Sparkle user guide

Koen van der Blom

January 7, 2021

1 Quick start

Follow these steps:

1.1 Install Sparkle

1.2 Prepare your configuration environment

1.3 Execute commands

1.1 Installing Sparkle

1. Copy the Sparkle files to your desired directory
2. Install Python 3.5 (other 3.x versions may work, but were not tested with the packages included in the `requirements.txt` file.
3. Install Swig 3.0 – With Anaconda: `conda install swig=3.0`
4. Install `requirements.txt`: `pip install -r requirements.txt` – With Anaconda: `/home/<username>/<anaconda_dir>/envs/<env_name>/bin/pip install -r requirements.txt`
5. Install `epstopdf` as described in Section 5.1.1

1.2 Configuration

Configuring an algorithm has the following minimal requirements for the algorithm (for an example of a solver directory see Section 2.3):

1.2.1 Make the solver executable work on Grace

1.2.2 An algorithm wrapper called `sprakle_smac_wrapper.py`

1.2.3 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 a solver directory see Section 2.3). 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.

1.2.1 Making your algorithm run on Grace

Add helpful tips when available

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.

1.2.2 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.

1.2.3 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.

1.3 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:

```
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/configure_solver_pbo-ccsat.sh` for an example of a sequence of commands to execute. Note that these example scripts should not be executed directly.

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

2 File structure

2.1 General

Before doing anything, the following subdirectories are present in the Sparkle directory:

```
Commands/
Components/
Documentation/
Examples/
Settings/
Test_Cases/
```

Based on various Sparkle commands, the following additional subdirectories may be generated:

```
    Configuration_Reports/
    Extractors/
    Feature_Data/
2.2 Instances/
    LOG/
    Performance_Data/
    Records/
    Reference_Lists/
2.3 Solvers/
    Sparkle_Portfolio_Selector/
    TMP/
```

2.2 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.

2.3 A typical solver directory

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.

3 Commands

Add missing subsections with details of other commands

Currently the following commands are available in Sparkle (listed alphabetically):

```
    add_feature_extractor.py
    add_instances.py
3.1 add_solver.py
    cleanup_current_sparkle_platform.py
    cleanup_temporary_files.py
    compute_features_parallel.py
    compute_features.py
    compute_marginal_contribution.py
3.2 configure_solver.py
    construct_sparkle_portfolio_selector.py
3.3 generate_report_for_configuration.py
    generate_report_for_test.py
    generate_report.py
3.4 initialise.py
    load_record.py
    remove_feature_extractor.py
    remove_instances.py
    remove_record.py
    remove_solver.py
    run_ablation.py
    run_solvers_parallel.py
    run_solvers.py
```

```
    run_sparkle_portfolio_selector.py
    run_status.py
    save_record.py
    system_status.py
3.6 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.

3.1 add_solver.py

Add a solver to the Sparkle platform.

Arguments:

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

3.2 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.

3.3 generate_report_for_configuration.py

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. When only a solver is given, Sparkle generates a report for the most recent configuration experiment with that solver.

3.4 initialise.py

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

3.5 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.

3.6 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>]
```

4 Settings

4.1 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 4.1.2).

4.1.1 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
```

4.1.2 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
```

4.1.3 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/Yahsp3 --instances-train Instances
   /Depots_train_few/
```

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.

5 Required packages

5.1 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.

5.1.1 `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] `http://ada.liacs.nl/`

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
   ```

5.1.2 General requirements (to be revised)

Check which of these old installation instructions actually apply (if any)

Currently Sparkle uses Python 2.7.

```
# Before starting Sparkle, please install the following packages with the specific versio
# pip3 install ConfigSpace==0.3.9 --user
# pip3 install smac==0.6.0 --user
# pip3 install install git+https://github.com/mlindauer/ASlibScenario --user
# pip3 install sphinx-bootstrap-theme==0.6.0 --user

# Also, please install the following software:
# pdflatex, latex, bibtex, swig, gnuplot, gnuplot-x11
```

5.2 Yahsp example

Describe how to make the Yahsp example work