Automated Algorithm Configuration and Selection with Sparkle

Koen van der Blom & Jeroen Rook

PPSN 2022

10 September, 2022

Universiteit Leiden
The Netherlands

UNIVERSITY OF TWENTE.

About us – Koen

Career

- PhD Leiden U.
- Postdoc Leiden U.
- Postdoc Sorbonne U.

Topics

- Multi-objective optimisation
- Meta-algorithms
- Software engineering for ML

About us – Jeroen

Career

- MSc Leiden U.
- PhD U. of Twente

Topics

- Multi-objective optimisation
- Multi-objective meta-algorithms

Format

- Please interrupt when you have questions!
- Don't hesitate to talk to us later in the conference!

- Slides available online:

  `https://liacs.leidenuniv.nl/~blomkvander/slides/ppsn_2022.pdf`

Outline

- 15:20-15:25 – Intro
- 15:25-15:40 – Algorithm configuration
- 15:40-15:55 – Algorithm selection
- 15:55-16:05 – 10 minute break
- 16:05-16:20 – The Sparkle platform
- 16:20-16:30 – Demo
- 16:30-16:45 – Discussion

  Times are estimates ;)

Meta-algorithms

Maximise the performance of the available algorithms

Algorithm Configuration
Many algorithms have (hyper)parameters and are made up of components that influence their performance

Algorithm Selection
There is not a single algorithm that solves all problems equally well. [Rice,'76]

(Automated) Algorithm Configuration

Algorithm Configuration

Sometimes also referred to as HPO or meta-optimization. We prefer AC.

- Algorithm A
- Many performance influencing parameters and components
  - E.g., selection strategy, tournament size, ...
- Which one to choose to get best performance?
  - E.g., solution quality, solving time, convergence rate, ...

Algorithm Configuration

Sometimes also referred to as HPO or meta-optimization. We prefer AC.

- Algorithm A
- Many performance influencing parameters and components
  - E.g., selection strategy, tournament size, ...
- Which one to choose to get best performance?
  - E.g., solution quality, solving time, convergence rate, ...

- Grid search
- Manual experimentation

Algorithm Configuration - Issues

- Search space can be very/infinitely large
- Bias/intuition for *good* configurations limits the potential
- Many parameter configurations are unproductive
- Has each algorithm received equal attention?

Example:

SOTA Algorithm A	**Your Algorithm B**
default parameters	months of expimentation

Automated Algorithm Configuration

Formulated as optimisation problem:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, Q(A_\theta, \mathcal{I})$$

- Θ Parameter space
- Q Performance measure
- A Algorithm
- $\mathcal{I}$ Set of problem instances: $\{I_1, \ldots, I_{10}\}$

Automated Algorithm Configuration

Formulated as optimisation problem:

$$\theta^* = \operatorname*{argmax}_{\theta \in \Theta} Q(A_\theta, \mathcal{I})$$

- Θ Parameter space
- Q Performance measure
- A Algorithm
- $\mathcal{I}$ Set of problem instances: $\{I_1, \dots, I_{10}\}$

Expensive functions calls $\rightarrow$ limited budget
Mixed-type and nested parameters

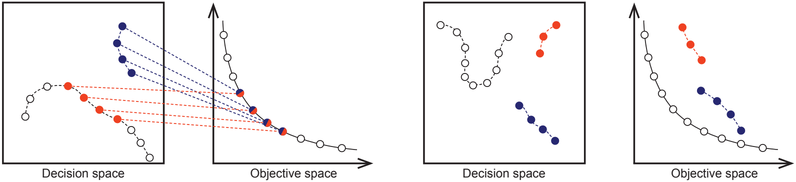
Automated Algorithm Configuration

- Systematically searches over the complete parameter space
- Tries out *unconventional* parameter combinations
- (Usually) gets better performing algorithms [Fawcett et al.,'11], [KhudaBukhsh et al.,'16], [Rook et al.,'22]
- Runs while you do other things
- Example AAC methods:
  - SMAC [Hutter et al.,'11]
  - irace [López-Ibáñez et al.,'16]
  - GGA++ [Ansótegui et al.,'15]
  - GPS [Pushak & Hoos,'20]
  - ...
  - Application of AAC at PPSN '22 [Trajanov et al.,'22]

Example of study with AAC

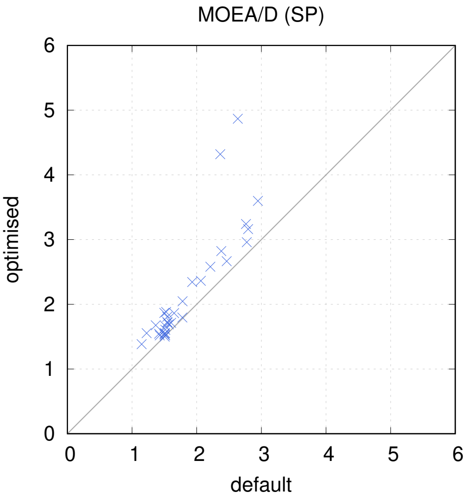
Benchmarking study of MO algorithms [Rook et al.,'22]

- 33 multi-modal multi-objective optimization problems
- 7 evolutionary MO algorithms
- 2 indicators:
  - Diversity in decision space (Solow-Polasky (SP))
  - Quality in objective space (Hypervolume (HV))



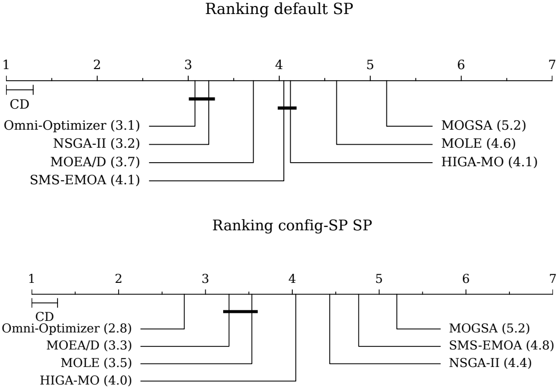
- How good are the algorithms in solving these instance types?

Example of study with AAC



Example of study with AAC

- Equal opportunity to get the best performance in each scenario
- Reproducible configuration search



How to integrate AAC in your research

- Allow for a fair comparison between different methods
- Validate if a newly designed component is truly favoured over existing components.
- Analyse algorithm parameter values under different circumstances

(Automated) Algorithm Selection

Why algorithm selection?

Choose the best algorithm for the problem (instance)

- Better performance than randomly chosen algorithm
- Which algorithm is the best?

Why algorithm selection?

Choose the best algorithm for the problem (instance)

- Better performance than randomly chosen algorithm
- Which algorithm is the best?

Configure to get the best algorithm?

- Configure multiple algorithms
- Which (configured) algorithm is the best?

Example problem: Boolean satisfiability (SAT)

Is there a satisfying assignment for a Boolean formula?

- $a \wedge \neg b$
- Satisfied when $a = \mathit{true}$ and $b = \mathit{false}$

Example problem: Boolean satisfiability (SAT)

Is there a satisfying assignment for a Boolean formula?

- $a \wedge \neg b$
- Satisfied when $a = \mathit{true}$ and $b = \mathit{false}$
- NP-complete problem
- Applications
  - Hard-/software verification
  - Scheduling
  - Model checking

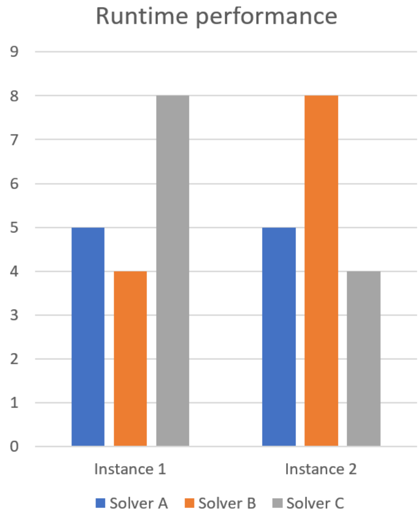
SAT solvers

Algorithm $\rightarrow$ (SAT) solver

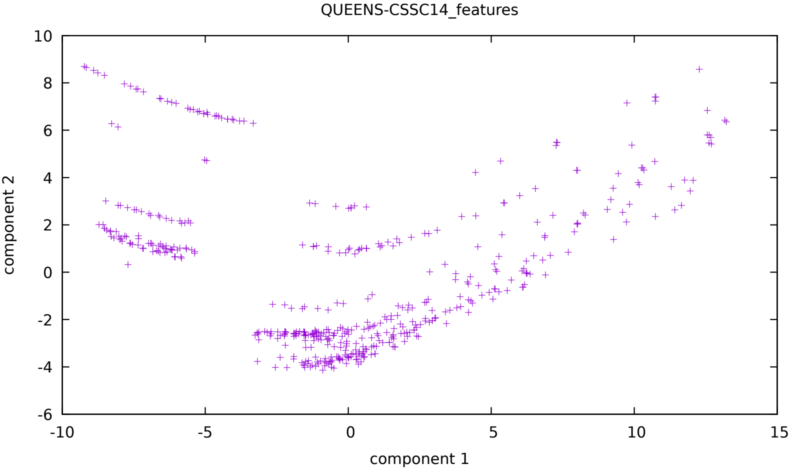
- Different solvers
- Different configuration of the same solver

Goal: Minimise runtime of the solver

Which solver is the best?

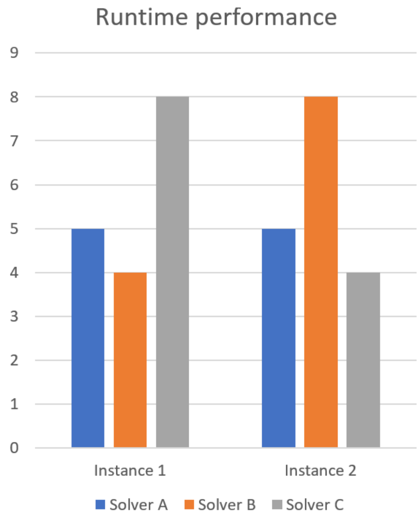


Heterogeneous problem instance sets



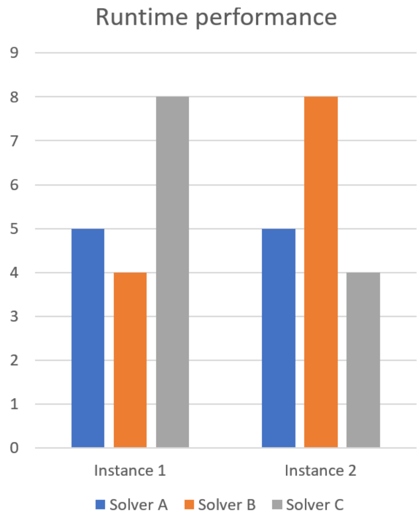
Single best solver

- Solver A is the fastest on average
- Best if we have to choose one solver for all instances



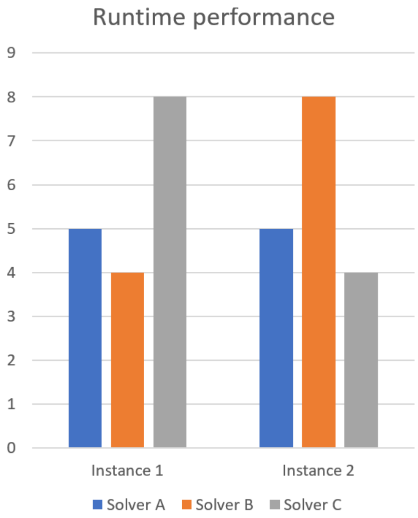
Single best solver: Can we do better?

- What if we can choose multiple solvers?



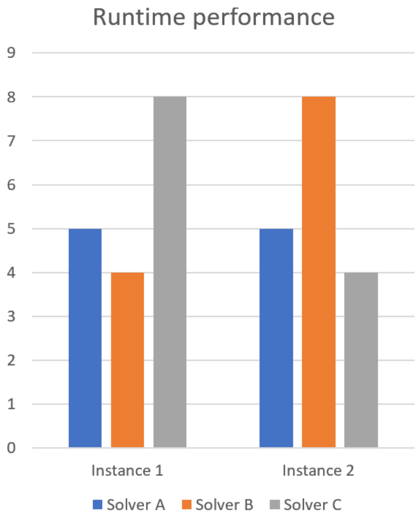
Single best solver: Can we do better?

- What if we can choose multiple solvers?
- Solver B is fastest for instance I_1



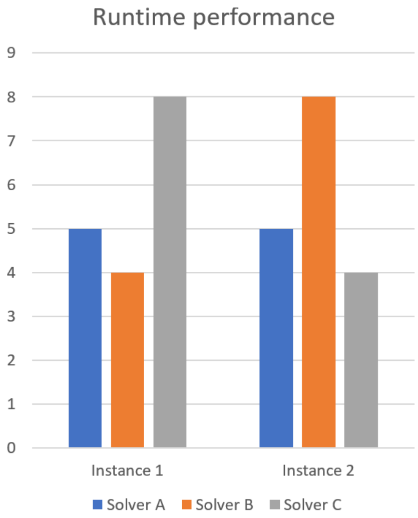
Single best solver: Can we do better?

- What if we can choose multiple solvers?
- Solver B is fastest for instance I_1
- Solver C is fastest for instance I_2



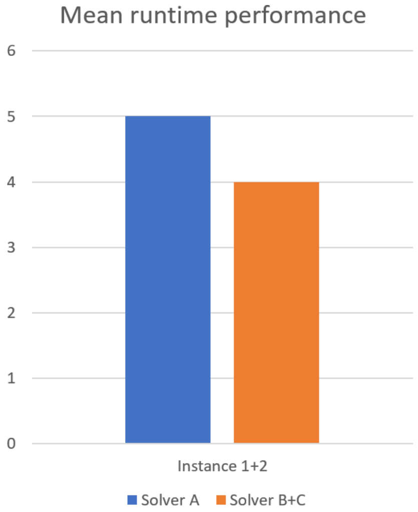
Single best solver: Can we do better?

- What if we can choose multiple solvers?
- Solver B is fastest for instance I_1
- Solver C is fastest for instance I_2
- Faster than the single best is possible!



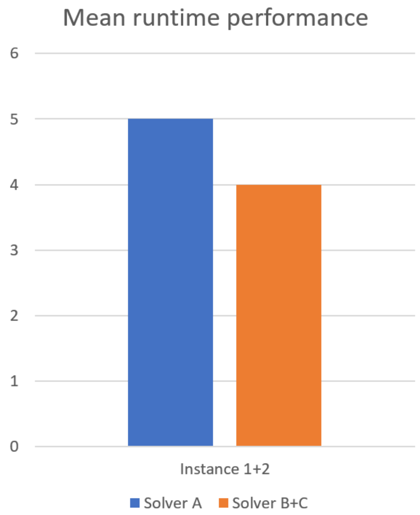
Per-instance algorithm selection

- Carefully choose between solvers B and C, depending on the instance



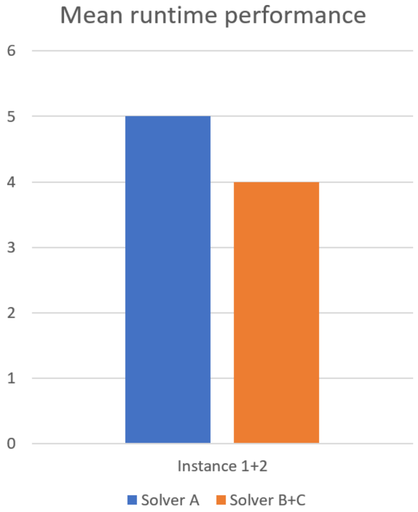
Per-instance algorithm selection

- Carefully choose between solvers B and C, depending on the instance
- Performance improved over the single best solver A



Per-instance algorithm selection

- Carefully choose between solvers B and C, depending on the instance
- Performance improved over the single best solver A
- Per-instance algorithm selection: Choose for every single problem instance



Algorithm selection: Predictions

- Instances $I_1, I_2 \rightarrow$ Best solver is known
- Instances $I_3, \ldots \rightarrow$ Unclear which solver is best

Algorithm selection: Predictions

- Instances $I_1, I_2 \rightarrow$ Best solver is known
- Instances $I_3, \ldots \rightarrow$ Unclear which solver is best
- Prediction using instance features (often problem specific)
  - Simple example: Number of variables in a Boolean formula
  - Note: Instance features should be sufficiently cheap

Algorithm selection: Predictions

- Instances $I_1, I_2 \rightarrow$ Best solver is known
- Instances $I_3, \ldots \rightarrow$ Unclear which solver is best

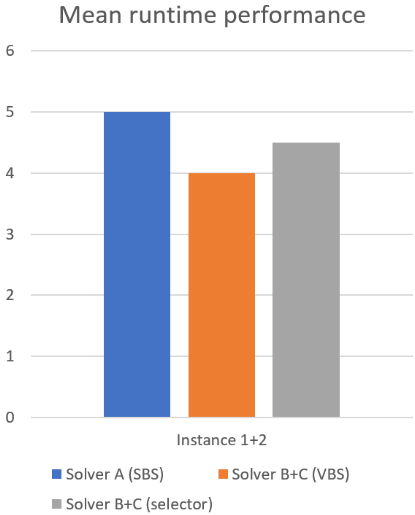
- Prediction using instance features (often problem specific)
  - Simple example: Number of variables in a Boolean formula
  - Note: Instance features should be sufficiently cheap

- Simple strategy
  - Compute features for new instance
  - Compare feature values to training set (I_1, I_2)
  - Use best solver for the most similar training instance

How good is an algorithm selector?

- Predictions are imperfect
- Ground truth
  - Best solver per instance
- Virtual best solver (VBS)
  - Always chooses correctly

Selector performance: Distance to the virtual best solver (VBS)



Algorithm selection in practice

Steps to prepare

- Choose solvers, problem instances, instance features
- Compute performance of all solvers on training set
- Compute instance features on training set

Steps to select a solver (basic idea for illustration)

- Compute features on new instance
- Compute distance of features to known instances
- Check which solver was fastest on the closest known instance

Problems with manual algorithm selection

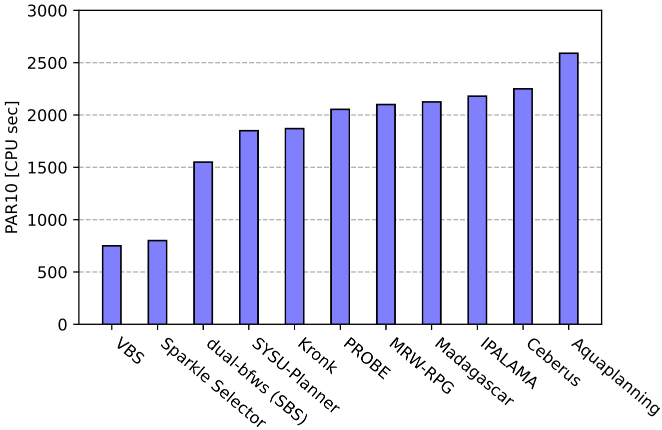
- Difficult to reproduce
  (without having an exact step-by-step description)
- Tedious
- Error prone

Automated algorithm selection (AAS)

- Use a selection algorithm
- Same behaviour when run again
  - (there may be non-determinism, but the steps are the same)
- Code of the selection algorithm describes how it works
- Trade human time for computation time

- Recent AAS survey paper [Kerschke et al.,'19]
- Example methods
  - AutoFolio [Lindauer et al.,'15]
  - Alors [Mısır & Sebag,'17]
  - ASAP [Gonard et al.,'17]
  - ...
  - At PPSN '22 [Kostovska et al.,'22], [Trajanov et al.,'22],[Prager et al.,'22]

Algorithm selection: Example



Results from the Sparkle Planning Challenge 2019 – `https://ada.liacs.nl/events/sparkle-planning-19/`

AAS summary

- Algorithm selection aims to improve performance by choosing the best algorithm for the job
- Per instance algorithm selection: For each problem instance decide which algorithm to use
- Selectors are not perfect, but (usually) better than the single best solver
- Automated algorithm selection makes it easier to apply

Automated algorithm configuration and selection

- Algorithm configuration (AC) → Optimise algorithm performance
- Algorithm selection (AS) → Best algorithm per problem instance
- Automation (AAC, AAS)
  - Better reproducibility
  - Save your time (less tedious)
  - Reduce mistakes

Automated algorithm configuration and selection

- Algorithm configuration (AC) $\rightarrow$ Optimise algorithm performance
- Algorithm selection (AS) $\rightarrow$ Best algorithm per problem instance
- Automation (AAC, AAS)
  - Better reproducibility
  - Save your time (less tedious)
  - Reduce mistakes

After the break: How to make using AAC and AAS even nicer

Break!

The Sparkle Platform

What is Sparkle?

Sparkle is a tool to facilitate the easy and correct use of meta-algorithms by non experts.

Sparkle assists with

- Set-up of experiments
- Reproducible execution
- Concise and complete reporting

One framework for multiple meta-algorithms

Automated algorithm configuration and selection

- Algorithm configuration (AC) $\rightarrow$ Optimise algorithm performance
- Algorithm selection (AS) $\rightarrow$ Best algorithm per problem instance
- Automation (AAC, AAS)
  - Better reproducibility
  - Save time (less tedious)
  - Reduce mistakes

Challenges when using AAC and AAS

- AAC and AAS are intrinsically complicated
- Tools assume expertise $\rightarrow$ E.g., best practices and pitfalls
  - E.g., for AAC, use multiple runs [Eggensperger et al.,'19]
  - E.g., for AAS, validation with the SBS and VBS
- Not easy to use for non-experts in AAC/AAS
- Incorrect use of AAC/AAS
  - Reduces their value
  - Worst case: Performance loss
- Partial automation
  - Error prone to write your own scripts

How Sparkle helps

- More accessible
  - Simple command line interface
  - Self-explanatory commands
- Integrated best practises + pitfall avoidance for AAC and AAS
- Detailed report
  - experimental procedure
  - result analysis
- Further automation (algorithm runs, feature computation, ...)
- Provide access to state-of-the-art AAC and AAS tools
  - AAC: SMAC [Hutter et al.,'11]
  - AAS: AutoFolio [Lindauer et al.,'15]
- Uses Slurm workload manager

Configuration process

Prepare algorithm, wrappers, parameter space, instances, configurator, ...

Configuration procedure & execution

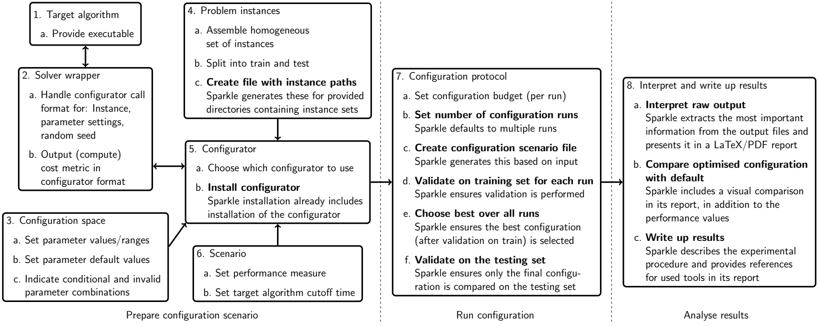
Result interpretation & analysis

Configuration process

Prepare algorithm, wrappers, parameter space, instances, configurator, . . .

Configuration procedure & execution

Result interpretation & analysis



AAC - Algorithm

- Algorithm executable

AAC - Algorithm

- Algorithm executable
- Parameter configuration space (PCS) file [Ramage & Hutter,'15]
  - Name, type and range `a integer [0,255]`
  - Conditional parameters `b | c in {foo}`
  - Forbidden combinations `a=0, c=foo`

AAC - Algorithm

- Algorithm executable
- Parameter configuration space (PCS) file [Ramage & Hutter,'15]
    - Name, type and range `a integer [0,255]`
    - Conditional parameters `b | c in {foo}`
    - Forbidden combinations `a=0, c=foo`
- Wrapper script – interface between Sparkle and algorithm
    - Format input to call configurator
      (instance, parameter settings, random seed)
    - Process output into configurator format
      (cost metric, run status)
    - Sparkle (and many AAC methods) include a template

AAC - Instances

- Homogeneous instance set
- Split into train + test
- Indicate instance paths to configurator

AAC - Instances

- Homogeneous instance set
- Split into train + test
- Indicate instance paths to configurator

Sparkle

- Paths forwarded automatically
- Directory with instance files
  - Multi-file instances possible $\rightarrow$ extra file with combinations
- In case of instance generators
  - Pre-generate instances
  - Handle instance loading in wrapper based on file contents

AAC - Configurator and scenario

- Configurator included with Sparkle (SMAC [Hutter et al.,'11])
  - No separate installation required

- Set performance measure – Runtime/quality
- Set target algorithm cutoff time
- Set configuration budget

- Sparkle: Set through settings file / command line options

AAC - Configuration procedure & execution

- Set number of configuration runs
- Set up configuration scenario
- Configure following the standard protocol [Styles et al.,'12]
  - Validate each run on training set
  - Choose best configuration over all runs
  - Validate best configuration on testing set

- Sparkle handles the scenario based on input
- Sparkle ensures the protocol is followed

AAC - Running the experiment - Input

- Resources/
  - `PbO-CSCCSAT/`
    - `PbO-CSCCSAT`
    - `sparkle_smac_wrapper.py`
    - `parameters.pcs`
  - `PTN/`
    - `instance1.cnf`
    - `...`
    - `instance10.cnf`
  - `PTN2/`
    - `instance11.cnf`
    - `...`
    - `instance20.cnf`

AAC - Running the experiment

1: `Commands/initialise.py`

2: `Commands/add_instances.py Resources/PTN/`

3: `Commands/add_instances.py Resources/PTN2/`

4: `Commands/add_solver.py --deterministic 0 Resources/PbO-CSCCSAT/`

5: `Commands/configure_solver.py --solver PbO-CSCCSAT --instance-set-train PTN`

6: `Commands/sparkle_wait.py`

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

8: `Commands/generate_report.py`

AAC - Result interpretation & analysis

- Interpret raw output
- Compare optimised configuration with default
- Write up results

AAC - Result interpretation & analysis

- Interpret raw output
- Compare optimised configuration with default
- Write up results

- Often very basic in AAC tools
  - E.g., only default + optimised performance value

AAC – Sparkle report

Sparkle generates a detailed report including:

- Used instance sets, target algorithm, configurator
- Experiment description (protocol, budgets, ...)
- Performance values + plots (#timeouts if optimising runtime)

AAC – Sparkle report

Sparkle generates a detailed report including:

- Used instance sets, target algorithm, configurator
- Experiment description (protocol, budgets, . . .)
- Performance values + plots (#timeouts if optimising runtime)

Configuration Report for the Solver PbO-CCSAT-Generic on the Training Instance Set PTN in *Sparkle*

Automatically generated by *Sparkle* (version: 1.0.0)

20th January 2022

1 Introduction

Sparkle [3] is a multi-agent problem-solving platform based on Programming by Optimisation (PbO) [2], and would provide a number of effective algorithm optimisation techniques (such as automated algorithm configuration, portfolio-based algorithm selection, etc) to accelerate the existing solvers.

This experimental report is automatically generated by *Sparkle*. This report presents experimental results on the scenario of configuring the solver PbO-CCSAT-Generic on the training instance set PTN.

2 Information about the Instance Set(s)

- Training set: **PTN**, consisting of 12 instances

3 Information about the Configuration Protocol

The configurator used in *Sparkle* is SMAC (*Sequential Model-based Algorithm Configuration*) [4], and the version of SMAC used in *Sparkle* is 2.10.03.

During the configuration process, *Sparkle* performs 10 independent SMAC runs for configuring the solver PbO-CCSAT-Generic on the trainging instance set PTN; the configuration objective is RUNTIME; the whole configuration time budget is 3600 seconds; the cutoff time for each run is 120 seconds.

Each independent run of SMAC would result in one optimised configuration. As a result, *Sparkle* would obtain 10 optimised configurations. Each of these was then evaluated on the entire training set, with one solver run per instance and a cutoff time of 120 seconds, and the configuration with the lowest PAR10 value was selected as the result of the configuration process.

4 Information about the Optimised Configuration

After the configuration process mentioned above, *Sparkle* obtained the optimised configuration. The details of the optimised configuration are described as below.

-gamma_hscore2 '351' -init_solution '1' -p_swt '0.20423712003341465' -perform_aspiration '1' -perform_clause_weight '1' -perform_double_cc '0' -perform_first_div '0' -perform_pac '1' -prob_pac '0.005730374136488115' -q_swt '0.6807207179674418' -sel_clause_div '1' -sel_clause_weight_scheme '1' -sel_var_break_tie_greedy '4' -sel_var_div '2' -threshold_swt '32'

1

5 Comparison between Configured Version and Default Version on the Training Instance Set

In order to investigate the performance on the training instance set, *Sparkle* would run the configured version of PbO-CCSAT-Generic and the default version of PbO-CCSAT-Generic on the training instance set. During this phase, each version was performed one run per instance with a cutoff time of 120 seconds. The results are reported as follows.

- **PbO-CCSAT-Generic (configured)**, PAR10: 3.2082931200663247
- **PbO-CCSAT-Generic (default)**, PAR10: 621.2856854001681

The empirical comparison between the PbO-CCSAT-Generic (configured) and PbO-CCSAT-Generic (default) on the training set of PTN is presented in Figure 1.

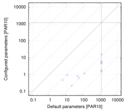


Figure 1: Empirical comparison between the PbO-CCSAT-Generic (configured) and PbO-CCSAT-Generic (default) on the training set of PTN.

Table 1 shows on how many instances the PbO-CCSAT-Generic (configured) and PbO-CCSAT-Generic (default) timed out (did not solve the instance within the cutoff time of 120 seconds) on the training set of PTN, as well as on how many instances both timed out.

configured	default	overlap
0	6	0

Table 1: Number of time-outs for PbO-CCSAT-Generic (configured), PbO-CCSAT-Generic (default), and for how many instances both timed out on the training set of PTN.

6 Parameter importance via Ablation

Ablation analysis [1] is performed from the PbO-CCSAT-Generic (default) to PbO-CCSAT-Generic (configured) to see which parameter changes between them contribute most to the improved performance. The ablation path is constructed and validated with the training set PTN. The set of

2

parameters that differ in the two configurations will form the ablation path. Starting from the default configuration, the path is computed by performing a sequence of rounds. In a round, each available parameter is flipped in the configuration and is validated on its performance. The flipped parameter with the best performance in that round, is added to the configuration and the next round starts with the remaining parameters. This repeats until all parameters are flipped, which is the best found configuration. The analysis resulted in the ablation path presented in Table 2.

Table 2: Ablation path from PbO-CCSAT-Generic (default) to PbO-CCSAT-Generic (configured) where parameters with higher importance are ranked higher.

Round	Flipped parameter	Source value	Target value	Validation result
0	-source-	N/A	N/A	610.48433
1	sel_var_div	3	2	
	sel_var_break_tie_greedy	2	4	
	gamma_hscore2	1000	351	116.32313
2	perform_pac	0	1	
	prob_pac	5.800000000000001E-4	0.005730374136488115	18.91441
3	p_swt	0.3	0.20423712003341465	122.56680
4	q_swt	0.0	0.6807207179674418	17.40350
5	threshold_swt	300	32	103.07659
6	perform_double_cc	1	0	3.85328
7	-target-	N/A	N/A	3.85717

References

[1] Chris Fawcett and Holger H. Hoos. Analysing differences between algorithm configurations through ablation. *J. Heuristics*, 22(4):431–458, 2016.

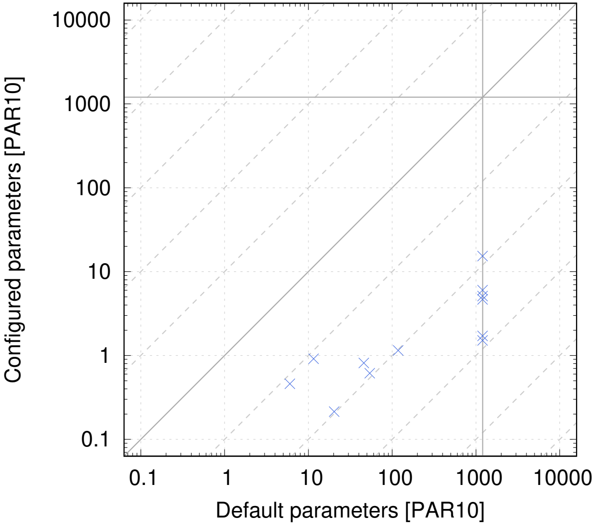
[2] Holger H. Hoos. Programming by Optimization. *Communications of the ACM*, 55(2):70–80, 2012.

[3] Holger H. Hoos. Sparkle: A pbo-based multi-agent problem-solving platform. Technical report, Department of Computer Science, University of British Columbia, 2015.

[4] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION 5)*, pages 507–523, 2011.

3

AAC – Performance plot

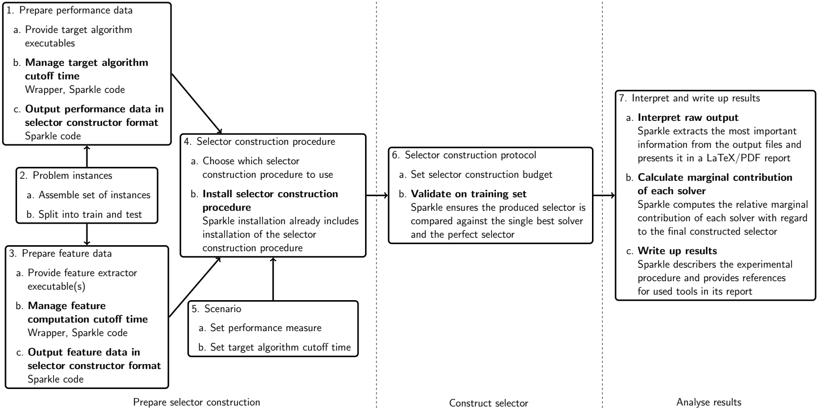


Selector (construction) process

Prepare algorithm, feature extractor, wrappers, instances, selector constructor, . . .

Selector construction procedure

Result interpretation & analysis

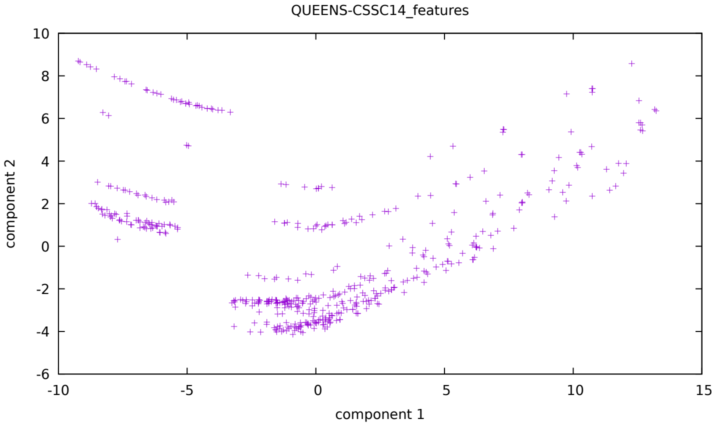


AAS - Instances

- Heterogeneous instance set
- Split into train + test

AAS - Instances

- Heterogeneous instance set
- Split into train + test



AAS - Prepare performance data

Performance data is input to construct a selector

- Provide target algorithm executables + wrappers (for each algorithm to be considered by the selector)
  - Sparkle provides wrapper template
- Manage target algorithm cutoff time
  - Handled by wrapper + Sparkle
- Run each algorithm on each training instance
- Output performance data in selector constructor format
  - Both automated by Sparkle

AAS - Prepare feature data

Feature data is input to construct a selector

- Provide feature extractor executable(s) + wrappers (for each extractor, if multiple are used)
- Manage feature extractor cutoff time
  - Handled by wrapper + Sparkle
- Run each feature extractor on each training instance
- Output feature data in selector constructor format
  - Both automated by Sparkle

AAS - Selector constructor and scenario

- Selector constructor included with Sparkle (AutoFolio [Lindauer et al.,'15])
  - No separate installation required

- Set performance measure — Currently runtime only
- Set target algorithm + feature extractor cutoff time

- Sparkle: Set through settings file / command line options

Automated algorithm selection (AAS)

- To use per-instance AAS we need a selector
- Selectors are problem specific
- Helpful: Approach to construct a selector

Automated algorithm selection (AAS)

- To use per-instance AAS we need a selector
- Selectors are problem specific
- Helpful: Approach to construct a selector

AutoFolio [Lindauer et al.,'15]

- Automatically chooses from different selection mechanisms
- Optimises parameter settings of selection mechanisms
- Result: Specialised selector for a specific problem
- Using... automated algorithm configuration! (SMAC)

AAS - Selector construction procedure & execution

- Set selector construction budget
- Validate on training set against
  - Single best solver (SBS)
  - Virtual best solver (VBS) – or: perfect selector, oracle

- Sparkle ensures validation happens

AAS - Running the experiment - Input

- Resources/
  - PbO-CSCCSAT/
    - PbO-CSCCSAT
    - sparkle_run_default_wrapper.py
  - CSCCSat/
    - ...
  - MiniSAT/
    - ...
  - PTN/
    - instance1.cnf
    - ...
    - instance10.cnf
  - Extractor/
    - sparkle_run_default_wrapper.py

AAS - Running the experiment

```
Commands/initialise.py
Commands/add_instances.py path/to/PTN/
Commands/add_solver.py --deterministic 0 path/to/PbO-CSCCSAT/
Commands/add_solver.py --deterministic 0 path/to/CSCCSat/
Commands/add_solver.py --deterministic 0 path/to/MiniSAT/
Commands/run_solvers.py --parallel
Commands/add_feature_extractor.py path/to/Extractor/
Commands/compute_features.py --parallel
Commands/sparkle_wait.py
Commands/construct_sparkle_portfolio_selector.py
Commands/run_sparkle_portfolio_selector.py path/to/PTN2/
Commands/generate_report.py
```

AAS - Result interpretation & analysis

- Interpret raw output
- Compare selector with SBS and VBS
- Write up results

AAS - Result interpretation & analysis

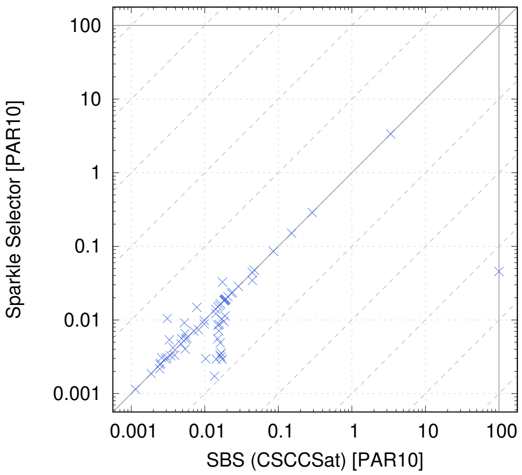
- Interpret raw output
- Compare selector with SBS and VBS
- Write up results

- Often very basic in AAS tools
  - E.g., only selector + SBS + VBS performance values

AAS - Sparkle report

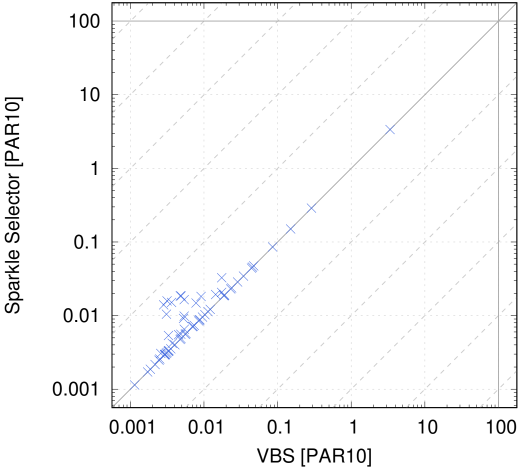
- Sparkle generates a detailed report including:
  - Used instance sets, target algorithms, feature extractor(s), selector construction technique
  - Experiment description (protocol, budgets, ...)
  - Performance values + plots

AAS - Performance plot – Selector vs. SBS



AAS - Performance plot – Selector vs. VBS

Cannot do better than being on the diagonal!



AAS - Marginal contributions

What does an algorithm contribute to a selector?

I.e.: To which extend does the selector performance depend on this algorithm?

AAS - Marginal contributions

What does an algorithm contribute to a selector?

I.e.: To which extend does the selector performance depend on this algorithm? Of interest to answer:

- How useful is my new algorithm?
- Who performed best in a competition?
- What is the state of the art?

AAS - Marginal contributions

What does an algorithm contribute to a selector?

I.e.: To which extend does the selector performance depend on this algorithm? Of interest to answer:

- How useful is my new algorithm?
- Who performed best in a competition?
- What is the state of the art?

Summary of things Sparkle helps with?

- Ensures that procedures are followed correctly and exactly
- Reproducible of experimental set-up
- Detailed reporting mechanism
- Automatically installs many tools and provides wrapper interfaces

Sparkle - Other topics

- Parameter importance, through ablation analysis [Fawcett,'16]
- Sparkle competitions
  - SAT
  - AI Planning
- Parallel algorithm portfolios
  - Run algorithms in parallel, get the best solution

Sparkle - Future work

- Other configurators + selector constructors
- Multi-objective support
- AAS for quality optimisation
- Combine AAC with AAS
- Extend to run locally and other workload managers

Demo

Sparkle is a tool to facilitate the easy and correct use of meta-algorithms by non experts.

- Paper accepted in TEVC, will be published soon
  - Koen van der Blom, Holger H. Hoos, Chuan Luo, Jeroen G. Rook, *Sparkle: Towards Accessible Meta-Algorithmics for Improving the State of the Art in Solving Challenging Problems*
- Contact information
  - `koen.vdblom@lip6.fr`
  - `j.g.rook@utwente.nl`

`https://bitbucket.org/sparkle-ai/sparkle/`

References I

Carlos Ansótegui, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney, *Model-based genetic algorithms for algorithm configuration*, Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), 2015, pp. 733–739.

Katharina Eggensperger, Marius Lindauer, and Frank Hutter, *Pitfalls and best practices in algorithm configuration*, Journal of Artificial Intelligence Research **64** (2019), 861–893.

Chris Fawcett, Malte Helmert, Holger Hoos, Erez Karpas, Gabriele Röger, and Jendrik Seipp, *FD-Autotune: Domain-specific configuration using fast downward*, PAL '11, 2011, pp. 13–20.

François Gonard, Marc Schoenauer, and Michèle Sebag, *Asap.v2 and asap.v3: Sequential optimization of an algorithm selector and a scheduler*, Proceedings of the Open Algorithm Selection Challenge (Marius Lindauer, Jan N. van Rijn, and Lars Kotthoff, eds.), Proceedings of Machine Learning Research, vol. 79, PMLR, 11–12 Sep 2017, pp. 8–11.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown, *Sequential model-based optimization for general algorithm configuration*, Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION 5), 2011, pp. 507–523.

Frank Hutter and Steve Ramage, *Manual for SMAC version v2.10.03-master*, 2015.

References II

Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann, *Automated algorithm selection: Survey and perspectives*, Evolutionary Computation **27** (2019), 3–45.

Ana Kostovska, Anja Jankovic, Diederick Vermetten, Jacob de Nobel, Hao Wang, Tome Eftimov, and Carola Doerr, *Per-run algorithm selection with warm-starting using trajectory-based features*, Parallel Problem Solving from Nature – PPSN XVII (Cham) (Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar, eds.), Springer International Publishing, 2022, pp. 46–60.

Ashiqur R. KhudaBukhsh, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown, *SATenstein: Automatically building local search SAT solvers from components*, Artificial Intelligence **232** (2016), 20–42.

Marius Thomas Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub, *AutoFolio: an automatically configured algorithm selector*, Journal of Artificial Intelligence Research **53** (2015), 745–778.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari, *The irace package: Iterated racing for automatic algorithm configuration*, Operations Research Perspectives **3** (2016), 43–58.

References III

Mustafa Mısır and Michèle Sebag, *Alors: An algorithm recommender system*, Artificial Intelligence **244** (2017), 291–314, Combining Constraint Solving with Mining and Learning.

Yasha Pushak and Holger H. Hoos, *Golden parameter search: Exploiting structure to quickly configure parameters in parallel*, Proceedings of the 2020 Genetic and Evolutionary Computation Conference (New York, NY, USA), GECCO '20, Association for Computing Machinery, 2020, p. 245–253.

Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke, *Automated algorithm selection in single-objective continuous optimization: A comparative study of deep learning and landscape analysis methods*, Parallel Problem Solving from Nature – PPSN XVII (Cham) (Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar, eds.), Springer International Publishing, 2022, pp. 3–17.

John R Rice, *The algorithm selection problem*, Advances in computers, vol. 15, Elsevier, 1976, pp. 65–118.

Jeroen Rook, Heike Trautmann, Jakob Bossek, and Christian Grimme, *On the potential of automated algorithm configuration on multi-modal multi-objective optimization problems*, GECCO '22 Companion, 2022, p. 356–359.

References IV

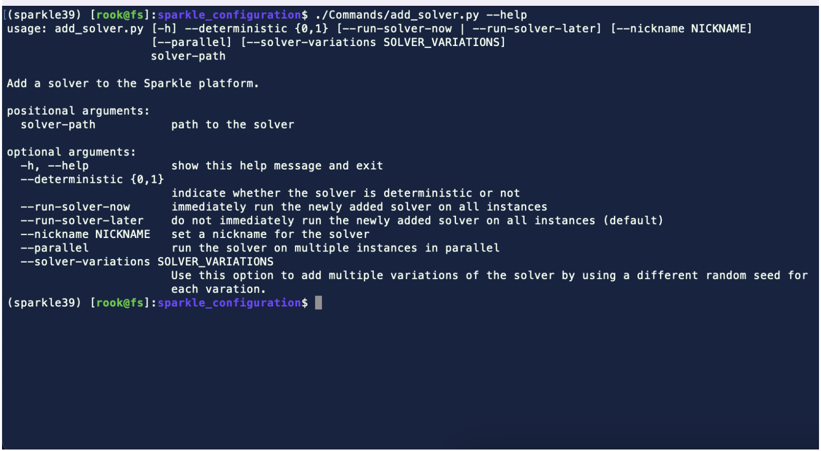
James Styles, Holger H. Hoos, and Martin Müller, *Automatically configuring algorithms for scaling performance*, Learning and Intelligent Optimization - 6th International Conference, LION 6, Paris, France, January 16-20, 2012, Revised Selected Papers (Youssef Hamadi and Marc Schoenauer, eds.), Lecture Notes in Computer Science, vol. 7219, Springer, 2012, pp. 205–219.

Risto Trajanov, Ana Nikolikj, Gjorgjina Cenikj, Fabien Teytaud, Mathurin Videau, Olivier Teytaud, Tome Eftimov, Manuel López-Ibáñez, and Carola Doerr, *Improving nevergrad's algorithm selection wizard ngopt through automated algorithm configuration*, Parallel Problem Solving from Nature – PPSN XVII (Cham) (Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar, eds.), Springer International Publishing, 2022, pp. 18–31.

Demo back-up – AAC

```
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/initialise.py
Start initialising Sparkle platform ...
Now recording current Sparkle platform in file Records/My_Record_2022-09-08-10:02:12_5934_6764862317228415399.zip ...
Current Sparkle platform found!
Current Sparkle platform recorded!
New Sparkle platform initialised!
(sparkle39) [rook@fs]:sparkle_configuration$
```

Demo back-up – AAC



Demo back-up – AAC

```
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/add_solver.py --deterministic 0 ../EA_Resources/MOEAD/
one pcs file detected, this is a configurable solver
Adding solver MOEAD done!
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/add_instances.py ../EA_Resources/MMMOOproblems/
```

Demo back-up – AAC

```
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/configure_solver.py --solver Solvers/MOEAD/ --instance-set-trai
n Instances/MMMOOproblems/ --validate
Callback script to launch validation is placed at Tmp/delayed_validation_MOEAD_MMMOOproblems_script.sh
Once configuration is finished, validation will automatically start as a Slurm job: 27000243
Running configuration in parallel. Waiting for Slurm job(s) with id(s): 27000242,27000243
(sparkle39) [rook@fs]:sparkle_configuration$
```

Demo back-up – AAC

```
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/configure_solver.py --solver Solvers/MOEAD/ --instance-set-trai
n Instances/MMMOOproblems/ --validate
Callback script to launch validation is placed at Tmp/delayed_validation_MOEAD_MMMOOproblems_script.sh
Once configuration is finished, validation will automatically start as a Slurm job: 27000243
Running configuration in parallel. Waiting for Slurm job(s) with id(s): 27000242,27000243
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/sparkle_wait.py
Waiting for 4 jobs...
All jobs done!
(sparkle39) [rook@fs]:sparkle_configuration$ ./Commands/generate_report.py
Generating report for configuration ...
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Report is placed at: Configuration_Reports/MOEAD_MMMOOproblems/Sparkle-latex-generator-for-configuration/Sparkle_Report
_for_Configuration.pdf
Generating report for configuration done!
(sparkle39) [rook@fs]:sparkle_configuration$
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/initialise.py
Start initialising Sparkle platform ...
Now recording current Sparkle platform in file Records/My_Record_2022-09-08-09:54:32_6913_5532464730768778891.zip ...
Current Sparkle platform found!
Current Sparkle platform recorded!
New Sparkle platform initialised!
(sparkle39) [rook@fs]:sparkle_selection$ 
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ls -l Examples/Resources/Solvers/CSCCSat/
total 1708
-rwxrwxr-x 1 rook rook 1638264 Sep  7 08:59 CSCCSat
-rwxrwxr-x 1 rook rook    1984 Sep  7 08:59 sparkle_run_default_wrapper.py
-rw-rw-r-- 1 rook rook   12156 Sep  7 08:59 src.zip
(sparkle39) [rook@fs]:sparkle_selection$
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ls -l Examples/Resources/Solvers/CSCCSat/
total 1708
-rwxrwxr-x 1 rook rook 1638264 Sep  7 08:59 CSCCSat
-rwxrwxr-x 1 rook rook    1984 Sep  7 08:59 sparkle_run_default_wrapper.py
-rw-rw-r-- 1 rook rook   12156 Sep  7 08:59 src.zip
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/add_solver.py --deterministic 0 Examples/Resources/Solvers/CSCCSat/

Adding solver CSCCSat done!
Removing Sparkle report Components/Sparkle-latex-generator/Sparkle_Report.pdf done!
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/add_solver.py --deterministic 0 Examples/Resources/Solvers/PbO-CCSA
T-Generic/
one pcs file detected, this is a configurable solver
Adding solver PbO-CCSAT-Generic done!
(sparkle39) [rook@fs]:sparkle_selection$
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ls -l ../EA_Resources/Barthel/ | head -n5
total 1439
-rw-rw-r-- 1 rook rook 13347 Sep  8 08:56 fla-barthel-200-1.cnf
-rw-rw-r-- 1 rook rook 13376 Sep  8 08:56 fla-barthel-200-2.cnf
-rw-rw-r-- 1 rook rook 13443 Sep  8 08:56 fla-barthel-200-3.cnf
-rw-rw-r-- 1 rook rook 13333 Sep  8 08:56 fla-barthel-200-4.cnf
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/add_instances.py ../EA_Resources/Barthel/
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/add_feature_extractor.py Examples/Resources/Extractors/SAT-features
-competition2012_revised_without_SatELite_sparkle/
Adding feature extractor SAT-features-competition2012_revised_without_SatELite_sparkle done!
(sparkle39) [rook@fs]:sparkle_selection$
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/run_solvers.py --parallel
Start running solvers ...
Cutoff time for each run on solving an instance is set to 60 seconds
The number of total running jobs: 110
Running solvers in parallel. Waiting for Slurm job(s) with id(s): 27000075,27000076
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/
about.py                                  remove_feature_extractor.py
add_feature_extractor.py                  remove_instances.py
add_instances.py                          remove_record.py
add_solver.py                             remove_solver.py
cleanup_current_sparkle_platform.py       run_ablation.py
cleanup_temporary_files.py                run_configured_solver.py
compute_features_parallel.py              run_solvers.py
compute_features.py                       run_sparkle_parallel_portfolio.py
compute_marginal_contribution.py          run_sparkle_portfolio_selector.py
configure_solver.py                       run_status.py
construct_sparkle_parallel_portfolio.py   save_record.py
construct_sparkle_portfolio_selector.py   sparkle_help/
generate_report.py                        sparkle_wait.py
initialise.py                             system_status.py
load_record.py                            test/
__pycache__/                              validate_configured_vs_default.py
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/compute_features.py --parallel
Start computing features ...
The number of total running jobs: 55
Computing features in parallel. Waiting for Slurm job(s) with id(s): 27000102,27000103
(sparkle39) [rook@fs]:sparkle_selection$
```

Demo back-up – AAS

```
(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/sparkle_wait.py
Waiting for 4 jobs...
All jobs done!
(sparkle39) [rook@fs]:sparkle_selection$
```

Demo back-up – AAS

```
construct_sparkle_portfolio_selector.py: error: argument -h/--help: ignored explicit argument 'elp'
[(sparkle39) [rook@fs]:sparkle_selection$ ./Commands/construct_sparkle_portfolio_selector.py
Start constructing Sparkle portfolio selector ...
Sparkle portfolio selector constructed!
Sparkle portfolio selector located at Sparkle_Portfolio_Selector/sparkle_portfolio_selector__@@SPARKLE@@__
Start computing each solver's marginal contribution to perfect selector ...
In this calculation, cutoff time for each run is 60 seconds
Computing virtual best performance for portfolio selector with all solvers ...
Virtual best performance for portfolio selector with all solvers is 55.49983175079534
Computing done!
Computing virtual best performance for portfolio selector excluding solver CSCCSat ...
Virtual best performance for portfolio selector excluding solver CSCCSat is 55.499540733828205
Computing done!
Marginal contribution (to Perfect Selector) for solver CSCCSat is 0.000291016967132407
Computing virtual best performance for portfolio selector excluding solver PbO-CCSAT-Generic ...
Virtual best performance for portfolio selector excluding solver PbO-CCSAT-Generic is 54.490685270413564
Computing done!
Marginal contribution (to Perfect Selector) for solver PbO-CCSAT-Generic is 1.009146480381773
******
Solver ranking list via marginal contribution (Margi_Contr) with regards to perfect selector
#1: PbO-CCSAT-Generic    Margi_Contr: 1.009146480381773
#2: CSCCSat      Margi_Contr: 0.000291016967132407
******
Marginal contribution (perfect selector) computing done!
Start computing each solver's marginal contribution to actual selector ...
In this calculation, cutoff time for each run is 60 seconds
Computing actual performance for portfolio selector with all solvers ...
Portfolio selector already exists for the current feature and performance data.
```

Demo back-up – AAS

```
Marginal contribution (to Perfect Selector) for solver CSCCSat is 0.000291016967132407
Computing virtual best performance for portfolio selector excluding solver PbO-CCSAT-Generic ...
Virtual best performance for portfolio selector excluding solver PbO-CCSAT-Generic is 54.490685270413564
Computing done!
Marginal contribution (to Perfect Selector) for solver PbO-CCSAT-Generic is 1.009146480381773
******
Solver ranking list via marginal contribution (Margi_Contr) with regards to perfect selector
#1: PbO-CCSAT-Generic    Margi_Contr: 1.009146480381773
#2: CSCCSat      Margi_Contr: 0.000291016967132407
******
Marginal contribution (perfect selector) computing done!
Start computing each solver's marginal contribution to actual selector ...
In this calculation, cutoff time for each run is 60 seconds
Computing actual performance for portfolio selector with all solvers ...
Portfolio selector already exists for the current feature and performance data.
Actual performance for portfolio selector with all solvers is 55.499540733828205
Computing done!
Computing actual performance for portfolio selector excluding solver CSCCSat ...
Actual performance for portfolio selector excluding solver CSCCSat is 55.499540733828205
Computing done!
Marginal contribution (to Actual Selector) for solver CSCCSat is 0.0
Computing actual performance for portfolio selector excluding solver PbO-CCSAT-Generic ...
Actual performance for portfolio selector excluding solver PbO-CCSAT-Generic is 54.490685270413564
Computing done!
Marginal contribution (to Actual Selector) for solver PbO-CCSAT-Generic is 1.0088554634146405
******
Solver ranking list via marginal contribution (Margi_Contr) with regards to actual selector
#1: PbO-CCSAT-Generic    Margi_Contr: 1.0088554634146405
#2: CSCCSat      Margi_Contr: 0.0
******
Marginal contribution (actual selector) computing done!
(sparkle39) [rook@fs]:sparkle_selection$
```

Demo back-up – AAS

```
This is BibTeX, Version 0.99d (TeX Live 2013)
The top-level auxiliary file: Sparkle_Report.aux
The style file: plain.bst
Database file #1: Sparkle_Report.bib
This is BibTeX, Version 0.99d (TeX Live 2013)
The top-level auxiliary file: Sparkle_Report.aux
The style file: plain.bst
Database file #1: Sparkle_Report.bib
Report is placed at: Components/Sparkle-latex-generator/Sparkle_Report.pdf
Report generated ...
(sparkle39) [rook@fs]:sparkle_selection$
```