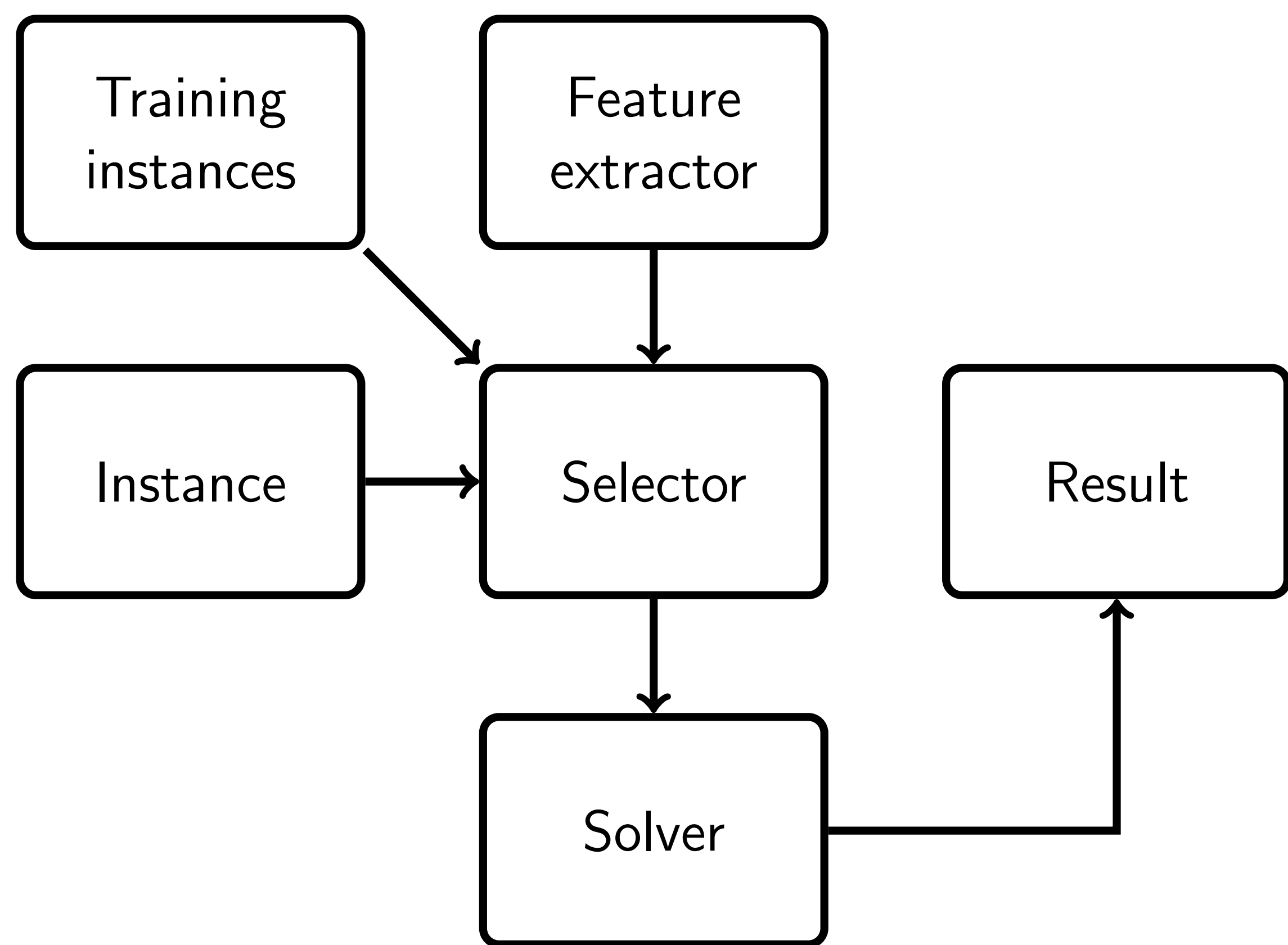


SPARKLE: TOWARDS AUTOMATED ALGORITHM CONFIGURATION FOR EVERYONE

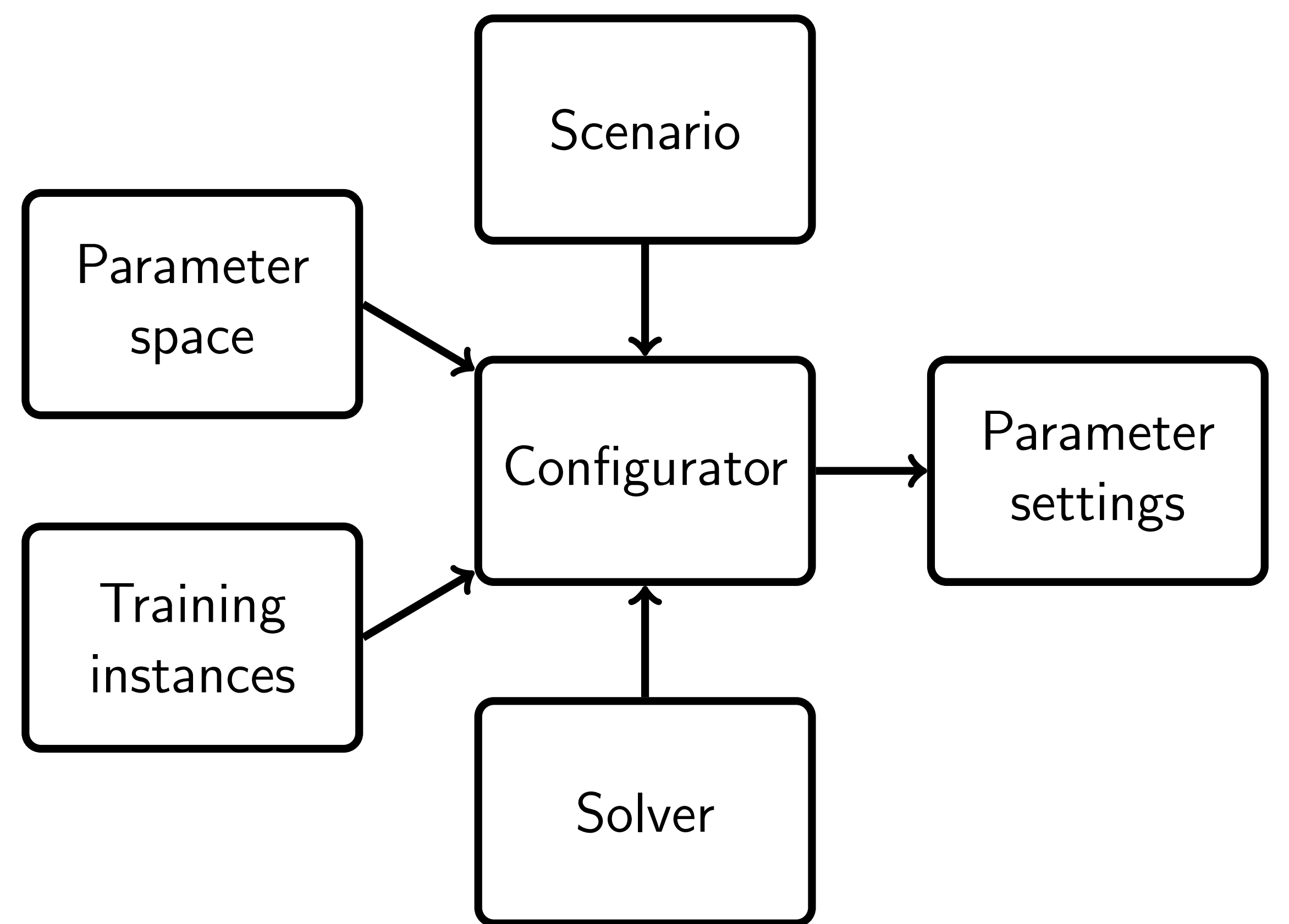
KOEN VAN DER BLOM¹, CHUAN LUO², AND HOLGER H. HOOS^{1,3}

¹LEIDEN UNIVERSITY, ²MICROSOFT RESEARCH ASIA, ³UNIVERSITY OF BRITISH COLUMBIA

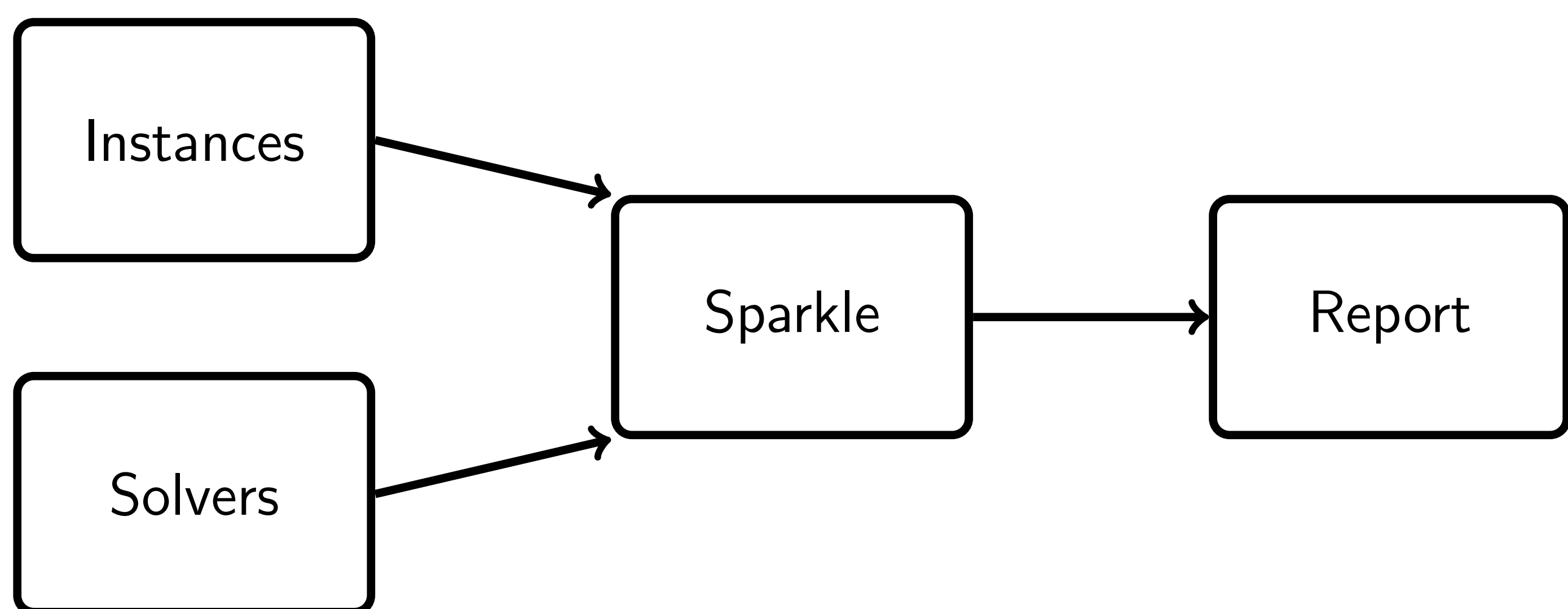
SELECTION



CONFIGURATION



SPARKLE



- Make meta-algorithmics more accessible
- Operate with simple commands
- Integrated best practices and avoidance of pitfalls

SIMPLE COMMANDS

- 1: Commands/initialise.py
- 2: Commands/add_instances.py -run-solver-later -run-extractor-later ../PTN/
- 3: Commands/add_solver.py -run-solver-later -deterministic 0 ../Pb0-CSCCSAT-Generic/
- 4: Commands/add_feature_extractor.py -run-extractor-later ../SAT-features-competition2012/
- 5: Commands/compute_features.py
- 6: Commands/configure_solver.py -solver Solvers/Pb0-CSCCSAT-Generic -instance_set Instances/PTN/
- 7: Commands/generate_report_for_configuration.py

REPORTING

The System Report for Sparkle

Automatically generated by Sparkle (version: Sparkle_SAT_Challenge_2018) 1st April 2019

1 Introduction

Sparkle [2] is a multi-agent problem-solving platform based on Programming by Optimisation (PbO) [3], and would provide a number of effective algorithm optimisation techniques (such as automated algorithm configuration, portfolio-based algorithm selection, etc) to accelerate the solving process. This experimental report is automatically generated by Sparkle. This report presents the system setup for Sparkle.

2 Experimental Preliminaries

In this section, we present the experimental preliminaries, including the list of solvers, the list of feature extractors, the list of instance classes, the information about experimental setup and the information about how to construct a portfolio-based algorithm selector in Sparkle.

2.1 Solvers

There are 2 solvers submitted in Sparkle, and the list of solvers is given as follows:

1. CSCCSat_wrapper_sparkle
2. Lingling_wrapper_sparkle
3. MiniSAT_wrapper_sparkle

2.2 Feature Extractors

There are 1 feature extractor(s) submitted in Sparkle, and the list of feature extractor(s) is given as follows:

1. SAT-features-competition2012_revised_without_SatELite_sparkle

2.3 Instance Classes

There are 10 instance class(es) submitted in Sparkle. All instance class(es) are classified into 1 instance class(es), and the list of instance class(es) is given as follows:

1. Sparkle_test_instances, number of instance(s): 10

2.4 Experimental Setup

Feature computation: We use all the feature extractors which are presented above to compute the feature vector for each instance. Each feature extractor will compute a feature vector for each instance. The final feature vector is the concatenation of all computed feature vectors. The cutoff time for feature vector computation on each instance is set to 30 seconds. The memory limit for feature vector computation on each instance is set to 256M MB.

Performance computation: Each solver will run one time on each instance. The cutoff time for each performance computation run is set to 10 seconds. The memory limit for each performance computation run is set to 4192 MB.

2.5 Constructing Portfolio-Based Algorithm Selector

Since the primary goal of Sparkle is to analyse the contribution of each solver to the real state of the set, Sparkle utilizes the concept of marginal contribution [4] to measure each solver's contribution to the perfect portfolio selector, also known as Virtual Best Solver (VBS), and the actual portfolio selector.

Now we give the protocol for calculating each solver's marginal contribution. Assume that we have a set of solvers S and a portfolio selector F constructed based on a subset of S . In this report, the performance of a solver or portfolio selector is measured by the final produced average runtime (PAR10), which conforms with the measurement used in SAT Competitions. Let $par(F)$ denote the PAR10 value achieved by averaging the complementary strategies of the algorithm in F . The absolute marginal contribution (Abs_Marg_Contr) for solver s is calculated as

$$abs(s) = \begin{cases} \log_2 \frac{par(S)}{par(F)} & \text{if } s \in F \\ 0 & \text{else} \end{cases} \quad (1)$$

Then, for each solver s , we calculate the relative marginal contribution (Rel_Marg_Contr), which is proportional to the absolute marginal contribution, with normalization. The relative marginal contribution for solver s is calculated as

$$rnc(s) = \frac{abs(s)}{\sum_{s \in S} abs(s)} \quad (2)$$

Finally, the contribution of each solver is evaluated by the relative marginal contribution.

3 Experimental Results

In this section, the related experimental results in Sparkle are presented and analyzed.

• Pb0-CSCSAT-Generic (configured), PAR10: 4.50000

• Pb0-CSCSAT-Generic (default), PAR10: 397.61601

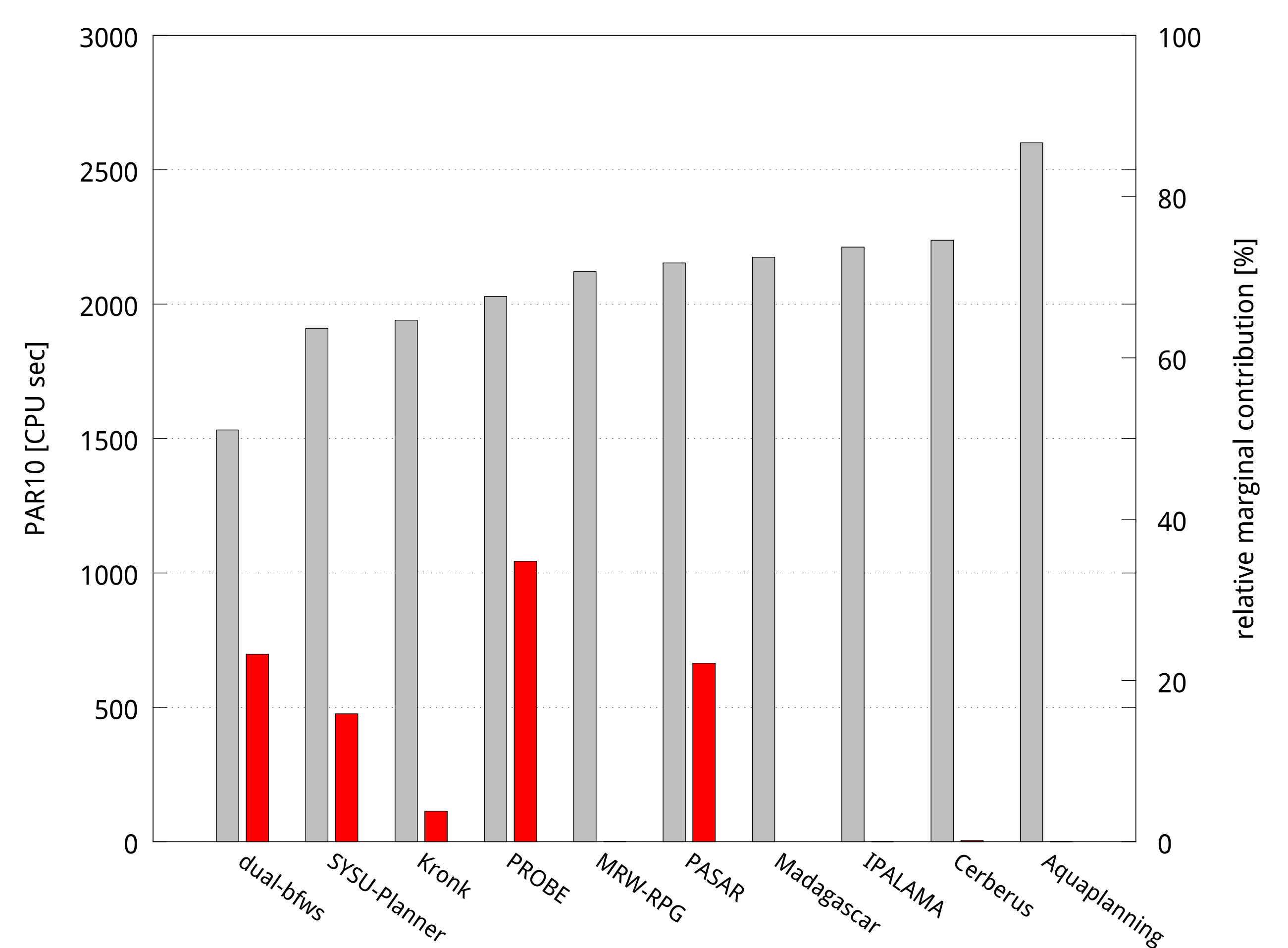
The empirical comparison between the Pb0-CSCSAT-Generic (configured) and Pb0-CSCSAT-Generic (default) on the training set of PTN is presented in Figure 2.

References

- [1] Holger H. Hoos, Programming by optimization, *Communications of the ACM*, 55(2):79–80, 2012.
- [2] Holger H. Hoos, Sparkle: A pbO-based multi-agent problem-solving platform, Technical report, Department of Computer Science, University of British Columbia, 2015.
- [3] Frank Hees, Holger H. Hoos, and Kevin Leyton-Brown, Sequential model-based optimization for general algorithm configuration, In *Proceedings of the 30th International Conference on Learning and Intelligent Optimization (LION 5)*, pages 407–424, 2011.

- Contribution per solver
- Cite selector
- Cite configurator
- Process description

COOPERATIVE COMPETITION



- Marginal contribution [Xu et al 2012]: How valuable is this solver to the selector

REFERENCES

- [1] Shapley, L. S. (1953). A value for n-person games. In *Contributions to the Theory of Games, volume II*, pages 307–317. Princeton University Press.
- [2] Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2012). Evaluating component solver contributions to portfolio-based algorithm selectors. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT 2012)*, LNCS 7317, pages 228–241.

FUTURE

- Further simplifications, such as inferring the parameter space
- Shapley value [Shapley 1953]?
- Your ideas?