An Experimental Comparison of Explore/Exploit Strategies for the Learning Classifier System XCS

Tim Hansmeier
Paderborn University, Germany
Department of Computer Science
tim.hansmeier@uni-paderborn.de

Marco Platzner
Paderborn University, Germany
Department of Computer Science
platzner@uni-paderborn.de

ABSTRACT

When determining the actions to execute, reinforcement learners are constantly faced with the decision of either exploiting existing knowledge or exploring new options, risking short-term costs but potentially improving performance in the long run. This paper describes and experimentally evaluates four existing explore/exploit strategies for the learning classifier system XCS. The evaluation takes place on three well-known learning problems – two multiplexers and one maze environment. An automized parameter optimization is conducted, showing that different environments require different parametrization of the strategies. Further, our results indicate that none of the strategies is superior to the others. It turns out that multi-step problems with scarce rewards are challenging for the selected strategies, highlighting the need to develop more reliable explore/exploit strategies to tackle such environments.

CCS CONCEPTS

• **Computing methodologies** → **Rule learning**; *Intelligent agents.*

KEYWORDS

XCS, Learning Classifier Systems, Exploration, Exploitation, Autonomous Computing Systems, Evolutionary Machine Learning

ACM Reference Format:
Tim Hansmeier and Marco Platzner. 2021. An Experimental Comparison of Explore/Exploit Strategies for the Learning Classifier System XCS. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449726.3463159

1 INTRODUCTION

Learning Classifier Systems (LCS) are increasingly being proposed to enable self-adaptive and autonomous behavior in technical systems, e.g. within the concept of Self-aware Computing [6] or Organic Computing [10]. However, to reach full autonomy, reinforcement learners must decide on their own when to exploit the existing knowledge by taking the most promising action and when to deliberately select an action that is not the apparent best to potentially gain additional knowledge. This decision is commonly referred to as explore/exploit dilemma, since obtaining new knowledge through exploration incurs a short-term performance loss, while too much exploitation of the already learned knowledge risks staying on an unnecessarily low level of performance in the long term [12]. Explore/Exploit (E/E) strategies are required to let reinforcement learners decide autonomously on their own what way to pursue depending on the environment and their learning progress.

Nowadays, the most common LCS is XCS [13]. About 25 years ago, Wilson presented a high-level overview of ten different E/E strategies [14], which are not necessarily restricted to XCS, but still quite suited to it. The most important distinction between the strategies is whether they are *global*, i.e. based on global performance metrics of the whole classifier population, or *local*, i.e. determining the E/E decision for each situational input individually. Even though the importance of reliable E/E strategies for XCS has been identified more than two decades ago, not much research has been conducted in this direction since then. Instead, an ϵ-greedy strategy with a fixed exploration probability of 0.5 is commonly used. And among the few E/E strategies that have been developed, the majority has mostly been evaluated in a single scenario only and without comparison to other strategies. Further, their parameterization is often discussed on a qualitative level only, giving practitioners no guidelines on how to apply the strategies to other learning problems.

This paper aims to narrow the existing gap in research by experimentally comparing one local and three global E/E strategies proposed in the literature. The evaluation takes place on learning problems well known in the LCS community – the Multiplexer and Maze environments. A considerable part of this evaluation is an automatized parameter optimization to identify suitable parameter configurations for each of the learning problems. Even though Multiplexers and Mazes are simulated toy problems used solely in research and thus do not exhibit any imperative need for a sophisticated E/E strategy, they still resemble well-studied problem environments. Hence, our experimental results can guide future research on the development of reliable E/E strategies. Further, we hope that XCS practitioners planning to employ one of the E/E strategies are able to utilize our findings by relating the characteristics of their application environments to the evaluated Multiplexer and Maze environments.

The paper continues by presenting the methodology of our literature search and a description of the selected E/E strategies in Section 2. The experimental setup is detailed in Section 3, while the experimental results are discussed in Section 4. Finally, the paper concludes with Section 5, summarizing our findings and outlining future work.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463159

2 EXPLORE/EXPLOIT STRATEGIES

Our study focuses on explore/exploit strategies that enable autonomy and determine *when* exploration is called for and not *how* the exploration should take place. Consequently, we focus on schemes that decide between employing pure exploitation, i.e. taking the action with the highest payoff prediction, or pure exploration, i.e. choosing a random action. Therefore, directed or biased exploration, e.g. by selecting actions that promise the largest gain in knowledge, is not part of this study, even though it can have a considerable impact [3]. Nevertheless, with most directed exploration techniques the question when exploration is called for needs to be answered as well. In addition, we restrict the study to E/E strategies designed specifically for XCS, as we deem strategies tailored to its unique learning mechanism as most useful.

2.1 Literature Study

Our main tool for searching published works on explore/exploit strategies for XCS was Google scholar and the results represent the state as of 23rd March 2021. We have employed three different search terms, namely *XCS "exploration exploitation"* (142 result entries), *XCS "exploration strategy"* (68 entries) and *XCS "explore exploit"* (157 entries). The terms given in quotation marks must exactly match but can still encompass special characters, e.g. "explore exploit" matches the phrase "explore/exploit". Further, we have considered all publications that, according to Google scholar, cite the seminal paper of Wilson [14]. Overall, this resulted in two explore/exploit strategies that match our criteria, i.e. the HECS strategy of McMahon et al. [9] and the meta-rules strategy of Rejeb et al. [11]. Further, we have searched all publications that reference these publications or are cited by them, but with no additional results. In addition to the HECS and Meta-rules strategies, we selected two error-based strategies from Wilson [14], one global and one local strategy, which also have served as baseline for comparison in [11].

Not considered due to our selection criteria has been, among others, the work of Bagnall and Smith [1], who propose a strategy that not only determines when to explore but combines it with directed exploration through temperature-based Boltzman weighting. Also not considered has been the E/E strategy based on a fuzzy system as proposed by Hamzeh and Rahmani [4], since it requires to know the lifetime of the system, represented by the number of total XCS iterations. For truly autonomous systems, we assume this information to be unknown or at least associated with a high degree of uncertainty.

2.2 Selected E/E Strategies

Meta-rules [11]. Exploration and exploitation is balanced by executing repeated cycles of n exploration runs, followed by m exploitation runs. After each such cycle, the performance during the exploration and exploitation runs, denoted by $Perf_{explore}$ and $Perf_{exploit}$, respectively, is used to increase or decrease the number of exploitation runs m. This is done by the following two meta-rules, where e_r is termed the exploration rate:

$$m = \begin{cases} m \cdot (1 - e_r) & \text{if } Perf_{explore} > Perf_{exploit} \\ m \cdot (1 + e_r) & \text{if } Perf_{explore} \leq Perf_{exploit} \end{cases} \quad (1)$$

Hence, the meta-rules balance exploration and exploitation by adapting the ratio of exploration to exploitation runs, while always maintaining a minimum amount of exploration through keeping n constant. To assure that the intervals between exploration periods are not growing too large, a maximum value of m can be specified. In multi-step environments, n and m represent numbers of complete runs and are unaffected by the number of steps done in these runs. Overall, the strategy is parameterized by three values: The number of exploration runs n, the initial value of exploitation runs m and the exploration rate e_r, ranging from 0 to 1, controlling the change of m.

Global Error [14]. At the beginning of a run, it is decided if an exploration run is conducted according to the exploration probability $p_{explore}$. As opposed to the common ϵ-greedy strategy, the exploration probability is not fixed but determined as

$$p_{explore} = min(1, G \cdot E_{global}) \quad (2)$$

where E_{global} is the moving average of the global prediction error and G a configurable gain factor. The global prediction error is the absolute difference between predicted and received payoff. In our implementation, we used the values in the prediction array as the predicted payoff, but other schemes are possible as well. For instance, the prediction of the classifier with the highest fitness in the action set could be used, which would make the strategy less susceptible to non-optimal classifiers, but could stop exploration too early. The strategy is parameterized by two parameters, namely the gain factor G and the moving average's windows size W. To make gain factors between environments with different reward schemes comparable, E_{global} is normalized to the range of rewards. In multi-step environments, the prediction errors during a run, based on the internal payoffs and not the immediate rewards, are averaged and inserted into the window as a single value to avoid that runs have a different impact on the global prediction error just because they are shorter or longer.

Local Error [14]. Since it is a local strategy, the exploration probability $p_{explore}$ is individually determined for each input that is received. For each action in the match set, the numerosity-weighted average of the prediction error values ϵ, as estimated by each classifier, is calculated. The average over the actions' error estimates is then used to set the exploration probability as

$$p_{explore} = min(1, G \cdot E_{local}) \quad (3)$$

where E_{local} is the average over the actions' error estimates and G a configurable gain factor, which is the only configurable parameter of this strategy. Again, E_{local} is normalized to the range of rewards.

HECS [9]. The HECS strategy was developed specifically for multi-step problems and distinguishes between two different exploration levels. The *accuracy induced* exploration level E_A ranges between -1 and +1 and is updated every step with

$$\Delta E_A = \Delta E_{max} \cdot F \cdot \left(\frac{2}{1 - e^{M_{PA}}} \cdot \left(e^{|O_s| \cdot M_{PA}} - 1 \right) + 1 \right) \quad (4)$$

where ΔE_{max} is the maximum possible change, F the fitness of the prediction, O_s the payoff over-/undershoot (positive/negative prediction error) scaled to the maximum range of rewards and M_{PA} a parameter that defines the tolerance to perfect accuracy. The

exploration probability, used to determine if an exploration run should be conducted, is determined as

$$p_{explore} = \left(0.5 - \frac{L_{exploit} \cdot E_A}{2}\right) \quad (5)$$

where $L_{exploit}$ takes a value between 0 and 1 and assures a minimum level of exploration for values below one. As fitness of the prediction, we take the numerosity-weighted average of the classifiers' fitnesses in the action set.

In multi-step problems, a second, *reward induced* explorer level E_R is used to escape from unsuccessful exploit trials. First, the maximum numbers of steps required to reach a reward is estimated as $n_{max} = log_\gamma \left(\frac{P_{min}}{P_{max}}\right)$, where P_{min} and P_{max} are the minimum and maximum predictions present in the population and γ the discount factor of XCS. In case P_{min} is negative or zero, n_{max} must be sanitized to a meaningful value. After n_{max} steps have passed, the reward induced explorer level E_R, initialized as 1, begins to change according to

$$\Delta E_R = \Delta E_{max} \cdot \frac{1}{e^{M_{RS}} - 1} \cdot sign(R_S) \cdot \left(e^{|R_S| \cdot M_{RS}} - 1\right) \quad (6)$$

where R_S is the immediate reward (punishment) that is received scaled to the maximum reward (punishment) and M_{RS} is a scaling factor that affects the sensitivity of ΔE_R to the magnitude of R_S. Hence, a reward increases E_R, while punishments decrease E_R. During an exploitation run, HECS switches to exploration mode with a probability $P_{SwitchToExplr} = 1 - E_R$. Hence, a series of exploitation steps without reward increases the probability of switching to exploration. Overall, the HECS strategy is parameterized by four values: ΔE_{max}, $L_{exploit}$, M_{PA} and M_{RS}.

3 EXPERIMENTAL SETUP

To investigate the characteristics of the E/E-strategies in different environments, we have selected three reinforcement learning problems well-known to the LCS community.

11-Multiplexer. A single-step problem, in which XCS receives 11 input bits, of which the first three are index bits that point to one of the remaining eight bits that XCS has to predict. If XCS outputs the correct value of the bit, it receives a reward of 1,000, otherwise a reward of 0. With the 30,000 iterations that we employ, it represents a rather simple problem, as XCS with the common ϵ-greedy strategy and a fixed exploration probability of 0.5 is able to fully solve the problem in its exploit trials after roughly 10,000 iterations. Hence, a well-suited E/E strategy is expected to do some exploration at the beginning and then switch to full exploitation once XCS is able to completely solve the problem.

20-Multiplexer. The 20-Multiplexer is similar to the 11-Multiplexer problem but more complex to solve due to the larger input space, as XCS receives an input of 20 bits containing four index bits. With the same number of 30,000 iterations, XCS with the common ϵ-greedy strategy is not able to derive a complete solution. Instead, it achieves a classification accuracy of roughly 85% at the end. Hence, it represents a case in which XCS is not able to find a perfect solution and tests the ability of the E/E strategies to carefully balance exploration and exploitation during the whole time to maximize overall performance.

```
R R R R R R R R
R . . R . . F R
R R . . R . . R
R R . R . . R R
R . . . . . . R
R R . R . . . R
R . . . . R . R
R R R R R R R R
```

Figure 1: The Maze4 environment. Empty fields are denoted by dots, while obstacles are represented by rocks ('R'). The target field is the food ('F') in the upper right corner.

Maze4. Maze environments are multi-step problems, where XCS is navigating an animat through a maze in order to reach a target, denoted as food. The employed Maze4 environment is shown in Figure 1. As input, XCS receives the types of the eight surrounding fields (empty, rock, or food) and then has to make a step in one of the eight directions. At the beginning of each run, the animat is placed randomly on an empty field and the goal of XCS is to learn the shortest path to the food from each field on the map. Upon reaching the food, XCS receives a reward of 1,000, and for every other step a reward of zero. XCS is learning the shortest path by backpropagating the reward received for finding the food to preceding classifiers through the use of a discount factor. We employ 3,000 runs, with a run ending after the food is reached or 30 steps have passed. In most cases, XCS finds the shortest path, on average having a length of 3.5 steps, with the ϵ-greedy strategy after roughly 1,500 runs.

Overall, the three environments evaluate different aspects of the selected E/E strategies. The 11-Multiplexer environment investigates the capability of a strategy to determine when the problem is fully solved by the classifier population and exploration should be stopped in favor of performance-maximizing exploitation. On the other hand, the 20-Multiplexer environment tests how well a strategy is able to balance between exploration and exploitation if the learning problem cannot be fully solved and thus neither full exploration nor full exploitation is called for at any point in time. The Maze4 environment is the only evaluated multi-step environment and should reveal any behavioral differences of the strategies on multi-step problems.

For all our experiments, we have used the Python implementation scikit-XCS [15], which we have extended with the capability of solving multi-step problems. We kept the default parameter settings of scikit-XCS[1] and used a discount factor γ of 0.71 in multi-step environments. For the 11-Multiplexer and Maze4 problem a population size N of 800 has been used, while a larger population of 2000 classifiers has been employed for the more complex 20-Multiplexer problem. All problems have been implemented as randomized reinforcement learning environments, i.e. the problem instances are randomly generated and not drawn from a data set as in supervised learning settings. It is important to note that we always consider and report the overall performance of XCS, i.e. both explore and exploit runs, as opposed to the majority of works in the field that only report performance during exploit runs.

[1]That is $\beta = 0.2$, $\alpha = 0.1$, $\nu = 5$, $\mu = 0.04$, $\delta = 0.1$, $p_I = 10$, $\epsilon_I = 0$, $f_I = 0.01$, $P_\# = 0.5$, $\epsilon_0 = 10$, $\chi = 0.8$, $\theta_{GA} = 25$, $\theta_{sub} = 20$, $\theta_{del} = 20$, $\gamma = 0.71$, *DoGaSubsumption = True*, *DoActionSetSubsumption = False*

3.1 Parameter Study

Since the selected E/E strategies do not have any obvious well-suited or even optimal parameter values, we have conducted a parameter optimization of the strategies on each of the three evaluation environments. We have optimized for each environment separately instead of global optimization over all environments, as this approach allows to identify strengths and weaknesses of the strategies in different scenarios. Further, it provides system designers knowing the complexity and problem type of their operational environment with more useful guidelines in setting the parameters.

As tool for the automated parameter optimization, we have employed irace [7]. The applied ranges of allowed parameter values are summarized in Table 1. The parameter m of the meta-rules strategy only represents its initial value, since it is adapted at run-time by the strategy. A possible maximum value for m has been excluded from the parameter study and instead be set to 10% of the total number of iterations. The parameter M_{RS} of the HECS strategy has not been considered in this parameter study, either, as it is only applicable to the Maze4 multi-step environment. However, in the Maze4 environment, only two possible rewards exist, where the reward of 0 can be considered as punishment. Hence, the value of R_S in Equation 6 is either -1 or +1 and M_{RS} has no effect on the change of the explorer level. Further, we have set n_{max} to 15 steps (50% of the maximum number of steps) in case P_{min} is zero.

As problem instances, 15 different seeds have been used to initialize the random number generators inside the problem environments. An important aspect is defining the target metric that irace is optimizing, as there exists no obvious choice for assessing the quality of the E/E strategies. In general, it depends on the application how the performance of XCS is quantified. For instance, in some environments, the performance at the beginning can be close to irrelevant, e.g. if the system has an initial setup period, but in other cases, the performance is equally important over the whole lifetime. We have opted for a tradeoff between these two cases by taking all iterations into account for the calculation of the target metric, but assigning earlier iterations a smaller weight, i.e. the first quarter of all iterations is assigned a weight of $\frac{1}{15}$, the second quarter a weight of $\frac{2}{15}$, the third quarter a weight of $\frac{4}{15}$ and the last quarter the highest weight of $\frac{8}{15}$. Our choice is motivated by the assumption that even though the performance of an autonomous system is relevant during the whole runtime, it is still expected, and consequently accounted for by the system designer, that an untrained system will initially yield suboptimal performance.

To obtain valid results quickly, irace has been executed in a parallelized fashion on the nodes of the PC2 compute cluster located at Paderborn University[2]. Each compute node contains two Intel Xeon Gold 6148F CPUs with 20 cores each. For the 11-Multiplexer environment, irace was given a computing budget of 150,000 CPU seconds, which irace used internally to determine the number of optimization iterations. Due to the higher problem complexity, the 20-Multiplexer and the Maze4 environment both have been optimized with a budget of 600,000 CPU seconds.

[2]pc2.uni-paderborn.de/hpc-services/available-systems/noctua/, accessed 01.04.2021

Table 1: Value ranges used in the parameter optimization.

Strategy	Param.	Datatype	Range	
			Lower	Upper
Meta-rules	n	integer	1	1000
	m	integer	1	1000
	e_r	real	0.01	0.99
Global Error	G	real	0.01	10
	W	integer	1	1000
Local Error	G	real	0.01	10
HECS	ΔE_{max}	real	0.01	1
	M_{PA}	real	−10	10
	$L_{exploit}$	real	0.5	1

4 EXPERIMENTAL RESULTS

This section summarizes the obtained experimental results. First, Subsection 4.1 presents and discusses the results of the parameter study. The experimental comparison of the strategies follows in Subsection 4.2, where the optimized parameters have been employed on each of the three evaluation problems. Finally, the sensitivity of the E/E-strategies to non-optimal parameter choices is investigated in Subsection 4.3, where the parameters that have been optimized for one evaluation problem are applied on the other environments.

4.1 Results Parameter Study

The parameter configurations resulting from irace's optimization are shown in Table 2, where notable differences between the different evaluation problems can be observed. In the 20-Multiplexer environment, the constant number n of exploration runs used in the meta-rules strategy is more than four times higher than for the 11-Multiplexer case, indicating the higher complexity of the 20-Multiplexer problem. In the Maze4 environment, both n and m are considerably smaller, as the environment employs only a tenth of the runs than the two multiplexer environments. On the other hand, the exploration rate e_r is considerably higher, resulting in a more aggressive adaption of m. This could be related to the fact that in multi-step problems each run consists of multiple iterations, allowing XCS to gain more knowledge during a single run.

For the global error strategy, the gain factor G, representing the sensitivity of the exploration probability to the prediction error, is ten times smaller for the 20-Multiplexer than for the 11-Multiplexer, again showing the higher complexity of the 20-Multiplexer environment. Since we normalize the prediction error to the maximum range of rewards, the gain factor of 0.655 also represents the upper bound of the exploration probability. For the Maze4 problem, the gain factor of 2.769 lies in between the other two values. In the case of the multiplexer problems, these observations apply to the gain factor of the local error strategy as well, but in the Maze4 environment, the optimized value of G is considerably higher.

That the optimal sensitivity of the exploration probability to the prediction error differs between the evaluation problems can also be observed for the HECS strategy, where the M_{PA} parameter is smaller for the 11-Multiplexer than for the 20-Multiplexer, with

Table 2: Optimized parameterization of each configuration for the evaluated scenarios.

	Meta-rules			Global Error		Local Error	HECS		
	n	m	e_r	G	W	G	ΔE_{max}	M_{PA}	$L_{exploit}$
11-Multiplexer	112	17	0.151	6.907	248	2.992	0.013	-8.336	0.988
20-Multiplexer	467	36	0.227	0.655	115	0.684	0.380	-2.187	0.935
Maze4	14	33	0.524	2.769	23	7.793	0.115	-5.894	0.987

the optimal value for the Maze4 environment ranging in between. The smaller the value of M_{PA} is, the more sensitive is the accuracy induced explorer level to the prediction error. Hence, these results are in line with the gain factors found for the global error strategy. The value for the parameter ΔE_{max}, the maximum change of the explorer level, is very small and close to zero in the case of the 11-Multiplexer, leading to small changes of the explorer level, while they are considerably larger for the other two environments. $L_{exploit}$ is close to one for all problems, leading to a low level of minimum exploration and potentially enabling HECS to reach close to optimal performance in all cases.

4.2 Experimental Comparison of E/E Strategies

To compare the E/E strategies, the optimized parameter configurations have been employed on each evaluation problem. To obtain statistically meaningful results, all reported results are averages over 50 independent trials, where the seeds that have been used to initialize the environments are different from those used in the parameter study.

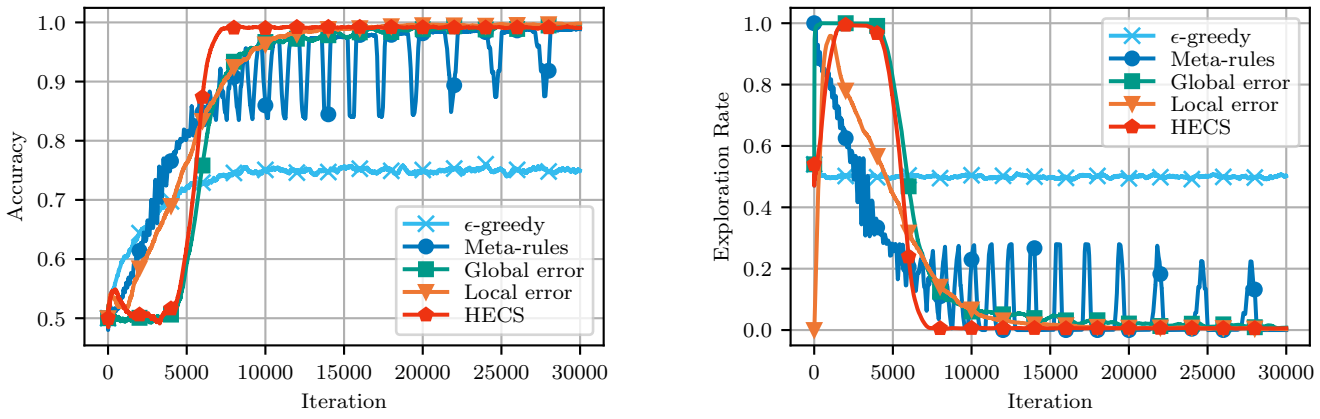
11-Multiplexer. Figure 2a shows the development of the classification accuracy for all E/E strategies on the 11-Multiplexer. The common ϵ-greedy strategy with a fixed exploration probability of 0.5 achieves an accuracy of 0.75 after approximately 10,000 iterations. The problem is then perfectly solved, but the exploration iterations negatively affect the achieved accuracy. On the other hand, all of the evaluated E/E strategies are able to achieve the goal of first doing exploration and then switching to (nearly) full exploitation. The distinct pattern of the meta-rules strategy is related to the alternating periods of exploration and exploitation. Since the number of explore iterations is not adapted, exploration is done even when the exploitation iterations achieve perfect accuracy. The accuracy of the local error strategy is continuously improving until perfect accuracy is reached. On the other hand, both the global error and HECS strategies achieve a very low accuracy at the beginning that suddenly increases after around 5,000 iterations, with the increase of HECS being considerably steeper.

Figure 2b shows the corresponding development of the exploration rate. The graphs basically mirror the development of the classification accuracy, as the exploration rate of the local error strategy is continuously decreasing, while the global error strategy first maintains an exploration rate of 1 which suddenly begins to decrease and eventually falls down to 0. The HECS strategy is behaving similarly, but its exploration rate is dropping even more quickly. Overall, the results show that the HECS strategy is best in determining the point when exploitation is called for to maximize performance, while the local error strategy seems to be most suited in case a continuously improving performance is desirable.

20-Multiplexer. Figure 3a shows the development of the classification accuracy on the more complex 20-Multiplexer. This time, both the global and the local error strategy achieve a continuous increase that is consistently better than the common ϵ-greedy strategy, with the global error strategy having a slight advantage. The meta-rules strategy is first achieving a worse accuracy, but then catches up and outperforms both the ϵ-greedy and the error-based strategies – at least during its exploitation periods. The HECS strategy depicts a similar behavior than on the 11-Multiplexer problem. At first, the accuracy is not improving, but towards the end there is a steep increase, eventually leading to the best accuracy in the field.

The development of the exploration rate shown in Figure 3b differs from those observed for the 11-Multiplexer. Both error-based strategies start with relatively low exploration rates of 0.2 to 0.3 – a consequence of the small gain factors, which make them more insensitive to high prediction errors. In addition, their exploration rates are only slowly decreasing over time. The HECS strategy is first applying an exploration rate of 1, which after approximately 15,000 iterations begins decreasing to reach a value close to 0 at the end, mirroring the development of the classification accuracy quite well. However, the 20-Multiplexer is still incompletely solved at the end of the experiment and would require additional exploration afterward to generate a complete solution. In case the operation period of the system is extended, HECS would not apply this and settle on a non-optimal classification accuracy. Hence, it seems that especially the HECS strategy is overfitted to the specific scenario and the target metric used in the parameter optimization, which assigns a higher weight to the performance in later iterations as described in Subsection 3.1.

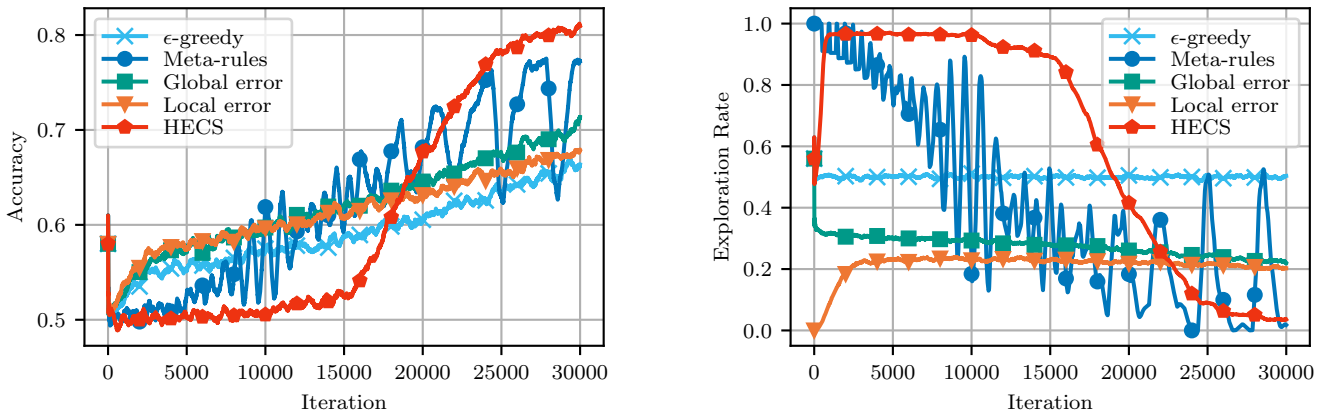
Maze4. The average number of steps until the food or the maximum number of steps is reached is shown in Figure 4a. Considering that on average a minimum number of 3.5 steps towards the food is required and that XCS is able to optimally solve the problem with the applied parameter settings, it can be concluded that all evaluated E/E strategies perform poorly. Until run 400, the HECS strategy is considerably outperforming all others but shows no improvement afterward. Both error-based strategies improve continuously, with the local error strategy being on the lead and achieving the lowest number of steps at the end. Even though the meta-rules strategy employs a constant number of exploration runs, its performance in the exploitation periods is not considerably better than those of the other strategies. An inspection of the exploration rates shown in Figure 4b leads to the reason for the poor performance since all strategies employ low rates of exploration from the beginning on. Even though the local error strategy is parameterized with a high



(a) Classification accuracy

(b) Exploration rate

Figure 2: Experimental results obtained on the 11-Multiplexer problem. Results are averages over 50 trials and shown as a moving average over 400 samples.



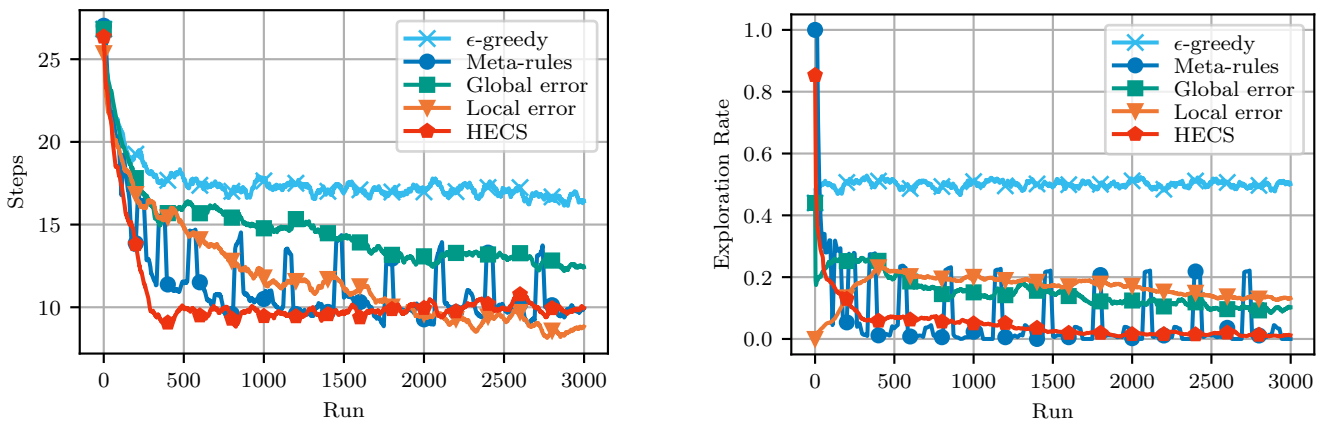
(a) Classification accuracy

(b) Exploration rate

Figure 3: Experimental results obtained on the 20-Multiplexer problem. Results are averages over 50 trials and shown as a moving average over 400 samples.

gain factor and should thus be rather sensitive to the estimated prediction errors, its resulting rate of exploration is initially increasing rather slowly, reaching a maximum of approximately 0.2 to then begin decreasing even slower. The global error strategy is reaching a similar exploration rate of roughly 0.2 almost immediately, which then slowly decreases as well. The HECS strategy applies the smallest amount of exploration of all evaluated strategies and employs close to zero exploration after approximately 2,000 runs.

We presume that the poor performance is related to the specific multi-step characteristics of the Maze environment, which make it challenging for accuracy- or error-based strategies, like the HECS, global error, and local error strategy, to assess the current capability of the classifier population to solve the problem. In the Maze environment, a positive reward is obtained only once at the end of a run and only in case the food is reached. On all other steps, an immediate reward of zero is received. To learn the shortest paths to the food, the reward received upon reaching the food is back-propagated to the preceding classifiers with the Q-learning-like internal reinforcement mechanism of XCS using the discount factor γ. For classifiers early in the path, i.e. those matching to positions distant to the food, the path needs to be repeatedly taken until a small portion of the reward is arriving at the classifier. Until then, these classifiers keep a perfectly accurate payoff prediction of zero. However, if the majority of classifiers in the population lead to a prediction error of zero, the error- and accuracy-based strategies are lured into believing that XCS is solving the problem quite well and thus apply a small exploration probability. Our assumption is supported by the development of the exploration rate of the local



(a) Number of Steps

(b) Exploration rate

Figure 4: Experimental results obtained on the Maze4 problem. Results are averages over 50 trials and shown as a moving average over 50 runs. The exploration rate of a run is defined as the percentage of exploration steps in the run.

Table 3: Standard deviation σ of the average over the last 500 runs of each E/E strategy in the Maze4 environment.

	ϵ-greedy	Meta-rules	Global Error	Local Error	HECS
σ	3.06	6.44	8.92	6.37	7.91

error strategy as shown in Figure 4b. Even though the strategy is parameterized with a high gain factor, the exploration rate starts at nearly zero, which is the initial prediction error of the majority of classifiers, and then increases very slowly, presumably in line with the backpropagation of the discounted reward.

The impact that low prediction errors have not only arises at the beginning of the trials. For instance, the HECS strategy is applying nearly no exploration in the second half of our experiment, which means that its accuracy induced exploration level seems to dictate no further exploration. The second, reward induced exploration level that is used to escape from unsuccessful exploit runs is not able to correct this, either, as the classifiers with a low prediction lead to a high estimate of the maximum number of steps n_{max}. Thus, as soon as XCS has evolved a solution requiring less than the current n_{max} steps, exploration is no longer applied and HECS settles on a suboptimal model. Even though the meta-rules strategy is not based on accuracy, but directly on the target metric, i.e. the number of steps, it is not reaching a superior performance, either, but this could be related to the mechanism of alternating periods of exploration and exploitation and its optimistic approach of increasing exploitation once it receives better performance than exploration.

However, all averaged results obtained in the Maze4 environment must be interpreted with caution. When inspecting single evaluation runs, which only differ through the applied random seed, we have noticed that the performance can differ considerably, e.g. in some case all strategies haven been able to achieve good results, while in others some strategies performed poorly. We have not been able to identify any obvious pattern or correlation between different strategies. To quantify this effect, we averaged the number of steps taken during the last 500 runs of every evaluation trial and then calculated the standard deviation of the 50 average values of each strategy, resulting in the standard deviations given in Table 3. The common ϵ-greedy strategy with its fixed exploration probability of 0.5 has a standard deviation of roughly 3 steps, representing 10% of the applied maximum limit of 30 steps. This can be considered as the baseline deviation or noise that is introduced due to randomness in XCS and the Maze4 environment, e.g. through different placements of the animat. The relatively high standard deviation is related to the fact that XCS is not always reliably solving the Maze4 problem within the employed number of runs, which may be caused by non-optimal parameter settings of XCS or overgeneralization in the classifier population, which is known to occur especially in multi-step environments [2]. However, all four E/E strategies lead to standard deviations that are even higher than the baseline, with the least affected strategies being the meta-rules and local error strategy. The deviations of the HECS and global error strategy are even higher, reaching a standard deviation that is, in the case of the global error strategy, nearly three times larger than the baseline. Even though the increased standard deviations observed with the E/E strategies can potentially be explained by the lower level of exploration, further intensifying the low GA frequency in states far from the target field, this does not give any reason for the differences between the four strategies. Deeper investigations have to be conducted to identify the source of this phenomenon and how E/E strategies can be designed to achieve a more reliable behavior.

4.3 Parameter Sensitivity

To assess how sensitive the strategies are to non-optimal parameters, we have evaluated the performance on each evaluation problem not only with the parameters optimized for the specific problem but also with the parameter configurations optimized for the other

Table 4: Change of the performance metric when non-optimized parameter configurations are applied. Rows represent the evaluation scenarios, columns the applied parameter configuration. The values in parentheses give the absolute value of the performance metric. The asterisk (*) marks differences statistically significant at a significance level of $1 - \alpha = 0.05$ according to a two-tailed Mann-Whitney U test [8].

Strategy	Meta-rules			Global Error			Local Error			HECS		
Config.	11-Mul.	20-Mul.	Maze4	11-Mul.	20-Mul.	Maze4	11-Mul.	20-Mul.	Maze4	11-Mul.	20-Mul.	Maze4
11-Mul.	- (0.94)	-5.3%* (0.89)	-20.2%* (0.75)	- (0.96)	-4.2%* (0.92)	-0.0%* (0.96)	- (0.97)	-4.1%* (0.93)	-1.0%* (0.96)	- (0.97)	-6.2%* (0.91)	-2.1%* (0.95)
20-Mul.	-1.5% (0.67)	- (0.68)	-8.8%* (0.62)	-21.5%* (0.51)	- (0.65)	-10.8%* (0.58)	-9.4%* (0.58)	- (0.64)	-18.8%* (0.52)	-23.2%* (0.53)	- (0.69)	-11.6%* (0.61)
Maze4	-65.3%* (17.56)	-126.8%* (24.09)	- (10.62)	+0.1% (13.48)	-8.9% (14.70)	- (13.50)	-30.6% (13.14)	-31.1%* (13.19)	- (10.06)	-5.3% (10.59)	+12.7% (8.78)	- (10.06)

two problems. As figure of merit, we have employed the target metric used in the parameter study (cmp. Subsection 3.1). The results are shown in Table 4. On the 11-Multiplexer, all strategies seem to be relatively insensitive to non-optimal parameters. However, it represents a simple problem, and thus the parameters mainly influence how quickly the maximum classification accuracy is reached and not so much if it is reached at all. A notable exception is the meta-rules strategy with its Maze4 configuration, which points to the obvious drawback of using parameters representing a number of iterations in environments with a vastly different number of total iterations.

On the other hand, a higher parameter sensitivity can be observed on the more complex 20-Multiplexer problem. The reported relative changes of the performance metric are a bit misleading in this case, as the absolute performance value has a lower limit of 0.5, achieved by random guessing. Hence, strategy/configuration pairs with an absolute value of around 0.5 do not apply any reasonable balance of exploration and exploitation. Both error-based strategies and HECS have one such configuration, while the meta-rules strategy shows at least some level of reasonable balance in all cases. Thus, the conclusion regarding the parameter sensitivity of the strategies is exactly inverted compared to the 11-Multiplexer case.

In the Maze4 environment, the meta-rules strategy shows by far the highest parameter sensitivity. The performance metric represents number of steps and thus should be as small as possible, as opposed to the multiplexer environments. The local error strategy shows a considerable amount of parameter sensitivity as well, while the global error and HEC strategies seem to be rather insensitive. For both strategies, the optimized parameter configurations are even outperformed by configurations optimized for other problems. This could be related to the strong influence that the randomness has in the Maze4 environment, which potentially mislead the parameter optimization, as the seeds used in the parameter optimization have been different from those used in the final evaluation. The strong influence of randomness is most likely also the reason why most observed differences lack statistical significance.

5 CONCLUSION

Our experimental comparison of four different explore/exploit strategies for XCS has not yielded a clear winner that achieved superior performance in all evaluated cases. If parameterized appropriately, the HECS strategy with its non-linear and accuracy-based equations seems to be best in determining a point when to switch to full exploitation. On the other hand, linear error-based strategies tend to lead to a more steady increase in performance by continuously decreasing the rate of exploration. Our parameter study showed that different environments require vastly different parameter configurations. Especially if the strategies are configured to be very sensitive to the prediction error and employed in environments that XCS is not able to completely solve, the overall performance is negatively affected. Naturally, our experimental evaluation has not been exhaustive and did not evaluate all use cases, e.g. changing or stochastic environments have not been considered. Another aspect that has not been investigated in this work is the influence of directed exploration, e.g. through roulette wheel selection.

Still, much more research has to be conducted to develop reliable explore/exploit strategies for XCS. Our results show that especially multi-step problems with scarce rewards are challenging, as the error or accuracy of the classifier population, which is the internal optimization target of XCS, can misguide the E/E strategies into applying too much exploitation too early. An outcome to this could be to base the E/E decisions directly on the target metric of the environment, as it is done in the meta-rules strategy or the reward induced exploration used in the HECS strategy to escape unsuccessful exploit runs. To realize reliable behavior on a wider range of environments, the parameter sensitivity of the strategies could be reduced by adapting the parameters at run-time to suitable values, e.g. by deriving them from performance or population state metrics [5]. Further, a meta-strategy could be developed that automatically switches to the E/E strategy being most appropriate for the current environment.

ACKNOWLEDGMENTS

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre On-The-Fly Computing (GZ: SFB 901/3) under the project number 160364472. The authors gratefully acknowledge the funding of this project by computing time provided by the Paderborn Center for Parallel Computing (PC²).

REFERENCES

[1] Anthony J. Bagnall and George D. Smith. 2005. A multiagent model of the UK market in electricity generation. *IEEE Transactions on Evolutionary Computation* 9, 5 (oct 2005), 522–536. https://doi.org/10.1109/TEVC.2005.850264

[2] A. M. Barry. 2002. The stability of long action chains in XCS. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 6, 3 (jun 2002), 183–199. https://doi.org/10.1007/s005000100115

[3] Martin V. Butz. 2001. Biasing Exploration in an Anticipatory Learning Classifier System. In *Advances in Learning Classifier Systems, 4th International Workshop, IWLCS 2001, San Francisco, CA, USA, July 7-8, 2001, Revised Papers (Lecture Notes in Computer Science)*, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson (Eds.), Vol. 2321. Springer, 3–22. https://doi.org/10.1007/3-540-48104-4_1

[4] Ali Hamzeh and Adel Rahmani. 2005. A Fuzzy System to Control Exploration Rate in XCS. In *Learning Classifier Systems, International Workshops, IWLCS 2003-2005, Revised Selected Papers (Lecture Notes in Computer Science)*, Tim Kovacs, Xavier Llorà, Keiki Takadama, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson (Eds.), Vol. 4399. Springer, 115–127. https://doi.org/10.1007/978-3-540-71231-2_9

[5] Tim Kovacs. 2002. Performance and population state metrics for rule-based learning systems. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, Vol. 2. IEEE Computer Society, 1781–1786. https://doi.org/10.1109/CEC.2002.1004512

[6] Peter R. Lewis, Marco Platzner, Bernhard Rinner, Jim Tørresen, and Xin Yao (Eds.). 2016. *Self-aware Computing Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-319-39675-0

[7] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. 2016. The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives* 3 (2016), 43–58. https://doi.org/10.1016/j.orp.2016.09.002

[8] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (mar 1947), 50–60. https://doi.org/10.1214/aoms/1177730491

[9] Alex McMahon, Dan Scott, Paul Baxter, and Will Browne. 2006. An autonomous explore/exploit strategy. In *Proceedings of AISB'06: Adaptation in Artificial and Biological Systems*, Vol. 2. ACM Press, New York, New York, USA, 192–201. https://doi.org/10.1145/1102256.1102280

[10] Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer (Eds.). 2011. *Organic Computing — A Paradigm Shift for Complex Systems*. Springer Basel. https://doi.org/10.1007/978-3-0348-0130-0

[11] Lilia Rejeb, Zahia Guessoum, and Rym M'Hallah. 2005. An Adaptive Approach for the Exploration-Exploitation Dilemma and Its Application to Economic Systems. In *Learning and Adaption in Multi-Agent Systems, First International Workshop, LAMAS 2005, Utrecht, The Netherlands, July 25, 2005, Revised Selected Papers (Lecture Notes in Computer Science)*, Karl Tuyls, Pieter Jan't Hoen, Katja Verbeeck, and Sandip Sen (Eds.), Vol. 3898. Springer, 165–176. https://doi.org/10.1007/11691839_10

[12] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction (2nd ed.)*. MIT Press. 427 pages.

[13] Stewart W. Wilson. 1995. Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3, 2 (jun 1995), 149–175. https://doi.org/10.1162/evco.1995.3.2.149

[14] Stewart W. Wilson. 1996. Explore/Exploit Strategies in Autonomy. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. The MIT Press. https://doi.org/10.7551/mitpress/3118.003.0040

[15] Robert F Zhang and Ryan J Urbanowicz. 2020. A Scikit-learn Compatible Learning Classifier System. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. ACM, New York, NY, USA. https://doi.org/10.1145/3377929.3398097