JMLR: Workshop and Conference Proceedings 1:1–3, 2017

SUNNY with Algorithm Configuration

Tong Liu T.LIU@UNIBO.IT
University of Bologna, Italy

Roberto Amadini ROBERTO.AMADINI@UNIMELB.EDU.AU
University of Melbourne, Australia

Jacopo Mauro MAURO.JACOPO@GMAIL.COM
University of Oslo, Norway

Abstract

The SUNNY algorithm is a portfolio technique originally tailored for Constraint Satisfaction Problems (CSPs). SUNNY allows to select a set of solvers to be run to solve a given CSPs and was proven effective in the context of the MiniZinc Challenge, i.e., the yearly international competition for CSP solvers. In 2015, SUNNY was compared with other solver selectors in the first ICON Challenge on algorithm selection with less satisfactory performance. In this paper we briefly describe the new version of the SUNNY approach for algorithm selection, that was submitted to the first Open Algorithm Selection Challenge.

1. SUNNY-OASC

SUNNY is a per instance algorithm scheduling strategy based on k-NN algorithm. Roughly speaking, for each test instance SUNNY selects k training instances which are similar to the test instance in terms of Euclidean Distance (on instance features). Based on the selected instances, SUNNY generates a schedule of solvers that maximize the number of instances solved by the selected solvers. Then, a time slot proportional to the fraction of solved instances is assigned to each solver. Finally, the proposed solvers are ordered according to the average solving time on the selected instances.

For a detailed description of the SUNNY approach we refer the interested reader to Amadini et al. (2014, 2015b, 2016). In the following we present SUNNY-OASC, an extension of the original SUNNY algorithm that enables the configuration of the neighborhood size k (an idea borrowed from Lindauer et al. (2016)) and a wrapper-based feature selection.

1.1. Execution modalities

SUNNY-OASC has two execution modalities: `autok` and `fkvar`. **WHAT IS T-SUNNY???**

- The `autok` is a variant of T-SUNNY as defined by Lindauer et al. (2016) where SUNNY-AS has been improved by training also on the size of the neighborhood k.[1]
- The `fkvar` trains both on the neighborhood size and the subset of features to consider by using a wrapper method (Kohavi and John, 1997). SUNNY is used as the evaluator

1. `autok` is slightly different than T-SUNNY since the reimplementation of T-SUNNY used a different algorithm to select the solvers to use. To choose the solvers we used the original SUNNY-AS algorithm.

and a greedy forward selection is adopted to select the subset of features for computing the neighborhood. The selection procedure is defined as follow: the unselected feature set is considered and we pick one feature at time, adding it to the selected feature set (initially empty) to form a test feature set. By also tuning the value k, SUNNY calculates the best PAR10 score that it can achieve with the test feature set. Based on the outcome, a new feature is added until the performance decrease or we have performed a given number of evaluations. Finally, `fkvar` produces a combination of features and a value k for which SUNNY performs the best on training data.

When the selection procedure ends, we also run SUNNY in `autok` modality. This is helpful for scenarios where the whole set of features is more relevant than the feature set selected by using the wrapper filtering method. In these cases, we simply use the setting computed by `autok`.

1.2. Representative instances

Since training is computationally expensive and may take a long time,[2] SUNNY-OASC is not used on all the instances available but only on some selected ones. The representative instances used for the training are selected as follows: (i) SUNNY-OASC first associates to each training instance the fastest solver for solving it, according to the training set; (ii) for each solver, instances are ordered from hard to easy in terms of runtime; (iii) for each solver, one instance at a time is picked until a global limit on the number of representative instances is reached.

1.3. Parameters for the Challenge

Bischl et al. (2016) and Amadini et al. (2015a) noted that a handful subset of features (e.g., 5 or less) is often enough for SUNNY to obtain competitive performance. For this reason, in `fkvar` we fixed 5 as the number of feature to select. In order to guarantee an acceptable execution runtime, for the `fkvar` approach we consider only 1500 instances to be included in the representative instance set. We also fixed k to vary between 3 and 30.

When `fkvar` is executed, we also run `autok` with $k \in [3, 80]$ as a backup. If SUNNY runs better with the entire feature set, we then use the result produced by `autok`.

For the `autok` version submitted, different to the one used when running SUNNY-OASC in the `fkvar` modality, we consider the full training set as effective training data.

2. Setup Instructions

The source code of SUNNY-OACS is available at Liu (2017) and requires Python v2. There are five folders: 'data' and 'results' contain oasc-challenge data and solution results respectively, 'src' contains the original SUNNY-AS scripts from Amadini and Mauro (2015), 'oasc' contains scripts that coordinate those in 'src' for training and testing, the folder 'main' contains the scripts that automatically call 'oasc' for the different execution modalities.

The program runs training and testing in sequence. Let us take `autok` approach as execution example. To run it, in the folder 'main' the command `sh make_oasc_tasks.sh` >

2. On a PC with Intel Core i5 and 8 GB RAM running Ubuntu, training only one fold out of 10 of the ASLib scenario PROTUS (4,000 instances) would take for instance 35 hours using the `fkvar` approach.

`tasks.txt` must be used to create the tasks. Then the training can be done by running `sh oasc_train.sh run_autok tasks.txt`. The test is performed by running `sh make_oasc_tasks.sh > tasks.txt` followed by `sh oasc_test.sh autok tasks.txt`.

To run `fkvar` it is enough to replace `autok` by `fkvar` in the above commands.

References

Roberto Amadini and Jacopo Mauro. SUNNY-AS, 2015. Available at https://github.com/CP-Unibo/sunny-as.

Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. SUNNY: a Lazy Portfolio Approach for Constraint Solving. *TPLP*, 14(4-5):509–524, 2014.

Roberto Amadini, Fabio Biselli, Maurizio Gabbrielli, Tong Liu, and Jacopo Mauro. Feature selection for SUNNY: A study on the algorithm selection library. In *ICTAI*, pages 25–32. IEEE Computer Society, 2015a.

Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. A Multicore Tool for Constraint Solving. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 232–238, 2015b.

Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. Portfolio approaches for constraint optimization problems. *Annals of Mathematics and Artificial Intelligence*, 76(1-2): 229–246, 2016.

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *ARTIFICIAL INTELLIGENCE*, 97(1):273–324, 1997.

Marius Lindauer, Rolf-David Bergdoll, and Frank Hutter. An Empirical Study of Per-instance Algorithm Scheduling. In *LION*, volume 10079 of *LNCS*, pages 253–259. Springer, 2016.

Tong Liu. SUNNY-OASC, 2017. Available at https://github.com/lteu/oasc.