yet

Each solver directory should contain the solver executable and a wrapper

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/Solvers/CSCCSat/
```

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/Solvers/PbO-CCSAT-Generic/
```

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/Solvers/MiniSAT/
```

3.3.4 Add feature extractor

Similarly, add a feature extractor, without immediately running it on the instances

```
Commands/add_feature_extractor.py --run-extractor-later Examples/Resources/
Extractors/SAT-features-competition2012_revised_without_SatELite_sparkle/
```

3.3.5 Compute features

Compute features for all the instances; add the `--parallel` option to run in parallel

```
Commands/compute_features.py
```

3.3.6 Run the solvers

Run the solvers on all instances; add the `--parallel` option to run in parallel

```
Commands/run_solvers.py
```

3.3.7 Construct a portfolio selector

To make sure feature computation and solver performance computation are done before constructing the portfolio use the `sparkle_wait` command

```
Commands/sparkle_wait.py
```

Construct a portfolio selector, using the previously computed features and the results of running the solvers

```
Commands/construct_sparkle_portfolio_selector.py
```

3.3.8 Generate a report

Generate an experimental report detailing the experimental procedure and performance information; this will be located at `Components/Sparkle-latex-generator/Sparkle_Report.pdf`

```
Commands/generate_report.py
```

3.4 Run the portfolio selector (e.g. on a test set)

3.4.1 Run on a single instance

Run the portfolio selector on a *single* testing instance; the result will be printed to the command line

```
Commands/run_sparkle_portfolio_selector.py Examples/Resources/Instances/PTN2/
plain7824.cnf
```

3.4.2 Run on an instance set

Run the portfolio selector on a testing instance *set*

```
Commands/run_sparkle_portfolio_selector.py Examples/Resources/Instances/PTN2/
```

3.4.3 Generate a report including results on the test set

Wait for the portfolio selector to be done running on the testing instance set

```
Commands/sparkle_wait.py
```

Generate an experimental report that includes the results on the test set, and as before the experimental procedure and performance information; this will be located at `Components/Sparkle-latex-generator/Sparkle_Report_For_Test.pdf`

```
Commands/generate_report.py
```

By default the `generate_report` command will create a report for the most recent instance set. To generate a report for an older instance set, the desired instance set can be specified with: `--test-case-directory Test_Cases/PTN2/`

3.5 Use Sparkle for algorithm selection with multi-file instances

3.5.1 Initialise the Sparkle platform

```
Commands/initialise.py
```

3.5.2 Add instances

Add instance files in a given directory, without running solvers or feature extractors yet. In addition to the instance files, the directory should contain a file `sparkle_instance_list.txt` where each line contains a space separated list of files that together form an instance.

```
Commands/add_instances.py --run-solver-later --run-extractor-later Examples/
Resources/CCAG/Instances/CCAG/
```

3.5.3 Add solvers

Add solvers (here for the constrained covering array generation (CCAG) problem) with a wrapper containing the executable name of the solver and a string of command line parameters, without running the solvers yet

Each solver directory should contain the solver executable and a wrapper

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/CCAG/Solvers/TCA/
```

```
Commands/add_solver.py --run-solver-later --deterministic 0 Examples/
Resources/CCAG/Solvers/FastCA/
```

3.5.4 Add feature extractor

Similarly, add a feature extractor, without immediately running it on the instances

```
Commands/add_feature_extractor.py --run-extractor-later Examples/Resources/
CCAG/Extractors/CCAG-features_sparkle/
```

3.5.5 Compute features

Compute features for all the instances; add the `--parallel` option to run in parallel

```
Commands/compute_features.py
```

3.5.6 Run the solvers

Run the solvers on all instances; add the `--parallel` option to run in parallel. For the CCAG (Constrained Covering Array Generation) problem we measure the absolute quality performance by setting the `--performance-measure` option, to avoid needing this for every command it can also be set in `Settings/sparkle_settings.ini`.

```
Commands/run_solvers.py --performance-measure QUALITY_ABSOLUTE
```

3.5.7 Construct a portfolio selector

To make sure feature computation and solver performance computation are done before constructing the portfolio use the `sparkle_wait` command

```
Commands/sparkle_wait.py
```

Construct a portfolio selector, using the previously computed features and the results of running the solvers. We again set the performance measure to absolute quality.

```
Commands/construct_sparkle_portfolio_selector.py --performance-measure
QUALITY_ABSOLUTE
```

3.5.8 [Coming soon] Generate a report

This is not yet implemented for quality performance

Generate an experimental report detailing the experimental procedure and performance information; this will be located at `Components/Sparkle-latex-generator/Sparkle_Report.pdf`. We again set the performance measure to absolute quality.

```
Commands/generate_report.py --performance-measure QUALITY_ABSOLUTE
```

3.6 [Coming soon] Run the portfolio selector (e.g. on the test set)

This is not yet implemented for quality performance

This is not yet implemented for multi-file instances

3.6.1 Run on a single instance

Run the portfolio selector on a *single* testing instance; the result will be printed to the command line. We again set the performance measure to absolute quality.

```
Commands/run_sparkle_portfolio_selector.py Examples/Resources/CCAG/Instances/
CCAG2/Banking2.model Examples/Resources/CCAG/Instances/CCAG2/Banking2.
constraints --performance-measure QUALITY_ABSOLUTE
```

3.6.2 Run on an instance set

Run the portfolio selector on a testing instance *set*. We again set the performance measure to absolute quality.

```
Commands/run_sparkle_portfolio_selector.py Examples/Resources/CCAG/Instances/
CCAG2/ --performance-measure QUALITY_ABSOLUTE
```

3.6.3 Generate a report including results on the test set

Wait for the portfolio selector to be done running on the testing instance set

```
Commands/sparkle_wait.py
```

Generate an experimental report that includes the results on the test set, and as before the experimental procedure and performance information; this will be located at `Components/Sparkle-latex-generator/Sparkle_Report_For_Test.pdf`. We again set the performance measure to absolute quality.

```
Commands/generate_report.py --performance-measure QUALITY_ABSOLUTE
```

By default the `generate_report` command will create a report for the most recent instance set. To generate a report for an older instance set, the desired instance set can be specified with: `--test-case-directory Test_Cases/CCAG2/`

CHAPTER

FOUR

OUTPUT PATHS

4.1 Configuration

Path: ../../Components/smac-v2.10.03-master-778/ File: <solver_name><*instance_set*>scenario.txt<*number_of_configuration_runs*>ex
Description: sbatch script that executes configuration experiments through Slurm Output: Slurm stderr: tmp/<solver_name><*instance_set*>scenario.txt<*number_of_configuration_runs*>exp_sbatch.sh.err Slurm stdout: tmp/<solver_name><*instance_set*>scenario.txt<*number_of_configuration_runs*>exp_sbatch.sh.txt Configurator log: results/<solver_name><*instance_set*>/<*solver_name*><instance_set>*scenario.txt*<number_of_configuration_runs>*exp_sbatch.sh_seed*
Execution directory: example_scenarios/<solver_name>/<run_id>/ Configurator detailed output: example_scenarios/cadical-sc17_for_sparkle/outdir_train_configuration/cadical-sc17_for_sparkle_train_scenario/
Validation log: log-val-1.txt

Other: Scenario file: example_scenarios/<solver_name>/<solver_name>_train_scenario.txt Last solver and instance set used for configuration: example_scenarios/cadical-sc17_for_sparkle/last_configuration.txt Last solver, training-, and testing instance sets used for validation: example_scenarios/cadical-sc17_for_sparkle/last_test_configured_default.txt Training instances: <instance_set>_train.txt

4.2 Validation

Path: ../../Components/smac-v2.10.03-master-778/ File: <solver_name>_<instance_set>*validation_sbatch.sh Output: Slurm stderr: tmp/<solver_name>*<instance_set>*validation_sbatch.sh.err Slurm stdout: tmp/<solver_name>*<instance_set>_validation_sbatch.sh.txt Validation log: results/<solver_name>_validation_train_train_default_scenario.txt Description: sbatch script that executes validation experiments through Slurm

CHAPTER

FIVE

COMMANDS

5.1 Commands package

5.1.1 Subpackages

Commands.sparkle_help package

Submodules

Commands.sparkle_help.argparse_custom module

class Commands.sparkle_help.argparse_custom.**SetByUser**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: argparse.Action

Commands.sparkle_help.argparse_custom.**set_by_user**(*args*, *arg_name: str*) → bool

Commands.sparkle_help.argparse_custom.**user_set_state**(*args*, *arg_name: str*) → sparkle_help.sparkle_settings.SettingState

Commands.sparkle_help.compute_features_core module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.reporting_scenario module

class Commands.sparkle_help.reporting_scenario.**ReportingScenario**

Bases: object

DEFAULT_config_instance_set_test = PosixPath('.')

DEFAULT_config_instance_set_train = PosixPath('.')

DEFAULT_config_solver = PosixPath('.')

DEFAULT_latest_scenario = 0

DEFAULT_reporting_scenario_path = PosixPath('Output/latest_scenario.ini')

DEFAULT_selection_portfolio_path = PosixPath('.')

DEFAULT_selection_test_case_directory = PosixPath('.')

get_config_instance_set_test() → pathlib.Path

get_config_instance_set_train() → pathlib.Path

get_config_solver() → pathlib.Path

get_latest_scenario() → Commands.sparkle_help.reporting_scenario.Scenario

get_selection_portfolio_path() → pathlib.Path

get_selection_test_case_directory() → pathlib.Path

none_if_empty_path(*path: pathlib.Path*)

path_setter(*section: str*, *name: str*, *value: pathlib.Path*)

read_scenario_ini(*file_path: pathlib.Path = PosixPath('Output/latest_scenario.ini')*)

set_config_instance_set_test(*value: pathlib.Path = PosixPath('.')*)

set_config_instance_set_train(*value: pathlib.Path = PosixPath('.')*)

set_config_solver(*value: pathlib.Path = PosixPath('.')*)

set_latest_scenario(*value: Commands.sparkle_help.reporting_scenario.Scenario = <Scenario.NONE: 0>*)

set_selection_portfolio_path(*value: pathlib.Path = PosixPath('.')*)

set_selection_test_case_directory(*value: pathlib.Path = PosixPath('.')*)

write_scenario_ini(*file_path: pathlib.Path = PosixPath('Output/latest_scenario.ini')*)

class Commands.sparkle_help.reporting_scenario.**Scenario**

Bases: enum.Enum

An enumeration.

CONFIGURATION = 2

NONE = 0

SELECTION = 1

from_str()

Commands.sparkle_help.run_solvers_core module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.run_sparkle_portfolio_core module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_add_configured_solver_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_add_configured_solver_help.**check_adding_solver_contain_pcs_fi**

Commands.sparkle_help.sparkle_add_configured_solver_help.**create_necessary_files_for_configu**

Commands.sparkle_help.sparkle_add_configured_solver_help.**get_pcs_file_from_solver_directory**

Commands.sparkle_help.sparkle_add_solver_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_add_solver_help.**check_adding_solver_contain_pcs_file**(*solver_dire* *str*) → bool

Commands.sparkle_help.sparkle_add_solver_help.**get_pcs_file_from_solver_directory**(*solver_directo* *str*) → str

Commands.sparkle_help.sparkle_basic_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_basic_help.**get_time_pid_random_string**()

Commands.sparkle_help.sparkle_command_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

class Commands.sparkle_help.sparkle_command_help.**CommandName**

Bases: enum.Enum

An enumeration.

ABOUT = 0

ADD_FEATURE_EXTRACTOR = 1

ADD_INSTANCES = 2

ADD_SOLVER = 3

CLEANUP_CURRENT_SPARKLE_PLATFORM = 4

CLEANUP_TEMPORARY_FILES = 5

COMPUTE_FEATURES = 6

COMPUTE_MARGINAL_CONTRIBUTION = 7

CONFIGURE_SOLVER = 8

CONSTRUCT_SPARKLE_PORTFOLIO_SELECTOR = 9

GENERATE_REPORT = 10

INITIALISE = 11

LOAD_RECORD = 12

REMOVE_FEATURE_EXTRACTOR = 13

REMOVE_INSTANCES = 14

REMOVE_RECORD = 15

REMOVE_SOLVER = 16

RUN_ABLATION = 17

RUN_SOLVERS = 18

RUN_SPARKLE_PORTFOLIO_SELECTOR = 19

RUN_STATUS = 20

SAVE_RECORD = 21

SPARKLE_WAIT = 22

SYSTEM_STATUS = 23

VALIDATE_CONFIGURED_VS_DEFAULT = 24

from_str()

Commands.sparkle_help.sparkle_compute_features_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_compute_features_help.**computing_features**(*feature_data_csv_path*, *mode*)

Commands.sparkle_help.sparkle_compute_features_help.**generate_missing_value_csv_like_feature**

Commands.sparkle_help.sparkle_compute_features_help.**get_feature_data_id**() → int

Commands.sparkle_help.sparkle_compute_features_help.**update_feature_data_id**()

Commands.sparkle_help.sparkle_compute_features_parallel_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_compute_features_parallel_help.**computing_features_parallel**(*fe*
m

Commands.sparkle_help.sparkle_compute_marginal_contribution_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**compute_actual_performance**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**compute_actual_selector_ma**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**compute_actual_selector_pe**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**compute_actual_used_time_**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**compute_perfect_selector_m**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**get_capvalue_list**(*performanc*
sparkle_he
→
List[float]

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**get_list_predict_schedule**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**print_rank_list**(*rank_list*, *mode*)

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**read_marginal_contributio**

Commands.sparkle_help.sparkle_compute_marginal_contribution_help.**write_marginal_contributi**

Commands.sparkle_help.sparkle_configure_solver_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

class Commands.sparkle_help.sparkle_configure_solver_help.**InstanceType**

Bases: enum.Enum

An enumeration.

TEST = 2

TRAIN = 1

Commands.sparkle_help.sparkle_configure_solver_help.**check_configuration_exists**(*solver_name: str*, *instance_set_name str*) → bool

Commands.sparkle_help.sparkle_configure_solver_help.**check_instance_list_file_exist**(*solver_nam str*, *instance_set_ str*)

Commands.sparkle_help.sparkle_configure_solver_help.**check_optimised_configuration_params**(*pa st*

Commands.sparkle_help.sparkle_configure_solver_help.**check_optimised_configuration_performa**

Commands.sparkle_help.sparkle_configure_solver_help.**check_optimised_configuration_seed**(*seed:*
str)

Commands.sparkle_help.sparkle_configure_solver_help.**check_validation_prerequisites**(*solver_nan*
str,
in-
*stance_set_*
str)

Commands.sparkle_help.sparkle_configure_solver_help.**clean_configuration_directory**(*solver_name*
str)

Commands.sparkle_help.sparkle_configure_solver_help.**create_configuration_directory**(*solver_nan*
str)

Commands.sparkle_help.sparkle_configure_solver_help.**create_file_scenario_configuration**(*solve*
str,
in-
stanc
str)

Commands.sparkle_help.sparkle_configure_solver_help.**create_file_scenario_validate**(*solver_name*
str,
in-
stance_set_n
str,
in-
stance_type:
Com-
mands.spark
de-
fault:
bool)
→
str

Commands.sparkle_help.sparkle_configure_solver_help.**create_smac_configure_sbatch_script**(*solv*
str,
in-
star
str)
→
str

Commands.sparkle_help.sparkle_configure_solver_help.**generate_ablation_callback_slurm_script**

Commands.sparkle_help.sparkle_configure_solver_help.**generate_configuration_sbatch_script**(*sb*
sc
na
re
su
nu
nu
sm

Commands.sparkle_help.sparkle_configure_solver_help.**generate_configure_solver_wrapper**(*solver_*
op-
ti-
*mised_*

Commands.sparkle_help.sparkle_configure_solver_help.**generate_generic_callback_slurm_script**

Commands.sparkle_help.sparkle_configure_solver_help.**generate_validation_callback_slurm_scr**

Commands.sparkle_help.sparkle_configure_solver_help.**get_optimised_configuration**(*solver_name: str*, *instance_set_nam str*) → Tuple[str, str, str]

Commands.sparkle_help.sparkle_configure_solver_help.**get_optimised_configuration_from_file**(

Commands.sparkle_help.sparkle_configure_solver_help.**get_optimised_configuration_params**(*solve str*, *instanc str*) → str

Commands.sparkle_help.sparkle_configure_solver_help.**get_smac_run_obj**() → str

Commands.sparkle_help.sparkle_configure_solver_help.**get_smac_settings**()

Commands.sparkle_help.sparkle_configure_solver_help.**get_solver_deterministic**(*solver_name*)

Commands.sparkle_help.sparkle_configure_solver_help.**handle_file_instance**(*solver_name: str*, *instance_set_name: str*, *file_postfix: str*)

Commands.sparkle_help.sparkle_configure_solver_help.**handle_file_instance_test**(*solver_name: str*, *instance_set_name: str*)

Commands.sparkle_help.sparkle_configure_solver_help.**handle_file_instance_train**(*solver_name: str*, *instance_set_name str*)

Commands.sparkle_help.sparkle_configure_solver_help.**prepare_smac_execution_directories_con**

Commands.sparkle_help.sparkle_configure_solver_help.**prepare_smac_execution_directories_val**

Commands.sparkle_help.sparkle_configure_solver_help.**remove_configuration_directory**(*solver_nam*
str)

Commands.sparkle_help.sparkle_configure_solver_help.**remove_validation_directories**(*solver_name*
str)

Commands.sparkle_help.sparkle_configure_solver_help.**remove_validation_directories_executio**

Commands.sparkle_help.sparkle_configure_solver_help.**submit_smac_configure_sbatch_script**(*sma*
str)
→
str

Commands.sparkle_help.sparkle_configure_solver_help.**write_optimised_configuration_pcs**(*solver_*
in-
*stance_*

Commands.sparkle_help.sparkle_configure_solver_help.**write_optimised_configuration_str**(*solver_*
in-
*stance_*

Commands.sparkle_help.sparkle_construct_portfolio_selector_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**construct_sparkle_portfolio**

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**data_unchanged**(*sparkle_portfolio*
path-
lib.Path)
→
bool

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**get_selector_fd_id**(*selector_di*
path-
lib.PurePat
→
int

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**get_selector_pd_id**(*selector_di*
path-
lib.PurePat
→
int

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**selector_exists**(*sparkle_portfol*
path-
lib.Path)
→
bool

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**write_selector_fd_id**(*sparkle_*
path-
lib.Path

Commands.sparkle_help.sparkle_construct_portfolio_selector_help.**write_selector_pd_id**(*sparkle_*
path-
lib.Path

Commands.sparkle_help.sparkle_csv_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

class Commands.sparkle_help.sparkle_csv_help.**Sparkle_CSV**(*csv_filepath*)
Bases: object

add_column(*column_name*, *value_list=[]*)

add_row(*row_name*, *value_list=[]*)

clean_csv()

static create_empty_csv(*csv_filepath*)

dataframe_get_specific_column(*column_name*)

dataframe_get_specific_column_isnull(*column_name*)

dataframe_get_specific_row(*row_name*)

dataframe_get_specific_row_isnull(*row_name*)

delete_column(*column_name*)

delete_row(*row_name*)

empty_column_name = '"'

get_column_name(*index*)

get_column_size() → int

get_row_name(*index*)

get_row_size() → int

get_value(*row*, *column*)

get_value_index(*row_index*, *column_index*)

is_empty()

list_columns()

list_get_specific_column(*column_name*)

list_get_specific_column_isnull(*column_name*)

list_get_specific_row(*row_name*)

list_get_specific_row_isnull(*row_name*)

list_rows()

rename_column(*ori_column_name*, *mod_column_name*)

rename_row(*ori_row_name*, *mod_row_name*)

save_csv(*csv_filepath: str*)

set_value(*row*, *column*, *value*)

set_value_index(*row_index*, *column_index*, *value*)

update_column(*column_name*, *value_list*)

update_csv()

update_row(*row_name*, *value_list*)

Commands.sparkle_help.sparkle_csv_merge_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_csv_merge_help.**feature_data_csv_merge**()

Commands.sparkle_help.sparkle_csv_merge_help.**performance_data_csv_merge**()

Commands.sparkle_help.sparkle_experiments_related_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_experiments_related_help.**init**()

Commands.sparkle_help.sparkle_feature_data_csv_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

class Commands.sparkle_help.sparkle_feature_data_csv_help.**Sparkle_Feature_Data_CSV**(*csv_filepat*

Bases: sparkle_help.sparkle_csv_help.Sparkle_CSV

bool_exists_missing_value()

calc_mean_over_all_non_missing_values_of_this_column(*column_name*)

combine(*second_sfdcsv*)

generate_mean_value_feature_vector()

get_bool_in_columns(*given_column_name*)

get_bool_in_rows(*given_row_name*)

get_extractor_path_from_feature(*given_column_name*)

get_feature_vector_string(*instance*)

get_list_processed_feature_computation_job()

get_list_recompute_feature_computation_job()

get_list_remaining_feature_computation_job()

impute_missing_value_of_all_columns()

impute_missing_value_of_this_column(*column_name*)

reload_and_combine_and_update(*second_sfdcsv*)

test()

Commands.sparkle_help.sparkle_file_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_file_help.**add_new_extractor_feature_vector_size_into_file**(*filep*
fea-
ture

Commands.sparkle_help.sparkle_file_help.**add_new_extractor_into_file**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**add_new_extractor_nickname_into_file**(*nickname*,
filepath)

Commands.sparkle_help.sparkle_file_help.**add_new_instance_into_file**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**add_new_solver_into_file**(*filepath*,
determinis-
tic='0')

Commands.sparkle_help.sparkle_file_help.**add_new_solver_nickname_into_file**(*nickname*,
filepath)

Commands.sparkle_help.sparkle_file_help.**append_string_to_file**(*file_path*,
string_value)

Commands.sparkle_help.sparkle_file_help.**checkout_directory**(*path*,
make_if_not_exist=True)

Commands.sparkle_help.sparkle_file_help.**create_new_empty_file**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_current_directory_name**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_directory**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_file_full_extension**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_file_least_extension**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_file_name**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_instance_list_from_reference**(*instances_path: pathlib.Path*) → List[str]

Commands.sparkle_help.sparkle_file_help.**get_last_level_directory_name**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_cnf_filename**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_cnf_filename_recursive**(*path*, *list_all_cnf_filename*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_csv_filename**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_directory**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_directory_recursive**(*path*, *list_all_directory*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_filename**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_filename_recursive**(*path*, *list_all_filename*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_jobinfo_filename**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_result_filename**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**get_list_all_statusinfo_filename**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**remove_from_solver_list**(*filepath*)

Commands.sparkle_help.sparkle_file_help.**remove_line_from_file**(*line_start: str*, *filepath: pathlib.Path*)

Commands.sparkle_help.sparkle_file_help.**rmdir**(*dir_name: pathlib.Path*)

Commands.sparkle_help.sparkle_file_help.**rmfile**(*file_name: pathlib.Path*)

Commands.sparkle_help.sparkle_file_help.**rmtree**(*directory: pathlib.Path*)

Commands.sparkle_help.sparkle_file_help.**write_extractor_feature_vector_size_mapping**()

Commands.sparkle_help.sparkle_file_help.**write_extractor_list**()

Commands.sparkle_help.sparkle_file_help.**write_extractor_nickname_mapping**()

Commands.sparkle_help.sparkle_file_help.**write_instance_list**()

Commands.sparkle_help.sparkle_file_help.**write_solver_list**()

Commands.sparkle_help.sparkle_file_help.**write_solver_nickname_mapping**()

Commands.sparkle_help.sparkle_file_help.**write_string_to_file**(*file_path*, *string_value*)

Commands.sparkle_help.sparkle_generate_report_for_configuration_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**check_results_exist**(*so*
in
st
in
st

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**construct_list_instanc**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**construct_list_instanc**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**generate_report_for_co**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**generate_report_for_co**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**generate_report_for_co**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**generate_report_for_co**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_PAR10_performance**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_ablationBool**(*solver*
in-
stance
in-
stance

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_ablation_table**(*solv*
in-
star
in-
star

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_customCommands**()

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_defaultConfigurati**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_defaultConfigurat**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_dict_instance_to_**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_dict_variable_to_**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_dict_variable_to_**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_dict_variable_to_**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_figure_configured**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_figure_configured**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_instance_path_fro**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_most_recent_test_**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_numInstanceInInsta**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_optimisedConfigura**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_optimisedConfigura**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_performance_measu**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_quality_plot_boun**

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_runtimeBool**()

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_sparkle**()

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_sparkleVersion**()

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_timeouts_test**(*solve*
in-
stanc
cut-
off)

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**get_timeouts_train**(*solv*
in-
star
cut-
off)

Commands.sparkle_help.sparkle_generate_report_for_configuration_help.**write_data_for_plot_a**

Commands.sparkle_help.sparkle_generate_report_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_generate_report_help.**generate_report**(*test_case_directory: str = None*)

Commands.sparkle_help.sparkle_generate_report_help.**get_PAR10RankingList**()

Commands.sparkle_help.sparkle_generate_report_help.**get_VBSPAR10**()

Commands.sparkle_help.sparkle_generate_report_help.**get_actualPAR10**()

Commands.sparkle_help.sparkle_generate_report_help.**get_customCommands**()

Commands.sparkle_help.sparkle_generate_report_help.**get_dict_actual_portfolio_selector_penal**

Commands.sparkle_help.sparkle_generate_report_help.**get_dict_sbs_penalty_time_on_each_instar**

Commands.sparkle_help.sparkle_generate_report_help.**get_dict_variable_to_value**(*test_case_director str = None*)

Commands.sparkle_help.sparkle_generate_report_help.**get_dict_vbs_penalty_time_on_each_instar**

Commands.sparkle_help.sparkle_generate_report_help.**get_featureComputationCutoffTime**()

Commands.sparkle_help.sparkle_generate_report_help.**get_featureExtractorList**()

Commands.sparkle_help.sparkle_generate_report_help.**get_figure_portfolio_selector_sparkle_vs**

Commands.sparkle_help.sparkle_generate_report_help.**get_figure_portfolio_selector_sparkle_vs**

Commands.sparkle_help.sparkle_generate_report_help.**get_instanceClassList**()

Commands.sparkle_help.sparkle_generate_report_help.**get_numFeatureExtractors**()

Commands.sparkle_help.sparkle_generate_report_help.**get_numInstanceClasses**()

Commands.sparkle_help.sparkle_generate_report_help.**get_numInstanceInTestInstanceClass**(*test_ca str*)

Commands.sparkle_help.sparkle_generate_report_help.**get_numSolvers**()

Commands.sparkle_help.sparkle_generate_report_help.**get_performanceComputationCutoffTime**()

Commands.sparkle_help.sparkle_generate_report_help.**get_solverActualRankingList**()

Commands.sparkle_help.sparkle_generate_report_help.**get_solverList**()

Commands.sparkle_help.sparkle_generate_report_help.**get_solverPerfectRankingList**()

Commands.sparkle_help.sparkle_generate_report_help.**get_sparkle**()

Commands.sparkle_help.sparkle_generate_report_help.**get_testActualPAR10**(*test_case_directory: str*)

Commands.sparkle_help.sparkle_generate_report_help.**get_testInstanceClass**(*test_case_directory: str*)

Commands.sparkle_help.sparkle_global_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_global_help.**get_seed**()

Commands.sparkle_help.sparkle_instances_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_instances_help.**check_existence_of_reference_instance_list**(*inst*
str)
→
boo

Commands.sparkle_help.sparkle_instances_help.**copy_instances_to_smac**(*list_instance_path*, *in-*
stance_dir_prefix:
str,
smac_instance_dir_prefix:
str,
train_or_test:
str)

Commands.sparkle_help.sparkle_instances_help.**copy_reference_instance_list**(*target_file:*
path-
lib.Path,
in-
stance_set_name:
str,
path_modifier:
str)

Commands.sparkle_help.sparkle_instances_help.**count_instances_in_reference_list**(*instance_set_nan*
str)
→
int

Commands.sparkle_help.sparkle_instances_help.**get_list_all_path**(*instances_directory*)

Commands.sparkle_help.sparkle_instances_help.**get_list_cnf_path**(*instances_directory*)

Commands.sparkle_help.sparkle_instances_help.**get_list_test_cnf_index**(*list_cnf_path*,
list_train_cnf_index)

Commands.sparkle_help.sparkle_instances_help.**get_list_train_cnf_index**(*list_cnf_path*)

Commands.sparkle_help.sparkle_instances_help.**remove_reference_instance_list**(*instance_set_name:*
str)

Commands.sparkle_help.sparkle_instances_help.**selecting_test_cnf**(*list_cnf_path*,
list_test_cnf_index,
cnf_dir_prefix,
smac_cnf_dir_prefix)

Commands.sparkle_help.sparkle_instances_help.**selecting_train_cnf**(*list_cnf_path*, *list_train_cnf_index*, *cnf_dir_prefix*, *smac_cnf_dir_prefix*)

Commands.sparkle_help.sparkle_job_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_job_help.**check_active_jobs_exist**() → bool

Commands.sparkle_help.sparkle_job_help.**check_job_exists**(*job_id: str*, *command: sparkle_help.sparkle_command_help.CommandNa* → bool

Commands.sparkle_help.sparkle_job_help.**check_job_is_done**(*job_id: str*) → bool

Commands.sparkle_help.sparkle_job_help.**check_job_is_done_slurm**(*job_id: str*) → bool

Commands.sparkle_help.sparkle_job_help.**cleanup_active_jobs**() → int

Commands.sparkle_help.sparkle_job_help.**delete_active_job**(*job_id: str*)

Commands.sparkle_help.sparkle_job_help.**delete_active_jobs**(*job_ids: List[str]*)

Commands.sparkle_help.sparkle_job_help.**expand_total_job_from_list**(*list_jobs*)

Commands.sparkle_help.sparkle_job_help.**get_active_job_ids**() → List[str]

Commands.sparkle_help.sparkle_job_help.**get_job_ids_for_command**(*command: sparkle_help.sparkle_command_help.Co* → List[str]

Commands.sparkle_help.sparkle_job_help.**get_num_of_total_job_from_list**(*list_jobs*)

Commands.sparkle_help.sparkle_job_help.**read_active_jobs**() → List[Dict[str, str]]

Commands.sparkle_help.sparkle_job_help.**sleep**(*n_seconds: int*)

Commands.sparkle_help.sparkle_job_help.**wait_for_all_jobs**()

Commands.sparkle_help.sparkle_job_help.**wait_for_dependencies**(*command_to_run: sparkle_help.sparkle_command_help.Comm*

Commands.sparkle_help.sparkle_job_help.**wait_for_job**(*job_id: str*)

Commands.sparkle_help.sparkle_job_help.**write_active_job**(*job_id: str*, *command: sparkle_help.sparkle_command_help.CommandNa*

Commands.sparkle_help.sparkle_job_parallel_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_job_parallel_help.**generate_job_sbatch_shell_script**(*sbatch_shell_s job_script, dependency_jobid_l*

Commands.sparkle_help.sparkle_job_parallel_help.**get_dependency_list_str**(*dependency_jobid_list*)

Commands.sparkle_help.sparkle_job_parallel_help.**running_job_parallel**(*job_script, dependency_jobid_list, command_name: sparkle_help.sparkle_command*) → str

Commands.sparkle_help.sparkle_logging module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_logging.**add_output**(*output_path: str, description: str*)

Commands.sparkle_help.sparkle_logging.**log_command**(*argv*)

Commands.sparkle_help.sparkle_logging.**update_caller**(*argv*)

Commands.sparkle_help.sparkle_logging.**update_caller_file_path**(*timestamp: str*)

Commands.sparkle_help.sparkle_performance_data_csv_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

class Commands.sparkle_help.sparkle_performance_data_csv_help.**Sparkle_Performance_Data_CSV**

Bases: sparkle_help.sparkle_csv_help.Sparkle_CSV

calc_score_of_solver_on_instance(*solver: str, instance: str, num_instances: int, num_solvers: int, capvalue: float = None*) → float

calc_vbs_penalty_time()

calc_virtual_best_performance_of_portfolio(*num_instances: int, num_solvers: int, capvalue_list: List[float] = None*) → float

calc_virtual_best_score_of_portfolio_on_instance(*instance: str, num_instances: int, num_solvers: int, capvalue: float = None*) → float

get_dict_vbs_penalty_time_on_each_instance()

get_list_processed_performance_computation_job()

get_list_recompute_performance_computation_job()

get_list_remaining_performance_computation_job()

get_maximum_performance_per_instance() → List[float]

get_solver_penalty_time_ranking_list()

test()

Commands.sparkle_help.sparkle_record_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_record_help.**cleanup_current_sparkle_platform**()

Commands.sparkle_help.sparkle_record_help.**detect_current_sparkle_platform_exists**()

Commands.sparkle_help.sparkle_record_help.**extract_sparkle_record**(*my_record_filename*)

Commands.sparkle_help.sparkle_record_help.**save_current_sparkle_platform**(*my_record_filename*)

Commands.sparkle_help.sparkle_run_ablation_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_run_ablation_help.**check_for_ablation**(*solver_name*, *instance_train_name*, *instance_test_name*)

Commands.sparkle_help.sparkle_run_ablation_help.**create_configuration_file**(*solver_name*, *instance_train_name*, *instance_test_name*)

Commands.sparkle_help.sparkle_run_ablation_help.**create_instance_file**(*instances_directory*, *ablation_scenario_dir*, *train_or_test*)

Commands.sparkle_help.sparkle_run_ablation_help.**generate_callback_slurm_script**(*solver_name*, *instance_train_nam*
instance_test_name
dependency=None)

Commands.sparkle_help.sparkle_run_ablation_help.**generate_slurm_script**(*solver_name*, *instance_train_name*, *instance_test_name*, *dependency=None*)

Commands.sparkle_help.sparkle_run_ablation_help.**generate_validation_callback_slurm_script**(*s*
i
s
i
s
d
d

Commands.sparkle_help.sparkle_run_ablation_help.**generate_validation_slurm_script**(*solver_name*, *instance_train_*
instance_test_na
dependency=None)

Commands.sparkle_help.sparkle_run_ablation_help.**get_ablation_scenario_directory**(*solver_name*, *instance_train_na*
instance_test_nam
exec_path=Fals

Commands.sparkle_help.sparkle_run_ablation_help.**get_ablation_table**(*solver_name*, *instance_train_name*, *instance_test_name*)

Commands.sparkle_help.sparkle_run_ablation_help.**get_slurm_params**(*solver_name*, *instance_train_name*, *instance_test_name*, *postfix=''*, *dependency=None*)

Commands.sparkle_help.sparkle_run_ablation_help.**prepare_ablation_scenario**(*solver_name*, *instance_train_name*, *instance_test_name*)

Commands.sparkle_help.sparkle_run_ablation_help.**print_ablation_help**()

Commands.sparkle_help.sparkle_run_portfolio_selector_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**call_solver_solve_instance_within**

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**call_sparkle_portfolio_selector_s**

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**call_sparkle_portfolio_selector_s**

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**generate_running_sparkle_portfoli**

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**get_list_feature_vector**(*extractor_pa* *in-* *stance_path*, *re-* *sult_path*, *cut-* *off_time_eac*

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**get_list_predict_schedule_from_fi**

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**print_predict_schedule**(*predict_sched*

Commands.sparkle_help.sparkle_run_portfolio_selector_help.**print_solution**(*raw_result_path*)

Commands.sparkle_help.sparkle_run_solvers_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_run_solvers_help.**check_solver_output_for_errors**(*raw_result_path: pathlib.Path*)

Commands.sparkle_help.sparkle_run_solvers_help.**get_performance_data_id**() → int

Commands.sparkle_help.sparkle_run_solvers_help.**get_quality_from_wrapper**(*result_list: List[str]*) → List[float]

Commands.sparkle_help.sparkle_run_solvers_help.**get_runtime_from_runsolver**(*runsolver_values_path: str*) -> (<class 'float'>, <class 'float'>)

Commands.sparkle_help.sparkle_run_solvers_help.**get_runtime_from_wrapper**(*result: str*) → float

Commands.sparkle_help.sparkle_run_solvers_help.**get_solver_call_from_wrapper**(*solver_wrapper_path: str*, *instance_path: str*) → str

Commands.sparkle_help.sparkle_run_solvers_help.**get_status_from_wrapper**(*result: str*) → str

Commands.sparkle_help.sparkle_run_solvers_help.**handle_timeouts**(*runtime: float*, *status: str*, *custom_cutoff: int = None*) -> (<class 'float'>, <class 'str'>)

Commands.sparkle_help.sparkle_run_solvers_help.**print**(*value*, *...*, *sep=' '*, *end='n'*, *file=sys.stdout*, *flush=False*)

Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments: file: a file-like object (stream); defaults to the current sys.stdout. sep: string inserted between values, default a space. end: string appended after the last value, default a newline. flush: whether to forcibly flush the stream.

Commands.sparkle_help.sparkle_run_solvers_help.**process_results**(*raw_result_path: str, solver_wrapper_path: str, run-solver_values_path: str) -> (<class 'float'>, <class 'float'>, typ-ing.List[float], <class 'str'>)*

Commands.sparkle_help.sparkle_run_solvers_help.**remove_faulty_solver**(*solver_path, in-stance_path*)

Commands.sparkle_help.sparkle_run_solvers_help.**run_solver_on_instance**(*solver_path: str, solver_wrapper_path: str, in-stance_path: str, raw_result_path: str, run-solver_values_path: str, cus-tom_cutoff: int = None*)

Commands.sparkle_help.sparkle_run_solvers_help.**run_solver_on_instance_and_process_results**(

Commands.sparkle_help.sparkle_run_solvers_help.**running_solvers**(*performance_data_csv_path*, *mode*)

Commands.sparkle_help.sparkle_run_solvers_help.**sat_get_result_status**(*raw_result_path: str*) → str

Commands.sparkle_help.sparkle_run_solvers_help.**sat_get_verify_string**(*tmp_verify_result_path*)

Commands.sparkle_help.sparkle_run_solvers_help.**sat_judge_correctness_raw_result**(*instance_path*, *raw_result_path*

Commands.sparkle_help.sparkle_run_solvers_help.**sat_verify**(*instance_path: str*, *raw_result_path: str*, *solver_path: str*) → str

Commands.sparkle_help.sparkle_run_solvers_help.**sparkle_sat_parser**(*raw_result_path: str*, *runtime: float*) → str

Commands.sparkle_help.sparkle_run_solvers_help.**update_performance_data_id**()

Commands.sparkle_help.sparkle_run_solvers_help.**verify**(*instance_path*, *raw_result_path*, *solver_path*, *status*)

Commands.sparkle_help.sparkle_run_solvers_parallel_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_run_solvers_parallel_help.**generate_running_solvers_sbatch_she**

Commands.sparkle_help.sparkle_run_solvers_parallel_help.**running_solvers_parallel**(*performance_*
num_job_in_p
mode)

Commands.sparkle_help.sparkle_run_status_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_run_status_help.**get_list_running_extractor_jobs**()

Commands.sparkle_help.sparkle_run_status_help.**get_list_running_solver_jobs**()

Commands.sparkle_help.sparkle_run_status_help.**print_running_extractor_jobs**(*mode=1*)

Commands.sparkle_help.sparkle_run_status_help.**print_running_portfolio_selector_jobs**()

Commands.sparkle_help.sparkle_run_status_help.**print_running_report_jobs**()

Commands.sparkle_help.sparkle_run_status_help.**print_running_solver_jobs**(*mode=1*)

Commands.sparkle_help.sparkle_settings module

class Commands.sparkle_help.sparkle_settings.**PerformanceMeasure**

Bases: enum.Enum

An enumeration.

QUALITY = 1

QUALITY_ABSOLUTE = 1

RUNTIME = 0

from_str()

class Commands.sparkle_help.sparkle_settings.**SettingState**

Bases: enum.Enum

An enumeration.

CMD_LINE = 3

DEFAULT = 1

FILE = 2

NOT_SET = 0

class Commands.sparkle_help.sparkle_settings.**Settings**(*file_path: pathlib.PurePath = None*)

Bases: object

DEFAULT_ablation_racing = False

DEFAULT_config_budget_per_run = 600

DEFAULT_config_number_of_runs = 25

DEFAULT_general_extractor_cutoff_time = 60

DEFAULT_general_penalty_multiplier = 10

DEFAULT_general_performance_measure = 0

DEFAULT_general_solution_verifier = 0

DEFAULT_general_target_cutoff_time = 60

DEFAULT_settings_path = PurePosixPath('Settings/sparkle_settings.ini')

DEFAULT_slurm_clis_per_node = 32

DEFAULT_slurm_number_of_runs_in_parallel = 25

DEFAULT_smac_target_cutoff_length = 'max'

add_slurm_extra_option(*name: str*, *value: str*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

get_ablation_racing_flag() → bool

get_config_budget_per_run() → int

get_config_number_of_runs() → int

get_general_extractor_cutoff_time() → int

get_general_penalty_multiplier() → int

get_general_performance_measure() → Commands.sparkle_help.sparkle_settings.PerformanceMeasure

get_general_solution_verifier() → Commands.sparkle_help.sparkle_settings.SolutionVerifier

get_general_target_cutoff_time() → int

get_penalised_time(*custom_cutoff: int = None*) → int

get_slurm_clis_per_node() → int

get_slurm_extra_options() → dict

get_slurm_number_of_runs_in_parallel() → int

get_smac_target_cutoff_length() → str

read_settings_ini(*file_path: pathlib.PurePath = PurePosixPath('Settings/sparkle_settings.ini')*, *state: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.FILE: 2>*)

set_ablation_racing_flag(*value: bool = False*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_config_budget_per_run(*value: int = 600*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_config_number_of_runs(*value: int = 25*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_general_extractor_cutoff_time(*value: int = 60*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_general_penalty_multiplier(*value: int = 10*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_general_performance_measure(*value: Commands.sparkle_help.sparkle_settings.PerformanceMeasure = <PerformanceMeasure.RUNTIME: 0>*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_general_solution_verifier(*value: Commands.sparkle_help.sparkle_settings.SolutionVerifier = <SolutionVerifier.NONE: 0>*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_general_target_cutoff_time(*value: int = 60*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_slurm_clis_per_node(*value: int = 32*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_slurm_number_of_runs_in_parallel(*value: int = 25*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

set_smac_target_cutoff_length(*value: str = 'max'*, *origin: Commands.sparkle_help.sparkle_settings.SettingState = <SettingState.DEFAULT: 1>*)

write_settings_ini(*file_path: pathlib.Path*)

write_used_settings()

class Commands.sparkle_help.sparkle_settings.**SolutionVerifier**

Bases: enum.Enum

An enumeration.

NONE = 0

SAT = 1

from_str()

Commands.sparkle_help.sparkle_slurm_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_slurm_help.**generate_sbatch_script_for_feature_computation**(*n_j fea- ture list_*

Commands.sparkle_help.sparkle_slurm_help.**generate_sbatch_script_for_validation**(*solver_name: str, instance_set_train_ str, instance_set_test_n str = None*) → str

Commands.sparkle_help.sparkle_slurm_help.**generate_sbatch_script_generic**(*sbatch_script_path*, *sbatch_options_list*, *job_params_list*, *srun_options_str*, *target_call_str*, *job_output_list=None*)

Commands.sparkle_help.sparkle_slurm_help.**get_slurm_options_list**(*path_modifier=None*)

Commands.sparkle_help.sparkle_slurm_help.**get_slurm_sbatch_default_options_list**()

Commands.sparkle_help.sparkle_slurm_help.**get_slurm_sbatch_user_options_list**(*path_modifier=None*)

Commands.sparkle_help.sparkle_slurm_help.**get_slurm_srun_user_options_list**(*path_modifier=None*)

Commands.sparkle_help.sparkle_slurm_help.**get_slurm_srun_user_options_str**(*path_modifier=None*)

Commands.sparkle_help.sparkle_slurm_help.**submit_sbatch_script**(*sbatch_script_name: str*, *command_name: sparkle_help.sparkle_command_help.Com*, *execution_dir: str = None*) → str

Commands.sparkle_help.sparkle_system_status_help module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.sparkle_help.sparkle_system_status_help.**get_file_modify_time**(*file_path*)

Commands.sparkle_help.sparkle_system_status_help.**print_extractor_list**(*mode=1*)

Commands.sparkle_help.sparkle_system_status_help.**print_instance_list**(*mode=1*)

Commands.sparkle_help.sparkle_system_status_help.**print_list_remaining_feature_computation_**

Commands.sparkle_help.sparkle_system_status_help.**print_list_remaining_performance_computati**

Commands.sparkle_help.sparkle_system_status_help.**print_portfolio_selector_info**()

Commands.sparkle_help.sparkle_system_status_help.**print_report_info**()

Commands.sparkle_help.sparkle_system_status_help.**print_solver_list**(*mode=1*)

Commands.sparkle_help.sparkle_system_status_help.**timestamp_to_time**(*timestamp*)

Module contents

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.2 Submodules

5.1.3 Commands.about module

5.1.4 Commands.add_feature_extractor module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.5 Commands.add_instances module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.6 Commands.add_solver module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.7 Commands.cleanup_current_sparkle_platform module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.8 Commands.cleanup_temporary_files module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.cleanup_temporary_files.**create_temporary_directories**()

Commands.cleanup_temporary_files.**remove_temporary_files**()

5.1.9 Commands.compute_features module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.compute_features.**compute_features_parallel**(*my_flag_recompute*)

5.1.10 Commands.compute_features_parallel module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.11 Commands.compute_marginal_contribution module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.compute_marginal_contribution.**compute_actual**(*flag_recompute: bool = False*) → List[Tuple[str, float]]

Commands.compute_marginal_contribution.**compute_marginal_contribution**(*flag_compute_perfect: bool*, *flag_compute_actual: bool*, *flag_recompute: bool*)

Commands.compute_marginal_contribution.**compute_perfect**(*flag_recompute: bool = False*) → List[Tuple[str, float]]

5.1.12 Commands.configure_solver module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.13 Commands.construct_sparkle_portfolio_selector module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.construct_sparkle_portfolio_selector.**delete_log_files**()

Commands.construct_sparkle_portfolio_selector.**delete_task_run_status**()

Commands.construct_sparkle_portfolio_selector.**generate_task_run_status**()

Commands.construct_sparkle_portfolio_selector.**judge_exist_remaining_jobs**(*feature_data_csv_path*, *performance_data_csv_path*)

Commands.construct_sparkle_portfolio_selector.**print_log_paths**()

5.1.14 Commands.generate_report module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

`Commands.generate_report.`**`delete_task_run_status`**`()`

`Commands.generate_report.`**`generate_task_run_status`**`()`

5.1.15 Commands.initialise module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.16 Commands.load_record module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.17 Commands.remove_feature_extractor module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.18 Commands.remove_instances module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.19 Commands.remove_record module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.20 Commands.remove_solver module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.21 Commands.run_ablation module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com

Holger H. Hoos, hh@liacs.nl

Jeroen Rook, j.g.rook@umail.leidenuniv.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.22 Commands.run_solvers module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

Commands.run_solvers.**construct_selector_and_report**(*dependency_jobid_list: List[str] = []*)

Commands.run_solvers.**run_solvers_parallel**(*flag_recompute*, *flag_also_construct_selector_and_report=False*)

5.1.23 Commands.run_sparkle_portfolio_selector module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.24 Commands.run_status module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.25 Commands.save_record module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.26 Commands.sparkle_wait module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.27 Commands.system_status module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.28 Commands.validate_configured_vs_default module

Software: Sparkle (Platform for evaluating empirical algorithms/solvers)

Authors: Chuan Luo, chuanluosaber@gmail.com Holger H. Hoos, hh@liacs.nl

Contact: Chuan Luo, chuanluosaber@gmail.com

5.1.29 Module contents

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

D

E

F

G

H

I

J

L

N

P

Q

R

S

T

U

V

W