



Discovering Rules for Rule-Based Machine Learning with the Help of Novelty Search

Michael Heider¹ · Helena Stegherr¹ · David Pätzel¹ · Roman Sraj¹ · Jonathan Wurth¹ · Benedikt Volger¹ · Jörg Hähner¹

Received: 31 March 2023 / Accepted: 28 July 2023
© The Author(s) 2023

Abstract

Automated prediction systems based on machine learning (ML) are employed in practical applications with increasing frequency and stakeholders demand explanations of their decisions. ML algorithms that learn accurate sets of rules, such as learning classifier systems (LCSs), produce transparent and human-readable models by design. However, whether such models can be effectively used, both for predictions and analyses, strongly relies on the optimal placement and selection of rules (in ML this task is known as model selection). In this article, we broaden a previous analysis on a variety of techniques to efficiently place good rules within the search space based on their local prediction errors as well as their generality. This investigation is done within a specific pre-existing LCS, named SupRB, where the placement of rules and the selection of good subsets of rules are strictly separated—in contrast to other LCSs where these tasks sometimes blend. We compare two baselines, random search and $(1, \lambda)$ -evolution strategy (ES), with six novelty search variants: three novelty-/fitness weighing variants and for each of those two differing approaches on the usage of the archiving mechanism. We find that random search is not sufficient and sensible criteria, i.e., error and generality, are indeed needed. However, we cannot confirm that the more complicated-to-explain novelty search variants would provide better results than $(1, \lambda)$ -ES which allows a good balance between low error and low complexity in the resulting models.

Keywords Rule set learning · Rule-based learning · Learning classifier systems · Evolutionary machine learning · Interpretable models · Explainable AI · Novelty search

Introduction

Autonomous decisions of agents are more common than ever as automation in many socio-technical settings, such as manufacturing, increases steadily. This advent of more independent and intelligent technical systems holds great potential in increasing safety and comfort of workers as well as general productivity. However, the roll-out of such systems is often slowed down or blocked when stakeholders, from operator to management level, do not trust that the systems perform on-par or even better than their human partners. Therefore, stakeholders expect the agents to provide

sensible explanations and insights into their decision making processes [1].

For the machine learning components of these agents, this implies that they have to offer strong interpretability, self-explaining or transparency-by-design capabilities. The use of learning classifier systems (LCSs) [2, 3] in these kinds of scenario was introduced in [4]. There, the unique suitability of LCSs is discussed, as this family of rule-based learning algorithms features both powerful machine learning capabilities applying to all commonly encountered settings as well as inherently transparent models that resemble human explanation processes due to their rule-like structure. Rules in an LCS are of an if-then form, where the *if* (or condition) constricts the section of the feature space this rule applies to (matches) and the *then* contains a simpler, therefore, more comprehensible, local model able to predict data points from that feature space partition. Conditions are usually optimized using evolutionary algorithms or similar stochastic search heuristics. The model's practical transparency (or

This article is part of the topical collection “Advances on Computational Intelligence 2022” guest edited by Joaquim Filipe, Kevin Warwick, Janusz Kacprzyk, Thomas Bäck, Bas van Stein, Christian Wagner, Jonathan Garibaldi, H. K. Lam, Marie Cottrell and Faiyaz Doctor.

Extended author information available on the last page of the article

explainability in a broader context) is primarily influenced by the effective number of rules and their placement within the feature space (the two tasks that constitute model selection in rule-based machine learning, cf. [3]). Therefore, the—preferably near optimal—placement of rules should be a primary concern when designing an LCS algorithm with explainability requirements.

In this work, we further expand the supervised rule-based learning system (SupRB) [5], a recently proposed LCS for regression tasks, with different methods to discover new rules that fit the data well and benchmark them against each other on a variety of real-world datasets. This article extends our previous work on this topic originally presented at ECTA/IJCCI 2022 [6].

Learning Classifier Systems and Rule Discovery

As previously introduced, LCSs are a family of—typically evolutionary—rule-based learning systems with a long research history [2, 3, 7]. Traditionally, they are categorized into two approaches:

- *Michigan-style systems*, which feature strong online learning capabilities and have evolved from reinforcement learning (RL) to all major learning schemes. They consist of a single population of rules on which the optimizer operates directly.
- *Pittsburgh-style systems*, that typically perform offline batch learning and primarily focus on supervised learning and feature a population of sets of rules, where the optimizer operates on those sets rather than the rules individually.

Note that there have been recent advances towards an updated classification system in [3] where batch learning versus online learning and single- versus multi-solution form the basis of classification. In that system, SupRB is classified as a multi-solution batch learning system.

Due to those very different outsets, their approaches to rule discovery and improvement differ substantially. For example, the most widespread Michigan-style system, the XCS classifier system [8], features two mechanisms to determine new rules. The first is the covering mechanism: Whenever the number of rules that match a new data point falls below a predetermined threshold, new rules are generated that match this data point. These rules are often randomly made more general than to just match this point specifically and, therefore, slightly differ when inserted into the population of rules. The second is the evolutionary algorithm which is invoked on matching rules that also propose the actually-chosen action (e.g., when following an *epsilon*-greedy policy

in RL). It utilizes crossover and mutation mechanisms appropriate for the types of input data and evolves the population in a steady-state manner.

For Pittsburgh-style systems, new rules can be directly added to an individual of the population (a set of rules), shared between individuals or be the product of a mutation (and more rarely rule-level crossover) operator. As the evolutionary algorithm of these systems operates on rule sets rather than rules directly, the fitness signal that guides evolution is not based on individual rule performance but rather the performance of a given set of rules, which can complicate rule discovery due to the added indirection.

The Supervised Rule-Based Learning System

The supervised rule-based learning system (SupRB) [5] is a rule set learning algorithm/an LCS with alternating phases of rule discovery and solution composition (additional details in [9]). The two phases combined solve the standard machine learning task of model selection (cf. [3]). Splitting into those two phases—in contrast to solving the task with a single optimizer or an optimizer and a post-hoc heuristic—allows more precise control of the optimization process as objectives are more directly related to the specific behavior of the component under optimization. In contrast, in Pittsburgh-style systems, a model's fitness that also controls changes made to its rules follows the interaction of its rules rather than their individual suitability, whereas, in Michigan-style systems, rule fitness often depends on neighboring rules (niching) and, due to their online learning nature, does not necessarily relate to the full dataset. In the *rule discovery* (RD) phase of SupRB, new rules are generated and locally optimized to produce a diverse pool of maximally general rules still exhibiting low errors. Subsequently, in the *solution composition* (SC) phase, an optimizer attempts to choose a well-performing subset from the pool of discovered rules. Ideally, this subset is small (exhibiting low complexity) but also produces a model (due to the mixing of rules in areas of overlap) with accurate predictions. The obvious tradeoff between those two objectives (similar to general versus accurate rules during RD) leads to a need to balance these objectives. The exact nature of the balance highly depends on the use case at hand and the inherent requirements for explanations and model readability (cf. [1]). Sometimes, more complex models can be acceptable, whereas, in other cases, e.g., when decisions need to be analyzed quickly, smaller models with less overlap are to be preferred.

Alternating between phases (rather than having a very long RD phase constructing a large pool followed by a single SC phase—which is similar to the process of supervised learning in Michigan-style systems followed by model compaction) allows to guide the optimizers better into more

interesting regions by giving them additional information about the current model behavior. In SupRB, when discovering new rules, the RD mechanism places new rules more likely into regions where the last model candidate from the SC process showed high prediction errors. This allows to fill unmatched areas (or areas of ill-fit local models) in the overall model that might otherwise go unnoticed or just never affected by pure randomness. Blindly fitting a multitude of rules would become quite expensive as the whole dataset needs to be matched and then batch-learned multiple times. In addition, it is hard to determine beforehand how many rules would be needed in the pool to later on find a satisfactory subset that solves the learning task.

As its transparency is a central aspect, SupRB's model is kept as simple and interpretable as possible [5]:

1. Rules' conditions use an interval-based matching: A rule k applies for example x iff $x_i \in [l_{k,i}, u_{k,i}] \forall i$ with l being the lower and u the upper bounds.
2. Rules' local models $f_k(x)$ are linear. They are fit using linear least squares with a l2-norm regularization (Ridge Regression) on the subsample matched by the respective rule.
3. When mixing multiple rules to make a prediction, a rule's experience (the number of examples matched during training and, therefore, included in fitting the submodel) and in-sample error are used in a weighted sum.

For the SC phase optimizer, we use a genetic algorithm (GA) for all experiments in this article, similar to [6] and [9]. It operates directly on a bitstring, where each element corresponds to whether a rule from the pool (if viewed as a list rather than a set) is part of this specific solution to the learning task or whether it is not. In previous work by [10], other metaheuristic optimizers were tested to compose solutions. They found that the choice of metaheuristic is—given sufficient computational budget—largely dependent on the learning task, although even there, differences were very small, which makes the GA a fitting general choice. As LCSs typically feature evolutionary computation methods, utilizing a GA is also the more traditional option. Tournament selection is used to select parents (in groups of two). An n -point crossover under a 90% probability recombines those parents, with n being set as part of hyperparameter tuning. The emerging children are then mutated using a probabilistic bit-flip mutation. Together with the fittest parents (*elitism*), these children form the new population. In our experiments, we used a population size of 5 elitists and 32 children. An individual's (solution to the learning task) fitness is determined based on its in-sample prediction error and its complexity (number of rules present) following [5]. Critically, this phase does not affect the position of

rules' bounds, which is in stark contrast to most previously proposed LCSs. Individuals can only be composed of rules in the pool and those rules remain unchanged by the GA's operators. Therefore, a specific bitstring will always express the same specific fitness during one training run, regardless of when it is evaluated during the training process.

The main subject of this article is the RD phase's optimizer, extending the analysis conducted in [6] by investigating additional algorithmic variants. In contrast to many typical optimization problems, we do not want to find a singular globally optimal rule but rather a set of localized rules that perform well in their particular feature space partition. Thus, we are attempting to find an unknown number of local optima without mapping the entire fitness landscape (or even all local optima). The discovered rules should be a diverse set to enable SC to compose a good overall model from them, which would be difficult from a set of very similar rules that do only cover a small part of the input space. However, as SC will select the subset of discovered rules most appropriate to solve the learning task, even heavily overlapping rules in the pool are not an issue. As described above for SC, a variety of different optimizers can achieve this. In the following, the different optimization approaches for RD that will be compared in the remainder of this article are presented. As we did previously [6], we build two baselines with a simple evolution strategy (cf. [5, 9]) and an even less sophisticated random search that only exploits information from the SC phase. We compare those with three different variants of novelty search, for each of which we test two substantially different ways of applying the archiving component (against which the novelty gets computed). For all heuristics, we utilize the same approach for calculating rule fitness by combination of two objectives, utilizing the in-sample error and the matched feature space volume, respectively (cf. [5]).

Evolution Strategy

Some form of evolutionary algorithm is the traditional choice for an LCS's optimization processes. Therefore, the first strategy employed for RD in SupRB, which was also used for the experiments in [10] and [9], is an *evolution strategy* (ES), specifically, a simplified $(1, \lambda)$ -ES [5].

The approach is summarized below and in Algorithm 1. The ES's initial individual is generated by selecting a random example from the training data around which a rule is placed, preferring those examples exhibiting a high in-sample error in the intermediate global solution. This individual serves both as the initial candidate for addition to the pool and the parent of the next generation. From this parent, we generate λ children with a non-adaptive mutation operator, which moves the upper and lower bounds further outwards by adding random values sampled from a half-normal

distribution. The child individual with the highest fitness becomes the parent for the next generation. If this individual showed a better fitness than the current candidate, it also becomes the new candidate. When for a fixed number of generations no new candidate has been found, the evolutionary search terminates. This ES produces one rule at a time, which allows it to be easily parallelized. One merit of this approach is in the explainability of both the search procedure and the resulting pool. In general, rules that have fitness independent from other rules are easier to understand for most non-experts (and even experts for more complex models). Whereas, in most current LCSs, the fitness assigned to each rule is highly dependent on the other rules it is surrounded by. Beyond these fitness-based considerations on the understandability of our learning process, the ES itself is also an easy to follow search method: expand the area (or hypervolume) an individual matches, evaluate the new individuals, choose the best new option and repeat.

Random Search

As an alternative to the strongly fitness-guided RD performed by the ES, we introduce a form of *random search* (RS). RS commonly serves as a baseline for testing the performance of other optimization algorithms. Furthermore, with the ulterior motive of finding diverse rules to add to the pool, RS provides an interesting approach where the fitness only plays a role in the selection of the final candidate but not in the generation of new rules.

In SupRB, RS (cf. Algorithm 2), similarly to ES, randomly selects a fixed number of data points with the probability of selection being weighted by their respective in-sample prediction errors in the last solution candidate (produced by the previous SC phase). We then place random bounds around those points based on half-normal distributions (to ensure we always match the selected point). To balance the computational cost between RD approaches, we produce

Algorithm 1 Rule Discovery with ES

```

1: procedure DISCOVER RULES(elitist)
2:   rules  $\leftarrow \emptyset$ 
3:   for  $i \leftarrow 1, n\_rules$  do  $\triangleright (1, \lambda)$ -ES for each new rule
4:     candidate, proponent  $\leftarrow$  INIT RULE(elitist)
5:     repeat
6:       children  $\leftarrow \emptyset$ 
7:       for  $k \leftarrow 1, \lambda$  do
8:         children  $\leftarrow$  children  $\cup$  MUTATE(proponent)
9:       end for
10:      proponent  $\leftarrow$  child with highest fitness
11:      if candidate's fitness < proponent's fitness then
12:        candidate  $\leftarrow$  proponent
13:         $j \leftarrow 0$ 
14:      else
15:         $j \leftarrow j + 1$ 
16:      end if
17:      until  $j = \delta$ 
18:      rules  $\leftarrow$  rules  $\cup$  candidate
19:    end for
20:    return rules
21: end procedure

```

substantially more rules initially than we would in an ES generation but a number roughly similar to the total number of children present in the ES. We then greedily select the rule(s) with the highest fitness to become part of the pool.

For our adaptation, we base the NS on a (μ, λ) -ES with elitism and follow the extensive experimental findings laid out by [13]. In each iteration, we select a list of λ parents out of the current population. These parents are paired, undergo

Algorithm 2 Rule Discovery with RS

```

1: procedure DISCOVER RULES(elitist)
2:   rules  $\leftarrow \emptyset$ 
3:   for  $i \leftarrow 1, n\_rules$  do
4:     candidates  $\leftarrow \emptyset$ 
5:     for  $k \leftarrow 1, \lambda$  do ▷ Randomly generate a large number of rules
6:       candidates  $\leftarrow$  candidates  $\cup$  INIT RULE(elitist)
7:     end for
8:     rules  $\leftarrow$  rules  $\cup$  candidate with highest fitness
9:   end for
10:  return rules
11: end procedure

```

Novelty Search

One of the central challenges of RD is that the optimizer’s objective is to find multiple rules that partition the feature space and, in their individually matched hypervolume, predict data points well. Contrastingly, optimizers operating on many typical optimization problems are expected to find a single global optimum (or at least a point very close to it). The RD’s primary objective can somewhat be viewed as the optimizer being tasked to map an unknown number of deeper local optima within the search space. In other words: with RD, we aim at finding a diverse set of well-performing rules for all areas of the feature space.

In this section, we describe a new approach towards discovering rules based on Lehman’s *novelty search* (NS) [11]. In this evolutionary search method, the optimizer tries to find individuals that exhibit new behavior previously unknown within the population, rather than being guided (or at least not being fully guided) by the typical fitness function. In SupRB, behavior of rules can be equated to what subsample of the training data they match, as we want to find a rule that predicts an area of the feature space currently unmatched or only matched by rules with high errors in this area. This makes NS a natural fit out of the field of quality-diversity optimization techniques. An application of MAP-elites [12], another common quality-diversity technique, is much less straight-forward in our use case as we are unaware where rules should be placed and what properties good rules should have before training was performed. Likely, we even want multiple rules with quite similar properties but located differently.

a uniform crossover and a half-normal mutation (cf. Section “[Evolution Strategy](#)”). The resulting children are then fitted and the best performing μ children are selected for the next population. In addition, we select a number of high-performing parents equal to the number of rules the NS is expected to produce within one RD phase as part of the new population (*elitism*). Performance of an individual can be based on novelty alone or on a combination of fitness (as used in the ES; cf. Section “[Evolution Strategy](#)”) and novelty, e.g., a linear combination.

For the novelty of a rule, we compare its match set and the match sets of the other rules. In this paper, we experiment with two different approaches at the selection of those other rules. Previously [6], we chose the rules in the pool and those in the current NS population, where we would compare with the other children for the selection or the parents for elitism, respectively. In the remainder of this paper, we refer to this as *NS-P*, with the *P* indicating that a comparison of past populations is only based on rules in the pool. In addition, we investigate another option called *NS-G*. In this approach, we use a more traditional archiving technique and compare with this archive and the current population (either children or parents, as previously). Initially during each RD phase, the entirety of the pool is copied into the archive. Therefore, *NS-G* automatically encompasses all comparisons made in *NS-P*. In addition, in each generation, ρ (which is subject to hyperparameter tuning) rules are selected from the children and put into this archive, giving *NS-G* its name. Thus, the archive grows with each generation and—in contrast to *NS-P*—*NS-G* is discouraged from exploring regions it has already explored in this RD phase. Note that, after an RD phase completes, the archive is reset to the then current

pool. This avoids a heavy computational load on one hand but is also more in line with the idea of independent optimization only based on the current solution to the problem. Would we account for previous RD phases, we might hinder the incorporation of information from this solution, as some region might have been touched but that rule not added to the pool as it was unimportant at the time but did become beneficial to further explore now.

The novelty score assigned to a rule is the average Hamming-distance between its match set and its k nearest neighbors' (most similar rules) from the respective comparison set. A value typically encountered for k with other NS applications in literature—e.g., [14]—is 15, although we tune between 10 and 20.

After a set number of iterations, we add a predefined number of rules from the current population to the pool and conclude this phase. Which rules get added can be randomized or based on the highest novelty(-fitness combination).

MCNS imposes additional pressure on the search to explore less vividly and focus more on rules that at least fulfill some minimal requirement. In our experiments, we set the minimal criterion to a minimum number (tuned between 5 and 15) of examples from the training data having to be matched by the rule to become viable, although we did also impose that at most one-fourth of the population should be removed because they missed the minimal criterion to prevent collapsing gene pools. We also use progressive minimal criteria novelty search [15], itself based on MCNS, as an option for combining fitness and novelty as the objective. Here, all individuals that do exhibit a fitness worse than the median fitness are removed automatically in each iteration of the search. This approach is not tied to MCNS and can be used in all three variants.

NSLC introduces a localized fitness-based pressure on the new generation. The idea is that, within a neighborhood (based on their behavior and not their position in the search

Algorithm 3 Rule Discovery with NS

```

1: procedure DISCOVER RULES(elitist) ▷ (based on  $\mu, \lambda$ )-ES
2:   population  $\leftarrow \emptyset$ 
3:   archive  $\leftarrow$  pool
4:   for  $i \leftarrow 1, \mu$  do
5:     population  $\leftarrow$  population  $\cup$  INIT RULE(elitist)
6:   end for
7:   for  $i \leftarrow 1, n\_iter$  do
8:     children  $\leftarrow \emptyset$ 
9:     parents  $\leftarrow$  SELECTION(population,  $\lambda$ ) ▷ Select  $\lambda$  parents
10:    for  $k \leftarrow 1, \lambda/2$  do
11:      child_1, child_2  $\leftarrow$  Crossover(parents[ $k$ ], parents[ $k + \lambda/2$ ])
12:      children  $\leftarrow$  children  $\cup$  MUTATE(child_1)  $\cup$  MUTATE(child_2)
13:    end for
14:    EVALUATE_FITNESS_AND_NOVELTY(children)
15:    best_children  $\leftarrow$  SELECT_BEST_CHILDREN(children,  $\mu$ )
16:    if NS-G then
17:      archive  $\leftarrow$  archive  $\cup$  SELECT_BEST_CHILDREN(children,  $\rho$ )
18:    end if
19:    best_parents  $\leftarrow$  SELECT_BEST_PARENTS(parents) ▷ Elitism
20:    population  $\leftarrow$  best_children  $\cup$  best_parents
21:  end for
22:  return  $n$  best rules from population
23: end procedure

```

In addition to the basic NS, we implemented and experimented with two variants: *minimal criteria novelty search* (MCNS) [11, 14] and *Novelty Search with Local Competition* (NSLC) [11]. For both variants, we also applied and benchmarked the ‘generational’ and ‘pool-only’ options for the novelty calculation.

space) of similar rules, the rules that exhibit high fitness should be chosen. A rule’s novelty score gets increased by a factor of b/κ , where b is the number of individuals within the neighborhood specified by κ that have a worse fitness than the rule currently evaluated. We tune κ in the same

Table 1 Overview of the regression datasets the eight rule discovery approaches for SupRB are compared on

Name	n_{dim}	n_{sample}
Combined cycle power plant (CCPP)	4	9568
Airfoil self-noise (ASN)	5	1503
Concrete strength (CS)	8	1030
Energy efficiency cooling (EEC)	8	768

range as k , as this does also specify a neighborhood of rules this rule is in competition with.

One disadvantage of NS-like approaches is that rule selection is no longer solely based on independent metrics (fitness) but rather on the independent fitness and the highly dependent (on other rules) novelty score.

Evaluation

To examine the differences between the rule discovery methods and to find the most versatile strategy, we evaluated those strategies within SupRB on several regression datasets.

Experiment Design

The experimental design of this article follows those of previous papers on SupRB [6, 9, 10]. SupRB is implemented¹ in Python 3.9, adhering to *scikit-learn* [16] conventions. The target is standardized and input features are transformed into the range $[-1, 1]$. While these transformations are reversible, they improve SupRB's training process as they help preventing rules to be placed in regions where no sample could be matched and remove the need to tune error coefficients in fitness calculations, respectively. Based on our assumptions about the number of rules needed, 32 cycles of alternating rule discovery and solution composition are performed, generating four (or 8 in case of NS variants) rules in each cycle for a total of 128 (256 for NS variants) rules. In addition, the GA is configured to perform 32 iterations with a population size of 32. To tune some of the more sensitive parameters, we performed a hyperparameter search using a Tree-structured Parzen Estimator in the Optuna framework [17] that optimizes average solution fitness on 4-fold cross-validation. We tuned the optimizers on each dataset independently for a fixed tuning budget of 360 core hours. The final evaluation, for which we report results in Sect. 4.2, uses 8-split Monte Carlo cross-validation, each with 25% of samples reserved as a validation set. Each learning algorithm is evaluated with 8 different random seeds for each 8-split cross-validation, resulting in a total of 64 runs.

¹ Recent code at: <https://github.com/heidmic/suprb>.

Table 2 Mean and standard deviation (over 64 runs, rounded to two decimal places) of MSE achieved by the eight RD approaches on the four datasets

	CCPP	ASN	CS	EEC
ES	0.07 ± 0.0	0.16 ± 0.03	0.14 ± 0.04	0.03 ± 0.01
RS	0.07 ± 0.0	0.23 ± 0.03	0.17 ± 0.03	0.06 ± 0.02
NS-G	0.07 ± 0.0	0.21 ± 0.02	0.15 ± 0.04	0.04 ± 0.01
MCNS-G	0.07 ± 0.0	0.19 ± 0.02	0.14 ± 0.05	0.04 ± 0.01
NSLC-G	0.07 ± 0.0	0.21 ± 0.02	0.15 ± 0.04	0.04 ± 0.01
NS-P	0.07 ± 0.0	0.22 ± 0.02	0.13 ± 0.02	0.04 ± 0.01
MCNS-P	0.07 ± 0.0	0.23 ± 0.02	0.13 ± 0.02	0.04 ± 0.01
NSLC-P	0.07 ± 0.0	0.20 ± 0.02	0.12 ± 0.02	0.04 ± 0.01

Best entry in each row (if one exists) marked in bold

As previously, we evaluate on four datasets part of the UCI machine learning repository [18]. An overview of sample size and dimensionality is given in Table 1. The combined cycle power plant (CCPP) [19, 20] dataset shows an almost linear relation between features and targets and can be acceptably accurately predicted using a single rule. Airfoil self-noise (ASN) [21] and concrete strength (CS) [22] are both highly non-linear and will likely need more rules to predict the target sufficiently. The CS dataset has more input features than ASN but is easier to predict overall. Energy efficiency cooling (EEC) [23] is another rather linear dataset, but has a much higher input features to samples ratio compared to CCPP. It should similarly be possible to model it using only few rules. Overall, these four datasets offer a mix of complex and more linear problems for different dimensionalities, therefore, enabling a more balanced testing whether the different RD techniques are really advantageous or just perform well on certain classes of problems.

Results

In the following, we abbreviate *SupRB using X as RD method* simply by *X*.

Table 3 Mean and standard deviation like Table 2 but for model complexity

	CCPP	ASN	CS	EEC
ES	4.52 ± 1.48	31.78 ± 2.21	24.95 ± 2.45	12.94 ± 2.10
RS	12.86 ± 3.13	32.98 ± 3.11	29.56 ± 2.47	30.17 ± 3.78
NS-G	7.58 ± 1.54	50.69 ± 5.83	40.83 ± 4.56	17.67 ± 3.33
MCNS-G	6.45 ± 1.30	42.45 ± 3.71	39.09 ± 3.65	19.17 ± 2.48
NSLC-G	7.58 ± 1.54	50.69 ± 5.83	40.83 ± 4.56	17.67 ± 3.33
NS-P	5.84 ± 0.91	18.22 ± 2.91	23.50 ± 3.47	6.92 ± 1.70
MCNS-P	6.02 ± 1.39	18.81 ± 2.46	22.38 ± 3.02	6.22 ± 1.13
NSLC-P	6.94 ± 1.41	29.75 ± 3.84	33.83 ± 3.28	10.16 ± 1.94

Best entry in each row (if one exists) marked in bold

Fig. 1 Violin plots of the (standardized) MSEs achieved in the 64 runs performed by the different RD methods on CCPP

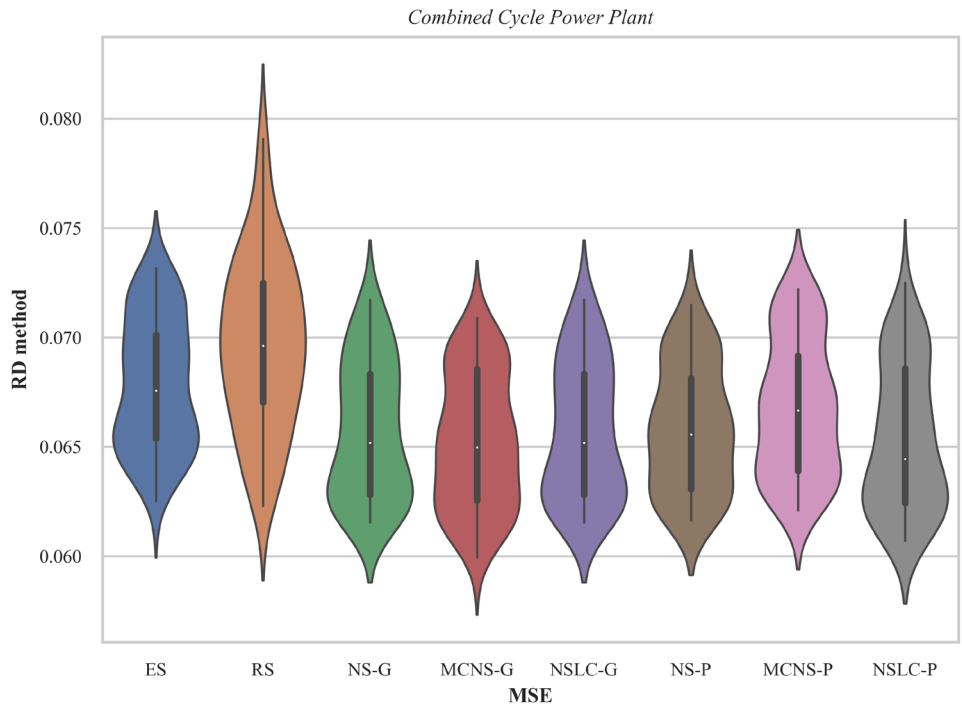
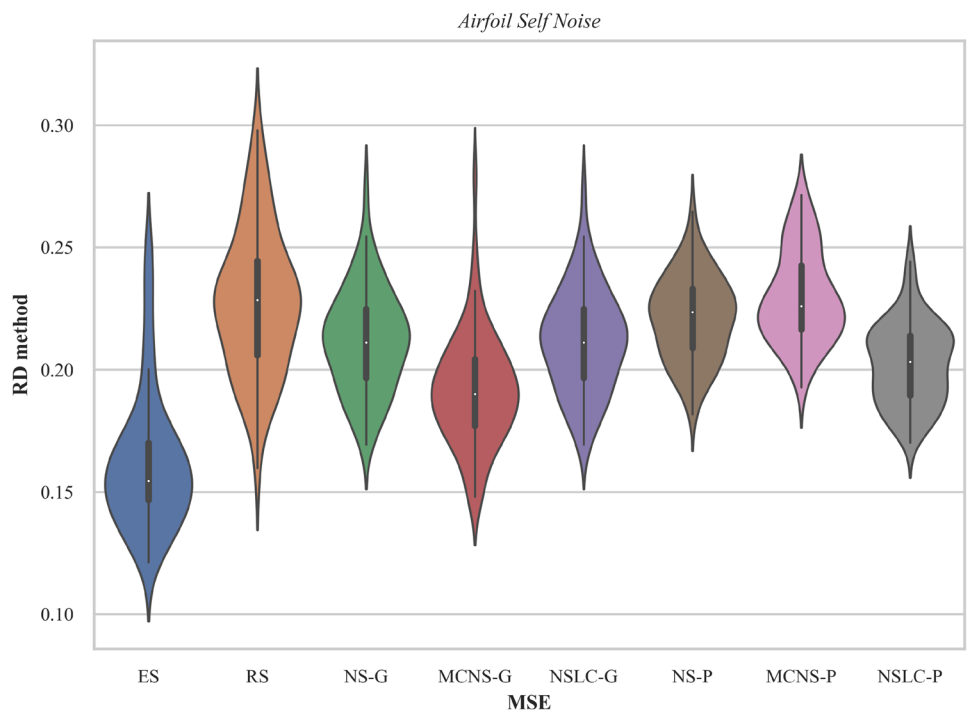


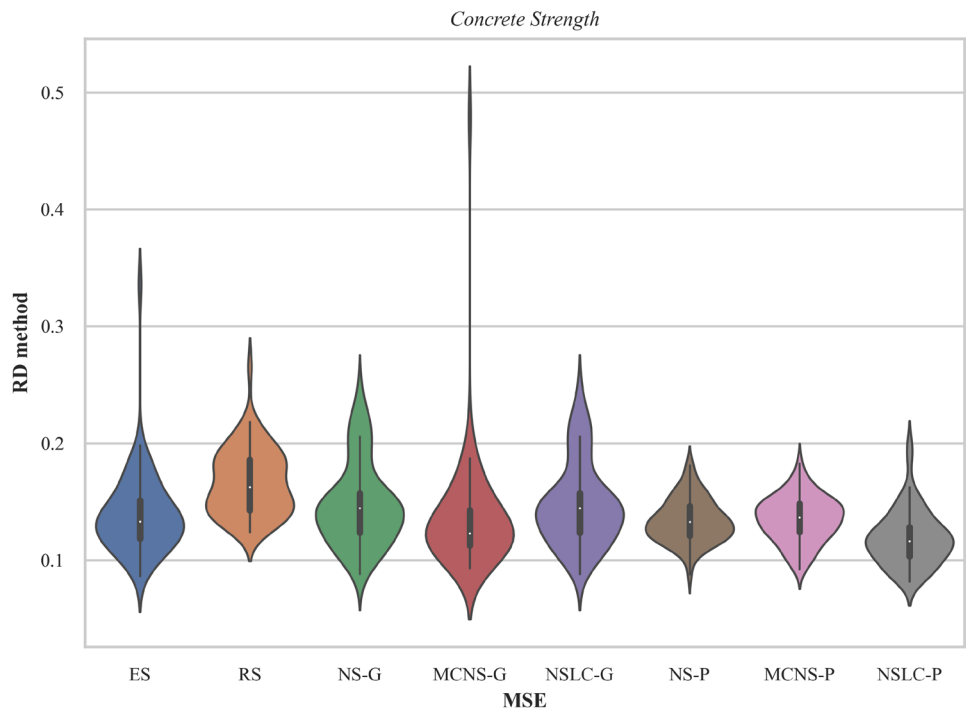
Fig. 2 Violin plots of the (standardized) MSEs achieved in the 64 runs performed by the different RD methods on ASN



Tables 2 and 3 give the means and standard deviations of model performances (measured using the standardized—individually per dataset—*mean squared error* on test data (MSE)) and model complexities (measured by the number of rules in the final elitist) achieved by the eight RD approaches when evaluated—as described in the previous section—on

the four real-world datasets. At a first glance, on all four datasets, RS shows the worst performance in terms of mean MSE and, with the exception of ASN and CS, the models it creates also have the highest model complexity. For ASN and CS, the models are still substantially more complex than the on average smallest models but about average

Fig. 3 Violin plots of the (standardized) MSEs achieved in the 64 runs performed by the different RD methods on CS



when compared to the other optimizers. The other optimization approaches vary in their results between datasets, with ‘pool-only’ novelty search variants showing a tendency for smaller models than their ‘generational’ counterparts.

In order to get a better overview, as well as include distributional information, we create violin plots to visualize the MSE results. For CCPP (Fig. 1), it can be seen that the

distributions of MSE values achieved look very similar for all optimizers, which is reinforced by identical values in the rounded mean and standard deviations given in Table 2. However, from a visual perspective, NSLC-P might be having a very slight edge (consider the y axis scale). Here, RS also seems to have a worse performance than the other optimizers. In terms of model complexity (Table 3), RS is clearly

Fig. 4 Violin plots of the (standardized) MSEs achieved in the 64 runs performed by the different RD methods on EEC

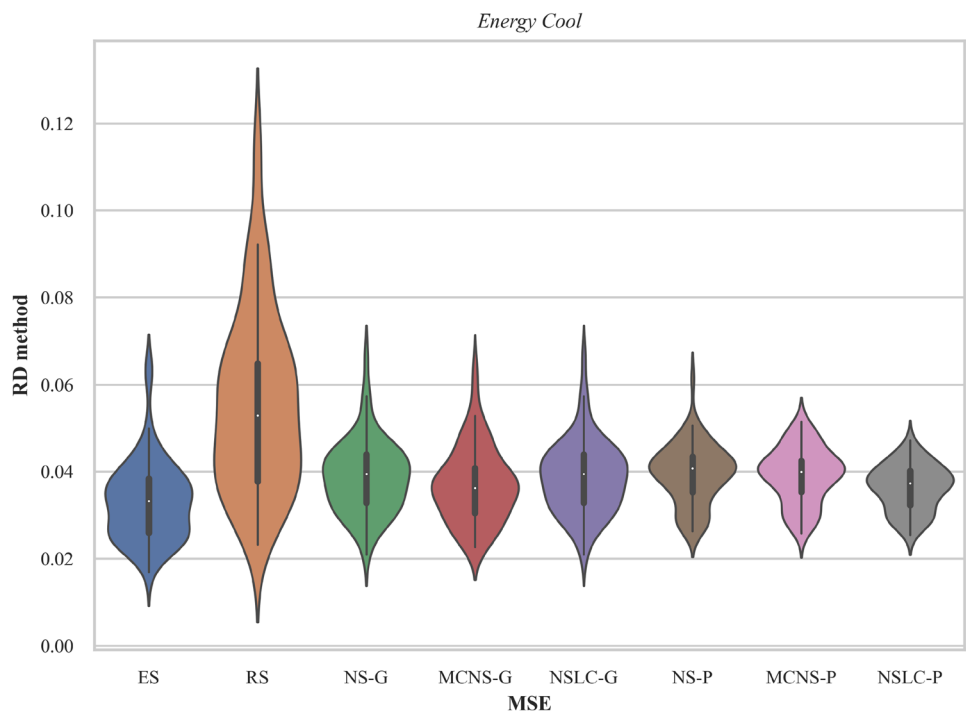


Fig. 5 Box plot (with standard 1.5 IQR whiskers and outliers) of the posterior distribution obtained from the model by Calvo et al. [25, 26] applied to the MSE data. An RD method having a probability value of q % says that the probability of that RD method performing the best with respect to MSE is q %

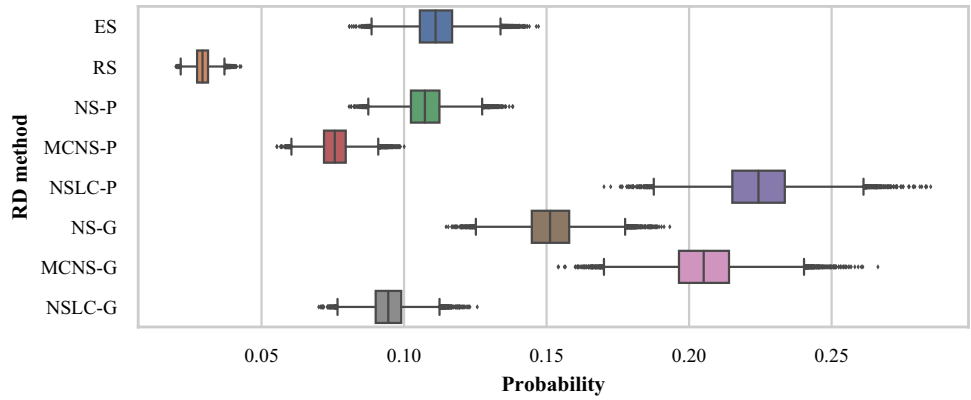


Fig. 6 Box plot like Fig. 5 but for the model complexity metric

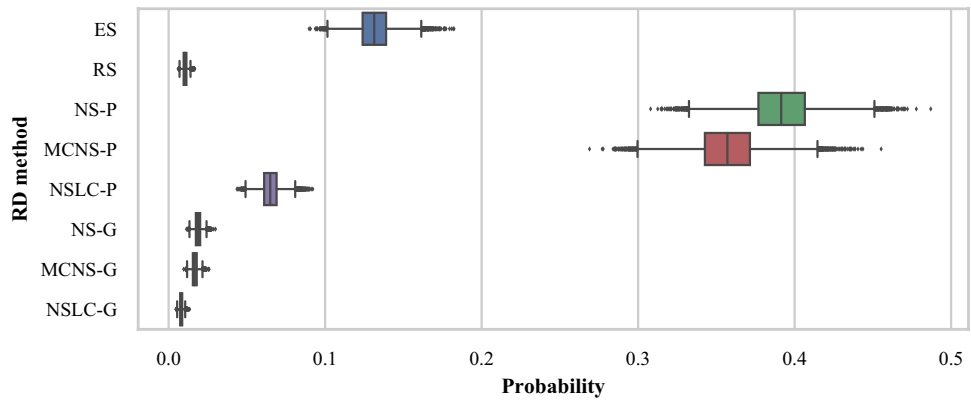


Fig. 7 Density plot of the posterior distribution obtained from Corani and Benavoli’s Bayesian correlated t -test [27] applied to the difference in MSE between NSLC-P and ES. Orange dashed lines and numbers indicate the 99 % HPDI (i.e., 99 % of probability mass lies within these bounds). HPDI bounds rounded to two significant figures. An earlier version of this figure was part of [6]

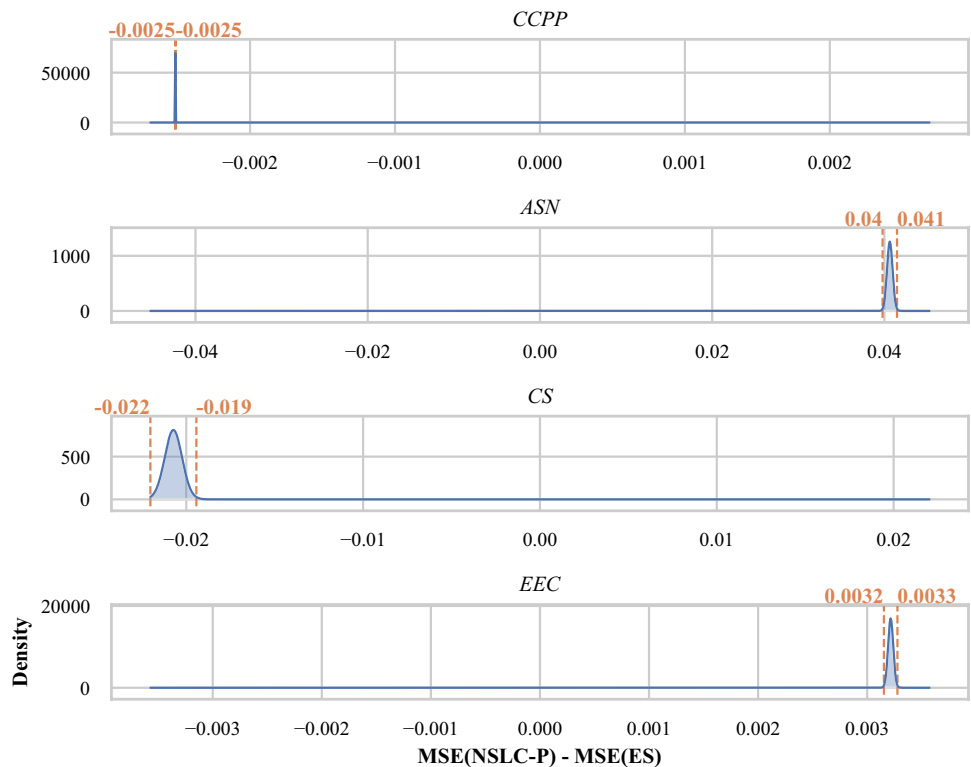
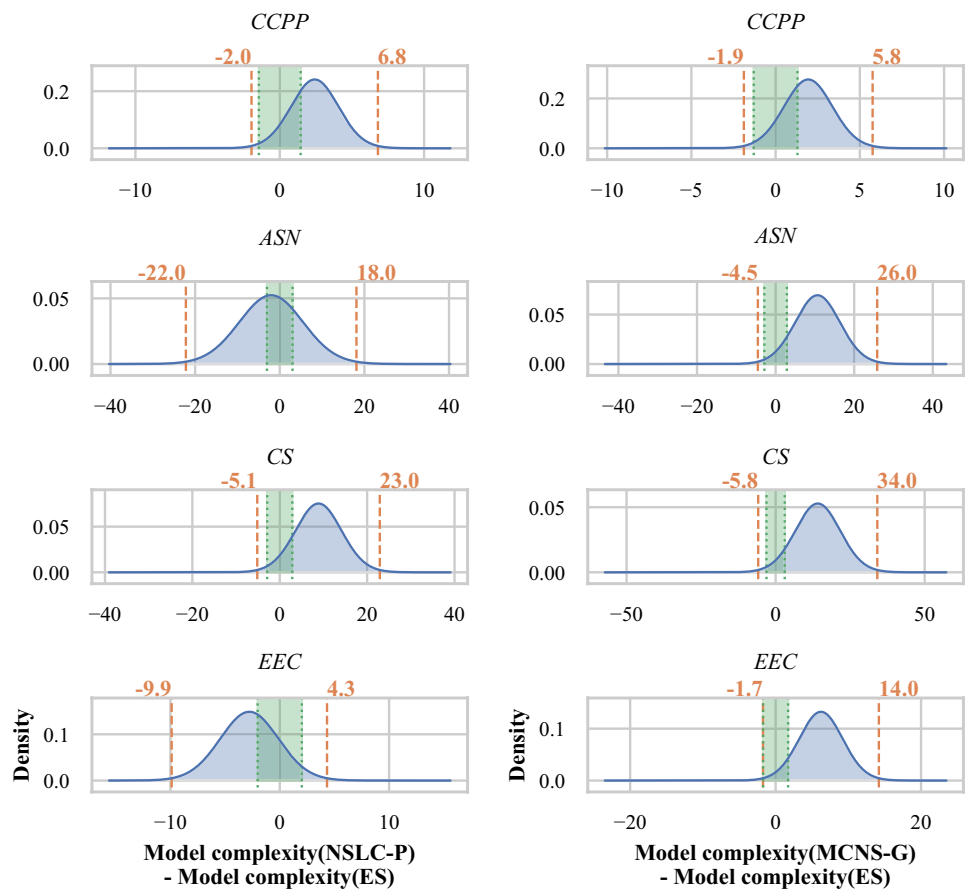


Fig. 8 Density plot like Fig. 7 but for the model complexity metric and with the region of practical equivalence (rope) marked by green dotted lines and a green area



(a) Comparing NSLC-P and ES. An earlier version of this figure was part of [6].

(b) Comparing MCNS-G and ES.

the worst approach, while ES substantially outperforms the others on average. The violin plot for the ASN dataset (Fig. 2) as well as means in Table 2 suggest that ES outperforms all other optimizers and that NSLC-P and MCNS-G are at least slightly better than the others. However, with respect to model complexity (Table 3), we observe that NS-P and MCNS-P find models of a low complexity, while the mean number of rules in the solutions found by NSLC-P, ES and RS is higher by more than ten, with the ‘generational’ archive variants being more than twice as complex than NS-P and MCNS-P. When regarding the CS dataset, Table 2 and Fig. 3 hint towards NSLC-P being the best of the considered methods, with all but RS performing similarly to each other but slightly worse than NSLC-P. In terms of model complexity (Table 3), ES, NS-P and MCNS-P are similar and result in much less complex solutions than the other optimizers. For the last dataset, EEC, ES is indicated again as the best performing method (Fig. 4). The model complexity, however, is slightly lower for NSLC-P and much lower for NS-P and MCNS-P, with the ‘generational’ archive variants being three times as complex as those two optimizers, while exhibiting almost identical errors.

Overall, the visual analysis combined with the rounded statistics of mean and standard deviation is not capable of providing us with a conclusive answer regarding which of the RD methods should be preferred on tasks like the ones considered. We thus investigate the gathered data more closely using Bayesian data analysis.²

We start by applying the model proposed by Calvo et al. [25, 26] to our data which, for each of the RD methods, provides us with the *posterior distribution over the probability of that RD method performing best*. We apply this model to both the MSE observations as well as the model complexity observations and provide box plots (Figs. 5 and 6) which show the most relevant distribution statistics. Both figures show that the locations of the distributions over the probabilities of performing the best (relative to a single

² We deliberately do not use null-hypothesis significance tests due to their many flaws and possible pitfalls—cf. e.g. [24].

Table 4 Probability (in percent, rounded to one decimal place) of ES or NSLC-P performing better or worse than the other—or practically equivalently—with respect to model complexity

	ES worse	Pract. equiv.	NSLC worse
CCPP	1.3	23.5	75.1
ASN	45.6	28.4	26.0
CS	1.4	12.4	86.2
EEC	65.6	29.6	4.9

metric) are rather close to each other. Therefore, the figures show that a very confident conclusive statement for a single RD method cannot be made given the data available.

When considering *model complexity* (Fig. 6), there is an about 75 %³ probability that one of NS-P and MCNS-P performs the best. These two algorithms are about as likely to build the smallest model (albeit, each at less than 40 % of the total probability mass which means that the probability of that candidate not being the best is above 60 %). However, since the probability of each of these *not* performing the best *MSE-wise* is 89 % or more, which is usually the more important metric, we posit to not consider them competitive enough to merit a closer analysis within the present work. Similarly, the very poor performance on complexity discards all the ‘generational’ archive variants, most notably, when considering not only the locations of the posterior distributions according to the Calvo model but also the raw numbers in Table 3. ES, the third most likely algorithm to produce the smallest model, is far from being a clear candidate but might at least be a better option, especially when accounting for the absolute numbers where it did perform at least average and even best on CCPP.

Looking at the MSEs, we find that, based on the existing data, the Calvo model does not support an automated decision. Furthermore, the Calvo model suggests that—except concerning RS—the optimizers do not considerably outrank each other at all. There might be slight tendencies towards NSLC-P, NS-G and MCNS-G, but even ES and the others behind it are not much less likely. Given the insights from the model complexities, which would discard NS-G and MCNS-G, this warrants a closer comparison of ES and NSLC-P.

We next consider the difference in performance between ES (since it is both the originally used RD method as well as the next-best runner up with respect to the probability of being the best MSE-wise and a good candidate for low model complexity) and NSLC-P (since it has the highest probability of having the lowest MSE) by applying Corani

and Benavoli’s Bayesian correlated *t*-test [27]. The resulting posteriors (given in Figs. 7 and 8a including 99 % high posterior density intervals (HPDIs)) are the *distribution of the difference between the considered metric for ES and the considered metric for NSLC-P* (practically, this equates to values below zero indicating ES having a higher (worse) metric value than NSLC-P). The difference in MSEs between NSLC-P and ES on the CCPP dataset is most certainly negligible in practice, which corroborates the earlier statement that all optimizers but RS perform similarly on that dataset. On the other hand, on ASN, ES outperforms NSLC-P by an MSE of *at least* 0.04 with a probability of over 99 %⁴ (99 % of probability mass lies between the dashed lines which are both considerably right of the center). The probability of ES being outperformed by NSLC-P on the CS dataset by an MSE of 0.019–0.022 is 99 %. On the last of the four datasets, EEC, we can, again, not really notice a significant difference (i.e., the two should be considered to perform equivalently with a probability of over 99 % if a difference in MSE of 0.0033 could be considered not practically relevant on this dataset). Recalling the information we have about the datasets, namely that EEC and CCPP are more linear (and, therefore, easier to solve) than the other two datasets, this is not too surprising. What we learn here is that, with the basic idea behind SupRB’s alternating phases and information sharing, we can solve such problems equally well, regardless of the specific optimizer employed. While an important lesson, it does not advance our effort towards a decision between ES and NSLC-P.

Things look differently when regarding the model complexities produced by the two approaches (Fig. 8a). We find that there is considerable probability mass on *each* side of 0 on all of the datasets, although the exact amount varies. While computing the integral to the right (or left) of 0 is possible, this would include differences which do not hold practical relevance. Thus, we opted for defining a *region of practical equivalence* (rope) centered around an equal complexity. To accommodate for the high difference of resulting model complexity among the learning tasks (i.e., they require different solution sizes), we define the rope to be task-dependent. Choosing this based on the task alone is difficult, as it would require detailed domain knowledge about the different processes that generated the data. Therefore, we let the collected data inform our choice for each task’s rope: We choose as the bounds of the rope the *mean of the standard deviations* of ES, NS-P, MCNS-P and NSLC-P (leaving out RS, NS-G, MCNS-G and NSLC-G since the results with

³ A typical minimal threshold for automated decision making is 80 % (or usually even more) for a single algorithm [24].

⁴ To elaborate: this states that we would expect that in over 99 % of future runs the difference of MSE between the two approaches was at least 0.04.

respect to solution complexity that these methods achieved are not even remotely competitive). In Fig. 8, a green area delimited by green dotted lines indicates the rope for each task. Based on the Bayesian correlated t -test model and the assumed ropes, we can now compute the probabilities for ES and NSLC-P performing practically equivalently or worse than the respective other. These can be found in Table 4. Upon close inspection, we can determine that, for ASN, the data is inconclusive. For CCPP and CS, the probabilities that ES produces less complex solutions than NSLC-P are 75.1 % and 86.2 %, respectively. There is *some* evidence hinting towards NSLC-P creating more compact solutions for EEC, although, this is, again, not fully conclusive, as the probability of it not being the case is 34.4 %.

We performed the same comparisons among ES, NS-P, MCNS-P and MCNS-G, finding largely similar results. Novelty search variants with a ‘pool-only’ archive outperform ES (or are outperformed by it) on the same datasets as NSLC-P was. The resulting plotted distributions look almost the same (with only variations in the scale). Therefore, these additional comparisons did also not yield any new insights. To illustrate why the Calvo model shown in Fig. 6 assigned MCNS-G such a low probability and what this means in practical terms when compared with ES, we show the distributions in Fig. 8b. ES clearly outperforms on complexity. On the corresponding MSE distribution, we found that, other than with ASN (where ES is better), none of the differences was practically significant, despite MCNS-G being the second most likely (admittedly, with a low probability) method to produce the best model (cf. Fig. 5).

We conclude our analysis with a summary of the key takeaways.

- On datasets like the ones we considered, NSLC-P has the highest probability of being the best option MSE-wise. However, the collected data is not at all conclusive with respect to that (there remains an around 77 % chance of NSLC-P *not* having the best MSE on such datasets).
- When considering model complexity alone, there is a high probability of either of NS-P or MCNS-P being the best—however, their MSEs are with a considerable probability not the best.
- The ‘generational’ archiving variants are contenders for the most likely best average MSE of a model, however, their model complexities fall considerably short of the better or even average performing RD methods.
- A closer comparison of ES with NSLC-P yielded:
 - ES outperforms NSLC-P MSE-wise with very high probability (> 99 %) by a presumably practically relevant amount on ASN, whereas this is the other way around on CS. However, on these two tasks, the data

are either not conclusive as to which method yields the better model complexities (ASN) or the method with the worse MSE performs better in that regard (CS).

- On the other two tasks, ES and NSLC-P perform practically equivalently MSE-wise. There is some light evidence for ES yielding better model complexities on the less difficult of these (CCPP) and for NSLC-P yielding better model complexities on the more difficult one (EEC).
- Similar close examinations among ES and the ‘pool-only’ novelty search variants did yield a very similar picture, albeit with slightly different raw numbers.
- Comparing with the most likely to rank best ‘generational’ archiving variant MCNS-G, we find clearly stronger complexities for the other RD methods and inconclusiveness on MSE data. Although, ES likely outperforms (MSE) practically significant on ASN, whereas the differences on the other three tasks cannot be deemed practically significant.
- Overall, we can say that, despite its simplicity when compared to the other approaches, the $(1, \lambda)$ -ES performs not worse when only considering *practically relevant differences*.

Conclusion

In this article, we extended a previous [6] investigation into the use of different evolutionary optimizers performing the task of rule discovery in the supervised rule-based learning system (SupRB), a recently proposed learning classifier system (LCS) for regression problems. This system’s key characteristic is the separation of finding a diverse pool of rules that fit the data well from their composition into a model that is both accurate and uses a minimal number of rules—a key advantage when employed in settings where model transparency is important, such as typical explainable artificial intelligence applications. By determining from intermediate solutions which areas of the problem space were already well covered, subsequent evolution can be guided towards other areas to form new rules there and, therefore, improve the overall diversity of the available rules.

The investigated methods were a $(1, \lambda)$ -evolution strategy (ES), a random search (RS) and three Novelty Search (NS) variants. For each of the three NS-based variants, we tested two differing approaches at building the archive against which the novelty is computed. The first was to only use the pool of rules available to solution composition in SupRB as the archive (denoted as *-P). In the second, we additionally archived a number of well-performing rules in each generation of the search heuristic (denoted as *-G). In total, this leads to a comparison of eight rule discovery approaches.

The NS-based variants all employed a (μ, λ) -ES, scored a rule's novelty on the basis of its match set and combined it with traditional fitness. Besides a traditional NS, we investigated minimal criteria novelty search (MCNS) and novelty search with local competition (NSLC).

After an evaluation of the eight methods on four real-world regression problems, we found that performances are close to equal in terms of prediction error, but—depending on the dataset—specific methods can be preferred, with ES and NSLC-P being the most interesting candidates. On the second important metric of complexity, these two methods perform about average but vastly outperform all of the 'generational' archive variants while obtaining competitive or better error scores. These findings were confirmed by a rigorous statistical analysis. Interestingly, we found that ES and NSLC-P reciprocally outperform each other on the considered learning tasks, i.e., where ES can be expected to perform better on error than NSLC-P, it can also be expected to yield larger solution sizes (a higher number of rules in the model) and vice versa.

Overall, our recommendation based on the gathered data is to employ either ES or NSLC-P, although, ES seems to be the preferential candidate for cases where explainability requirements of users consider model construction important, due to its greater algorithmic simplicity and the fact that rules are generated independently of the status of other rules.

Funding Open Access funding enabled and organized by Projekt DEAL. Partial financial support for this work was received from the Bavarian Ministry of Economic Affairs, Regional Development and Energy and the Deutsche Forschungsgemeinschaft (DFG).

Data availability Raw data used for this study is available at <https://archive.ics.uci.edu/>. The code of SupRB is available at <https://github.com/heidmic/suprb> while the code used to produce the experimental results of SupRB is available at <https://github.com/heidmic/suprb-experimentation>.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.


References

- Heider M, Stegherr H, Nordsieck R, Hähner J. Learning classifier systems for self-explaining socio-technical-systems. arXiv Accepted for Publication in the Journal of Artificial Life (2022). <https://doi.org/10.48550/ARXIV.2207.02300>.
- Urbanowicz RJ, Moore JH. Learning classifier systems: a complete introduction, review, and roadmap. *J Artif Evol Appl*. 2009.
- Heider M, Pätzel D, Stegherr H, Hähner J. In: Eddaly M, Jarboui B, Siarry P, editors. A metaheuristic perspective on learning classifier systems. Springer, Singapore, 2023. p. 73–98. https://doi.org/10.1007/978-981-19-3888-7_3.
- Heider M, Nordsieck R, Hähner J. Learning classifier systems for self-explaining socio-technical-systems. In: Stein A, Tomforde S, Botev J, Lewis P (eds) Proceedings of LIFELIKE 2021 Co-located with 2021 Conference on Artificial Life (ALIFE 2021) (2021). <http://ceur-ws.org/Vol-3007/>.
- Heider M, Stegherr H, Wurth J, Sraj R, Hähner J. Separating rule discovery and global solution composition in a learning classifier system. In: Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion) 2022. <https://doi.org/10.1145/3520304.3529014>.
- Heider M, Stegherr H, Pätzel D, Sraj R, Wurth J, Volger B, Hähner J. Approaches for rule discovery in a learning classifier system. In: Proceedings of the 14th International Joint Conference on Computational Intelligence—ECTA, SciTePress, Setúbal, Portugal; 2022. pp. 39–49. <https://doi.org/10.5220/0011542000003332>. INSTICC.
- Urbanowicz RJ, Browne WN. Introduction to learning classifier systems. 1st ed. Berlin, Heidelberg: Springer; 2017. <https://doi.org/10.1007/978-3-662-55007-6>.
- Wilson SW. Classifier fitness based on accuracy. *Evol Comput*. 1995;3(2):149–75.
- Heider M, Stegherr H, Wurth J, Sraj R, Hähner J. Investigating the impact of independent rule fitnesses in a learning classifier system. In: Mernik M, Eftimov T, Črepinšek M, editors. Bioinspired optimization methods and their applications. Cham: Springer; 2022. p. 142–56.
- Wurth J, Heider M, Stegherr H, Sraj R, Hähner J. Comparing different metaheuristics for model selection in a supervised learning classifier system. In: Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion) 2022. <https://doi.org/10.1145/3520304.3529015>.
- Lehman J. Evolution through the search for novelty. PhD thesis, University of Central Florida; 2012.
- Mouret J, Clune J. Illuminating search spaces by mapping elites. *CoRR*; 2015. [arXiv:1504.04909](https://arxiv.org/abs/1504.04909).
- Gomes J, Mariano P, Christensen AL. Devising effective novelty search algorithms: a comprehensive empirical study. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. GECCO '15, Association for Computing Machinery, New York, NY, USA; 2015. p. 943–950. <https://doi.org/10.1145/2739480.2754736>.
- Lehman J, Stanley KO. Revising the evolutionary computation abstraction: minimal criteria novelty search. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. GECCO '10, Association for Computing Machinery, New York, NY, USA; 2010. p. 103–110. <https://doi.org/10.1145/1830483.1830503>.
- Gomes J, Urbano P, Christensen AL. Progressive minimal criteria novelty search. In: Pavón J, Duque-Méndez ND, Fuentes-Fernández R, editors. Advances in artificial intelligence—IBERAMIA 2012. Berlin Heidelberg, Heidelberg: Springer; 2012. p. 281–90.

16. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. Scikit-learn: machine learning in Python. *J Mach Learn Res.* 2011;12:2825–30.
17. Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19*, Association for Computing Machinery, New York, NY, USA; 2019. 10/gf7mzz.
18. Dua D, Graff C. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences; 2017.
19. Kaya H, Tüfekci P. Local and global learning methods for predicting power of a combined gas & steam turbine. In: *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE*; 2012.
20. Tüfekci P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *Int J Electr Power Energy Syst.* 2014;60:126–40. 10/gn9s2h.
21. Brooks T, Pope D, Marcolini M. Airfoil self-noise and prediction. Technical Report RP-1218, NASA; 1989.
22. Yeh I-C. Modeling of strength of high-performance concrete using artificial neural networks. *Cement Concr Res.* 1998;28(12):1797–808. 0/dxm5c2.
23. Tsanas A, Xifara A. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy Build.* 2012;49:560–7. 10/gg5vzx.
24. Benavoli A, Corani G, Demšar J, Zaffalon M. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *J Mach Learn Res.* 2017;18(1):2653–88.
25. Calvo B, Ceberio J, Lozano JA. Bayesian inference for algorithm ranking analysis. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '18*, Association for Computing Machinery, New York, NY, USA; 2018. p. 324–325. <https://doi.org/10.1145/3205651.3205658>.
26. Calvo B, Shir OM, Ceberio J, Doerr C, Wang H, Bäck T, Lozano JA. Bayesian performance analysis for black-box optimization benchmarking. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '19*, Association for Computing Machinery, New York, NY, USA; 2019. p. 1789–1797. <https://doi.org/10.1145/3319619.3326888>.
27. Corani G, Benavoli A. A Bayesian approach for comparing cross-validated algorithms on multiple data sets. *Mach Learn.* 2015;100(2):285–304. <https://doi.org/10.1007/s10994-015-5486-z>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Michael Heider¹  · Helena Stegherr¹ · David Pätzel¹ · Roman Sraj¹ · Jonathan Wurth¹ · Benedikt Volger¹ · Jörg Hähner¹

✉ Michael Heider
michael.heider@uni-a.de

Helena Stegherr
helena.stegherr@uni-a.de

David Pätzel
david.paetzel@uni-a.de

Roman Sraj
roman.sraj@uni-a.de

Jonathan Wurth
jonathan.wurth@uni-a.de

Benedikt Volger
benedikt.volger@uni-a.de

Jörg Hähner
joerg.haehner@uni-a.de

¹ Organic Computing Group, Universität Augsburg, Am Technologiezentrum 8, Augsburg 86159, Germany