Mechanisms to Alleviate Over-Generalization in XCS for Continuous-Valued Input Spaces

Alexander R. M. Wagner¹ · Anthony Stein¹

Received: 15 July 2021 / Accepted: 10 February 2022 / Published online: 28 February 2022
© The Author(s) 2022

Abstract

In the field of rule-based approaches to *Machine Learning*, the *XCS classifier system* (XCS) is a well-known representative of the *learning classifier systems* family. By using a *genetic algorithm* (GA), the XCS aims at forming rules or so-called classifiers which are as general as possible to achieve an optimal performance level. A too high generalization pressure may lead to over-general classifiers degrading the performance of XCS. To date, no method exists for *XCS for real-valued input spaces* (XCSR) and *XCS for function approximation* (XCSF) to handle over-general classifiers ensuring an accurate population. The *Absumption* mechanism and the *Specify* operator, both developed for XCS with binary inputs, provide a promising basis for over-general handling in XCSR and XCSF. This paper introduces adapted versions of Absumption and Specify by proposing different identification and specialization strategies for the application in XCSR and XCSF. To determine their potential, the adapted techniques are evaluated in different classification problems, i.e., common benchmarks and real-world data from the agricultural domain, in a multi-step problem as well as different regression tasks. Our experimental results show that the application of these techniques leads to significant improvements of the accuracy of the generated classifier population in the applied benchmarks, data sets, multi-step problems and regression tasks, especially when they tend to form over-general classifiers. Furthermore, considering the working principle of the proposed techniques, the intended decrease in overall classifier generality can be confirmed.

Keywords Learning classifier systems · XCS classifier system · Absumption · Over-general classifiers · Agricultural data

Introduction

The development and use of explainable models in the field of *artificial intelligence* (AI) are intended to promote confidence and transparency regarding the reliability of AI procedures among end users and thus increase their acceptance. Regarding the creation of explainable models, *learning classifier systems* (LCS) feature a fundamental advantage

as they rely on local rules to describe the state space of an environment. These rules allow a simple mapping between states and the system's decisions and improve its interpretability. The most widely studied LCS in terms of formal theoretical analysis [22] and empirical evaluation is currently the *XCS classifier system* (XCS) [39]. It represents an evolutionary rule-based online machine learning technique characterized by an inherent generalization pressure, which is hypothesized to result in accurate and maximally-general rules. These rules are also called classifiers in context of XCS. A classifier advocates an action for a specific region of the state space of an environment under consideration, specified by the condition of the classifier.

Yet, an exclusive pursuit of generalization should be treated with caution, as an excessive pressure to generalize or the lack of a pressure to specialize is likely to impede or prevent the creation of accurate and maximally-general rules. As shown in several papers [2, 15, 18], under certain circumstances an excessive generalization pressure occurs in XCS, leading to the formation

This article is part of the topical collection “Applications of bioinspired computing (to real world problems)” guest edited by Aniko Ekart, Pedro Castillo and Juanlu Jiménez-Laredo.

✉ Alexander R. M. Wagner
a.wagner@uni-hohenheim.de

Anthony Stein
anthony.stein@uni-hohenheim.de

¹ Department of Artificial Intelligence in Agricultural Engineering and Computational Science Lab (CSL), University of Hohenheim, Garbenstraße 9, 70599 Stuttgart, Germany

of so-called over-general classifiers that reduce the performance and accuracy of XCS. Over-general classifiers refer to classifiers that are only capable of advocating the correct action for a subset of its condition as they match incompatible regions or niches of the considered state space [15]. A more detailed description of over-generalization in XCS is provided in section “Generalization and Over-Generalization in XCS”.

By using intervals in the continuous real-valued input space to define a classifier condition, the XCS variants *XCS for real-valued input spaces* (XCSR) [36] as well as the *XCS for function approximation* (XCSF) [37] theoretically permit infinite variations in one dimension of the condition. In contrast the traditional XCS with binary inputs only allows three different states. Thus, the concepts of generalization and over-generalization differ significantly in traditional XCS and the variants XCSR and XCSF, whereas XCSR and XCSF are characterized by a considerably increased complexity due to the large number of variations. A more detailed description of XCS, XCSR and XCSF is provided in section “Real-Valued XCS in a Nutshell”.

Until now, no solution has been proposed for the problem of identifying and handling over-general classifiers in XCSR and XCSF guaranteeing an accurate population during the learning phase. In contrast, for traditional XCS, the *Assumption* mechanism [21] and the *Specify* operator [17] have been proposed offering a basis for XCSR and XCSF. Since both methods aim at solving the same problem, we refer to them as *Over-Generality Handling* (OGH) in this paper.

This work is part of our overarching aspiration to deal with the still unsolved issue of over-generalization in XCS when applied to real-valued problem spaces, cf. e.g., [28]. We focus on XCS(R) as it forms the basis for many descended systems such as UCS [3] and ExSTraCS [33], while additionally investigating the effects of OGH on the variant XCSF. We also deem the capability of XCS to learn complete state-action mappings, more formally $X \times A \rightarrow P$, as a crucial feature in our pursuit of readying robust and reliable online learning systems allowing for interpretability but also self-reflection of the evolved knowledge [27, 29].

The contributions of this work are: (1) The evaluation of the application of *Assumption* and of *Specify* to gain insights into their impact on the learning performance of XCSR and XCSF. (2) The introduction of identification and specialization strategies for OGH in real-valued problem spaces. (3) A comparative study of the use of *Assumption* and of *Specify* in XCSR, applied to several well-known toy problems (single- and multi-step), and agricultural

real-world classification tasks. (4) A comparative study of the use of *Assumption* and of *Specify* in XCSF, applied to three well-known regression tasks from the field of global optimization, while assessing different over-generalization tendency levels.

We proceed with a brief overview of approaches to remove inappropriate classifiers in LCS in the following section followed by necessary background information on XCS and OGH in the next section. The subsequent section introduces the adapted versions of the OGH approaches for XCS in real-valued problem spaces, i.e., XCSR and XCSF, including new specialization and identification strategies. In “Evaluation”, the results of empirical studies on various classification tasks, on a multi-step problem and three regression tasks are reported and thoroughly discussed in the next section. We conclude with a short summary and a description of future work in “Conclusion”.

This article is an extension of an original conference paper published at the Evostar 2021 conference [34]. It additionally includes: new experiments on the application of OGH to XCSF for regression tasks; a more thorough discussion of both the new results and the results of the original conference paper; more elaborate descriptions of the algorithms of the proposed techniques using pseudocode.

Related Work

Early on, over-generalization was part of the research in the field of XCS [14], but so far this has only resulted in procedures for XCS with binary inputs. The *Assumption* method [21] is the most recent procedure for XCS with binary inputs, especially designed for single-step problems. *Assumption* is based on the inconsistency of an over-general classifier’s condition, since it contains incompatible niches or niches, which cannot be predicted by the same action. After identification, a more specific version of an over-general is generated by changing a Don’t-Care symbol # to a specific value. The *Specify* operator [17], a previously introduced procedure for XCS with binary inputs, was developed to handle the problem of over-generals in multi-step environments. This approach employs the prediction error to identify over-generals, since the inevitably arising oscillation in the prediction of an over-general is recognizable this way. The specialization of an over-general classifier is performed similarly to *Assumption*. In addition to the previously mentioned procedures developed especially for XCS with binary inputs, there are further specialization mechanisms in other LCSs. In the field of evolutionary learning systems, rule specialization methods have also been introduced, such as the

so-called memetic operators in BioHEL [8]. The *Mutespec* operator [10], introduced for the ALECSYS classifier system, bases the identification of over-generalers on the variance of the rewards received by a classifier. It has to be noted that over-general classifiers are termed oscillating classifiers in the publications of Specify and Mutespec. Furthermore, ACS2 [6] applies a specialization mechanism, which, however, was not specifically designed to identify over-generalers. Finally, an approach for the specialization of rules beyond LCSs, introduced in [11], is the *Specialize* operator for a genetic learning system operating on high-level symbolic rules.

In addition to approaches that focus on the specialization of rules, further approaches are available to remove unnecessary or inappropriate rules from the population in order to achieve an optimal solution. In [39], the idea of *condensation* was discussed and further enhanced in [16], which employs the GA with the genetic operators turned off in order to remove classifiers with a low level of fitness from the population. The *compaction* algorithm [9, 40] forms a compact rule set after the learning phase of XCS is completed by finding the smallest subset of the population that achieves maximum performance. In [5], the compaction algorithm has been extended by *closest classifier matching*, which is applied at the beginning of the compaction process to avoid that the resulting population no longer covers the entire problem space.

Background

In this section, we start by briefly providing the necessary general information about XCS and its general learning

scheme for the sake of facilitated comprehension. We then give a brief introduction to over-generalization in XCS, which includes a definition of over-general classifiers along with a description of their negative impact on XCS. However, a certain familiarity with *temporal difference learning* in general is assumed due to space constraints. For a more detailed introduction to temporal difference learning and XCS, the reader is referred to [31] and [7, 36], respectively.

Real-Valued XCS in a Nutshell

The evolutionary rule-based machine learning system XCS was introduced by Wilson in [35, 39] and belongs to the family of Michigan-style LCSs, which are attributed to Holland [12]. As XCS heavily relies on a niche-relative variant of Holland's *genetic algorithm* (GA), which employs a measure of relative accuracy of the rule's prediction as fitness, XCS represents the origin of the family of accuracy-based LCSs [32]. In contrast, the GA of its so-called strength-based relatives, such as ZCS [38], directly employs the rule's prediction value, e.g., an estimate of expected cumulated reward in reinforcement learning settings, as fitness of the GA. In the context of LCS, the prediction value is also called strength. As the GA enables continuous optimization of the structure of the rules, XCS is able to adapt to changes in its environment while in use, which is an important factor for reliable deployment in real-world applications, such as field robotics. Thus the evolutionary learning approach of XCS constitutes a particular strength in adapting to an environment.

Algorithm 1: XCS learning loop, adapted from [7]

```

1 Observe environmental state  $\vec{\sigma}_t$  at time  $t$ 
2 Create match set  $[M] \subseteq [P]$  for  $\vec{\sigma}_t$  with  $[M] := \{cl_i \mid \vec{\sigma}_t \in cl_i.C\}$ 
3 Create  $PA$  from  $[M]$  with  $PA(a) = \frac{\sum_{cl \in [M] \mid cl.a=a} cl.p * cl.F}{\sum_{cl \in [M] \mid cl.a=a} cl.F}$ 
4 if  $U_{[0,1]} < \epsilon$  then
5   | Select random action, i.e.,  $a_{exec} = \text{rand}(A)$ 
6 else
7   | Select best action, i.e.,  $a_{exec} = \text{argmax}_a PA(a)$ 
8 end if
9 Create action set  $[A] \subseteq [M]$  with  $[A] := \{cl_i \mid cl_i.a = a_{exec}\}$ 
10 Execute  $a_{exec}$  on and observe reward  $r_t$  from the environment
11 if  $[A]_{t-1} \neq \emptyset$  then
12   |  $P \leftarrow r_{t-1} + \gamma \cdot \text{argmax}_a PA(a)$ 
13   | Update each  $cl \in [A]_{t-1}$  using  $P$  by means of Widrow-Hoff delta rule
14   | if  $t = \frac{\sum_{cl \in [A]_{t-1}} cl.num * cl.ts}{\sum_{cl \in [A]_{t-1}} cl.num} > \theta_{GA}$  then
15     | Run GA on  $[A]_{t-1}$  considering  $\vec{\sigma}_{t-1}$ 
16   | end if
17 end if
18 if end of episode then
19   |  $P \leftarrow r_t$ 
20   | Update each  $cl \in [A]$  using  $P$  by means of Widrow-Hoff delta rule
21   | if  $t = \frac{\sum_{cl \in [A]} cl.num * cl.ts}{\sum_{cl \in [A]} cl.num} > \theta_{GA}$  then
22     | Run GA on  $[A]$  considering  $\vec{\sigma}_t$ 
23     |  $[A]_{t-1} \leftarrow \emptyset$ 
24   | end if
25 else
26   |  $[A]_{t-1} \leftarrow [A]$ 
27   |  $r_{t-1} \leftarrow r_t$ 
28   |  $\vec{\sigma}_{t-1} \leftarrow \vec{\sigma}_t$ 
29 end if

```

As proposed by Wilson's *generalization hypothesis* [35], XCS aims to construct a complete, accurate and maximally-general state-action map, more formally defined as $X \times A \rightarrow P$. The set of all possible environmental states σ_t is called *input space* X . The input space is also known as state space. The *action space* A represents all possible actions and P denotes the *payoff space*. In XCS the state-action map is represented by a population $[P] := \{cl_i\}_{i \in \mathbb{N}}$ of rules or classifiers, usually just called cl and defined as $cl := (C, a, p, \epsilon, F)$. The classifier condition $cl.C \subseteq X$ defines a subset of X , which is matched by the classifier. $cl.a \in A$ represents the action advocated by cl to be performed by the system when the current state falls into the classifier's condition. $cl.p$ denotes the prediction of cl , which represents an estimate of the payoff received from the environment in the case that $cl.a$ is performed within the classifier condition. The use of local learning in XCS by means of classifiers facilitates both the interpretability and the reflection

of the acquired knowledge, e.g., whether knowledge gaps or contradictions exist. In $cl.\epsilon$, an estimate of the absolute *prediction error* is stored. $cl.F$ represents the *fitness* of cl , estimating the *niche-relative* accuracy of the payoff prediction of the classifier. Furthermore, additional book-keeping parameters are maintained in XCS: the number of performed reinforcements of a cl is stored in $cl.exp$ and the number of successful subsumption operations in $cl.num$. To control the GA invocation, a timestamp parameter $cl.ts$ is updated each time a cl was a candidate for the GA. For more details about the classifiers and their attributes and parameters, the interested reader is referred to [7].

Algorithm 1 presents a detailed description of a single pass of the main learning loop of XCS. As long as the defined termination criteria are not satisfied, e.g., a maximum number of learning steps, the loop is repeated. When applied to single-step problems, i.e., classification, the actions are usually selected by alternating explore and

exploit steps instead of an ϵ -greedy approach. $U_{[0,1]}$ draws a value uniformly at random from the given interval. For a more in-depth explanation of the learning loop of XCS, the reader is referred to [7].

The basic structure of XCS features a great flexibility regarding possible tasks, also reflected in the good extensibility of XCS. Thus, many extensions to XCS have already been introduced to expand its application range, cf. e.g., [33, 36, 37]. Two extensions of particular interest in the context of this work are: *XCS for real-valued inputs* (XCSR) [36] and *XCS for function approximation* (XCSF) [37]. XCSR introduces so-called interval predicates (l_i, u_i) to define a classifier condition combined with minor modifications to deal with real-valued inputs. l_i and u_i denote a lower and upper bound, respectively, for the dimension i of the input space X . As a result of using interval predicates to define a classifier condition, the conditions have the geometric form of axis-parallel hyperrectangles.

XCSF builds on the extension of XCSR and additionally employs a single dummy action and classifier predictions based on a predictive model instead of a simple scalar. The predictive model usually employs a linear function $h(\sigma_t)$ that computes the classifier prediction as a function of the environmental state σ_t , which is an n -dimensional vector of the input space. The function is defined as $h(\sigma_t) = (\sigma_t^* - l^*) \times w$. σ_t^* refers to σ_t extended by a leading 1. l^* refers to vector $l = (l_1, \dots, l_n)$, which comprises the lower bounds of the interval predicates of the classifier, extended by a leading 0. $w = (w_0, \dots, w_n)$ denotes an $n+1$ -dimensional weight vector. As proposed in [20], updating the weights is now commonly done using *Recursive Least Squares* (RLS), rather than updating the weights using a modified delta rule as originally proposed. For a more detailed description of XCSR and XCSF, the reader is referred to [36] and [37], respectively.

Generalization and Over-Generalization in XCS

Based on Wilson's description in [35], a system uses generalization whenever it treats seemingly different situations in the same way if these situations lead to equivalent after-effects for the system when performing the same action. A major problem for LCS is the unintended formation of over-general classifiers, i.e., a classifier that is just capable of advocating the correct action for a subset of its condition [15], as already noted in section "Introduction". These classifiers are capable of degrading the performance of a system since they may result in the selection of an incorrect action and propagate incorrect information. Since over-general classifiers match incompatible niches, they possess an oscillating prediction as they are only locally accurate, which usually causes them to be inaccurate.

As conjectured by Wilson's *Generalization Hypothesis* [35], explaining the process of generating accurate and maximally-general classifiers in XCS, over-general classifiers should be removed from the population due to their inaccuracy. However, under certain circumstances the counteracting fitness signal is not sufficient, which leads to over-general classifiers caused by the generalization pressure present in XCS. Since XCS is able to tolerate small oscillations of the classifier prediction due to the error threshold hyperparameter ϵ_0 , it is possible that an inaccurate classifier is considered accurate if the oscillation is small enough [15]. Thus, as described by Lanzi in [18], an over-general classifier must be observed often enough or at all by XCS to cause sufficient oscillation of the prediction for the classifier to be recognized as inaccurate.

We refer to this problem as *unequally observed environmental niches* from the point of view of the classifiers, which can be caused by *non-uniform sampling* or *unequally sized niches*. The problem of unequally sized niches can also occur in environments with equally sized niches if different types of niches are not or not almost equally contained in a classifier. Another problem that causes XCS to consider over-general classifiers as accurate is the issue of *indistinguishable payoff levels* of different environmental niches, which is associated with multi-step problems [2]. This issue can arise due to the application of a discounted reward or the presence of *dominant* classifiers, which are over-general classifiers that result in a misleading fitness signal. In comparison to binary inputs, these problems are generally more serious for real-valued inputs, since, on the one hand, the number of possible observations in the possibly continuous input range is substantially increased. On the other hand, an infinite number of variations of the intervals are theoretically possible in the individual dimensions of a condition. For a more detailed description of generalization and over-generalization in the context of XCS, we refer the reader to [2, 15, 18] and [4].

Over-Generality Handling for Real-Valued XCS

To address the challenge of over-generalization in real-valued problem spaces by identifying and suitably handling over-general classifiers, we design two *Over-Generality Handling* (OGH) approaches, one based on the *Assumption* mechanism in [21] and the other based on the *Specify* operator in [17]. If OGH is enabled, it is invoked subsequently to the reinforcement in each iteration of XCS' main loop. First, we will introduce the basic structure and the identification strategies of both approaches, as the specialization strategies, creating more specific versions of an identified over-general classifier, are shared between both variants of OGH.

Algorithm 2: Absumption for real-valued XCS

Input: $[A]$:= classifiers in action set
 $[P]$:= classifier population

```

1  $[overG]$  := detect over-general classifiers in  $[A]$  by identification strategy
2 forall  $cl_{og}$  in  $[overG]$  do
3   if  $REA$  of  $cl_{og} < 1$  then
4     remove  $cl_{og}$  from  $[P]$ 
5   else
6      $[spec]$  := create more specific versions of  $cl_{og}$  by specialization strategy
7     remove  $cl_{og}$  from  $[P]$  and  $[A]$ 
8     add all classifiers from  $[spec]$  to  $[P]$  and  $[A]$ 
9   end if
10 end forall

```

As shown in Algorithm 2, the adapted version for real-valued problem spaces uses a more generic basic structure compared to the original Absumption [21], i.e., both the identification and the specialization strategy employed are exchangeable. Based on the *REA* ratio of an over-general classifier cl_{og} , Absumption decides, whether a cl_{og} in $[overG]$ is removed from the population or decomposed into more specific versions. *REA* is the ratio of cl_{og} 's experience to the dimensionality of its condition, i.e., an indicator of useful information contained in cl_{og} . The threshold value is set to 1 as described in [21]. Thus, if the *REA* of cl_{og} is at least 1,

the specialization strategy applied decomposes cl_{og} into an amount of more specific versions defined by $cl_{og}.num$.

The first strategy we propose for detecting over-general classifiers, called *Inconsistency*, is based on the mechanism proposed in [21]. The algorithmic description of Inconsistency is shown in Algorithm 3. It utilizes the idea of over-general classifiers matching incompatible niches by tracking the *number of positive rewards* (NPR) and the *number of negative rewards* (NNR). If $NPR * NNR > 0$ is satisfied for a classifier, it is considered to be over-general and is added to $[overG]$.

Algorithm 3: *Inconsistency* identification strategy

Input: $[A]$:= classifiers in action set

```

1  $[overG]$  := empty set
2 forall  $cl$  in  $[A]$  do
3   if  $cl.NPR * cl.NNR > 0$  then
4     add  $cl$  to  $[overG]$ 
5   end if
6 end forall
7 return  $[overG]$ 

```

Algorithm 4: *Prediction Deviation Variance* identification strategy

Input: $[A]$:= classifiers in action set
 $[P]$:= classifier population

```

1  $[overG]$  := empty set
2  $dev^* := \frac{1}{|[P]|} \sum_{cl \in [P]} cl.dev$ 
3 forall  $cl$  in  $[A]$  do
4   if  $cl.exp > \theta_{OGIdent}$  and  $cl.dev > 2 * dev^*$  then
5     add  $cl$  to  $[overG]$ 
6   end if
7 end forall
8 return  $[overG]$ 

```

As Inconsistency can be applied only to single-step classification problems, we introduce the second identification strategy, *Prediction Deviation Variance*, enabling its application to multi-step problems as well as to regression tasks. This strategy performs the identification based on the increased prediction oscillation of an over-general classifier. In this strategy, for each classifier cl , the absolute deviation $|cl.p - r_t|$, where r_t denotes the current reward at time t , is tracked in a FIFO-buffer within cl , denoted as $cl.dev$. As described in Algorithm 4, a classifier cl is only checked for over-generality by this strategy if $cl.exp > \theta_{OGCIdent}$ is satisfied. If the variance of the absolute deviations in $cl.dev$ is greater than twice the average deviation variances of the

entire population, denoted as dev^* , cl is regarded as over-general and is added to $[overG]$. This condition was determined in preliminary experiments and does not claim to be optimal, but has been found to yield improved results.

Specify for real-valued XCS is based on the version for binary inputs in [17], already employing an identification strategy applicable in multi-step problems as well as in single-step classification and regression tasks. The process of Specify for real-valued XCS is shown in Algorithm 5. This OGH approach uses an indirect mechanism to detect over-general classifiers based on the condition $\epsilon_{[A]} \geq 2 * \epsilon_{[P]}$ proposed in [17].

Algorithm 5: Specify for real-valued XCS

Input: $[A]$:= classifiers in action set
 $[P]$:= classifier population

```

1 if average experience in  $[A] \geq \theta_{OGCIdent}$  then
2    $\epsilon_{[A]} := \frac{\sum_{cl \in [A]} cl.\epsilon}{\sum_{cl \in [A]} cl.num}$ 
3    $\epsilon_{[P]} := \frac{\sum_{cl \in [P]} cl.\epsilon}{\sum_{cl \in [P]} cl.num}$ 
4   if  $\epsilon_{[A]} \geq 2 * \epsilon_{[P]}$  then
5      $cl := \epsilon$ -proportionate selection of one classifier in  $[A]$ 
6      $cl_{spec} :=$  create one more specific version of  $cl$  by specialization strategy
7     add  $cl_{spec}$  to  $[P]$ 
8   end if
9 end if
  
```

We propose two specialization strategies returning the required amount of more specific versions of an identified classifier cl_{id} utilizing the center point c of cl_{id} 's hyperrectangular condition $cl_{id}.C = (l^{cl_{id}}, u^{cl_{id}})$, i.e., $c_i = l_i^{cl_{id}} + \frac{1}{2}(u_i^{cl_{id}} - l_i^{cl_{id}})$ for $i = 1 \dots n$. The first method is the *New Condition Specialization* (NCS) strategy. Its mode of operation is shown in Algorithm 6. A new classifier cl_{new} is generated by computing the so-called interval predicate (l_i, u_i) of condition $cl_{new}.C$ for each

dimension $i = 1 \dots n$ as follows: $l_i = \max\{l_i^*, c_i - U_{[0, r_0)}\}$ and $u_i = \min\{u_i^*, c_i + U_{[0, r_0)}\}$. l_i^* and u_i^* denote the minimum and maximum bounds of the problem space for dimension i , respectively. In $U_{[0, r_0)}$, the given standard spread parameter r_0 is excluded. The other attributes of cl_{new} are initialized with the attribute values of cl_{id} , except $cl_{new}.exp = 0$ and $cl_{new}.num = 1$. However, as a simple and fast strategy, it does not guarantee the creation of $cl_{new}.C$ inside $cl_{id}.C$ and thus may result in the generation of $cl_{new}.C$ that is not fully contained within $cl_{id}.C$.

Table 1 Overall results for classification tasks, i.e., reward, system error, population size and generality

Benchmark Problems				
6-RMP	Reward	Sys. Err.	$ [P] $	Generality
XCSR w/ Abs	935.07 $\downarrow \pm 9.92$ (B)	119.73 $\uparrow \pm 14.56$ (A)	408.62 $\uparrow \pm 23.95$ (A)	0.00067 $\downarrow \pm 0.00006$ (C)
XCSR w/ Specify	968.10 $\uparrow \pm 3.97$ (A)	77.88 $\uparrow \pm 6.67$ (B)	323.96 $\uparrow \pm 11.89$ (B)	0.00113 $\downarrow \pm 0.00004$ (B)
Standard XCS	967.96 ± 6.07 (A)	65.82 ± 10.43 (C)	279.93 ± 17.03 (C)	0.00118 ± 0.00007 (A)
CBP(3,3)	Reward	Sys. Err.	$ [P] $	Generality
XCSR w/ Abs	899.39 $\downarrow \pm 6.72$ (B)	209.16 $\uparrow \pm 7.35$ (A)	1103.25 $\uparrow \pm 20.99$ (A)	0.00363 $\uparrow \pm 0.00003$ (A)
XCSR w/ Specify	936.25 $\uparrow \pm 4.27$ (A)	130.47 $\uparrow \pm 6.05$ (B)	765.16 $\uparrow \pm 14.75$ (B)	0.00356 $\uparrow \pm 0.00011$ (B)
Standard XCS	935.61 ± 5.57 (A)	119.05 ± 7.85 (C)	689.61 ± 17.99 (C)	0.00312 ± 0.00010 (C)
Mario	Reward	Sys. Err.	$ [P] $	Generality
XCSR w/ Abs	907.33 $\uparrow \pm 2.45$ (A)	156.24 $\downarrow \pm 2.69$ (C)	3544.48 $\uparrow \pm 26.96$ (A)	0.00557 $\downarrow \pm 0.00014$ (C)
XCSR w/ Specify	891.98 $\uparrow \pm 3.80$ (B)	169.79 $\downarrow \pm 3.62$ (B)	3291.98 $\uparrow \pm 24.87$ (B)	0.00709 $\downarrow \pm 0.00019$ (B)
Standard XCS	866.50 ± 5.28 (C)	190.84 ± 4.98 (A)	2690.30 ± 31.56 (C)	0.00939 ± 0.00034 (A)
Real-World Problems				
Paddy Leaf	Reward	Sys. Err.	$ [P] $	Generality
XCSR w/ Abs	828.09 $\uparrow \pm 4.57$ (A)	217.95 $\downarrow \pm 4.61$ (C)	4416.52 $\uparrow \pm 29.31$ (A)	0.00117 $\downarrow \pm 0.00003$ (C)
XCSR w/ Specify	808.03 $\uparrow \pm 12.38$ (B)	277.43 $\downarrow \pm 10.34$ (B)	3145.65 $\uparrow \pm 57.59$ (B)	0.00661 $\downarrow \pm 0.00882$ (B)
Standard XCS	688.16 ± 34.67 (C)	398.29 ± 29.64 (A)	1711.14 ± 145.73 (C)	0.01855 ± 0.00334 (A)
Horse Colic	Reward	Sys. Err.	$ [P] $	Generality
XCSR w/ Abs	990.89 $\uparrow \pm 0.66$ (A)	31.93 $\downarrow \pm 1.89$ (B)	5102.72 $\downarrow \pm 26.60$ (B)	$2.75 \times 10^{-11} \downarrow \pm 1.42 \times 10^{-11}$ (A)
XCSR w/ Specify	984.18 $\downarrow \pm 2.73$ (B)	54.26 $\uparrow \pm 5.03$ (A)	5139.62 $\uparrow \pm 26.87$ (A)	$3.54 \times 10^{-11} \downarrow \pm 1.50 \times 10^{-11}$ (A)
Standard XCS	984.83 ± 2.42 (B)	54.00 ± 4.72 (A)	5104.23 ± 28.37 (B)	$3.63 \times 10^{-11} \pm 1.53 \times 10^{-11}$ (A)
Soybean Disease	Reward	Sys. Err.	$ [P] $	Generality
XCSR w/ Abs	895.71 $\uparrow \pm 6.41$ (A)	187.89 $\downarrow \pm 14.02$ (B)	21471.02 $\uparrow \pm 33.76$ (B)	$6.05 \times 10^{-17} \uparrow \pm 5.06 \times 10^{-17}$ (A)
XCSR w/ Specify	866.61 $\downarrow \pm 10.25$ (B)	208.64 ± 12.65 (A)	21987.54 $\uparrow \pm 33.79$ (A)	$5.34 \times 10^{-17} \downarrow \pm 4.48 \times 10^{-17}$ (A)
Standard XCS	871.53 ± 8.83 (B)	200.99 ± 12.28 (A)	21470.48 ± 28.94 (B)	$5.46 \times 10^{-17} \pm 3.11 \times 10^{-17}$ (A)

All entries represent the means ± 1 SD of the 30 repetitions. Letters in brackets denote groups with statistically significant differences, i.e., a p value $< \alpha = 0.05$. Arrows indicate an increase or decrease in a metric and bold entries highlight significant improvements, both with respect to the baseline

Algorithm 6: *New Condition specialization strategy*

Input : $cl_{id} :=$ classifier to be specialized
 $num_cl_{new} :=$ number of classifiers to be generated
 $dims :=$ dimensionality of classifier conditions

```

1  $[spec] :=$  empty set
2 while  $|[spec]| < num\_cl_{new}$  do
3   forall  $i = 1 \dots dims$  do
4      $c_i = l_i^{cl_{id}} + \frac{u_i^{cl_{id}} - l_i^{cl_{id}}}{2}$ 
5      $l_i = \max\{l_i^*, c_i - U_{[0, r_0]}\}$ 
6      $u_i = \min\{u_i^*, c_i + U_{[0, r_0]}\}$ 
7   end forall
8    $C_{spec} :=$  generate specialized condition using  $\vec{l}$  and  $\vec{u}$ 
9   generate  $cl_{spec}$  as copy of  $cl_{id}$ 
10   $cl_{spec}.C = C_{spec}$ 
11   $cl_{spec}.exp = 0$ 
12   $cl_{spec}.num = 1$ 
13  add  $cl_{spec}$  to  $[spec]$ 
14 end while
15 return  $[spec]$ 

```

This drawback is solved by the *Inside Condition Specialization* (ICS) strategy, which is shown in Algorithm 7. In case of ICS, (l, u) of $cl_{new}.C$ are determined differently: $l_i = \max\{l_i^{cl_{id}}, l_i^*, c_i - U_{[0, r_0)}\}$ and $u_i = \min\{u_i^{cl_{id}}, u_i^*, c_i + U_{[0, r_0)}\}$. Thus, $cl_{new}.C$ is fully contained inside $cl_{id}.C$, as the constraints $l_i^{cl_{id}} \leq l_i$ and $u_i \leq u_i^{cl_{id}}$ are satisfied. The other attributes of cl_{new} are initialized analogously to NCS.

generate accurate complete mappings and, thus, to reliably perform classification tasks and stream mining of online real-world data. Furthermore, OGH is designed to be easily applicable to any system descended from XCSR, a fact highlighted by our experiments in XCSF.

All experiments have been repeated for 30 i.i.d. runs with individual random seeds. The repetition means and the observed standard deviations of the conducted experiments are given in Tables 1, 2, 3, 4, 5, 6 below. In all conducted

Algorithm 7: *Inside Condition specialization strategy*

Input : $cl_{id} :=$ classifier to be specialized
 $num_cl_{new} :=$ number of classifiers to be generated
 $dims :=$ dimensionality of classifier conditions

```

1  $[spec] :=$  empty set
2 while  $|[spec]| < num\_cl_{new}$  do
3   forall  $i = 1 \dots dims$  do
4      $c_i = l_i^{cl_{id}} + \frac{u_i^{cl_{id}} - l_i^{cl_{id}}}{2}$ 
5      $l_i = \max\{l_i^{cl_{id}}, l_i^*, c_i - U_{[0, r_0)}\}$ 
6      $u_i = \min\{u_i^{cl_{id}}, u_i^*, c_i + U_{[0, r_0)}\}$ 
7   end forall
8    $C_{spec} :=$  generate specialized condition using  $\vec{l}$  and  $\vec{u}$ 
9   generate  $cl_{spec}$  as copy of  $cl_{id}$ 
10   $cl_{spec}.C = C_{spec}$ 
11   $cl_{spec}.exp = 0$ 
12   $cl_{spec}.num = 1$ 
13  add  $cl_{spec}$  to  $[spec]$ 
14 end while
15 return  $[spec]$ 

```

Evaluation

In this section the results of a variety of experiments are summarized we carried out to evaluate the impact of Assumption- and Specify-based OGH on the performance on XCS variants for real-valued problem spaces. We performed experiments with XCSR on two multi-step problems and on six single-step problems, i.e., classification tasks. These tasks comprise three well-known toy problems and three real-world data sets from the agricultural domain, as the authors' research is concerned with the usage of AI in agricultural applications. In addition, we conducted experiments with XCSF in three well-known regression tasks from the field of global optimization. Typically, other types of modern LCSs, e.g., ExSTraCS [33] or BioHEL [1], are preferred for classification tasks and for mining of data sets. However, we have chosen XCSR as it enables reliable learning at runtime, i.e., online, in systems facing real-world conditions, due to creating a complete $X \times A \rightarrow P$ mapping. We intend to show OGH improving the ability of XCSR to

experiments, the performance of XCSR and XCSF was evaluated using 3 metrics: (1) The system error, i.e., the average error of system prediction in relation to either the actual reward received in classification and multi-step tasks or the actual function value in regression tasks. (2) The average number of macro classifiers or the average size of the population, indicated by $|[P]|$. (3) The average volume of classifier conditions in $[P]$, i.e., the average classifier generality. In addition, the performance of XCSR was also evaluated using a fourth metric, the average reward achieved at the end of the learning period.

In the evaluation experiments, both a configuration with the Specify-based and with the Assumption-based OGH approach have been evaluated. They are compared with XCSR and XCSF instances without our proposed OGH techniques, which are configured according to the settings yielding the latest improvements on the considered problem types as reported in the literature, serving as baseline in our experiments. Since the configurations are dependent on the problem types as well as the individual problem instances,

Fig. 1 Learning curve plots of the conducted experiments using XCSR

the exact configuration details are provided with the descriptions of the employed problem instances in the following sections: the benchmark problems in section “Results in Benchmark Problems”, the real-world data sets in section “Results in Real-World Problems”, the multi-step problems

in section “Results in Multi-step Problems” and the regression problems in section “Results in Regression Tasks”.

For statistical evaluation, we analyzed the experimental results by means of different statistical tests for significance: A standard *ANOVA* paired with a *Tukey-HSD* post-hoc test

Table 2 Overall results of conducted multi-step experiments, i.e., reward, system error, population size, generality and steps to goal

GridWorld(0.07)	Reward	Sys. Err.	$ [P] $	Generality	Steps
Determined over	First 205,000 steps				Entire run
XCSR w/ Abs	-0.536 $\uparrow$ (A)	3.10 $\downarrow$ (C)	5120.70 $\uparrow$ (A)	0.00328 $\downarrow$ (C)	20.04 $\downarrow$ (B)
	± 0.003	± 0.08	± 77.08	± 0.00017	± 0.32
XCSR w/ Specify	-0.538 $\uparrow$ (A)	4.28 $\downarrow$ (B)	3627.34 $\uparrow$ (B)	0.02234 $\downarrow$ (B)	21.32 $\downarrow$ (B)
	± 0.005	± 0.74	± 65.95	± 0.00682	± 1.37
Standard XCS	-0.549 (B)	5.27 (A)	2873.86 (C)	0.03807 (A)	23.80 (A)
	± 0.007	± 1.32	± 221.72	± 0.01423	± 6.37
GridWorld(0.05)	Reward	Sys. Err.	$ [P] $	Generality	Steps
Determined over	First 285,000 steps				Entire run
XCSR w/ Abs	-0.534 $\uparrow$ (A)	4.14 $\downarrow$ (C)	5306.61 $\uparrow$ (A)	0.00300 $\downarrow$ (C)	29.41 $\downarrow$ (C)
	± 0.003	± 0.17	± 129.26	± 0.00018	± 1.09
XCSR w/ Specify	-0.537 $\uparrow$ (B)	6.00 $\downarrow$ (B)	3600.12 $\uparrow$ (B)	0.03148 $\downarrow$ (B)	33.60 $\downarrow$ (B)
	± 0.003	± 0.92	± 150.25	± 0.00961	± 3.80
Standard XCS	-0.543 (C)	8.01 (A)	2510.71 (C)	0.05421 (A)	42.95 (A)
	± 0.004	± 1.56	± 326.01	± 0.01368	± 7.86

Entries are defined analogously to Table 1

was conducted to test for statistically significant differences if a test for *homoscedasticity* was positive. Otherwise, we employed the robust *Welch-ANOVA* in combination with a *Games-Howell* post-hoc test. Figure 1 and 2 show plots of the learning curves of the compared configurations over the entire learning period along with the standard deviation in form of error bars. As commonly done in the LCS literature, the learning progress of the single-step problems is shown in form of reward achieved and system prediction error or, in case of the regression tasks for XCSF, only in form of the system prediction error. For the multi-step problems, the learning progress is depicted by means of required steps to the goal.

Results in Benchmark Problems

First, we performed an evaluation of three well-known benchmark problems, as their known problem structure facilitates an analysis with respect to specific characteristics and provides the basis for comparison with results from the literature: (1) *Real k-multiplexer problem* (RMP) [36], (2) *Checkerboard problem* (CBP) [30], and (3) *Mario* classification problem [27]. RMP poses a challenging problem of real-valued binary classification with properties like feature interaction (*epistasis*) and multiple niches with same actions (*heterogeneity*) [33]. For each dimension x_i , with $i = 1, \dots, k$, the input space of RMP is defined in the interval $0.0 \leq x_i < 1.0$. A threshold θ specifies the two partitions of the bit values in each dimension. In this paper we applied

RMP with $\theta = 0.5$ and $k = 6$, denoted as 6-RMP. CBP is a well-known benchmark for LCS, designed to provide an increased complexity compared to RMP [30]. The underlying problem is to predict a multidimensional checkerboard pattern consisting of black and white areas. In this paper, we used a variant of CBP with 3 dimensions and 3 divisions per dimension denoted as CBP(3,3). Mario is a two-dimensional real-valued multi-class classification problem in the form of a 16x16 pixel art of Super Mario. Unlike CBP it features seven different actions, comprising the various colors, and allows for different levels of generalization in the niches, e.g., the blue trousers compared to the yellow knobs. Analogously to RMP, the input space of CBP and Mario is also defined in the interval $0.0 \leq x_i < 1.0$ for each dimension x_i .

A single repetition of an experiment was run over 100,000 alternating explore/exploit steps. In each step, the input space of the benchmark problems was sampled using a uniform distribution. We applied a binary reward scheme for each of the benchmark problems. Each correctly predicted action of a sampled situation led to a reward of 1000, otherwise 0. The intervals of a classifier condition were encoded by the unordered bound hyperrectangular representation [30]. The configuration and hyperparameters of OGH were determined in preliminary experiments. In all benchmark problems, we applied the specialization strategy NCS for both OGH approaches and the *Inconsistency* identification strategy of Absumption. For Specify, $\theta_{OGCIdent}$ was set to 10 for 6-RMP and to 50 for CBP(3,3) and Mario. We adopted further hyperparameters from the literature, which

Table 3 Overall results of conducted experiments on Eggholder function and Sine-in-Sine function using RLS-based predictions, i.e., system error, population size and generality

Over	First 50,000 steps			Entire run		
Eggholder 2D $r_0=0.1$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0764 ↓ (C)	3475.30 ↑ (A)	8.82×10^{-4} ↓ (C)	0.0649 ↑ (A)	3581.16 ↑ (A)	6.43×10^{-4} ↓ (C)
	± 0.0008	± 55.05	$\pm 0.25 \times 10^{-4}$	± 0.0004	± 37.79	$\pm 0.07 \times 10^{-4}$
XCSF w/ Specify	0.0797 ↓ (B)	2705.16 ↑ (B)	11.17×10^{-4} ↓ (B)	0.0616 ↓ (C)	2817.21 ↑ (B)	7.64×10^{-4} ↓ (B)
	± 0.0008	± 16.72	$\pm 0.26 \times 10^{-4}$	± 0.0004	± 11.66	$\pm 0.08 \times 10^{-4}$
Standard XCS	0.0808 (A)	2603.51 (C)	11.57×10^{-4} (A)	0.0621 (B)	2675.86 (C)	7.94×10^{-4} (A)
	± 0.0009	± 22.54	$\pm 0.30 \times 10^{-4}$	± 0.0004	± 14.35	$\pm 0.10 \times 10^{-4}$
Eggholder 2D $r_0=0.5$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.1065 ↓ (C)	3123.13 ↑ (A)	50.29×10^{-4} ↓ (B)	0.0749 ↓ (C)	3490.44 ↑ (A)	17.17×10^{-4} ↓ (B)
	± 0.0067	± 126.13	$\pm 19.68 \times 10^{-4}$	± 0.0016	± 39.56	$\pm 4.93 \times 10^{-4}$
XCSF w/ Specify	0.1806 ↓ (B)	1748.28 ↑ (B)	235.95×10^{-4} ↓ (A)	0.0927 ↓ (B)	2611.99 ↑ (B)	66.54×10^{-4} ↓ (A)
	± 0.0188	± 200.75	$\pm 95.86 \times 10^{-4}$	± 0.0091	± 44.88	$\pm 30.87 \times 10^{-4}$
Standard XCS	0.1924 (A)	1551.57 (C)	253.48×10^{-4} (A)	0.1045 (A)	2386.63 (C)	74.49×10^{-4} (A)
	± 0.0146	± 144.38	$\pm 97.68 \times 10^{-4}$	± 0.0157	± 53.72	$\pm 37.77 \times 10^{-4}$
Eggholder 2D $r_0=1.0$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.1258 ↓ (B)	2739.35 ↑ (A)	161.86×10^{-4} ↓ (B)	0.0802 ↓ (C)	3396.52 ↑ (A)	45.22×10^{-4} ↓ (B)
	± 0.0093	± 178.73	$\pm 64.87 \times 10^{-4}$	± 0.0024	± 53.16	$\pm 16.23 \times 10^{-4}$
XCSF w/ Specify	0.2264 ↓ (A)	886.23 ↑ (B)	731.23×10^{-4} ↓ (A)	0.1259 ↓ (B)	2360.74 ↑ (B)	245.56×10^{-4} ↓ (A)
	± 0.0102	± 248.20	$\pm 242.53 \times 10^{-4}$	± 0.0198	± 120.58	$\pm 116.87 \times 10^{-4}$
Standard XCS	0.2280 (A)	862.97 (B)	736.51×10^{-4} (A)	0.1642 (A)	2092.89 (C)	296.71×10^{-4} (A)
	± 0.0084	± 220.80	$\pm 238.47 \times 10^{-4}$	± 0.0383	± 138.40	$\pm 164.56 \times 10^{-4}$
Sine-in-Sine 2D $r_0=0.1$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0836 ↑ (A)	3612.79 ↑ (A)	8.13×10^{-4} ↓ (C)	0.0648 ↑ (A)	3624.22 ↑ (A)	6.15×10^{-4} ↓ (C)
	± 0.0014	± 33.67	$\pm 0.12 \times 10^{-4}$	± 0.0008	± 21.50	$\pm 0.04 \times 10^{-4}$
XCSF w/ Specify	0.0825 ↓ (B)	2529.79 ↑ (B)	11.06×10^{-4} ↓ (B)	0.0570 ↓ (B)	2679.26 ↑ (B)	7.69×10^{-4} ↓ (B)
	± 0.0017	± 19.79	$\pm 0.22 \times 10^{-4}$	± 0.0006	± 9.65	$\pm 0.07 \times 10^{-4}$
Standard XCS	0.0833 (AB)	2432.36 (C)	11.44×10^{-4} (A)	0.0572 (B)	2520.05 (C)	8.06×10^{-4} (A)
	± 0.0015	± 22.73	$\pm 0.22 \times 10^{-4}$	± 0.0005	± 14.86	$\pm 0.08 \times 10^{-4}$
Sine-in-Sine 2D $r_0=0.5$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.1608 ↓ (C)	3214.01 ↑ (A)	42.84×10^{-4} ↓ (B)	0.0936 ↓ (C)	3517.84 ↑ (A)	15.13×10^{-4} ↓ (B)
	± 0.0125	± 73.43	$\pm 7.78 \times 10^{-4}$	± 0.0033	± 24.13	$\pm 1.94 \times 10^{-4}$
XCSF w/ Specify	0.2270 ↓ (B)	2016.02 ↑ (B)	88.58×10^{-4} ↓ (A)	0.1006 ↓ (B)	2594.14 ↑ (B)	27.71×10^{-4} ↓ (A)
	± 0.0407	± 118.26	$\pm 30.97 \times 10^{-4}$	± 0.0106	± 27.82	$\pm 7.82 \times 10^{-4}$
Standard XCS	0.2636 (A)	1758.61 (C)	96.23×10^{-4} (A)	0.1108 (A)	2333.62 (C)	29.98×10^{-4} (A)
	± 0.0524	± 124.69	$\pm 33.13 \times 10^{-4}$	± 0.0142	± 35.29	$\pm 8.45 \times 10^{-4}$
Sine-in-Sine 2D $r_0=1.0$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.2056 ↓ (B)	2922.40 ↑ (A)	87.44×10^{-4} ↓ (B)	0.1070 ↓ (C)	3459.38 ↑ (A)	26.39×10^{-4} ↓ (B)
	± 0.0216	± 147.76	$\pm 24.62 \times 10^{-4}$	± 0.0054	± 46.50	$\pm 6.16 \times 10^{-4}$
XCSF w/ Specify	0.4677 ↓ (A)	1217.04 ↑ (B)	402.45×10^{-4} ↓ (A)	0.1824 ↓ (B)	2393.49 ↑ (B)	117.85×10^{-4} ↓ (A)
	± 0.1058	± 402.12	$\pm 200.91 \times 10^{-4}$	± 0.0663	± 133.02	$\pm 75.60 \times 10^{-4}$
Standard XCS	0.5069 (A)	1044.47 (B)	415.34×10^{-4} (A)	0.2216 (A)	2113.95 (C)	127.08×10^{-4} (A)
	± 0.0837	± 279.86	$\pm 193.41 \times 10^{-4}$	± 0.0828	± 109.25	$\pm 78.56 \times 10^{-4}$

Entries are defined analogously to Table 1

can be found for 6-RMP¹, CBP(3,3)² and Mario³ in [27, 30, 36], respectively. For CBP(3,3), a value of 0.5 was selected for r_0 in deviation from [30] to artificially create a tendency of XCSR to over-generalize in this problem.

According to Table 1, the application of Absumption or Specify in all considered problems is accompanied by a significant increase of the population size and by a significant reduction of the generality, except for CBP(3,3). This can be easily explained by the underlying concept of OGH, as the decomposition of over-general classifiers can lead to many redundant over-specific classifiers. The significantly improved generality in CBP(3,3) can be possibly attributed to OGH enabling XCSR to solve the problem from the over-specific side, facilitating the formation of accurate classifiers with increased generality. Regarding the reward and system error metrics, in case of 6-RMP and CBP(3,3), XCSR benefits neither from Specify nor from Absumption. Compared to standard XCS, the application of Specify results in a non-significantly different reward value and an increased system error. In contrast, the application of Absumption reduces the reward value and leads to the highest system error of all configurations. One possible explanation of the results is the basic structure of these problems posing a challenge to OGH. The equally sized niches with strictly defined boundaries require the intervals of a classifier condition to be accurately inside a niche in order for a classifier not to be regarded as over-general. Due to the random decomposition of identified over-general classifiers, it cannot be guaranteed that newly created conditions will be contained inside a niche. Especially for Absumption, this effect seems to have a major impact, as the strict evaluation of inaccuracies and their subsequent treatment seem to reduce the learning speed, as shown in the learning curve diagrams in Fig. 1a, b. For Specify, this problem seems less affecting, which may be due to the indirect identification of at most one over-general classifier per iteration. It would therefore be of interest to realize a more focused specialization pressure by OGH, which will be subject of future research. In Mario, the results are quite different, as this problem enables different generalizations and is known to lead to over-general classifiers due to its more inconsistent underlying structure consisting of unequal-sized niches. Both OGH approaches increase the performance of XCSR and lead to a significant improvement of the reward and the system error metrics compared to the standard configuration. Special attention is

drawn to Absumption, which significantly outperforms all other configurations in the aforementioned metrics. This is also reflected in the learning curves of Mario (cf. Fig. 1c) showing a significantly increased learning speed by OGH, especially by Absumption. Thus, OGH seems to enable a significant refinement of the learned model of a problem affected by over-general classifiers, resulting in increased accuracy and performance.

To summarize, in our experiments, the application of OGH in XCSR mainly provides significant improvements regarding the system error and reward metrics for problems characterized by the formation of over-general classifiers and no significant benefits for other problems. It is also associated with an increase in population size and, in most cases, a decrease in generality in our experiments.

Results in Real-World Problems

In addition, an evaluation with real-world data was performed to assess the effects of OGH when applied to XCSR in real-world settings. Since the authors are conducting research on the use of AI in agricultural applications, three available data sets covering different agricultural domains were selected for evaluation: (1) *Paddy Leaf*, (2) *Horse Colic* and (3) *Soybean Disease*. The *Paddy Leaf*⁴ data set from Kaggle represents a multi-class classification task based on the average RGB values of 6000 paddy leaf pictures used to perform nitrogen fertilizer recommendation. The data set is balanced and consists of 3 real-valued attributes, indicating the average color channel values of red, green and blue present in a picture, and 4 different labels. The task of the *Horse Colic*⁵ data set from the UCI repository is a binary classification, whether a lesion of a horse suffering from a colic was surgical or not based on different health attributes. The data set is unbalanced and consists of 368 instances with 21 different health attributes, either real-valued, integer-valued or boolean, and a total of 30% missing attribute entries. In case of missing entries, each attribute has been assigned a default value not being within the value range of the respective attribute. The last data set is *Soybean Disease*⁶ from the UCI repository. The task is to classify 19 different soybean diseases from 35 integer-valued marker attributes also containing missing entries, handled analogously to the *Horse Colic* data set. The data set consists of 683 instances and is highly unbalanced. W.l.o.g., we normalized the attributes to the range [0, 1] for all data sets.

Analogously to the benchmark problems before, a single repetition of an experiment was run over 50,000 alternating

¹ 6-RMP: $N = 800$, $\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\nu = 5$, $\theta_{mna} = 2$, $\theta_{GA} = 12$, $\theta_{del} = 20$, $\theta_{sub} = 20$, $\epsilon_0 = 10$, $\chi = 0.8$, $\mu = 0.04$, $p_{ini} = 10.0$, $\epsilon_{ini} = 0.0$, $F_{ini} = 0.01$, $\epsilon_{red} = 0.25$, $F_{red} = 0.1$, $m_0 = 0.1$, $r_0 = 1.0$.

² CBP(3,3): Analogous to 6-RMP, except: $N = 2000$, $r_0 = 0.5$.

³ Mario: Analogous to 6-RMP, except: $N = 7000$, $\beta = 0.3$, $\theta_{mna} = 6$, $\theta_{GA} = 30$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $r_0 = 0.1$.

⁴ <https://www.kaggle.com/torikul140129/paddy-leaf-images-aman> (05.11.2020).

⁵ <https://data.world/uci/horse-colic> (05.11.2020).

⁶ <https://data.world/uci/soybean-large> (05.11.2020).

explore/exploit steps. In each step, an instance is drawn uniformly at random from the used data set with replacement. We also applied a binary reward scheme, i.e., a reward of 1000 for each correctly predicted action, i.e., class, otherwise 0. For the evaluation experiments with real-world data sets, a standard parameterization⁷ of XCSR was used. The conditions were encoded by unordered bound hyperrectangular representation. θ_{mna} was set depending on the number of available classes: In Paddy Leaf it was set to 4, in Horse Colic to 2 and in Soybean Disease to 19. Due to the high number of attributes and available classes in Soybean Disease, $N = 25,000$ to increase the population size. The experiments for the three real-world data sets shared the same configuration for OGH: The specialization strategy ICS was employed by both approaches, the *Inconsistency* identification strategy was applied in Absumption and, for Specify, $\theta_{OGCIdent}$ was set to 50.

Considering the results in Table 1, the application of Absumption results in significant performance improvements regarding the reward and system error metrics in all conducted experiments. It leads to a significant increase in learning performance and superior results compared to the other configurations. As shown in the learning curve plots in Fig. 1d–f, Absumption also causes a faster and more accurate learning of the problems, which is most evident in Paddy Leaf. Specify also leads to advantages in Paddy Leaf in terms of faster learning and a significant improvement of the reward and system error metrics, however, less pronounced compared to Absumption. In the other data sets, Specify does not yield any advantage, possibly due to the just indirect identification of over-general classifiers. Compared to the benchmark problems, only in one real-world data set, i.e., Paddy Leaf, OGH shows a significant increase in population size and a significant reduction in generality. Similarly to the benchmark problems, this can be attributed to the underlying concept of OGH and is more pronounced in case of Absumption. For the other two data sets, Specify causes a significantly increased population size, unlike Absumption. This can be attributed to the direct identification of over-general classifiers by Absumption, only performing necessary decompositions, which is not guaranteed in case of Specify. Regarding generality, there are no significant differences between the configurations in Horse Colic and Soybean Disease.

In summary, for real-world data sets, the application of OGH, especially Absumption, in XCSR causes significant improvements of the reward and system error metrics in our experiments. The increase in population size and decrease

in generality due to OGH was less pronounced in our experiments. Based on the results, we expect OGH also provides advantages beyond the agricultural domain for problems featuring characteristics comparable to the applied data sets, such as unbalanced data or missing entries. However, we left this for future work.

Results in Multi-step Problems

To preliminarily evaluate the potential of OGH for multi-step problems, we conducted experiments in two differently configured variants of the *GridWorld* environment, which is based on the *Puddles* environment introduced in [19]. The problem space of GridWorld is two dimensional and each dimension x_i is defined in the interval $0.0 \leq x_i < 1.0$. In GridWorld, the task to be learned is to reach the goal in position (1, 1) in as few steps as possible. In each episode the agent starts in a random position within the environment except the goal. The agent is allowed to move within the environment with a given step size in four directions, i.e., left, right, up and down. For each step taken, the agent receives a negative reward or punishment of -0.5, except the step leading to the goal, resulting in a reward of 0. The environment also contains so-called puddles leading to an additional punishment of -2 for each puddle the agent is in. An episode ends either if the goal has been reached or if 200 steps have been taken.

In our experiments, we applied a step size of 0.07 and 0.05 denoted as GridWorld(0.07) and GridWorld(0.05), respectively. Each repetition of the experiments was run over 10,000 episodes. As a fixed number of episodes is defined for the repetitions, the configurations and repetitions differ in the number of steps. Thus, the results of the metrics already used in the single-step experiments are determined over the first 205,000 steps and 285,000 steps for GridWorld(0.07) and GridWorld(0.05), respectively. In addition to these metrics, the performance of XCSR is evaluated over the entire repetition run using the steps to goal metric, which calculates as the 100-episode mean steps required to reach the goal. XCSR only used explore steps in combination with an ϵ -decay action selection regime, parameterized as follows: $\epsilon = 1.0$, $\epsilon_{fin} = 0.02$ and a decay fraction of 10% of the episodes, i.e., in the reported experiments the ϵ was decayed from 1.0 to 0.02 over the first 1000 episodes. GridWorld(0.07) and GridWorld(0.05) shared the same parameterization⁸, based on the settings for Puddles (0.1) in [19]. The conditions of the classifiers were encoded by unordered bound hyperrectangular

⁷ Analogous to 6-RMP, except: $N = 6400$, $\theta_{GA} = 48$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\epsilon_0 = 1.0$, $\epsilon_{red} = 1.0$, $m_0 = 0.5$, $r_0 = 1.0$.

⁸ Analogous to 6-RMP, except: $N = 10,000$, $\gamma = 0.95$, $\theta_{mna} = 4$, $\theta_{GA} = 50$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\epsilon_0 = 0.005$, $m_0 = 0.25$, $r_0 = 0.5$.

representation. The classifiers employed a computed prediction based on a linear prediction model using *recursive least squares* [20] (RLS) with a parameterization of $\lambda_{RLS} = 1.0$ and $\delta_{RLS} = 1.0$. The settings of OGH were determined in preliminary experiments: For both experiments, Absumption employed the *Prediction Deviation Variance* identification strategy with a FIFO-buffer size of 200 and both OGH approaches used the specialization strategy ICS as well as a $\theta_{OGCIdent}$ of 100.

As can be seen from Table 2, XCSR using OGH significantly outperforms the standard configuration in terms of the steps to goal, reward and system error metrics in both configurations of GridWorld. Analogously to Mario and the real-world problems, Absumption significantly surpasses Specify in terms of the aforementioned metrics, except for the reward and system error metrics in GridWorld(0.07). The learning curve of GridWorld(0.05) in Fig. 1h shows a faster reduction of the required steps by the OGH approaches compared to the standard configuration, with Absumption being clearly superior to Specify. In GridWorld(0.07) (cf. Fig. 1g) the difference between OGH and the standard configuration is less pronounced, since GridWorld(0.07) already seems to be well solvable by the standard configuration. The significant improvement of the steps to goal, the reward and the system error metrics indicates the application of both Specify and Absumption in GridWorld enables XCSR to evolve a more accurate model of the underlying problem faster. Absumption provides additional benefits since it is superior to Specify. In terms of population size and generality, both OGH approaches cause a significant increase in population size and a significant decrease in generality. Once again, this can be attributed to the underlying concept of OGH, the decomposition of over-general classifiers. The intensified occurrence of this effect in Absumption can be explained by the direct identification of over-general classifiers, resulting in more classifiers being decomposed per iteration of XCSR compared to Specify.

In conclusion, the use of OGH and especially Absumption results in significantly improved steps to goal, reward and system error metrics in our experiments in multi-step environments, indicating faster learning of an accurate model. However, due to the mode of operation of OGH, these advantages are accompanied by an increase in population size and a decrease in generality in our experiments.

Results in Regression Tasks

This part of the evaluation focuses on applying OGH in XCSF when solving regression tasks from the field of global optimization. The reported results regarding regression problems were obtained by evaluations on three test

functions⁹ already applied in the literature for XCSF assessment:

$$\begin{aligned} f_1(x_1, x_2) &= -(x_2 + 47) \sin\left(\sqrt{\left|\frac{x_1}{2} + (x_2 + 47)\right|}\right) - \\ &\quad x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right), -512 \leq x_1, x_2 \leq 512 \\ f_2(x_1, x_2) &= \sin(4\pi(x_1 + \sin(\pi x_2))), 0 \leq x_1, x_2 \leq 1 \\ f_3(x) &= \max\{\exp(-10a^2), \exp(-50b^2), 1.25 \exp(-5(a^2 + b^2))\}, \\ a &= \frac{1}{\lfloor \frac{n}{2} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_i, b = \frac{1}{\lceil \frac{n}{2} \rceil} \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^n x_i, -1 \leq x_i \leq 1 \end{aligned}$$

The first function f_1 , a common 2-dimensional benchmark in global optimization called *Eggholder* function [13], proved to be quite complex to approximate with XCSF in [26] due to its strong and repetitive curvature and multi-modality in both dimensions. These properties are also featured by the 2-dimensional *Sine-in-Sine* function f_2 [24], which thereby exhibits a higher degree of variation of the curvature and was applied for XCSF assessment in [24, 25]. The next function f_3 , referred to as *Cross* function [25], was chosen for the analysis concerning the approximation capacity of XCSF in [25] and is characterized by its mixture of linear and nonlinear subspaces. In this work, we apply f_3 in its 3-dimensional shape based on the generalized version proposed in [23].

Each experiment repetition features 200,000 explore steps, or learning steps in context of XCSF. In each step, the test functions' input space was randomly sampled following a uniform distribution. The settings of OGH were determined in preliminary experiments: For both experiments, a $\theta_{OGCIdent}$ of 100 as well as the specialization strategy ICS were applied for both OGH approaches and Absumption employed the *Prediction Deviation Variance* identification strategy with a FIFO-buffer size of 200. The intervals of a classifier condition were encoded by the unordered bound hyperrectangular representation. In XCSF, classifiers learn a model for predicting the payoff as a function of the input instead of simply holding a scalar estimate. In this paper, we applied both a linear prediction model using RLS to update the prediction values and a *radial basis function* (RBF) based interpolation approach as introduced in [29]. RLS was applied using a parameterization of $\lambda_{RLS} = 1.0$ and $\delta_{RLS} = 1.0$ and RBF using a *Thin-Plate-Spline* basic function, defined as $\theta(r) = r^2 \log(r)$. To interpolate $cl.p$, in each experiment, the classifiers contain a set $cl.Sp$ of up to 50 samples s_i representing situation-reward pairs, more formally defined as $s_i = (x_i, r_i)$. x_i denotes an input or feature vector

⁹ W.l.o.g., we normalized the domains of all functions to $\forall_i x_i \in [0, 1]$. The co-domains of the functions were rescaled to $y \in [-1, 1]$.

Table 4 Overall results of conducted experiments on Cross function using RLS-based predictions, i.e., system error, population size and generality

Over	First 50,000 steps			Entire run		
Cross 3D $r_0 = 0.1$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0588 $\uparrow$ (A)	4037.26 $\uparrow$ (A)	2.22×10^{-4} $\downarrow$ (B)	0.0360 $\uparrow$ (A)	4268.92 $\uparrow$ (A)	2.83×10^{-4} $\downarrow$ (C)
	± 0.0010	± 37.99	$\pm 0.04 \times 10^{-4}$	± 0.0006	± 88.50	$\pm 0.04 \times 10^{-4}$
XCSF w/ Specify	0.0553 (B)	3718.01 $\uparrow$ (B)	2.57×10^{-4} (A)	0.0297 (B)	3403.34 $\uparrow$ (B)	3.74×10^{-4} $\downarrow$ (B)
	± 0.0010	± 21.11	$\pm 0.07 \times 10^{-4}$	± 0.0003	± 15.46	$\pm 0.05 \times 10^{-4}$
Standard XCS	0.0553 (B)	3710.87 (B)	2.57×10^{-4} (A)	0.0297 (B)	3375.77 (B)	3.78×10^{-4} (A)
	± 0.0011	± 26.25	$\pm 0.06 \times 10^{-4}$	± 0.0003	± 21.71	$\pm 0.05 \times 10^{-4}$
Cross 3D $r_0 = 0.5$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0467 $\downarrow$ (C)	4084.22 $\uparrow$ (A)	8.13×10^{-4} $\downarrow$ (C)	0.0334 $\uparrow$ (A)	4090.23 $\uparrow$ (A)	4.82×10^{-4} $\downarrow$ (C)
	± 0.0016	± 59.04	$\pm 0.97 \times 10^{-4}$	± 0.0006	± 70.31	$\pm 0.29 \times 10^{-4}$
XCSF w/ Specify	0.0504 $\downarrow$ (B)	2949.38 $\uparrow$ (B)	17.06×10^{-4} $\downarrow$ (B)	0.0302 $\downarrow$ (C)	3122.14 $\uparrow$ (B)	8.05×10^{-4} $\downarrow$ (B)
	± 0.0024	± 57.71	$\pm 1.99 \times 10^{-4}$	± 0.0006	± 27.81	$\pm 0.53 \times 10^{-4}$
Standard XCS	0.0540 (A)	2728.71 (C)	18.97×10^{-4} (A)	0.0312 (B)	2976.99 (C)	8.78×10^{-4} (A)
	± 0.0032	± 63.28	$\pm 2.32 \times 10^{-4}$	± 0.0010	± 22.19	$\pm 0.66 \times 10^{-4}$
Cross 3D $r_0 = 1.0$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0609 $\downarrow$ (C)	3863.85 $\uparrow$ (A)	20.72×10^{-4} $\downarrow$ (C)	0.0384 $\downarrow$ (B)	4027.68 $\uparrow$ (A)	8.11×10^{-4} $\downarrow$ (C)
	± 0.0049	± 92.93	$\pm 4.91 \times 10^{-4}$	± 0.0014	± 56.78	$\pm 1.24 \times 10^{-4}$
XCSF w/ Specify	0.0811 $\downarrow$ (B)	2565.82 $\uparrow$ (B)	53.72×10^{-4} $\downarrow$ (B)	0.0395 $\downarrow$ (B)	3062.49 $\uparrow$ (B)	17.53×10^{-4} $\downarrow$ (B)
	± 0.0116	± 168.78	$\pm 18.66 \times 10^{-4}$	± 0.0029	± 40.79	$\pm 4.70 \times 10^{-4}$
Standard XCS	0.1017 (A)	2125.67 (C)	67.64×10^{-4} (A)	0.0468 (A)	2818.22 (C)	21.84×10^{-4} (A)
	± 0.0170	± 167.41	$\pm 22.35 \times 10^{-4}$	± 0.0057	± 53.10	$\pm 5.97 \times 10^{-4}$

Entries are defined analogously to Table 1

at time t and r_t denotes the associated reward or function value. To ensure robust interpolation, during interpolation of $cl.p$ for a situation $\mathbf{x}$, it is checked whether the interpolated function value $f(\mathbf{x})$ satisfies $r_{min} < f(\mathbf{x}) < r_{max}$, where r_{max} and r_{min} denote the largest and smallest reward so far observed among all $s_i \in cl.Sp$. If the conditions are met, $f(\mathbf{x})$ is chosen as $cl.p$, otherwise *Nearest Neighbor Interpolation* is applied as fallback. The mixing strategy used in our experiments to compute the system prediction based on the considered classifier predictions is analogous to the strategy used in [29]. Further hyperparameters¹⁰ of XCSF were taken from the literature, in particular [29]. To assess the presence of a specialization pressure by OGH in XCSF, a tendency to over-generalize was artificially created in XCSF. For this purpose, different levels for the initial spread value (r_0) for newly generated conditions were selected, namely 0.1, 0.5, and 1.0, which reflect a stronger tendency to over-generalization with increasing values. Usually, smaller r_0 values are chosen only in case of existing prior knowledge about the

environment and corresponding necessity, since a too small choice unnecessarily generates many transient, redundant classifiers. Thus, especially for environments with little prior knowledge, there is a more severe over-generalization problem due to the default choice of larger r_0 values. In Tables 3, 4, 5 and 6, the r_0 values used in the different experiments are provided with the respective results. Also a lower bound for r_0 of 0.005 was applied, i.e., a minimal spread was applied to each newly created classifier.

XCSF Using RLS Prediction

As shown in Tables 3 and 4, representing the results for the experiments using RLS predictions, in case of Eggholder, Absumption is significantly superior to the other configurations in the first 50,000 steps in each tested experiment setting. Specify performs worse compared to Absumption, but also causes significantly improved levels at r_0 values of 0.1 and 0.5. Over the entire run, this situation changes slightly: ultimately, Absumption leads to a significant reduction in system error in case of the higher r_0 values, 0.5 and 1.0. For $r_0 = 0.1$, Absumption causes a slight increase of the system error. Therefore, it can be seen that Absumption can realize a specialization pressure in Eggholder in case of higher r_0

¹⁰ $N = 6400$, $\alpha = 1$, $\beta = 0.1$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 50$, $\theta_{del} = 20$, $\theta_{sub} = 20$, $\epsilon_0 = 0.01$, $\chi = 1.0$, $\mu = 0.04$, $p_{ini} = 10.0$, $\epsilon_{ini} = 0.0$, $F_{ini} = 1.0$, $\epsilon_{red} = 1.0$, $F_{red} = 0.1$, $m_0 = 0.1$.

Fig. 2 Learning curve plots of the conducted experiments using XCSF

values and, thus, can capture the existing multi-modality and the strong curvature of this test function. Nevertheless, Absumption seems to act too strictly in case of low r_0 values, possibly generating more inaccurate classifiers due to the random decomposition of over-general classifiers in OGH.

Specify, in contrast, leads to a significant reduction in system error at all r_0 values tested, though it does not quite achieve the level of Absumption in case of $r_0 = 0.5$ and 1.0 . This is probably due to the random decomposition of only one classifier in the action set that is considered to be over-general

Table 5 Results of conducted experiments on Eggholder function and Sine-in-Sine function using interpolation-based predictions, i.e., system error, population size and generality

Over	First 50,000 steps			Entire run		
Eggholder 2D $r_0 = 0.1$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0101 $\uparrow$ (A) ± 0.0096	3831.90 $\uparrow$ (A) ± 29.69	$12.68 \times 10^{-4} \downarrow$ (B) $\pm 0.25 \times 10^{-4}$	0.0043 $\uparrow$ (A) ± 0.0024	3967.92 $\uparrow$ (A) ± 19.54	$11.45 \times 10^{-4} \downarrow$ (C) $\pm 0.17 \times 10^{-4}$
XCSF w/ Specify	0.0100 (A) ± 0.0096	3357.94 $\uparrow$ (B) ± 31.80	$16.85 \times 10^{-4} \downarrow$ (A) $\pm 0.28 \times 10^{-4}$	0.0040 (A) ± 0.0024	3534.27 $\uparrow$ (B) ± 14.47	$15.78 \times 10^{-4} \downarrow$ (B) $\pm 0.18 \times 10^{-4}$
Standard XCS	0.0100 (A) ± 0.0096	3345.81 (B) ± 26.97	16.91×10^{-4} (A) $\pm 0.35 \times 10^{-4}$	0.0040 (A) ± 0.0024	3502.88 (C) ± 14.75	16.00×10^{-4} (A) $\pm 0.20 \times 10^{-4}$
Eggholder 2D $r_0 = 0.5$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0116 $\downarrow$ (B) ± 0.0013	3687.17 $\uparrow$ (A) ± 43.81	$30.21 \times 10^{-4} \downarrow$ (C) $\pm 4.75 \times 10^{-4}$	0.0047 $\downarrow$ (B) ± 0.0003	3935.62 $\uparrow$ (A) ± 18.49	$16.52 \times 10^{-4} \downarrow$ (C) $\pm 1.26 \times 10^{-4}$
XCSF w/ Specify	0.0135 $\downarrow$ (B) ± 0.0018	2891.88 $\uparrow$ (B) ± 58.17	$52.45 \times 10^{-4} \downarrow$ (B) $\pm 9.20 \times 10^{-4}$	0.0050 $\downarrow$ (B) ± 0.0005	3428.07 $\uparrow$ (B) ± 20.46	$25.32 \times 10^{-4} \downarrow$ (B) $\pm 2.42 \times 10^{-4}$
Standard XCS	0.0184 (A) ± 0.0049	2522.59 (C) ± 130.49	66.72×10^{-4} (A) $\pm 15.74 \times 10^{-4}$	0.0063 (A) ± 0.0013	3299.97 (C) ± 35.10	29.10×10^{-4} (A) $\pm 4.09 \times 10^{-4}$
Eggholder 2D $r_0 = 1.0$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0195 $\downarrow$ (B) ± 0.0073	3505.95 $\uparrow$ (A) ± 151.83	$75.63 \times 10^{-4} \downarrow$ (C) $\pm 44.77 \times 10^{-4}$	0.0066 $\downarrow$ (B) ± 0.0018	3895.19 $\uparrow$ (A) ± 43.44	$28.07 \times 10^{-4} \downarrow$ (C) $\pm 11.27 \times 10^{-4}$
XCSF w/ Specify	0.0256 $\downarrow$ (B) ± 0.0118	2693.70 $\uparrow$ (B) ± 248.84	$130.76 \times 10^{-4} \downarrow$ (B) $\pm 75.25 \times 10^{-4}$	0.0080 $\downarrow$ (B) ± 0.0030	3396.30 $\uparrow$ (B) ± 60.10	$45.36 \times 10^{-4} \downarrow$ (B) $\pm 18.91 \times 10^{-4}$
Standard XCS	0.0454 (A) ± 0.0208	1925.71 (C) ± 323.63	182.30×10^{-4} (A) $\pm 96.63 \times 10^{-4}$	0.0133 (A) ± 0.0058	3161.03 (C) ± 93.82	58.48×10^{-4} (A) $\pm 24.81 \times 10^{-4}$
Sine-in-Sine 2D $r_0 = 0.1$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0080 $\uparrow$ (A) ± 0.0177	3957.79 $\uparrow$ (A) ± 46.92	$11.84 \times 10^{-4} \downarrow$ (B) $\pm 0.32 \times 10^{-4}$	0.0023 $\uparrow$ (A) ± 0.0044	4145.36 $\uparrow$ (A) ± 34.61	$10.81 \times 10^{-4} \downarrow$ (C) $\pm 0.24 \times 10^{-4}$
XCSF w/ Specify	0.0079 (A) ± 0.0177	3226.44 $\uparrow$ (B) ± 18.73	$18.04 \times 10^{-4} \downarrow$ (A) $\pm 0.39 \times 10^{-4}$	0.0022 (A) ± 0.0044	3377.42 $\uparrow$ (B) ± 14.83	$18.15 \times 10^{-4} \downarrow$ (B) $\pm 0.23 \times 10^{-4}$
Standard XCS	0.0079 (A) ± 0.0177	3189.23 (C) ± 21.19	18.21×10^{-4} (A) $\pm 0.37 \times 10^{-4}$	0.0022 (A) ± 0.0044	3317.46 (C) ± 14.33	18.61×10^{-4} (A) $\pm 0.23 \times 10^{-4}$
Sine-in-Sine 2D $r_0 = 0.5$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0078 $\downarrow$ (B) ± 0.0015	3831.96 $\uparrow$ (A) ± 56.84	$26.21 \times 10^{-4} \downarrow$ (C) $\pm 3.06 \times 10^{-4}$	0.0022 $\downarrow$ (B) ± 0.0004	4097.45 $\uparrow$ (A) ± 36.31	$14.97 \times 10^{-4} \downarrow$ (C) $\pm 0.08 \times 10^{-4}$
XCSF w/ Specify	0.0086 $\downarrow$ (B) ± 0.0017	2762.44 $\uparrow$ (B) ± 37.99	$51.35 \times 10^{-4} \downarrow$ (B) $\pm 5.63 \times 10^{-4}$	0.0024 $\downarrow$ (B) ± 0.0004	3274.61 $\uparrow$ (B) ± 19.52	$27.36 \times 10^{-4} \downarrow$ (B) $\pm 1.61 \times 10^{-4}$
Standard XCS	0.0122 (A) ± 0.0040	2391.70 (C) ± 107.57	66.77×10^{-4} (A) $\pm 12.52 \times 10^{-4}$	0.0033 (A) ± 0.0010	3107.53 (C) ± 33.25	31.69×10^{-4} (A) $\pm 3.28 \times 10^{-4}$
Sine-in-Sine 2D $r_0 = 1.0$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0170 $\downarrow$ (B) ± 0.0070	3743.19 $\uparrow$ (A) ± 63.66	$52.78 \times 10^{-4} \downarrow$ (C) $\pm 17.98 \times 10^{-4}$	0.0045 $\downarrow$ (B) ± 0.0018	4084.19 $\uparrow$ (A) ± 28.45	$21.74 \times 10^{-4} \downarrow$ (C) $\pm 4.50 \times 10^{-4}$
XCSF w/ Specify	0.0203 $\downarrow$ (B) ± 0.0093	2793.09 $\uparrow$ (B) ± 69.98	$93.75 \times 10^{-4} \downarrow$ (B) $\pm 27.11 \times 10^{-4}$	0.0053 $\downarrow$ (B) ± 0.0023	3308.45 $\uparrow$ (B) ± 26.48	$38.70 \times 10^{-4} \downarrow$ (B) $\pm 6.95 \times 10^{-4}$
Standard XCS	0.0507 (A) ± 0.0332	1889.73 (C) ± 228.96	164.85×10^{-4} (A) $\pm 53.62 \times 10^{-4}$	0.0130 (A) ± 0.0087	2997.15 (C) ± 70.72	56.32×10^{-4} (A) $\pm 13.66 \times 10^{-4}$

Entries are defined analogously to Table 1

Table 6 Results of conducted experiments on Cross function using interpolation-based predictions, i.e., system error, population size and generality

Over	First 50,000 steps			Entire run		
Cross 3D $r_0 = 0.1$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0092 $\uparrow$ (A) ± 0.0020	4635.27 $\uparrow$ (A) ± 15.24	2.67×10^{-4} $\downarrow$ (B) $\pm 0.06 \times 10^{-4}$	0.0048 $\uparrow$ (A) ± 0.0005	4966.73 $\uparrow$ (A) ± 7.03	3.81×10^{-4} $\downarrow$ (B) $\pm 0.07 \times 10^{-4}$
XCSF w/ Specify	0.0089 (A) ± 0.0019	4488.75 $\uparrow$ (B) ± 16.59	3.21×10^{-4} $\downarrow$ (A) $\pm 0.08 \times 10^{-4}$	0.0044 (B) ± 0.0005	4562.88 $\uparrow$ (B) ± 10.53	5.30×10^{-4} $\downarrow$ (A) $\pm 0.10 \times 10^{-4}$
Standard XCS	0.0089 (A) ± 0.0019	4484.91 (B) ± 14.95	3.22×10^{-4} (A) $\pm 0.08 \times 10^{-4}$	0.0044 (B) ± 0.0005	4558.09 (B) ± 8.94	5.33×10^{-4} (A) $\pm 0.09 \times 10^{-4}$
Cross 3D $r_0 = 0.5$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0072 $\downarrow$ (A) ± 0.0006	4672.29 $\uparrow$ (A) ± 84.01	8.16×10^{-4} $\downarrow$ (B) $\pm 0.63 \times 10^{-4}$	0.0042 $\uparrow$ (A) ± 0.0002	4935.65 $\uparrow$ (A) ± 91.84	5.70×10^{-4} $\downarrow$ (B) $\pm 0.31 \times 10^{-4}$
XCSF w/ Specify	0.0072 $\downarrow$ (A) ± 0.0006	3950.02 $\uparrow$ (B) ± 68.55	14.83×10^{-4} $\downarrow$ (A) $\pm 1.50 \times 10^{-4}$	0.0040 (B) ± 0.0002	4381.05 $\uparrow$ (B) ± 24.61	8.92×10^{-4} $\downarrow$ (A) $\pm 0.46 \times 10^{-4}$
Standard XCS	0.0073 (A) ± 0.0006	3918.38 (B) ± 72.72	15.11×10^{-4} (A) $\pm 1.68 \times 10^{-4}$	0.0040 (B) ± 0.0002	4371.30 (B) ± 24.25	8.97×10^{-4} (A) $\pm 0.53 \times 10^{-4}$
Cross 3D $r_0 = 1.0$	Sys. Err.	$ [P] $	Generality	Sys. Err.	$ [P] $	Generality
XCSF w/ Abs	0.0092 $\downarrow$ (B) ± 0.0011	4496.16 $\uparrow$ (A) ± 86.37	16.13×10^{-4} $\downarrow$ (C) $\pm 3.45 \times 10^{-4}$	0.0047 $\downarrow$ (B) ± 0.0003	4859.02 $\uparrow$ (A) ± 97.55	7.95×10^{-4} $\downarrow$ (C) $\pm 0.93 \times 10^{-4}$
XCSF w/ Specify	0.0103 $\downarrow$ (B) ± 0.0011	3646.75 $\uparrow$ (B) ± 72.14	31.73×10^{-4} $\downarrow$ (B) $\pm 5.22 \times 10^{-4}$	0.0047 $\downarrow$ (B) ± 0.0003	4321.14 $\uparrow$ (B) ± 21.21	13.36×10^{-4} $\downarrow$ (B) $\pm 1.39 \times 10^{-4}$
Standard XCS	0.0127 (A) ± 0.0027	3327.59 (C) ± 181.77	38.08×10^{-4} (A) $\pm 8.26 \times 10^{-4}$	0.0054 (A) ± 0.0007	4231.62 (C) ± 52.21	15.22×10^{-4} (A) $\pm 2.28 \times 10^{-4}$

Entries are defined analogously to Table 1

and the following creation of only one new specialized classifier. Thus, Specify is less strict than Absumption. Looking at the learning curve plots in Fig. 2a, b, the experiments with r_0 values of 0.5 and 1.0 show that Absumption leads to a strong reduction of the system error, especially at the beginning. The configuration with Specify shows an increased reduction of the error only at a later time step of the experiment, but can significantly undercut the default configuration. In terms of population size, both OGH approaches lead to a significant increase in this metric in all cases over the entire course. This can be attributed to the decomposition of identified over-general classifiers, resulting in a larger number of transient classifiers. This decomposition effect can also be seen as the major cause for the reduced generality, as both OGH approaches cause a decrease in this metric value.

For the Sine-in-Sine function, the experiments show that the OGH approaches exert a positive effect on the system error only in case of the higher r_0 values tested, i.e., 0.5 and 1.0, and, thus, in case of a stronger over-generalization tendency. Already at the beginning of the experiment run, but also over the entire course of the experiment, Absumption leads to the largest reductions in the system error in these experiment configurations, followed by Specify showing also significant reductions in the metric. The learning curve

plot of the experiment with an r_0 value of 1.0 in Fig. 2c also illustrates the significant reduction in system error at the beginning of the experiment due to Absumption. Specify also shows a smaller but still significant reduction of the system error as the experiment progresses. In addition, Specify can even slightly undercut Absumption at the end of the experiment. Thus, also in case of Sine-in-Sine it becomes apparent that the OGH approaches and especially Absumption in case of the higher tested r_0 values enable XCSF to learn the underlying problem environment or test function more accurately. This equally suggests the presence of specialization pressure due to OGH. In case of the experiment configuration with $r_0 = 0.1$, Absumption leads to a slight deterioration of the system error and Specify leads to no significant improvement. Thus, there seems to be only a slight over-generalization tendency in this experiment configuration and Absumption performs worse due to its probably too strict mode of operation in combination with the random decomposition. As with Eggholder, both OGH approaches cause a significant increase in population size for Sine-in-Sine, which, like previously, is due to the increased generation of transient classifiers by the decomposition of over-general classifiers. Likewise, both OGH approaches lead to

a reduction in generality due to the inherent formation of specialized classifiers in OGH.

Similar to Sine-in-Sine, OGH does not lead to any improvement in the Cross function configuration with $r_0 = 0.1$, with Absumption again leading to a slight increase in system error. Even with $r_0 = 0.5$, Absumption causes a small but significant increase in system error over the entire run, although XCSF with Absumption has the lowest error value in the first 50,000 steps. Presumably, a too strict mode of operation of Absumption as well as the random-based decomposition in OGH is responsible for this result. Compared to Absumption, Specify can lead to significant reductions of the system error at $r_0 = 0.5$ as well as at $r_0 = 1.0$. Thus, also in this context advantages are shown by the less strict mode of operation of Specify. Only at $r_0 = 1.0$, XCSF with Absumption achieves a significant improvement in system error. These results are also reflected in the learning curve plot in Fig. 2d. Absumption causes the strongest reduction in system error at the beginning, but is undercut by Specify at about 50,000 steps. As before, Absumption causes a significant increase in population size as well as a significant reduction in generality for all r_0 values examined. In case of Specify, a significant increase in population size or reduction in generality occurs only at $r_0 = 1.0$, in the other cases Specify does not cause significant changes. The increase or reduction can probably again be attributed to an increased formation of redundant, specialized classifiers.

To conclude, the results in the investigated test functions show that OGH can realize a specialization pressure in case of RLS-based predictions and leads to improvements especially in configurations with increased over-generalization tendency. In this context, it becomes apparent that Specify leads to improvements in most cases or does not negatively affect the performance in the other cases. Absumption, on the other hand, has a greater potential for improvement in most cases due to its stricter mode of operation, but can also lead to a deterioration in performance.

XCSF Using Interpolated Prediction

In Tables 5 and 6, the results for the experiments with interpolated predictions show that OGH leads to significant improvements in the system error over the entire run for the experiment configurations with $r_0 = 0.5$ and 1.0 in case of Eggholder. This is also evident in the learning curve plots for these configurations in Fig. 2e, f. Absumption features a slightly faster reduction in system error at the beginning, but is quickly matched by Specify. At $r_0 = 0.1$, OGH leads to no significant change in system error. Thus, in case of higher over-generalization tendencies, OGH enables more accurate learning of the test function. In terms of population size for the experiments with interpolated prediction, OGH is also shown to lead to a significant increase in population

size, with Absumption showing the highest values. This is again probably due to the formation of transient, redundant classifiers. For generality, it is found that OGH leads to a significant reduction in this metric and Absumption has the lowest values. Thus, a specialization pressure by OGH is realized here as well, leading to the formation of more specialized classifiers. It is also shown that the performance of the applied interpolated predictions can be increased by the formation of specialized classifiers. The smaller conditions increase the density of the stored sampling points and, thus, an improvement of the interpolation performance can be achieved.

For the sine-in-sine function, an almost identical pattern as for the Eggholder function is obtained. OGH causes a significant reduction of system error in case of $r_0 = 0.5$ and 1.0 and no significant improvement in case of $r_0 = 0.1$. In Fig. 2g, the learning curve plot of the Sine-in-Sine experiment configuration with $r_0 = 1.0$ also shows a significant reduction of the system error due to both OGH approaches. The use of OGH also results in a significant increase in population size and a significant reduction in generality in Sine-in-Sine. Thus, this test function also confirms that both OGH approaches can realize a specialization pressure, which leads to increases in performance especially in the presence of a stronger over-generalization tendency.

In case of the Cross function, a significant reduction of the system error by both OGH approaches occurs at an r_0 value of 1.0, also underlined by the learning curve plot in Fig. 2h. Both OGH variants lead to a faster reduction of the system error at the beginning of the experiment. Considering the results for $r_0 = 0.1$ and 0.5, it can be seen that in this case Specify leads to no significant change in the system error, but Absumption causes a small but still significant increase in this metric. This can again probably be attributed to the stricter mode of operation of Absumption. Therefore, for the Cross function, once again the less strict mode of operation of Specify is shown to be beneficial. For population size and generality, Absumption leads to a significant increase and decrease, respectively, in all cases. In contrast, Specify causes a significant increase in population size or decrease in generality in case of $r_0 = 1.0$ and otherwise it does not lead to any significant changes. Thus, it appears that OGH is able to develop benefits by realizing the specialization pressure in the cross function using interpolated predictions only in case of strong over-generalization tendencies.

In summary, the experiments with the interpolation-based prediction show that also in this case a specialization pressure can be realized by OGH. Similar to the RLS-based predictions, the less strict operation of Specify shows advantages. In case of improvements by OGH, Specify can achieve similarly good values as Absumption, but in the other cases, unlike Absumption, it does not lead to any deterioration of the system error.

Key Findings and Discussion

The results of the evaluation experiment support the hypothesis that OGH is a viable approach to the problem of over-generalization in XCS-based systems for continuous-valued problem domains. As evident in the generality metric by the significantly lowered values, OGH is able to cause a specialization pressure in XCSR and XCSE.

In case of the classification tasks, the advantage of OGH becomes apparent when XCSR has to cope with the studied real-world data sets or the examined toy problem environments featuring unbalanced classes due to unequally sized environmental niches, i.e. the Mario environment. The results in the Mario environment indicate that OGH is able to prevent the formation of over-general classifiers caused by the unequally sized environmental niches. It also seems to enable XCS to more accurately represent small niches of an environment, such as the yellow buttons in the Mario environment, indicating a more accurate state-action map.

This indicates that OGH can improve the online learning process of XCSR. In XCSR, a local learning is performed as local submodels in the form of the classifiers are trained instance by instance for the underlying classification task in this learning process. Using OGH, these local models become more fine-grained, allowing for a more accurate adaptation to the classification task. Also, updates to the corresponding submodels affect smaller areas of the entire learned model, which reduces unnecessary updates and increases the accuracy of the overall model. Thus, OGH enables an improvement of the adaptive modeling capability present in XCSR, which results from the online learning process as well as local learning. Furthermore, the accelerated learning of the classification tasks in the real world data sets, especially in Paddy Leaf, due to OGH reveals that OGH can lead to an increase in data efficiency. Early in the learning process, OGH can prevent the formation of over-general classifiers that lead to an incorrect representation of the underlying problem, adversely affect classifiers that are actually correct, and ultimately slow down and hinder the learning process. Thus, using OGH, XCS requires fewer data samples in order to achieve high system performance.

In the toy problems featuring balanced classes, i.e. 6-RMP or CBP(3,3), the OGH approaches lead to no improvement of the overall learning performance. Unexpectedly, OGH causes even the highest generality in the Checkerboard environment. Since the Checkerboard environment lacks sufficient specialization pressure, this seems to be due to the specialization pressure by OGH. This pressure leads to the problem being solved rather from the over-specific side, which facilitates XCS to generate accurate and maximally-general classifiers. However, OGH seems to degrade the accuracy of the learned model in these environments, as it

causes an increase of the system error. This may be due to the random creation of specific versions of the conditions by NCS which may result in an unfocused specialization pressure. The cause of the deterioration in accuracy should be investigated more closely in future work, as should other specialization strategies that enable more focused specialization pressure. Nonetheless, OGH seems to be a distinct advantage in classification problems with more complex structures, such as the Mario environment and probably the Checkerboard environment with more divisions per dimension.

Similarly, for XCSE, the pressure of specialization by OGH leads to performance improvements in solving the studied regression tasks. Especially for configurations that have a stronger tendency to form over-general classifiers due to having a large value for r_0 , the specialized classifiers, generated due to the specialization pressure of OGH, led to a more accurate modeling of the respective test function and, thus, a significant reduction of the system error. Especially when using linear approximators as classifier prediction, the more fine-grained local models allow a significantly better representation of the high multi-modality as well as the strong and repetitive curvature in the Eggholder function and the Sine-in-Sine function. Also, the obliqueness of the Cross function was better reflected when using OGH. For the interpolated classifier predictions, the increase in performance was mainly accompanied by significant reductions in generality. Thus, the specialization pressure of OGH causes an increase in the density of sampling points within the interpolated predictions. This enables the interpolation function to better approximate the studied function. Nevertheless, it also turned out that in case of a presumably already sufficient density, e.g. in the case of the configurations with $r_0 = 0.1$, no further increases of the learning performance are obtained, since a sufficiently accurate approximation can be achieved apparently by means of the interpolated classifier predictions in the standard configuration. Still, especially in the absence of knowledge about the function under investigation and a consequently larger choice of the r_0 value, OGH provides a good approach to obtain a more accurate model of the function under investigation.

The statements for the regression tasks also apply to the results of XCSR in the GridWorld environment, since in this case, analogous to XCSE, a function is approximated, i.e., the Q-function underlying this multi-step environment or rather reinforcement learning task. Due to the more fine-grained local models, the Q-function, which can be highly non-linear and complex in structure, is approximated more accurately, resulting in a more appropriate choice of actions in the environment. Thus, the increased specialization pressure due to OGH significantly improves the performance of XCSR, as the further reduction of the number of required

steps can be attributed to the further refinement of the created state-action map. The refined map leads to a more accurate selection of suitable actions. This is also reflected by the rapidly increasing reward and the rapidly decreasing error, which indicate that OGH enables XCS to quickly predict short and puddle-avoiding paths.

In summary, OGH is able to contribute a significant advantage for XCSR and XCSF in the studied classification tasks in the form of the toy problems and the real-world data sets, as well as the examined multi-step tasks and the studied regression tasks.

Conclusion

We presented two *Over-Generality Handling* (OGH) approaches adapted for *XCS with real-valued inputs* (XCSR) and *XCS for function approximation* (XCSF), one based on the Absumption mechanism and one based on the Specify operator. The presented approaches provide promising means to deal with the challenge of over-general classifiers arising in XCS-based systems designed for real-valued inputs. This is particularly evident in reinforcement learning settings that demand for long action-chains and only provide sparse reward signals.

To enable the use of OGH in XCSR and XCSF, two specialization strategies were introduced enabling a decomposition of over-general classifiers into more specific ones in real-valued problem space. A new identification strategy for over-general classifiers was proposed to enable the application of Absumption in multi-step problems. To fathom the potential of OGH, the Absumption- and Specify-based OGH was evaluated in multi-step problems and in single-step problems, i.e., classification tasks based on common benchmark problems and on real-world data sets.

The presented results of the conducted empirical studies showed the application of OGH results in considerable improvements in benchmark problems, tending towards the formation of over-general classifiers, in our experiments. Regarding the considered real-world data sets from the agricultural domain, especially the Absumption-based OGH caused significant improvements. In the evaluated multi-step problems, the application of OGH in general and Absumption in particular led to a significant performance increase of XCSR. In case of XCSF, OGH led to a reduction of the system error in most cases for both RLS-based predictions and interpolation-based predictions. In the experiments conducted, it became apparent that Absumption resulted in stronger reductions, but in some cases also caused deterioration of the system error. Specify, on the other hand, led to smaller reductions compared to Absumption, but

in no case caused a deterioration of the system error. Moreover, it turned out the underlying concept of OGH can result in an increased number of specific classifiers during the learning phase leading to an increase of the population size and a reduction of the generality metric.

The overall goal we strive for is to enhance XCS' ability of learning in environments that mislead the system to suffer from over-generalization, i.e., long chain sequential decision problems. Therefore, in our future work, we aim to realize a more precise localization of incompatible niches within over-general classifiers. In addition, based on these new identification methods, a more systematic decomposition of over-general classifiers will be implemented, using a more sophisticated heuristic instead of the currently used random-based decomposition. The results in 6-RMP and CBP motivate an investigation of whether OGH has difficulties with such tasks in general, or whether the results are due to a possibly too unfocused specialization pressure. Thus, in our future work, we also plan to study OGH in more complex variants of RMP and CBP, such as: RMP with eleven or more dimensions; CBP with either systematically increased division per dimensions or additional irrelevant attributes initialized with a Gaussian distribution. Furthermore, we will evaluate OGH on additional and also larger real-world data sets.

Author contributions Conceptualization: AW, AS; methodology: AW, AS; implementation: AW, AS; validation: AW, AS; resources: AS; writing—original draft preparation: AW; writing—review and editing: AW, AS; visualization: AW, AS and supervision: AS

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest Alexander R.M. Wagner and Anthony Stein 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

- Bacardit J, Burke EK, Krasnogor N. Improving the scalability of rule-based evolutionary learning. *Memet Comput.* 2009;1(1):55–67.
- Barry AM. The Stability of Long Action Chains in XCS. *Soft Comput.* 2002;6(3):183–99.
- Bernadó-Mansilla E, Garrell-Guiu JM. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evol Comput.* 2003;11(3):209–38.
- Butz MV, Kovacs T, Lanzi PL, Wilson SW. How XCS Evolves Accurate Classifiers. In: *Proc. of GECCO'01*. pp. 927–934. GECCO'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA 2001.
- Butz MV, Lanzi PL, Wilson SW. Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction. *IEEE T Evolut Comput.* 2008;12(3):355–76.
- Butz MV, Stolzmann W. An Algorithmic Description of ACS2. In: *Proc. of IWLCs '01*. pp. 211–229. Springer Berlin Heidelberg, Berlin, Heidelberg 2002.
- Butz MV, Wilson SW. An Algorithmic Description of XCS. In: *Soft Comput.* pp. 253–272. Springer Berlin Heidelberg, Berlin, Heidelberg 2001.
- Calian DA, Bacardit J. Integrating memetic search into the bihel evolutionary learning system for large-scale datasets. *Memet Comput.* 2013;5(2):95–130.
- Dixon PW, Corne DW, Oates MJ. A ruleset reduction algorithm for the xcs learning classifier system. In: *Learning Classifier Systems*. pp. 20–29. Springer Berlin Heidelberg, Berlin, Heidelberg 2003.
- Dorigo M. Genetic and Non-Genetic Operators in ALECSYS. *Evol Comput.* 1993;1(2):151–64.
- Grefenstette JJ, Ramsey CL, Schultz AC. Learning Sequential Decision Rules Using Simulation Models and Competition. *Mach Learn.* 1990;5(4):355–81.
- Holland JH, Reitman JS. Cognitive Systems Based on Adaptive Algorithms. In: *Pattern-Directed Inference Systems*, pp. 313 – 329. Academic Press 1978.
- Jamil M, Yang XS. A literature survey of benchmark functions for global optimisation problems. *Int J Math Model Numer Optim.* 2013;4(2):150–94.
- Kovacs T. Evolving Optimal Populations with XCS Classifier Systems. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, UK 1996.
- Kovacs T. Towards a Theory of Strong Overgeneral Classifiers. In: *FOGA 6*, pp. 165 – 184. Morgan Kaufmann, San Francisco 2001.
- Kovacs T. Xcs classifier system reliably evolves accurate, complete, and minimal representations for boolean functions. In: *Soft Computing in Engineering Design and Manufacturing*. pp. 59–68. Springer London, London 1998.
- Lanzi PL. A Study on the Generalization Capabilities of XCS. In: *Proc. of ICGA'97*. 1997;418–425.
- Lanzi PL. An Analysis of Generalization in the XCS Classifier System. *Evol Comput.* 1999;7(2):125–49.
- Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: XCS with Computed Prediction in Continuous Multistep Environments. In: *IEEE CEC 2005*. 2005;3:2032–2039.
- Lanzi PL, Loiacono D, Wilson SW, Goldberg DE. Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension. *Evol Comput.* 2007;15(2):133–68.
- Liu Y, Browne WN, Xue B. Absumption to Complement Subsumption in Learning Classifier Systems. In: *Proc. of GECCO '19*. pp. 410–418. ACM, New York, NY, USA 2019.
- Pätzelt D, Stein A, Hähner J. A Survey of Formal Theoretical Advances Regarding XCS. In: *Proc. of GECCO '19*. p. 1295–1302. Association for Computing Machinery, New York, NY, USA 2019.
- Stalpl P. Analysis and design of machine learning techniques: evolutionary solutions for regression, prediction, and control problems. Wiesbaden: Springer Science & Business Media; 2014.
- Stalpl PO, Butz MV. How fitness estimates interact with reproduction rates: Towards variable offspring set sizes in xcsf. In: *Learning Classifier Systems*. pp. 47–56. Springer Berlin Heidelberg, Berlin, Heidelberg 2010.
- Stalpl PO, Rubinsztajn J, Sigaud O, Butz MV. Function approximation with LWPR and XCSF: a comparative study. *Evol Intell.* 2012;5(2):103–16.
- Stein A, Eymüller C, Rauh D, Tomforde S, Hähner J. Interpolation-based classifier generation in xcsf. In: *IEEE CEC 2016*. pp. 3990–3998. IEEE, Vancouver, BC, Canada 2016.
- Stein A, Maier R, Hähner J. Toward Curious Learning Classifier Systems: Combining XCS with Active Learning Concepts. In: *Proc. of GECCO '17*. pp. 1349–1356. ACM, New York, NY, USA 2017.
- Stein A, Maier R, Rosenbauer L, Hähner J. XCS Classifier System with Experience Replay. In: *Proc. of GECCO '20*. p. 404–413. ACM, New York, NY, USA 2020.
- Stein A, Menssen S, Hähner J. What about Interpolation? A Radial Basis Function Approach to Classifier Prediction Modeling in XCSF. In: *Proc. of GECCO '18*. p. 537–544. ACM, New York, NY, USA 2018.
- Stone C, Bull L. For Real! XCS with Continuous-Valued Inputs. *Evol Comput.* 2003;11(3):299–336.
- Sutton RS, Barto AG. Reinforcement Learning: An Introduction. MIT press 2018.
- Urbanowicz RJ, Moore JH. Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. *J Artif Evol App* 2009;2009(1):1–1:25.
- Urbanowicz RJ, Moore JH. ExSTraCS 2.0: Description and Evaluation of a Scalable Learning Classifier System. *Evol Intell.* 2015;8(2):89–116.
- Wagner ARM, Stein A. On the effects of absumption for xcs with continuous-valued inputs. In: Castillo PA, Jiménez Laredo JL, editors. *Applications of Evolutionary Computation*. Cham: Springer International Publishing; 2021. p. 697–713.
- Wilson SW. Generalization in the XCS Classifier System. In: *Proc. of GP '98*. Morgan Kaufmann, Madison, Wisconsin, USA 1998.
- Wilson SW. Get Real! XCS with Continuous-Valued Inputs. In: *Proc. of IWLCs '99*. pp. 209–219. Springer Berlin Heidelberg, Berlin, Heidelberg 2000.
- Wilson SW. Classifiers that Approximate Functions. *Nat Comput.* 2002;1(2):211–34.
- Wilson SW. ZCS: A Zeroth Level Classifier System. *Evolutionary Computation* 1994;2(1):1–18. <https://doi.org/10.1162/evco.1994.2.1.1>
- Wilson SW. Classifier Fitness Based on Accuracy. *Evol Comput.* 1995;3(2):149–75.
- Wilson SW. Compact Rulesets from XCSI. In: *Proc. of IWLCs '00*. pp. 197–208. Springer Berlin Heidelberg, Berlin, Heidelberg 2002.

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