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 kNN portfolio technique originally tailored for CSP problems. It achieved excellent results consecutively in the Minizin Challenge in the last three years. In 2015, SUNNY for the first time forked a branch for ASLIB [1] problems, however, due to the lack of insufficient training process, it performed poorly in the ICON-Challenge 2015 [2]. After several efforts of investigation on algorithm configuration and on ASLIB problem features, we proposed an integrated training process for SUNNY and bring SUNNY into the OASC challenge. The final results were encouraging and competitive.

1. Description

1.1. SUNNY

SUNNY is an per instance algorithm scheduling strategy based on kNN . Briefly 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 cover the k instances as much as possible. Then, a time slot, which is propotional to the fraction of solved instances, is assigned to each solver. Finally, the proposed solvers get ordered according to the number of k instances solved. In the tool SUNNY-AS (see Amadini et al. (2015); Amadini and Mauro (2015)), we selected a subset of solvers, which solve the k instances as least time as possible.

SUNNY-OASC is an extension of the original SUNNY algorithm with the idea of configuring neighbourhood size k (borrowed from the work of Lindauer et al. (2016)) and wrapper-based feature selection.

1.2. Execution modalities

SUNNY-OASC has two execution modalities: `autok` and `fkvar`.

1. See Bischl et al. (2016)
2. See Kotthoff (2015)

- 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.[3]

- The `fkvar` instead trains for the neighborhood value k and the subset of features to consider by using a wrapper method Kohavi and John (1997). SUNNY is used as the evaluator and a greedy forward selection is adopted to select the subset of features for computing the neighborhood. The selection cycle is defined as follow: the unselected feature set is considered and we pick one feature at time, adding it to the selected features 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. In the end, `fkvar` produces a combination of features and a value k for which SUNNY performs the best on training data.

  When the selection cycle ends, we run a backup with the `autok` modality. This is helpful for those scenarios where all features are more relevant than the selected features for SUNNY performance.

1.3. Representative instances

For training performance reasons [4], 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 follow: i) SUNNY-OASC first associates each training instance to a solver that solves it in the least time, according to the ground data, ii) for each solver instances are ordered from hard to easy in terms of runtime, iii) for each solver one instance at the time is picked until a global limit on the number of representative instances is reached.

1.4. Parameters for the Challenge

In literature Bischl et al. (2016); Amadini et al. (2015), it is suggested 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 have chosen to consider only 1500 and not more instances to be included in the representative instance set. We also fixed the interval of k as [3,30].

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 as a backup, when running SUNNY-OASC in the `fkvar` modality, we consider the full training set as effective training data (i.e., more than 1500 instances are used to train if available).

3. `autok` is slightly different than T-SUNNY since the reimplementation of T-SUNNY used a different algorithm to select the solvers to use. To chose the solvers we used the original SUNNY-AS algorithm.
4. e.g. on a PC with Intel Core i5(3.30 GHz) and 8 GB RAM, running Ubuntu, training one fold (10 folds in total) of scenario PROTUS($4,000$ instances) would take 35 hours in fkvar approach

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 > 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". After training, the test is run by "sh make_oasc_tasks.sh > tasks.txt" and later by "sh oasc_test.sh autok tasks.txt".

To run `fkvar` it is sufficient to replace `autok` by `fkvar` in the previous commands.

References

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

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, 2015.

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.

Lars Kotthoff. Icon challenge on algorithm selection. *arXiv preprint arXiv:1511.04326*, 2015.

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.