Comparing different Metaheuristics for Model Selection in a Supervised Learning Classifier System

Jonathan Wurth
Jonathan.Wurth@uni-a.de
Universität Augsburg
Augsburg, Germany

Michael Heider
Michael.Heider@uni-a.de
Universität Augsburg
Augsburg, Germany

Helena Stegherr
Helena.Stegherr@uni-a.de
Universität Augsburg
Augsburg, Germany

Roman Sraj
Roman.Sraj@uni-a.de
Universität Augsburg
Augsburg, Germany

Jörg Hähner
Joerg.Haehner@uni-a.de
Universität Augsburg
Augsburg, Germany

ABSTRACT

In the context of tasks that highly involve human interaction and expert knowledge, i.e., operator guidance in manufacturing, the possibility of decision verifications by the user is a key requirement for inspiring confidence in a system and its predictions. Rule-based Machine Learning offers one way to create such systems, with Learning Classifier Systems being a family of algorithms whose models are by design human-interpretable. Obtaining a rule base as compact and accurate as possible is a mandatory prerequisite to increasing comprehensibility, and metaheuristics such as Genetic Algorithms or Particle Swarm Optimization are powerful approaches to reducing the size of large rule bases. In particular, this paper will analyze five population-based metaheuristics and their ability to compose solutions (rule subsets) as part of a newly developed rule-based learning system, the Supervised Rule-based Learning System (SupRB). The experiments suggest that all metaheuristics can significantly reduce the complexity and filter out obstructive rules, increasing the prediction quality in the process.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by regression**; **Rule learning**; **Genetic algorithms**.

KEYWORDS

metaheuristics, genetic algorithms, rule-based learning, learning classifier systems, regression

1 INTRODUCTION

Learning Classifier Systems (LCSs) are a popular choice for obtaining a set of human-interpretable rules, but similar to other rule-based approaches they can suffer from creating more rules than are needed to model the problem sufficiently. This is especially the case in complex problems, where the number of rules is already high and understanding the (hypothetical) optimal rule set is non-trivial. If additionally many similar (redundant) rules are part of the rule set, e.g., because several independently operating techniques to generate rules were combined, or expert knowledge is incorporated into the training process, making even simple statements about the accuracy and usefulness of individual rules is increasingly difficult. Interpreting such rule sets in their entirety gets confusing very quickly, especially if mixing models are involved. Reducing the size of rule sets and removing unnecessary or even sub-par rules is, therefore, a sensible choice and directly correlates with an increase in interpretability and often accuracy.

Such approaches are known under the term *Rule Compaction* in the context of Michigan-style LCSs and mostly performed as post-processing [14]. The compaction itself is often performed deterministically based on estimated values or by applying heuristic procedures. Some Pittsburgh-style LCSs such as [3] perform an intermediate rule pruning step, albeit being similarly heuristic. In this paper, a new, more general perspective will be taken on this task: selecting a minimal subset of rules while maintaining other objectives like accuracy is a typical model selection [6], or model pruning problem, and such problems can be solved efficiently using metaheuristics.

An obvious metaheuristic to apply here is a Genetic Algorithm (GA), especially because the LCS is a system originally designed to take advantage of the GA's ability to evolve and enhance its individuals, or rules [10]. However, other metaheuristics like Particle Swarm Optimization (PSO) or Ant Colony Optimization (ACO) are similarly able to handle hard optimization problems, which includes selection tasks. Therefore, this paper specifically compares five different metaheuristics on *composing solutions* as part of the Supervised Rule-based Learning System (SupRB), which in general tries to generate a rule set (solution) as compact and accurate as possible. The metaheuristics considered here include a GA, ACO, PSO, Grey Wolf Optimizer (GWO), and Artificial Bee Colony Algorithm (ABC), which all have been demonstrated to be successful at solving similar selection tasks [e.g. 8, 19, 23].

2 MODEL SELECTION IN SUPRB

The fundamental concept of SupRB is to split the LCS's usual process of finding a maximally concise and accurate rule set into two subtasks: Rule Discovery (RD) and Solution Composition (SC).

The RD generates rules, or classifiers, which fit a part of the input space using some submodel, and inserts them into a global population of rules: the (rule) *pool*. To keep interpretability high, the rules simply consist of rectangular bounds and a linear submodel. By default, SupRB employs an Evolution Strategy (ES) to discover locally optimal rules, preferring to generate rules in parts of the input space where the in-sample error of the current global solution is the highest. The process of how exactly rules are generated is not the focus of this paper, and details regarding the RD and the exact structure of rules can be found in [9]. The rules in the pool are nevertheless assumed to have the following properties:

- Rules in the pool meet some minimum standard, i.e. they are at least accurate enough so that knowledge about the part of the input space they match can be extracted.
- Many similar rules are part of the pool.
- Rules may (greatly) overlap.
- They are not modified or removed from the pool.

SC is the second component, which selects a minimal and maximally accurate subset of rules from the pool, mixes them according to some mixing model, and thus creates a valid model of the whole space. This model is also referred to as a (global) *solution*, or in the context of population-based metaheuristics, as an *individual*. The component performs a special kind of model selection, where not explicitly enumerated models are considered, but all possible subsets of rules 2^P from the pool P. Encoding these subsets can easily be achieved by using a binary string, or genome, G, where a 1 at index i encodes that the rule at index i is part of the subset, and 0 encodes that it is not. The following string thus represents the subset of choosing the first, third and last rule in the pool:

$$G = 101000\ldots001 \tag{1}$$

As rules are only appended to the pool and existing rules are not modified, a solution vector stays valid by padding it with zeros.

A central problem in SC is that the exact number of rules required for an adequate solution is unknown, so there is no way to know when the global optimum is found. The overall goal of SC is therefore finding a good subset of rules, while simultaneously keeping the computational cost reasonable. This also motivated the alternating nature of RD and SC, which makes feedback on the quality of individual rules and the global solution immediate and simplifies the design process. As SC is not only performed at the end but also during the training process, knowledge about the global solution can then again be incorporated into the RD process, for example in the form of choosing regions of the input space that are not yet predicted sufficiently.

3 SELECTING SUBSETS OF THE POOL

The effective goal of SC is to minimize both the Mean Squared Error (MSE) and complexity C of an individual, i.e. the number of rules in the subset. The fitness function used in this paper was taken from [22] and combines two objectives o_1, o_2 in its base form, with α weighting the importance of o_1 against o_2 :

$$F(o_1, o_2) = \frac{(1+\alpha^2)\cdot o_1 \cdot o_2}{\alpha^2 \cdot o_1 + o_2}\,. \tag{2}$$

The first objective is the so-called Pseudo-Accuracy (PACC), which squashes the (possibly unbounded) MSE into the $(0, 1]$ domain with 1 corresponding to an MSE of 0:

$$o_1 = \mathrm{PACC} = \exp(-\mathrm{MSE}\cdot\beta)\,, \tag{3}$$

where $\beta = 1$ acts as control over the slope of the curve. The normalized complexity C_{norm} given by

$$o_2 = C_{\mathrm{norm}} = \frac{N_{\max} - C}{N_{\max}} \tag{4}$$

is used as second objective, where C is the complexity and $N_{\max}$ is the number of rules the pool will contain at the end of training. Both objectives are by definition within the interval $[0, 1]$ and maximizing the presented fitness function will minimize the MSE along with the complexity. The choice of $\alpha = 0.3$ was adopted from [22] and observed to allow selecting few rules for easy and many for harder problems, all while maintaining a sensible error. However, the influence of α is something that should be investigated in more detail in future research.

Only the GA [15] operates on a binary space by default, so the other metaheuristics are binary adaptations of either graph-based (ACO [21, 23]) or real-valued representations (GWO [8], PSO [13], ABC [11, 19]). Each metaheuristic contains some interchangeable components, or operators, which are chosen as part of the hyperparameter tuning on a dataset basis. They are additionally equipped with an external single elitist storage, which does not influence the population and only saves the individual with the highest fitness. This elitist is padded using zeros, while the individuals in the population are extended using uniformly random bits to increase diversity.

4 EXPERIMENTAL SETUP

SupRB and the five metaheuristics are implemented in Python 3.9, adhering to *scikit-learn*[1] [18] conventions[2].The evaluation is performed on four datasets part of the UCI Machine Learning Repository [7], namely the Combined Cycle Power Plant (CCPP) [12] dataset, Airfoil Self-Noise (ASN) [4], Concrete Strength (CS) [24], and Energy Efficiency Cooling (EEC) [20]. The input features are transformed into the range $[-1, 1]$, while the target is standardized.

SupRB performs 32 RD-SC cycles in total, generating four rules in each cycle for a total of 128 rules. The population size and number of iterations were similarly set to 32, respectively, as all three parameters were observed to have negligible influence after a sufficient size. The parameters of the RD-ES are tuned on every dataset beforehand, and the hyperparameters of the metaheuristics are similarly tuned on every dataset[2]. The hyperparameter tuning is done using a Tree-structured Parzen Estimator implemented in the Optuna framework[3] [1], calling the objective function consisting of 4-fold cross validation 128 times per tuning process.

[1] https://scikit-learn.org [2] The implementation of SupRB can be found at https://doi.org/10.5281/zenodo.6460701 and https://github.com/heidmic/suprb while more details on the experimental setup are featured at https://github.com/heidmic/suprb-experimentation/tree/GECCO2022_experiments.

[3] https://optuna.readthedocs.io/

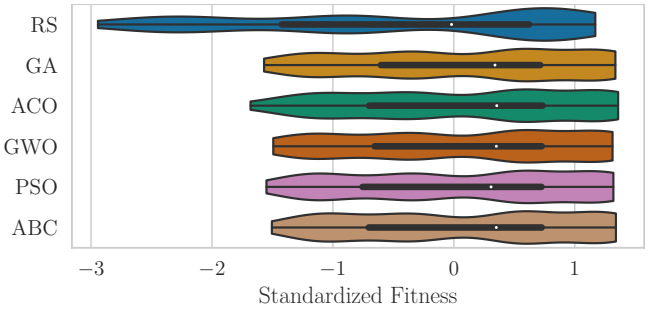

Figure 1: Distribution of the elitist fitness, standardized over all datasets. A 99% significance Wilcoxon signed-rank test has rejected the null hypothesis of equivalent fitness distributions of the random search and each metaheuristic, respectively.

The evaluation uses 8-split cross-validation with a random split of 25 % of the samples reserved as a validation set. Note that the tuning data is included here, as the metaheuristics optimize the training MSE and an overly optimistic validation error does not influence their evaluation. Each metaheuristic is evaluated eight times using different seeds for each 8-split cross-validation, resulting in a total of 64 runs. Although identical seeds are used throughout the experiments, incorporating the feedback of the SC in the process of RD means that different rules are generated in each run. In addition to the five optimizers, a Random Search (RS) is performed, finding rule subsets using an identical experimental setup.

5 RESULTS AND DISCUSSION

The experiments show that all five metaheuristics are able to compose good solutions and are very similar in their performance. The metrics considered for evaluation are the training error (MSE), complexity, and the fitness of the elitist solution. The mean and standard deviation of the MSE and complexity for each optimizer and problem are listed in Table 1. Note that the metaheuristics optimize only the fitness and decrease PACC and C_{norm}, i.e., training error and complexity, indirectly. A higher MSE or lower complexity for elitists with lower fitness is therefore caused by the influence of the other objective, respectively.

Figure 1 presents the distribution of elitist fitness values, standardized over all datasets. It can be observed that all metaheuristics outperform the RS and this is equally confirmed by a 99% significance Wilcoxon signed-rank test comparing the fitness distribution of the RS and each metaheuristic, respectively. On all problems and metaheuristics, the null hypothesis of identical distributions was rejected. A similar test with the GA as a baseline has indicated no significance for either hypothesis, which is consistent with the visual intuition gained from the distribution plots. The specific performances vary from dataset to dataset, but no general trend can be attributed. The ABC algorithm repeatedly shows the lowest complexity independent of fitness value, which might indicate a slight bias towards selecting fewer rules. This phenomenon should be investigated in more experiments.

The advantages of metaheuristics over simple stochastic methods are especially apparent on the more difficult problems, while they only show a slightly better performance on CCPP and EEC, both in terms of MSE and complexity. On ASN and CS on the other hand, all optimizers were able to significantly reduce the number of rules, but only the metaheuristics were able to select ∼1–6 fewer rules and achieve a difference in average MSE of up to 0.07, which corresponds to an increase of about 0.07 R^2 on the validation set (0.752 for RS vs. 0.821 for PSO, on average.). This difference is only expected to increase for larger problems. It is also a sign that evaluating a total of $32 \cdot 32 \cdot 32 = 32,768$ individuals over the whole training process is more than sufficient for the lower-dimensional datasets considered here, which can be interpreted as a guideline for future experiments.

The experiments have shown that there is no significant difference in choosing another metaheuristic over the GA in terms of performance, at least for these small datasets. Vice versa, the GA is not inherently superior to the other metaheuristics on the task of SC. This means that other properties can be considered in the metaheuristic selection process, such as computational effort needed or sensibility of parameters on different datasets.

A negative example in this regard is the ACO, as constructing a solution using intermediate information such as heuristic values of rules already selected is much more computationally expensive than, e.g., selecting, combining, and mutating individuals. This heuristic approach may only be profitable for larger problems, but the runtime complexity is also expected to scale exponentially with problem size, making this an even bigger issue. One possibility to bypass this is to scale down the number of individuals evaluated per RD-SC cycle, which is something the experiments suggest should still be sufficient. Regarding sensitivity of parameters, especially GWO has established itself to be very robust, achieving competitive performance using almost identical configurations on all datasets. Also, while overall some metaheuristic components seem to dominate others[4], the choice of parameters is not always as simple. Especially ACO and PSO define many real-valued parameters that highly influence solution quality, so GA, GWO and ABC are a better choice if no parameter tuning can be performed, as all of them show a trend towards robust parameters across datasets.

6 RELATED WORK

The task of what is labeled "Solution Composition" in this paper is not a new problem and can be found under various names throughout the field of LCSs and ML in general. The previously mentioned *Rule Compaction* is, similarly to SC, meant to remove redundant or incorrect rules from a greater rule set, typically the population of a Michigan-style LCS. A key difference to the approach proposed in this paper is that these strategies employ heuristics specific to the particular LCS architecture they were developed for and focus much more on maintaining an identical level of performance. A comparison of such algorithms can, for example, be found in [14].

The metaheuristic approach of composing a solution out of a large rule base is conceptually very similar to selection or pruning tasks, e.g., selecting subsets of input features or pruning ensembles using metaheuristics. They all share the goal of minimizing the number of rules/features/learners while maintaining or even

[4] More details on these circumstances can be found in the experiment setup at https://github.com/heidmic/suprb-experimentation/tree/GECCO2022_experiments.

Table 1: Overview of the experimental results. Every optimizer is evaluated on 64 runs per dataset, with the training MSE calculated on the standardized target. The results are rounded to three decimals and the best values are highlighted in bold.

	CCPP		ASN		CS		EEC	
	MSE	Complexity	MSE	Complexity	MSE	Complexity	MSE	Complexity
RS	0.064 ± 0.001	2.891 ± 0.799	0.204 ± 0.019	30.188 ± 7.546	0.122 ± 0.011	27.016 ± 5.954	0.030 ± 0.004	14.766 ± 3.745
GA	**0.063 ± 0.001**	2.656 ± 0.623	0.140 ± 0.014	26.422 ± 2.487	0.093 ± 0.011	22.312 ± 2.624	0.020 ± 0.003	12.812 ± 1.726
ACO	**0.063 ± 0.001**	2.609 ± 0.581	0.139 ± 0.012	26.547 ± 2.468	0.094 ± 0.011	22.453 ± 2.189	**0.018 ± 0.003**	13.016 ± 2.020
GWO	**0.063 ± 0.001**	2.656 ± 0.623	**0.136 ± 0.011**	29.781 ± 3.856	**0.092 ± 0.011**	24.609 ± 3.063	0.019 ± 0.003	13.391 ± 2.628
PSO	**0.063 ± 0.001**	2.594 ± 0.555	0.139 ± 0.010	27.094 ± 2.736	0.094 ± 0.010	22.969 ± 2.330	0.020 ± 0.003	12.594 ± 2.083
ABC	**0.063 ± 0.001**	**2.578 ± 0.558**	**0.136 ± 0.009**	**26.125 ± 2.728**	0.096 ± 0.010	**21.359 ± 2.081**	0.020 ± 0.003	**11.562 ± 1.670**

increasing accuracy. Diao and Shen [5] evaluate several nature-inspired metaheuristics such as a GA and ACO on feature selection and find that all are capable of finding good quality solutions. SC is also naturally an instance of model selection [6].

Several rule-based learning systems exist that employ metaheuristics other than a GA to evolve their rule sets, one of the most commonly known ones being the AntMiner [17]. It uses ACO to find optimal and simultaneously minimal rules and can loosely be interpreted as a Michigan-style LCS. Furthermore, several hybrid rule-based learning systems utilizing Michigan- and Pittsburgh-style phases with different metaheuristics in each phase can be found in the literature [e.g. 2, 16].

7 CONCLUSION

This paper has investigated the applicability of different metaheuristics for constructing solutions, i.e., rule subsets, as a central component in the newly developed Supervised Rule-based Learning System (SupRB). Specifically, five metaheuristics, namely a GA, ACO, GWO, PSO, and ABC were adapted to the task of Solution Composition (SC). They were evaluated in four experiments, using real-world datasets from the UCI ML Repository. The results have shown that all metaheuristics are capable of filtering redundant and unsuitable rules, significantly reducing the overall solution complexity and boosting prediction quality. This enables the use of other criteria than performance to guide the metaheuristic selection process, and a brief intuition was given in this regard. The methods introduced in this paper are specifically not only applicable to LCS, but any interval-based rule set regardless of origin, which opens up the possibility of implementation in other rule-based systems.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. 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, 2623–2631. https://doi.org/10/gf7mzz

[2] Sarab AlMuhaideb and Mohamed El Bachir Menai. 2014. HColonies: A New Hybrid Metaheuristic for Medical Data Classification. *Applied Intelligence* 41, 1 (July 2014), 282–298. https://doi.org/10/f58ghc

[3] Jaume Bacardit. 2004. *Pittsburgh Genetic-Based Machine Learning in the Data Mining Era: Representations, Generalization, and Run-Time.* Ph. D. Dissertation. PhD thesis, Ramon Llull University, Barcelona.

[4] Thomas Brooks, Dennis Pope, and Michael Marcolini. 1989. Airfoil Self-Noise and Prediction. (Aug. 1989).

[5] Ren Diao and Qiang Shen. 2015. Nature Inspired Feature Selection Meta-Heuristics. *Artificial Intelligence Review* 44, 3 (Oct. 2015), 311–340. https://doi.org/10/gn732t

[6] Jie Ding, Vahid Tarokh, and Yuhong Yang. 2018. Model Selection Techniques: An Overview. *IEEE Signal Processing Magazine* 35, 6 (Nov. 2018), 16–34. https://doi.org/10/gf8ck7

[7] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[8] Eid Emary, Hossam M. Zawbaa, and Aboul Ella Hassanien. 2016. Binary Grey Wolf Optimization Approaches for Feature Selection. *Neurocomputing* 172 (Jan. 2016), 371–381. https://doi.org/10/gmf2xg

[9] Michael Heider, Helena Stegherr, Jonathan Wurth, Roman Sraj, and Jörg Hähner. 2022. Separating Rule Discovery and Global Solution Composition in a Learning Classifier System. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion)*. https://doi.org/10.1145/3520304.3529014

[10] John H. Holland. 1992. Genetic Algorithms. *Scientific American* 267, 1 (1992), 66–73. https://doi.org/10/bmbqnb

[11] Dervis Karaboga and Bahriye Basturk. 2007. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization* 39, 3 (2007), 459–471. https://doi.org/10/c25dm6

[12] Heysem Kaya and Pinar Tufekci. 2012. Local and Global Learning Methods for Predicting Power of a Combined Gas & Steam Turbine.

[13] James Kennedy and Russell C. Eberhart. 1997. A Discrete Binary Version of the Particle Swarm Algorithm. In *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 5. 4104–4108 vol.5. https://doi.org/10/b7gp8s

[14] Yi Liu, Will N. Browne, and Bing Xue. 2021. A Comparison of Learning Classifier Systems' Rule Compaction Algorithms for Knowledge Visualization. *ACM Transactions on Evolutionary Learning and Optimization* 1, 3 (Aug. 2021), 10:1–10:38. https://doi.org/10/gn8gjt

[15] Seyedali Mirjalili. 2019. Genetic Algorithm. In *Evolutionary Algorithms and Neural Networks: Theory and Applications*, Seyedali Mirjalili (Ed.). Springer International Publishing, Cham, 43–55. https://doi.org/10.1007/978-3-319-93025-1_4

[16] Bijaya Kumar Nanda and Satchidananda Dehuri. 2020. Ant Miner: A Hybrid Pittsburgh Style Classification Rule Mining Algorithm. *International Journal of Artificial Intelligence and Machine Learning (IJAIML)* 10, 1 (2020), 45–59. https://doi.org/10/gn4t23

[17] Rafael Parpinelli, Heitor Lopes, and Alex Freitas. 2002. An Ant Colony Algorithm for Classification Rule Discovery. In *Data Mining*. IGI Global, 191–208. https://doi.org/10/cvhd4d

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, and Vincent Michel. 2011. Scikit-Learn: Machine Learning in Python. *The Journal of Machine Learning Research* 12 (Nov. 2011), 2825–2830.

[19] Clodomir J. Santana, Mariana Macedo, Hugo Siqueira, Anu Gokhale, and Carmelo J. A. Bastos-Filho. 2019. A Novel Binary Artificial Bee Colony Algorithm. *Future Generation Computer Systems* 98 (Sept. 2019), 180–196. https://doi.org/10/gnxb6q

[20] Athanasios Tsanas and Angeliki Xifara. 2012. Accurate Quantitative Estimation of Energy Performance of Residential Buildings Using Statistical Machine Learning Tools. *Energy and Buildings* 49 (2012), 560–567. https://doi.org/10/gg5vzx

[21] Youchuan Wan, Mingwei Wang, Zhiwei Ye, and Xudong Lai. 2016. A Feature Selection Method Based on Modified Binary Coded Ant Colony Optimization Algorithm. *Applied Soft Computing* 49 (2016), 248–258. https://doi.org/10/f9knrr

[22] Qing Wu, Zheping Ma, Jin Fan, Gang Xu, and Yuanfeng Shen. 2019. A Feature Selection Method Based on Hybrid Improved Binary Quantum Particle Swarm Optimization. *IEEE Access* 7 (2019), 80588–80601. https://doi.org/10/gnxcfb

[23] Yunshuang Xiao and Shuyu Chen. 2020. RBFACO: A New Feature Selection Algorithm. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2884–2890. https://doi.org/10/gnfbnv

[24] I-Cheng Yeh. 1998. Modeling of Strength of High-Performance Concrete Using Artificial Neural Networks. *Cement and Concrete Research* 28, 12 (Dec. 1998), 1797–1808. https://doi.org/10/dxm5c2