Towards self-explaining assistance systems in tomorrow's factories

Michael Heider

Angaben zur Veröffentlichung / Publication details:

Heider, Michael. 2024. "Towards self-explaining assistance systems in tomorrow's factories." In Organic Computing - Doctoral Dissertation Colloquium 2023, edited by Sven Tomforde and Christian Krupitzer, 84–94. Kassel: kassel university press. https://doi.org/10.17170/kobra-202402269661.

Intelligent
Embedded Systems

26

S. Tomforde | C. Krupitzer (Eds.)

Organic Computing

Doctoral Dissertation Colloquium 2023

Organic Computing

kassel
university
press

Intelligent
Embedded Systems

Band 26

Herausgegeben von
Prof. Dr. Bernhard Sick, Universität Kassel

Organic Computing

Doctoral Dissertation Colloquium 2023

S. Tomforde, C. Krupitzer (Editors)

Bibliographic information published by Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at http://dnb.dnb.de.

ISBN 978-3-7376-1169-5
DOI: https://doi.org/10.17170/kobra-202402269661

kassel university press

Printing Shop: Print Management Logistik Service, Kassel
Printed in Germany

7

Towards Self-Explaining Assistance Systems in Tomorrow's Factories

Michael Heider

Universität Augsburg, Am Technologiezentrum 8, 86159 Augsburg, Germany
michael.heider@informatik.uni-augsburg.de
✉0000-0003-3140-1993

Abstract. With the advent of highly digitized manufacturing environments, more options towards automation become available. While, previously, operators often adjusted settings manually (and directly in hardware), it becomes increasingly possible to make those adjustments at control panels semi-remotely. Additionally, machines are becoming more complex and more interdependent, increasing the demand on the operator.

To alleviate this, digital agents based on artificial intelligence methods should be employed. However, these agents might face mistrust if they are not sufficiently transparent in how predictions and forecasts are made.

In this paper, a doctoral project focussing on inherently explainable machine learning methods is motivated based on previous results regarding explainability requirements in real-world industrial settings.

Keywords: Explainable AI, Socio-technical Systems, Rule-based Learning, Evolutionary Computation.

7.1 From the Status Quo in Manufacturing to Intelligent Agents

One of the key issues of many current industrial manufacturing plants is the lack of consistent and digital data streams. In many cases, we see steps towards such setups, however, they are often hindered by data being distributed among multiple (often incompatible) databases with ERP and MES systems not synchronised or setup to create compatible data. As this and the general IoT idea have been well known for a few years now, we expect those issues to be resolved in the near future.

Concurrently, for the past few decades, we have seen trends towards more and more automated production. Demographic change and comparatively high wages are key push factors that drive managing stakeholders towards adopting more autonomously operating systems — at least for their European plants. However, to increase their efficiency further, the generated data should not only be used at specific points of a plant but rather be interconnected.

This raises the question of how operators will partake in these new production settings. Operators obviously still have to be specialists, but with fewer colleagues

they have to be responsible for multiple machines at once. Additionally, on-boarding can be more difficult. Processes are often complex to learn, which can take months to years, especially for edge cases, and with more responsibilities the mental load can be overwhelming as soon as multiple problems arise. While, for the foreseeable future, manufacturing will remain a socio-technical rather than a pure cyber-physical setting, operators should be assisted wherever possible.

Two key aspects of operator assistance are:

1. Increase their time to respond by warning them early that a system is leaving the acceptance space.
2. Assist the operators in making decisions towards solving a disturbance by creating more intelligent control mechanisms.

Operator Assisting Agents (OAAs) often interplay with one or many control mechanisms (CMs). While the CMs control parts of a machine or line or act in a hierarchical setup with high and low level CMs, currently, they are not operating alone but always in interplay with a human operator. While we can envision that CMs will be sufficiently advanced to operate fully autonomously in the future, we expect this to take quite some time, as with almost all industrial machines some adjustments have to be done in hardware and can not be made in software and some cases are quite complex, for which the required data has not been collected and the appropriate algorithms might not even have been developed. One approach towards constructing these partial or even fully autonomous CMs is found in Organic Computing (OC) [20]. This research direction aims at creating technical systems with favourable properties by taking inspiration from nature. Such properties can include robustness, flexibility, adaptivity, explainability and many more, although we typically expect these properties to emerge and be maintained during a systems lifetime without extensive human interaction or complicated a-priori system design. However, for now, we want to transfer workload from said operator towards our virtual agent, expanding the socio-technical systems, cf. [8], which can be argued to constitute systems with OC-like properties.

The key component of any OAA will be a model of the process on which predictions can be made or suggested actions to mitigate disturbances can be proposed. Ideally, such models are built from all available sources. If we think of machine learning (ML)–based OAAs — which will be the focus of this paper — they will primarily be trained on historic data. However, there are many other sources to build OAA models from, e.g. conceptual, tacit and procedural knowledge that has been extracted from operators or engineering handbooks [17, 22, 23]. These alternative sources often provide the data in the form of (if-then) rules, graphs, or ontologies. They can also be combined with historic data in neuro-symbolic or knowledge-infused learning [18] to build more advanced models. If the additional data is available in the form of rules these can also be integrated into rule set learning algorithms [25] to combine them with historic data and build especially reliable models (cf. the last paragraph of Section 7.2 for a more detailed discussion).

7.2 Transparent Agents and their ML internals

Regardless of their level of autonomy, CMs and OAAs are expected to be transparent decision makers that can reason why they performed certain actions or gave certain predictions [11], i.e. we expect them to be equipped with self-explaining properties. Stakeholders of all levels have a high interest in understanding what is happening on the floor as their boni and even their very positions in the company might depend on that. However, they are often of non-technical (or at least non–data science) backgrounds, which introduces new challenges towards explaining CM or OAA choices.

With the current state of explainable AI (XAI), we can safely assume that deep learning approaches are not able to fulfil our requirements for non–computer vision tasks[1]. Note that this paper uses the XAI-related terms, best defined by Barredo Arrieta et al. [2], somewhat liberally despite their slight nuances as the casual reader will not benefit from such an in-depth differentiation of closely related concepts[2].

Models that are based on rules are a type of model that is generally regarded as explainable [2]. These models learn *input space partitions* and respectively responsible *submodels* from data. They can easily be read by humans, which makes them ideal for post-hoc analysis. There are, however, some limitations to this, mostly based on the number of parameters in the models. This can simply be the number of rules but also how the rules are encoded and what submodels are used. It should be obvious that constant or linear submodels are likely easier to follow than multi-layer perceptrons. With decision trees, a type of rule-based model that ensures that rules do not overlap and many readers should be familiar with, the depth and leaf node count of the trees are very obvious limitations to the interpretability of these models.

One approach to learn a model that comprises a set of rules are rule set learning algorithms (RSLs [25]; also known as learning classifier systems or LCSs, however, going forward this term is not used as it is very commonly — and mistakenly — assumed to produce classification models, whereas for OAAs, we often deal with a variety of tasks, including regression). Their models are closely related to other rule-based models and should be thought of as generalizations of decision tree models [9]. While a large variety of RSLs has been proposed in the past — for an overview see, e.g. [10, 24, 26] — they all produce models of a similar type using metaheuristic optimization tools to solve the problem space partitioning problem. For an extensive discussion of the role of the optimizer(s) in RSLs, cf. Heider et al. [9]. Most importantly, two main tasks need to be performed during training, each of which can be seen as having two subtasks:

Model selection consists of

1. choosing an appropriate number of rules for the task at hand and

[1] It should be stressed that interpretability and self-explainability are hot topics and under active investigation by many larger players, e.g. OpenAI for their LLMs.

[2] For a more detailed view, readers are highly encouraged to read the original paper and differentiate the concepts of the work presented here.

2. assigning each rule a part of the input space (i. e. choosing *matching functions*).

The set of matching functions is also called the model's *model structure*.

Model fitting is concerned with selecting the best-performing submodels and combining them optimally, given a fixed model structure. It usually consists of:

3. fitting each submodel to the data its rule is responsible for. It is often beneficial to do this such that each submodel is trained independently of the others, since that way, upon getting new training examples or changing the model structure (model selection is typically done iteratively and alternately with model fitting), only partial retraining is necessary [7].
4. fitting the mixing weights[3] such that areas where the responsibilities of several rules overlap are dealt with optimally.

Assuming the usage of RSLs inside the OAAs, the next step is to determine the use case–specific explainability requirements. A template to assess them has been proposed and, subsequently, tested by Heider et al. [8, 11]. In this work, a variety of different stakeholders from an industry partner that wants to implement OAAs into their factory were interviewed. The evaluation of the interview results found that RSLs are a unique fit for the desired application. However, the model requirements are quite stern, with very small models being desired by most stakeholders. Some of the key points were:

- Rule bounds should be interval-based.
- Submodels should be constant or linear, with few coefficients being greater than zero.
- Only up to three rules should partake in inference for a specific input which influences mixing.
- The overall number of rules should be limited. Stakeholders were not unanimous on the size but 20 to 100 seem to be the very upper bound.

Interestingly, the actual model construction process, i.e. training, seemed less of interest than the analysis of the models and reasoning behind model outputs. One somewhat unexpected result of the interviews was that, besides the OAA application, these models will gladly be used by process engineers to garner deeper understanding of what is really happening on the machine (rather than the physics-inspired models known from theoretical research). Based on the findings, we extensively discussed how a practitioner can construct an RSL to build such models.

An additional advantage of RSLs (compared to deep learning approaches) is built on the additional sources of knowledge we might have in our factory setting (cf. the last paragraph of Section 7.1). As these might construct rules or data structures easily converted into rules, these could potentially be integrated into our trained model

[3] By relaxing the exclusivity assumption encountered in trees, RSLs allow — potentially multiple — rules to overlap. As each of these rules has their own submodel (the 'then' of our if-then rule), we need to apply a strategy to combine those. This is usually called *mixing* and often done via a weighted-sum with weights based on in-sample rule performance or other constant metrics.

rather seamlessly. We can use our model to validate such rules based on data or use them to cover edge cases or areas of the input space that are poorly sampled, i.e. our operators, process engineers, or handbooks often 'know' that some areas might lead to undesirable results or even damage our machine. Without these rules, a reinforcement learning–based agent might try to explore here and a supervised learning–based agent might show very noisy or even plain uninformed predictions in poorly sampled areas, which could lead a decision making system to propose a bad or potentially harmful machine parametrization from this area. Besides using the existing source of knowledge, we can also request our stakeholders to validate or invalidate certain rules that we received from training or to supply default behaviour for spaces without rules present (typically because there was no or little data).

RSLs also have a long history of being used in OC systems due to the aforementioned properties, cf. [3, 21]. This makes the work presented in this chapter especially relevant for that research community in that it might have a lot of use cases where currently the XCS Classifier System (XCS) [27] is used.

7.3 An RSL to Construct Better Explainable Models

Based on our foundational understanding on RSLs [9] and our knowledge on what requirements models should fulfil [11], we did, subsequently, design an RSL. We named it the Supervised Rule-based Learning System (SupRB) [16]. The key feature that differentiates it from other RSLs is that we independently search for good rule bounds instead of solving the problem of placing the bounds for all rules simultaneously, as it is for example done in XCS [27] or GAssist [1]. Thereby, we also separated the two subtasks of model selection (cf. Section 7.2), which is novel in the field of RSLs.

Figure 7.1 illustrates on a simulated one dimensional data set how SupRB produces a large set of rules on varying subsets of the data and then selects some of these rules for the final model. To improve convergence and potentially find even better solutions to the learning task, SupRB does not perform a singular search for rules and then compacts them (like many Michigan-style LCS would do) but alternates phases of rule discovery and solution composition. By performing alternating phases rule discovery can exploit knowledge from the last proposed solution. I.e. rules are more likely placed in areas of the feature space where the last solution showed poor training errors in. The training process thus can be described with a very simple state machine, cf. Figure 7.2. In its originally proposed version, we use an Evolution Strategy to discover rules and a Genetic Algorithm to compose subsets of rules into compact solutions to a learning task.

SupRB is a general (and provably universal[4]) function approximator. It is designed for offline batch learning and, while it was only tested on regression tasks [12–15, 29], it is easily extensible to solve classification tasks as well, following the remarks of Heider et al. [14, 15].

[4] Note that some assumptions need to be made for this and that runtimes converge to infinity.

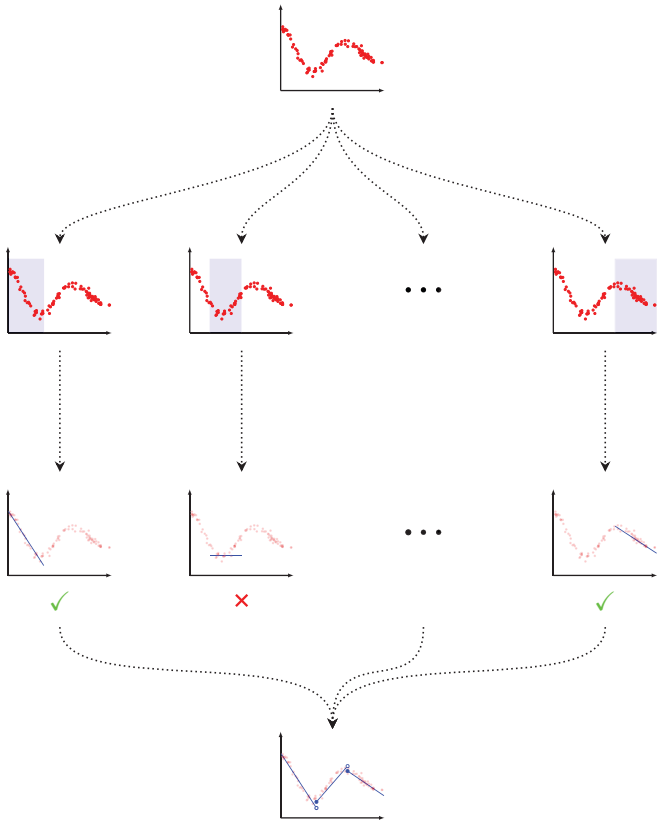

Fig. 7.1. Extracting a rule-based model from data by discovering appropriate areas of the input space to place linear models in, creating individual rules. Afterwards, a subset of these rules is selected (denoted by cross and checkmark) to compose the output model from. In the example, we utilize three rules and use a mixing model where only the fittest rule per area is used for predictions. This figure first appeared in [14].

We already proposed some extensions [12, 29] to the basic components of SupRB [16] and compared [15] the algorithm to the XCSF classifier system [28], which would be the most common choice for an RSL for function approximation. Later on, we extended the comparisons of [15] by also contrasting and benchmarking SupRB to the most common rule-based machine learning approaches: Decision Trees and Random Forests, which are ensemble models composed from a multiple (often hundreds) Decision Trees [14]. We expected SupRB to perform similarly (in regards to mean prediction errors) to XCSF, albeit with smaller rule set sizes. We assumed that it would perform better than Decision Trees in at least prediction errors and worse than Random Forests on that metric but that SupRB would staunchly beat Random

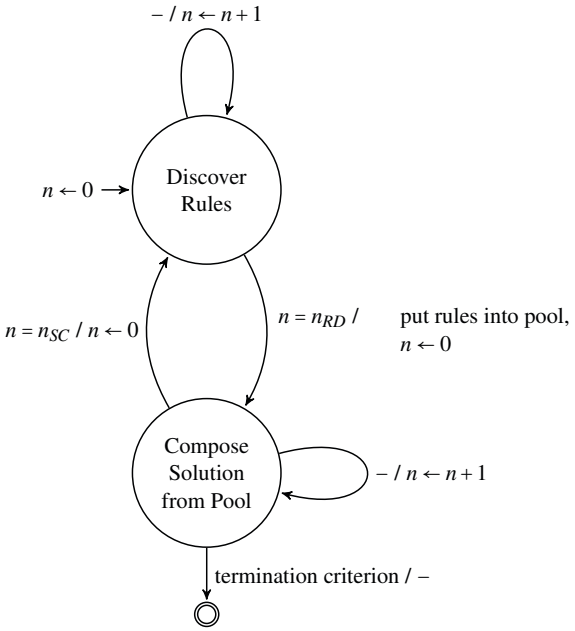

Fig. 7.2. Rule discovery and solution composition alternation SupRB. n_{SC} is the number of iterations for the solution composing optimizer and n_{RD} is the number of iterations performed for rule discovery. This figure first appeared in [16].

Forests in regards to the number of parameters (and therefore rules) contained in the model. After performing a large set of experiments and a very rigorous statistical analysis based on Bayesian statistical modelling, e.g. Corani and Benavoli's Bayesian correlated t-test [6], we found that our assumptions were largely correct. To demonstrate the explainability of a SupRB model, we included the extensive discussion of an exemplary rule into this article. We discussed how to read a SupRB rule based on a tabular presentation and how to interpret its parameters.

In [29], my student Jonathan Wurth investigated whether other metaheuristics than the previously used Genetic Algorithm should be used in the solution composition component of SupRB. For this, we implemented some of the most common metaheuristic optimizers: Ant Colony Optimization, Artificial Bee Colony, Particle Swam Optimization. Additionally, we included a combinatorial variant of the Gray Wolf Optimizer and a simple Random Search into the benchmarking process. We found that, while some optimizers can find better solutions on some learning tasks, there is no general trend beyond reasonable doubt (based on statistical analyses) on which optimizer should be used for solution composition (other than not using Random Search). Therefore, we decided to stick with the Genetic Algorithm for all subsequent work as it is the traditional choice for RSLs.

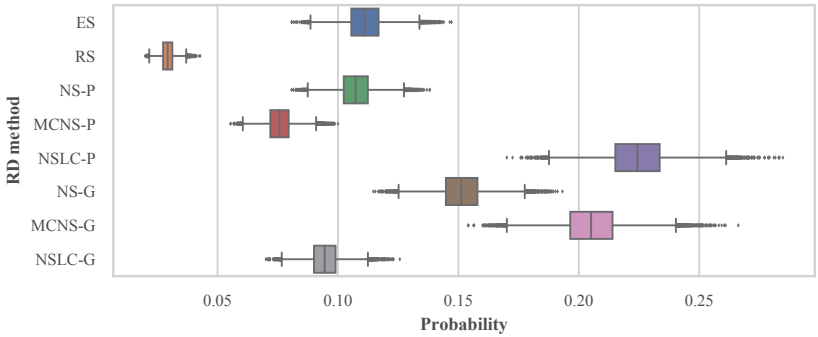

Fig. 7.3. Box plot (with standard 1.5 IQR whiskers and outliers) of the posterior distribution obtained from the model by Calvo et al. [4, 5] applied to the MSE data. An RD method having a probability value of q % says that the probability of that RD method performing the best with respect to MSE is q %. This figure first appeared in [13].

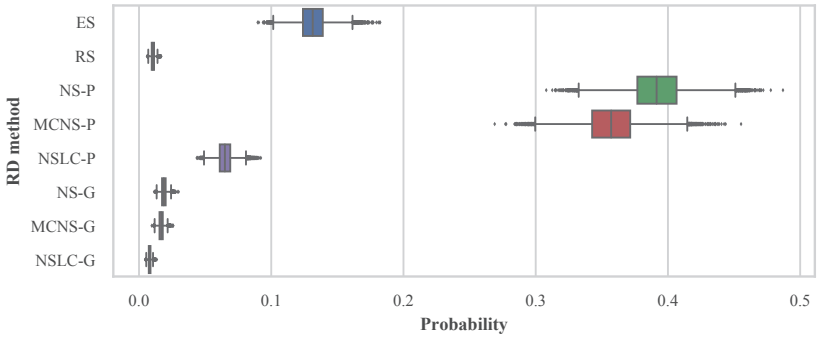

Fig. 7.4. Box plot like Figure 7.3 but for the model complexity (number of parameters in a model) metric. This figure first appeared in [13].

We also investigated different metaheuristics to replace the very simple $(1,\lambda)$–Evolution Strategy originally used for rule discovery [12, 13]. We compared the $(1,\lambda)$–Evolution Strategy with a Random Search (with equal function evaluation budget) and a variety of (μ,λ)–Evolution Strategies utilizing Novelty Search. Novelty Search [19] is a concept of not only (or not at all) targeting the traditional objective fitness, i.e. the error and generality of a rule, to decide which individuals should survive and be used in the evolutionary operators but rather to promote behavioural distinctiveness. In the case of SupRB, our search for novelty emphasized creating rules that did match new subsets of the feature space. We tested a large variety of options and extensions of Novelty Search in [13], finding that the original $(1,\lambda)$–Evolution Strategy is not clearly outperformed but seemingly creates rules from which

a good balance of accurate yet small models can be composed. Figures 7.3 and 7.4 show the results of a Bayesian model comparison test. Typical automated decision thresholds for such tests are 80%. We can therefore safely assume that no clear optimality between the metaheuristic approaches exists. Overall, ES or NSLC-P seem like the most likely candidates. Given ES's vastly superior runtime, we propose the usage of this optimizer for rule discovery in SupRB whenever insufficient computing budget exists to tune for the optimal components as we did find a strong learning task dependency in specific performance orderings on given metrics.

The next step is to investigate the impact of a variety of mixing models on the training success. Additionally, one could investigate different approaches towards solving the problem of optimizing multiple objectives, i.e. model size and prediction error. The use of two optimizers has introduced a multitude of potential hyperparameters. Although our experiments found that most of them only need to be 'sufficiently large' their value is not clear from the beginning and higher values constitute a strong tradeoff against training time. One could investigate the use of self-adaptive operators in the optimizers to alleviate this problem. Currently, SupRB uses ridge regression, however, while regularizing coefficient magnitudes, it does not fulfil the requirement of few features having any coefficient.

Finally, SupRB should be tested on factory data, combined with other knowledge sources, and be integrated into an OAA, as well as finding its way into OC applications and other ML tasks where explainability is of essence. The integration into OC-like systems should come relatively natural and, potentially, OAAs can be transformed into autonomously acting OC systems with the help of SupRB models. This is out-of-scope for this thesis as it is primarily concerned with the ML algorithmic and optimization aspects but will be researched regardless.

7.4 Conclusion

In this chapter, I introduced the need for operator assisting agents (OAAs) in factories and motivated the use of explainable models for decision making. The focus was then on machine learning–based OAAs with models constructed by rule set learning algorithms (RSLs) and I reiterated the tasks that an optimizer needs to solve in these algorithms. Bridging back to the factory setting, a previous study on explainability requirements for the use of RSLs in OAAs was presented. From this, an extensive overview on existing research towards a specific RSL that can more easily fulfil these requirements than other RSLs was given. Finally, this chapter highlighted the most interesting avenues for potential future research.

References

1. Bacardit, J.: Pittsburgh genetics-based machine learning in the data mining era: representations, generalization, and run-time. Ph.D. thesis, PhD thesis, Ramon Llull University, Barcelona (2004)

2. Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 82–115 (2020)
3. Cakar, E., Fredivianus, N., Hähner, J., Branke, J., Müller-Schloer, C., Schmeck, H.: Aspects of learning in oc systems. In: Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.) Organic Computing—A Paradigm Shift for Complex Systems, pp. 237–251. Birkhäuser Basel (2011)
4. Calvo, B., Ceberio, J., Lozano, J.A.: Bayesian inference for algorithm ranking analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 324–325. GECCO '18, Association for Computing Machinery, New York, NY, USA (2018)
5. Calvo, B., Shir, O.M., Ceberio, J., Doerr, C., Wang, H., Bäck, T., Lozano, J.A.: Bayesian performance analysis for black-box optimization benchmarking. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1789–1797. GECCO '19, Association for Computing Machinery, New York, NY, USA (2019)
6. Corani, G., Benavoli, A.: A bayesian approach for comparing cross-validated algorithms on multiple data sets. Machine Learning 100(2), 285–304 (Sep 2015)
7. Drugowitsch, J.: Design and Analysis of Learning Classifier Systems - A Probabilistic Approach, Studies in Computational Intelligence, vol. 139. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
8. 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/
9. Heider, M., Pätzel, D., Stegherr, H., Hähner, J.: A metaheuristic perspective on learning classifier systems. In: Eddaly, M., Jarboui, B., Siarry, P. (eds.) Metaheuristics for Machine Learning: New Advances and Tools, pp. 73–98. Springer Nature Singapore, Singapore (2023)
10. Heider, M., Pätzel, D., Wagner, A.R.M.: An overview of LCS research from 2021 to 2022. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 2086–2094 (2022)
11. Heider, M., Stegherr, H., Nordsieck, R., Hähner, J.: Assessing Model Requirements for Explainable AI: A Template and Exemplary Case Study. Artificial Life pp. 1–19 (08 2023)
12. 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,. pp. 39–49. INSTICC, SciTePress, Setúbal, Portugal (2022)
13. Heider, M., Stegherr, H., Pätzel, D., Sraj, R., Wurth, J., Volger, B., Hähner, J.: Discovering rules for rule-based machine learning with the help of novelty search. SN Computer Science 4(6) (2023)
14. Heider, M., Stegherr, H., Sraj, R., Pätzel, D., Wurth, J., Hähner, J.: SupRB in the context of rule-based machine learning methods: A comparative study. Applied Soft Computing 147, 110706 (2023), https://www.sciencedirect.com/science/article/pii/S156849462300724X
15. 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. (eds.) Bioinspired Optimization Methods and Their Applications. pp. 142–156. Springer International Publishing, Cham (2022)

16. 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)
17. Hörner, L., Schamberger, M., Bodendorf, F.: Using tacit expert knowledge to support shop-floor operators through a knowledge-based assistance system. Computer Supported Cooperative Work-The Journal of Collaborative Computing (2022)
18. Kursuncu, U., Gaur, M., Sheth, A.: Knowledge infused learning (K-IL): Towards deep incorporation of knowledge in deep learning. In: Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice (AAAI-MAKE) (2020)
19. Lehman, J.: Evolution Through the Search for Novelty. Ph.D. thesis, University of Central Florida (2012)
20. Müller-Schloer, C., Schmeck, H., Ungerer, T.: Organic Computing—A Paradigm Shift for Complex Systems. Birkhäuser Basel (2011)
21. Müller-Schloer, C., Tomforde, S.: Organic Computing—Technical Systems for Survival in the Real World. Springer International Publishing, Cham (2017)
22. Nordsieck, R., Heider, M., Hoffmann, A., Hähner, J.: Reliability-based aggregation of heterogeneous knowledge to assist operators in manufacturing. 2022 IEEE 16th International Conference on Semantic Computing (ICSC) pp. 131–138 (2022)
23. Nordsieck, R., Heider, M., Hummel, A., Hähner, J.: A closer look at sum-based embeddings for knowledge graphs containing procedural knowledge. In: Alam, M., Buscaldi, D., Cochez, M., Osborne, F., Reforgiato Recupero Diego (eds.) Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2022) co-located with the 21th International Semantic Web Conference (ISWC 2022) (2022), https://ceur-ws.org/Vol-3342/
24. Pätzel, D., Heider, M., Wagner, A.R.M.: An overview of LCS research from 2020 to 2021. In: Krawiec, K. (ed.) GECCO '21: Genetic and Evolutionary Computation Conference, Companion Volume, Lille, France, July 10-14, 2021. pp. 1648–1656. ACM (2021)
25. Pätzel, D., Heider, M., Hähner, J.: Towards principled synthetic benchmarks for explainable rule set learning algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion). ACM (2023)
26. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: A complete introduction, review, and roadmap. Journal of Artificial Evolution and Applications (2009)
27. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 149–175 (1995)
28. Wilson, S.W.: Compact Rulesets from XCSI. In: Advances in Learning Classifier Systems, pp. 197–208. Springer Berlin Heidelberg (2002)
29. 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)