CRACS: Compaction of Rules in Anticipatory Classifier Systems

Romain Orhand
Icube Laboratory - UMR CNRS 7357
Illkirch, France
University of Strasbourg
Strasbourg, France
rorhand@unistra.fr

Pierre Collet
Icube Laboratory - UMR CNRS 7357
Illkirch, France
University of Strasbourg
Strasbourg, France
pierre.collet@unistra.fr

Pierre Parrend
Icube Laboratory - UMR CNRS 7357
Illkirch, France
EPITA
Le Kremlin-Bicêtre, France
pierre.parrend@unistra.fr

Anne Jeannin-Girardon
Icube Laboratory - UMR CNRS 7357
Illkirch, France
University of Strasbourg
Strasbourg, France
anne.jeannin@unistra.fr

ABSTRACT

Rule Compaction of populations of Learning Classifier Systems (LCS) has always been a topic of interest to get more insights into the discovered underlying patterns from the data or to remove useless classifiers from the populations. However, these techniques have neither been used nor adapted to Anticipatory Learning Classifier Systems (ALCS). ALCS differ from other LCS in that they build models of their environments from which decision policies to solve their learning tasks are learned. We thus propose CRACS (Compaction of Rules in Anticipatory Classifier Systems), a compaction algorithm for ALCS that aims to reduce the size of their environmental models without impairing these models or the ability of these systems to solve their tasks. CRACS relies on filters applied to classifiers and subsumption principles. The capabilities of our compaction algorithm have been studied with three different ALCS on a thorough benchmark of 23 mazes of various levels of environmental uncertainty. The results show that CRACS reduces the size of populations of classifiers while the learned models of environments and the ability of ALCS to solve their tasks are preserved.

CCS CONCEPTS

• **Computing methodologies → Rule learning**; *Knowledge representation and reasoning*.

KEYWORDS

Anticipatory Learning Classifier System, Machine Learning, Knowledge Extraction, Compaction

ACM Reference Format:
Romain Orhand, Pierre Collet, Pierre Parrend, and Anne Jeannin-Girardon. 2023. CRACS: Compaction of Rules in Anticipatory Classifier Systems. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583133.3596352

1 INTRODUCTION

Artificial Intelligence models such as ChatGPT can have huge implications for science and society [24]. However, these models rely on "black boxes" lacking transparency and interpretability: their behavior is particularly difficult to capture [20]. Although it is possible to develop tools to provide explanatory elements about these black boxes, we advocate in favor of using intrinsically interpretable models. Rule-based machine learning models are models that generate rules describing the behavior of agents or characterizing data sets. The interpretability of the rule-based models is thus admitted in many fields where they have been used [1].

Learning Classifier Systems (LCS) are Rule-Based Machine Learning models designed to learn tasks in complex and continuously changing environments like real-world ones [10]. LCS build their populations of rules (called classifiers) thanks to two components: one devoted to discovering new rules and the other one to adapting rules to learning tasks [21]. Among these systems, we are interested in Anticipatory Learning Classifier Systems (ALCS) used in uncertain environments for reinforcement learning tasks.

ALCS generate their classifiers through successive comparisons of their perceptions of environments: ALCS are thus able to represent the consequences of their decisions through the `{conditions, actions, effects}` tuples in their classifiers (or $\{C, A, E\}$ tuples) [9]. Contrary to other LCS, ALCS thus build representations of their learning environments, from which decision policies capable of solving their learning tasks are learned. Environmental representations learned by ALCS are decoupled from the resolution of learning tasks *per se*: ALCS would only need to develop new decision policies from their environmental models if a learning task were to change. The design of ALCS allows them to provide more explanatory insights about their decisions. For example, a user can predict the consequences of other decisions that could have been made and compare them with the ones chosen by an ALCS.

However, environmental uncertainty impacts the ability of ALCS to develop representations of their environments and to solve their learning tasks. Environmental uncertainty can take the form of actions with non-deterministic consequences, situations that cannot be distinctly identified (as with the Perceptual Aliasing Issue) or noisy environmental rewards [18]. These uncertainties often result in building more classifiers since finding the most appropriate ones is more challenging for ALCS (and LCS).

As with any rule-based approach, an increase in the number of rules reduces the interpretability of these approaches [22]. Algorithms aiming at extracting knowledge from populations of classifiers have therefore been developed for many LCS to alleviate this rise: both compaction [22] and condensation [25] target at removing useless classifiers while other techniques merge similar classifiers into new ones representing their characteristics [11]. However, algorithms to extract knowledge from populations of classifiers have not been designed for ALCS, contrary to other LCS.

This paper thus introduces CRACS (Compaction of Rules in Anticipatory Classifier Systems), a compaction algorithm for ALCS that aims to reduce the size of their representations of uncertain environments without impairing these representations or the ability of these systems to solve their tasks.

A review of ALCS and techniques to extract knowledge from populations of classifiers of LCS is presented in section 2 to introduce CRACS in section 3. Section 4 presents a study of the capabilities of CRACS through a thorough benchmarking on the different mazes used as test-beds in the literature. The results achieved by CRACS are discussed in section 5 before concluding in section 6.

2 EXTRACTING KNOWLEDGE: REVIEW

Techniques to extract knowledge from populations of classifiers are based on two opposing approaches, depending on whether they are used with learning data (data-driven) or not (rule-driven). When environments are uncertain, it seems tricky to use approaches that rely on data from these environments to extract knowledge. The changing nature of uncertain environments implies that these data cannot fully represent learning tasks. Moreover, [11] show that rule-driven approaches highlight the classifiers that effectively describe the problem while data-driven approaches suffer from overfitting: data-driven approaches may use overly specific classifiers or ignore previously learned knowledge to create a representation that best matches the data.

The three main techniques for extracting knowledge from LCS are compaction, condensation and building new rules (through merging or clustering for example).

Originally proposed by [25], condensation disables rule discovering components of LCS to let them remove unfitted classifiers while learning to solve their tasks. [12] demonstrated that this technique enables LCS to build representations of boolean functions with a minimal number of classifiers, with the least possible overlap, while completely describing the problem. Condensation was applied by [13] to an LCS dedicated to data mining, reducing the sizes of populations of classifiers by about 94%, with very small impacts on the resolution of the problems. [4] has finally adapted this technique to be used with an LCS that solves function approximation problems. The use of condensation is thus constraining and also unsuitable to our needs since it subscribes to data-driven approaches. For condensation to be effective, it requires that LCS have a complete representation of its data, otherwise, condensation could not make populations converge to the most compact set of classifiers. In practice, however, it is difficult to obtain or know whether representations built by LCS are complete for uncertain environments. This would imply that the training data perfectly describes such changing environments.

The building of new rules was introduced by [11]: classifiers are first grouped according to their similarity and new classifiers representing the characteristics of the groups are generated. This technique depends on a measure of similarity between two classifiers and on a clustering algorithm that uses this measure. It has been employed by [23], [27] and [16] which combines visualization tools with statistical tools to guide knowledge discovery by identifying discriminating conditional attributes of classifiers.
Many ALCS already use generalization mechanisms on sets of classifiers corresponding to the same environmental situations, thus avoiding any loss of knowledge for these sets [3]. If these techniques were used on the same sets of classifiers, they would be redundant with the ones already implemented in ALCS. Otherwise, these techniques could impair the built environmental representations depending on how classifiers could be grouped, which is not desirable.

Finally, compaction removes overlapping classifiers, classifiers with low fitness to learning tasks or classifiers with conflicting conditions from the populations. Compaction was first suggested by [26] and consists of a 4-step algorithm :

(1) Classifiers are sorted according to a property that allows them to be ranked by their relevance to solving a learning task, such as their numerosity, fitness or experience.
(2) A subset of classifiers is formed from the set of sorted classifiers, such that performance on a training set is maximized: the first classifier of the sorted set forms a subset, and if performance is not maximized, the second classifier of the sorted set is added to this subset, *etc.*
(3) Classifiers that have been added to the previous subset without improving its performance are deleted.
(4) Classifiers with overlapping conditional structures are removed from the previously formed subset, preferring those related to the largest number of instances in the training set.

Several propositions related to [26] were then made: [7] changed the last step of this compaction algorithm by preferring the most general classifiers; [6] proposed to use the product of one classifier numerosity multiplied by its reward to form classifier subsets in step 2 of this algorithm; [8] modified the three first step of [26]'s algorithm by using experience and prediction reward of classifiers. But these algorithms rely on training data and thus fall under the data-driven approach that is not desired; [22] went further by using the product of numerosity, reward and generality of classifier in step 2 of this algorithm; [22] also suggested to use the fitness of classifiers to sort the population and then, to only remove overlapping classifiers by promoting classifiers whose fitness is the largest.

Other compaction algorithms do not rely on training data. [4] proposed to directly delete classifiers whose situations described by their conditional structure are included in a classifier whose prediction error is lower. [22] proposed a filter adapted to LCS performing classification tasks where classifiers are removed from the population if their prediction does not outperform chance, or if these classifiers have overfitted the training data. [14] have proposed a compaction algorithm that seeks to identify, for the same LCS having to the same problem several times, classifiers common to all populations. [15] then refined this idea under two new compaction algorithms whose goal is to identify all correct and unsubsumable classifiers (what they called the "natural solution"), instead of identifying the smallest, correct, non-overlapping set of classifiers that solves a learning task (that is an optimal solution): both algorithms differ in their use of training data to identify over-general rules (thus making one of them fall under the data-driven approach).

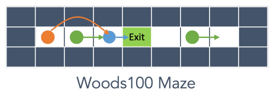
Since ALCS were originally intended for reinforcement learning tasks, they attempt to learn both a transition model of the environment and a decision policy built from this model. The other LCS focus on setting a decision policy. The object to be compacted is thus different according to the systems (the model or the decision policy) and could explain why adaptations of these techniques are expected for ALCS. This could also be a reason why these compaction techniques have not been used with ALCS.

CRACS (Compaction of Rules in Anticipatory Classifier Systems) is hereby introduced as **the first** rule-driven compaction technique generic to any kind of ALCS. CRACS adapts two ideas from [4] and [22] to reduce the sizes of the environmental representations learned by ALCS without impairing these representations or the ability of these systems to solve their tasks.

3 CRACS

ALCS classifiers are mainly made of a $\{C, A, E\}$ tuple (consisting of a condition component C, an action component A and an effect component E), a mark M that specifies states for which the classifier has failed to anticipate, a measurement of the quality of anticipation q, a prediction of the expected reward, and lastly, an experience exp indicating the number of times a classifier has been used [5].

CRACS attempts to take into account all possible representations of $\{C, A, E\}$ tuples to adapt to any ALCS operating in uncertain environments (partially observable and non-deterministic). Three different $\{C, A, E\}$ representations exist to provide ALCS with capabilities to manage environmental uncertainty: BACS uses Behavioral Sequences consisting of sequences of actions to tackle the Perceptual Aliasing Issue [17]; PEPACS uses (Probability-)Enhanced Predictions to enable the building of several consequences in effect components of classifier so that the system can build complete representations of uncertain environments [18]; BEACS enhances and merges Probability-Enhanced Predictions with Behavioral Sequences to better enable ALCS to handle uncertain environments [19]. The figure 1 shows examples of different $\{C, A, E\}$ representations. The hash is a wildcard that corresponds to all possible items in the condition and indicates there are no changes in the effect. Both Cl_2 and Cl_3 depict classifiers having enhanced predictions. Cl_2 anticipates two sets of changes that only differ at the right position of an agent where one set of changes has been predicted 1503 times and the other one 1501 times. Cl_3 also anticipates two possible changes at the right position of an agent where a wall (resp. a path) is expected with 48% occurring probability (resp. 52%). Colors correlate classifiers with environmental situations to which they correspond.



Cl_1: ##Exit#####	A: →	E: ##Path#####
Cl_2: ##Path###Path#	A: →	E: (##Wall#####:1503, ##Exit#####:1501)
Cl_3: ##Path###Path#	A: →	E: ##{Exit : 48%, Wall : 52%}#####
Cl_4: ######Wall#	A: → →	E: ##Exit###Path#

Figure 1: Illustration of different ALCS classifiers tuples in a maze environment, if the provided observations are limited to the eight squares adjacent to each position starting from the North and clockwise. Walls are represented with dark blue cells while paths are represented with white cells.

CRACS adapts the idea of filtering out classifiers from [22]. In particular, the use of Behavioral Sequences in ALCS increases the size of populations of classifiers [17]. Many classifiers with such sequences can be created without being tested or participating in the system's decision policies. For example, the experience of such classifiers is then lower than the threshold θ_{exp} set by the user: this threshold enables the classifiers to be considered as sufficiently experimented. Effect components of these classifiers can also predict that their use of Behavioral Sequences does not lead to any environmental changes (*e.g.* an agent taking a step forward and then a step back). However, environmental changes are expected when the Behavioral Sequences aims at bridging states related to the Perceptual Aliasing Issue.
Removing these classifiers would reduce the number of classifiers in the populations while making it easier to identify the relevant Behavioral Sequences classifiers for population and decision policy study.

CRACS also adapts from [4] the idea of removing overlapping classifiers to reduce the size of the populations and ease their interpretability. Implementing this idea requires defining how one classifier can subsume another, while taking into account their structure, *i.e.* the use of Behavioral Sequences or the presence of several consequences in their effect components with Enhanced Predictions.

CRACS then consists of two steps where, for each classifier cl in the population:

- cl is compared to every other classifier in the population to determine if it can be subsumed and therefore removed from the population ;
- if cl cannot be subsumed by another classifier of the population, cl goes through a filter that removes classifiers with Behavioral Sequences if their effect components do not anticipate any changes or if they are not sufficiently experienced according to the θ_{exp} parameter of ALCS.

The pseudo-code for CRACS is given by the algorithm 1, and the function to determine if one classifier subsumes another is given by the algorithm 2. The compaction of the population of classifiers is performed once, before exploiting the population to solve a learning task. It takes as arguments the population of classifiers Cls, the user thresholds θ_{exp} and θ_{r} each determining whether a classifier has been sufficiently tested by the system or whether the quality of a classifier is sufficiently high (meaning that the consequences of its action can be reliably predicted), and finally the length L characterizing perceptions of the environment. CRACS

Algorithm 1 Compaction of an ALCS population of classifiers

```
function CompactPopulation($Cls$,$L$,$\theta_{\text{exp}}$,$\theta_{\text{r}}$)
    for each $cl_1 \in Cls$ do
        $keepCl_1 \leftarrow True$
        for each $cl_2 \in Cls$ do
            if IsSubsumed($cl_1$,$cl_2$,$\theta_{\text{r}}$) == $True$ then
                $keepCl_1 \leftarrow False$
                break
            end if
        end for
        if $keepCl_1 == True$ and
          $length(A_{\text{cl}_1}) > 1$ and
          ($exp_{\text{cl}_1} < \theta_{\text{exp}}$ or $E_{\text{cl}_1} = \{\#\}^L$) then
            $keepCl_1 \leftarrow False$
        end if
        if $keepCl_1 == False$ then
            Discard $cl_1$ from $Cls$
        end if
    end for
```

first performs subsumption between classifiers (lines 4 to 9), before performing the proposed filtering for the classifiers having Behavioral Sequences (lines 10 to 12). The execution time of this algorithm, although quadratic with the size of the population of classifiers, is still negligible compared to the time taken by ALCS to learn [4].

Algorithm 2 takes as argument two classifiers cl_1, cl_2, and the user threshold θ_r. A classifier cl_2 subsumes cl_1 if:

- The set of environmental situations described by the condition of cl_1, noted C_{cl_1}, is included in the set of environmental situations described by the condition of cl_2, noted C_{cl_2}.
- The actions A_{cl_1} of cl_1 and the actions A_{cl_2} of cl_2 are identical.
- The set of environmental changes described by the effect of cl_1, noted E_{cl_1}, is included in the set of environmental changes described by the effect of cl_2, noted E_{cl_2}.
- cl_2 is reliable, meaning its quality is larger than the related user threshold θ_r, or the quality of cl_2 is larger than the quality of cl_1.
- cl_2 has not been marked by any environmental situations or its mark M_{cl_2} strictly corresponds to that of cl_1, noted M_{cl_1}.

By ensuring that the condition and effect components of cl_1 are included in those of cl_2 for the same actions, any environmental transition described by one classifier to be deleted from the population is then necessarily described by another classifier. CRACS is also generic to any representation used in the $\{C, A, E\}$ tuples of the ALCS classifiers as long as it is possible to define how one condition or effect can include another.

Relying on classifiers not marked by an environmental situation indicates that their anticipations have always been correct. If two classifiers are marked, checking that their marks match indicates that these classifiers were used in the same environmental situations. Using the marks allows us to remove classifiers that are merely intermediate products of other classifiers, while minimizing an impairment of the environmental representations constructed by ALCS.

Algorithm 2 Subsumption of ALCS classifiers

```
function IsSubsumed($cl_1$,$cl_2$,$\theta_{\text{r}}$)
    if $C_{\text{cl}_1} \subseteq C_{\text{cl}_2}$ and
      $A_{\text{cl}_1} == A_{\text{cl}_2}$ and
      $E_{\text{cl}_1} \subseteq E_{\text{cl}_2}$ and
      ($q_{\text{cl}_2} > \theta_{\text{r}}$ or $q_{\text{cl}_2} \geq q_{\text{cl}_1}$) then
        if $M_{\text{cl}_2} == \varnothing$ then
            return $True$
        end if
        if $M_{\text{cl}_2} == M_{\text{cl}_1}$ then
            return $True$
        end if
    end if
    return $False$
```

CRACS was tested with a set of ALCS using different representations of $\{C, A, E\}$ tuples. The validation of CRACS was done using a maze benchmark because they are widely used as reinforcement learning benchmark [2]: the experimental protocol and the achieved results are presented in the following section.

4 PERFORMANCE IN MAZE ENVIRONMENTS

4.1 Experimental protocol

The mazes used to validate CRACS can be characterized according to different measures such as the average distance to the exit or the type of Perceptual Aliasing Issue an agent can experience. [2] proposes a measure of the complexity of mazes that quantifies how difficult it is for an agent to learn in these environments. The complexity is computed as the ratio of how long an agent trained by Q-Learning takes to reach the exit to the average distance to the exit (for further details, refer to [2]). Thus, the complexity of a maze mainly depends on the size of the maze, on the average distance to the exit and on environmental uncertainties like the Perceptual Aliasing Issue. The more a maze is uncertain, the higher this complexity. We used the maze benchmark from [17] as these mazes were clearly described in the literature and often used in different experimental protocols. The benchmark is made up of 23 mazes of different complexities (due to the occurrence of states related to the Perceptual Aliasing Issue for instance) summed up in the table 1.

Our experimental protocol is set to address the following questions:

- To what extent are the sizes of ALCS populations of classifiers reduced by CRACS?
- How does CRACS alter the environmental representations constructed by ALCS?
- To what extent does CRACS influence the ability of ALCS to perform their learning task?

Three different ALCS (BACS, PEPACS and BEACS) are used to construct a complete and accurate representation of their environment, by moving one grid-cell at a time and in either eight adjacent positions, while attempting to reach the exit as fast as possible. To make the learning and the solving of the task more complex, the results of actions have a 25% chance of being uncertain, in which case an ALCS performs a random action without knowing which

Maze	Complexity	Maze	Complexity
MazeE2	> 250	Woods101.5	> 225
Maze10	> 150	Woods102	> 150
Woods100	> 150	Woods101	> 150
MazeF4	> 125	MazeE1	> 100
Maze7	> 100	MazeB	< 10
Littman89	< 10	MiyazakiA	< 10
MiyazakiB	< 10	MazeD	< 10
Littman57	1	Cassandra4x4	1
Maze4	$1*$	Maze5	$1*$
MazeA	$1*$	MazeF1	$1*$
MazeF2	$1*$	MazeF3	$1*$
Woods14	$1*$		

Table 1: Maze complexities from [2] sorted by descending order. $*$ indicates that no values were provided by [2] because corresponding mazes does not have the Perceptual Aliasing Issue. We can expect their complexity to be 1, as an agent employing *Q-learning* can optimally reach the exit of mazes in any position.

one. Their perceptive capabilities are limited to the eight squares adjacent to each position. Their starting position in the mazes is random (but distinct from the exit).

For each maze of the benchmark, 30 runs were performed using each of these three ALCS. A run firstly consists of a succession of 5000 trials, that are constrained explorations until the exit or the maximal number of actions (100) are reached: ϵ is set to 0.8 for the ϵ-greedy policy used to select actions; the learning rate of the rule discovery component and the reinforcement component, β, is set to 0.05; the PEP learning rate of PEPACS is set to 0.01; the maximal length of the Behavioral Sequences of BACS and BEACS is set to 3. A deep copy of the populations of classifiers is then made to study the impact of CRACS on the copies when the original populations serve as a control.
Then, the ALCS are switched to pure exploitation (*i.e.* no use of rule discovery component) and have 500 trials to bootstrap an efficient decision policy ($\epsilon = 0.2$, $\beta = 0.05$) and 500 more trials to stabilize the rewards ($\epsilon = 0$, $\beta = 0.05$), before recording the number of actions required by the ALCS to reach the exit for 500 more trials ($\epsilon = 0$, $\beta = 0.05$). Other parameters not described here are initialized to their default values.

The source code of the three different ALCS, the mazes and the experimental workflows is available on GitHub.

4.2 Metrics

To determine how CRACS affects environmental representations, knowledge ratios, average specificities of reliable classifiers, and average EP-accumulated errors are collected for each experiment (EP stands for Enhanced Predictions).
The knowledge ratio is the ratio of correct transitions learned by at least one reliable classifier to all possible transitions. Only transitions that led to environmental changes are included.
The average specificity of reliable classifiers is the average ratio of wildcards in the condition components of classifiers whose quality is larger than the user threshold θ_r. The average specificities of the reliable classifiers allow us to calculate the *generalization rate*:

$$1 - \frac{\text{Average specificity of compacted reliable classifiers}}{\text{Average specificity of control reliable classifiers}}$$

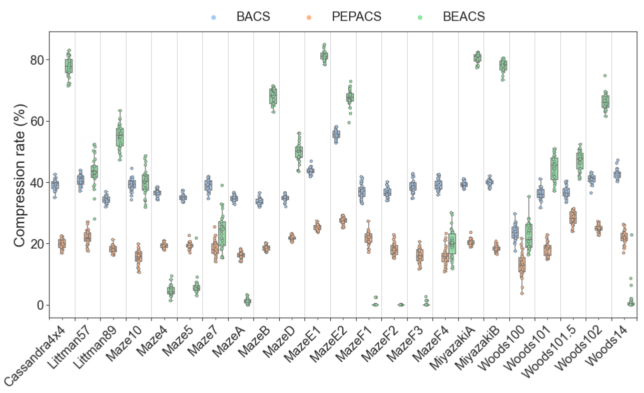


Figure 2: Compression rates for each maze environment and for each classifier system.

The *generalization rate* assesses the increase or decrease in generalization of reliable classifiers between control and compacted populations.
The average EP-accumulated error measures the accuracy related to the probabilities of anticipating a change within the effect components of classifiers [19]. This measure is thus different from the knowledge ratio which only provides information about the presence of a change in an effect component.

The sizes of the populations of classifiers and the number of steps needed to reach the exit of the mazes are also collected for each experiment to respectively address the first and the last questions.

In particular, the sizes of the populations of classifiers are used to compute the *compression rate* between the control and compacted populations:

$$1 - \frac{\text{Size of compacted population}}{\text{Size of control population}}$$

The *compression rate* quantifies the reduction in the number of classifiers between the analyzed populations.

All metrics were averaged over the 30 runs for each environment. The obtained averages were compared with p-values computed by Welch t-tests with (Welch-) Satterthwaite degrees of freedom (significance threshold 0.05).

4.3 Results

To what extent are the sizes of ALCS populations of classifiers reduced by CRACS?.

Figure 2 shows the compression rates following the use of the CRACS algorithm for each environment and for each ALCS. CRACS significantly reduces the sizes of populations of classifiers constructed by BEACS for 19 of the 23 mazes, where the compression rates can go up to 80%. The populations constructed for *MazeF1*, *MazeF2*, *MazeF3* and *Woods14* are not modified by the algorithm. The sizes of populations of classifiers in BACS are reduced for all environments, with a reduction up to 55%. Finally, the sizes of populations of PEPACS are also significantly reduced across all environments, to a maximum of 28%.

How does CRACS alter the environmental representations constructed by ALCS?.

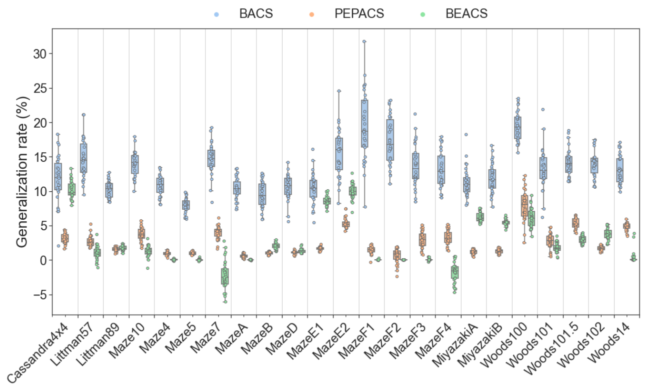


Figure 3: Generalization rates for each maze environment and for each classifier system.

The knowledge ratios of the control populations of classifiers are strictly identical to the knowledge ratios of the compacted populations, for all environments and all ALCS. The compaction algorithm did not remove any classifiers that would have induced knowledge losses in the environmental representations constructed by the ALCS. Thus, any environmental transition described by a classifier of a control population is found in a classifier of the corresponding compacted population.

The average EP-accumulated errors then test whether the compacted populations of ALCS can correctly indicate the probabilities of anticipating upcoming situations for each environmental transition. The average EP-accumulated errors of the control and compacted populations of BEACS show no significant difference in any of the environment. The average EP-accumulated errors of control and compacted populations of BACS also show no significant differences. The average EP-accumulated errors of the compacted populations of PEPACS are 2% larger than those of the control populations for 11 of the 23 environments and show no significant differences for the remaining 12 environments.

The mean specificities of the reliable classifiers in the BEACS compacted populations are higher for 2 environments (*Maze7* and *MazeF4*), show no significant differences for 9 environments, and are lower than those of the control populations for the remaining 12 environments. The average specificities of the reliable classifiers in the BACS compacted populations are lower than those of the control populations for all environments. The mean specificities of the reliable classifiers of the PEPACS compacted populations are lower than those of the control populations except for *MazeF2* where they are identical.

Figure 3 shows the generalization rates achieved by the control and compacted populations of the three ALCS. In other words, this figure clarifies the variations observed in the average specificities of reliable classifiers. CRACS increases the generalization of BEACS reliable classifiers by up to 10% for 12 environments. Conversely, for *Maze7* and *MazeF4*, this algorithm specializes the reliable classifiers by about 3%. CRACS increases the generalization of BACS reliable classifiers by up to 17.5%. Finally, CRACS increases the generalization of PEPACS reliable classifiers up to 7.5%.

To what extent does CRACS influence the ability of ALCS to perform their learning task?

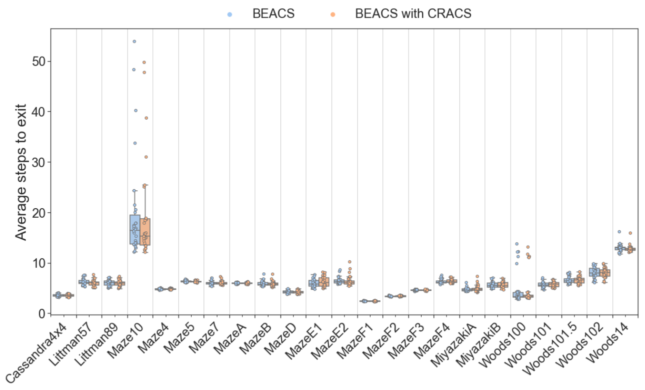


Figure 4: Numbers of steps needed for BEACS to reach the exit of each maze, before and after having compacted the populations with CRACS.

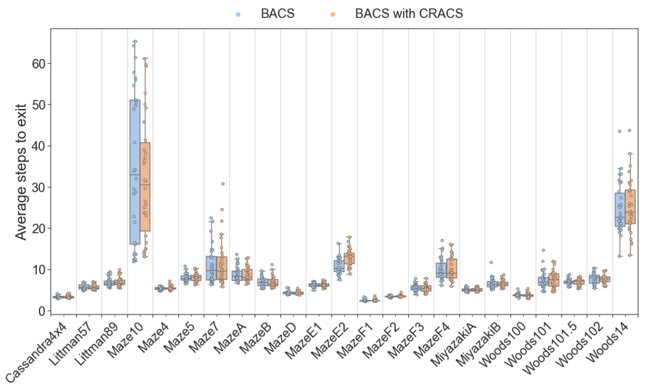


Figure 5: Numbers of steps needed for BACS to reach the exit of each maze, before and after having compacted the populations with CRACS.

The compaction algorithm does not change the ability of BEACS to reach the maze output for all environments (figure 4).

The compaction algorithm does not change the ability of BACS to reach the maze exit for 22 of the 23 environments (figure 5). This capacity is modified for the *MazeE2* maze where the algorithm increases the average number of steps by 1.75 additional step, which corresponds to a relative increase of 14%.

The compaction algorithm does not change the ability of PEPACS to reach the exit of mazes for 21 of the 23 environments (figure 6). The algorithm increases the average number of steps for the maze *MiyazakiA* by 0.2 additional steps (a relative increase of 4.6%). Finally, the algorithm decreases the average number of steps for the maze *MazeE2* by 1.4 additional step (a relative decrease of 4.8%).

5 DISCUSSION

The greater the environmental uncertainty or the larger the size of the environment, the greater the compaction.

The compression rates seem on average higher the more uncertain the environments are or the larger the sizes of the environment are, as it implies more numerous and complex environmental transitions to be discovered and exploited. This result can be explained

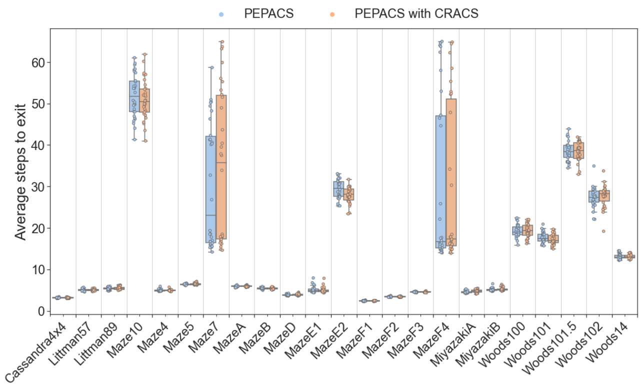


Figure 6: Numbers of steps needed for PEPACS to reach the exit of each maze, before and after having compacted the populations with CRACS.

by the use of a structure allowing to anticipate several environmental situations in the case of BEACS and PEPACS, and by the uncontrolled use of Behavioral Sequences in BACS.

With BEACS and PEPACS, the construction of classifiers able to correctly anticipate all the environmental situations for a given situation and action implies a greater number of intermediate classifiers describing part of the situations to be anticipated. These intermediate classifiers may still be present in the population at the end of the training: they are then effectively deleted by our compaction algorithm.

With BACS, many classifiers using Behavioral Sequences are created without the system testing them. The filtering introduced in our compaction algorithm is thus more important than subsumption in reducing the size of the classifier populations.

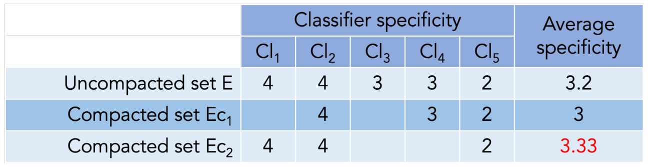
Environmental representations are preserved.

The results show that the environmental representations built by the three ALCS are not impaired by CRACS.

Indeed, the knowledge ratios show that no environmental transition is lost. The descriptions of these transitions remain, overall, of good quality, in particular for BEACS where they are unchanged (the average EP-accumulated errors have no significant difference). This is particularly important for BEACS because this system was designed to allow such descriptions, which BACS and even PEPACS cannot achieve [19]. The small relative differences of the average EP-accumulated errors of BACS and PEPACS, which are at most 2%, are thus negligible.

The average specificities of the reliable classifiers are also unchanged or lower for the three systems. CRACS thus allows the environmental representations to rely on the most general classifiers of the populations: the generalization rates of figures 3 quantify this effect. This result remains valid, especially for BEACS, for the two environments *Maze7* and *MazeF4* where these specificities increase. This increase is not due to the deletion of the most general reliable classifiers and gives further evidence that CRACS works well. Figure 7 illustrates the loss of specificity observed in these environments, where classifiers with specificity below the average specificity of a subset result in an increase when deleted.

Environmental uncertainty such as uncertain actions seems to decrease the generalization rates of BEACS and PEPACS achieved



	Classifier specificity					Average specificity
	Cl_1	Cl_2	Cl_3	Cl_4	Cl_5	
Uncompacted set E	4	4	3	3	2	3.2
Compacted set Ec_1		4		3	2	3
Compacted set Ec_2	4	4			2	3.33

Figure 7: Illustration of the increase in average specificity of reliable classifiers observed in *Maze7* and *MazeF4* with BEACS.

with CRACS when those of BACS seem to increase. What is highlighted here is a consequence of environmental uncertainty and not a limitation of the proposed algorithm. Uncertain actions imply that several environmental situations are to be anticipated by ALCS for a given action and situation. Thus, several changes are to be anticipated, leading to the further specialization of the classifiers to describe these changes. This makes the ALCS classifiers less generalizable, hence the lower generalization rates.

However, this reasoning is incomplete with BACS: it does not fully explain the increase in generalization rates. Uncertain actions lead to the creation of many classifiers having Behavioral Sequences since all environmental situations trigger their creation (even if they are not necessary). These classifiers will become more and more specific since the errors associated with their use will be more numerous, as the number of actions to be performed is larger. CRACS, therefore, has a more important role, since it acts directly on these classifiers *via* filtering and subsumption. Filtering removes more classifiers having Behavioral Sequences that would not have been tested by BACS and whose specificity is greater because of the errors associated with their use. Subsumption then favors the classifiers having Behavioral Sequences created before the aforementioned overspecificity occurs. BEACS, which also uses Behavioral Sequences, does not have this drawback, since their creation is conditioned by mechanics that seek to determine their necessity [19].

ALCS's ability to solve tasks is preserved.

The results show that CRACS does not impact the ability of ALCS to reach the exit of mazes as quickly as possible, despite some differences observed (figure 5 *MazeE2* with BACS, figure 6 *MiyazakiA* and *MazeE2* with PEPACS).

These differences may result from the fact that CRACS does not take into account the rewards of the ALCS classifiers, as well as from characteristics specific to certain environments (like the type of Perceptual Alising issue, the sizes of these environments, ...). The decision policies established by the ALCSs during the exploration of the environments may be altered by CRACS, at the risk of promoting suboptimal decision policies. The experimental protocol seeks to allow the ALCS to develop the most efficient decision policy possible for the task. This protocol is common across environments: the same resources are allocated to the ALCS when they seek to solve their learning task, regardless of environmental properties. Thus, decision policies after population compression and exploitation may remain suboptimal.

Another possibility to explain these differences, especially for BACS, is the filtering performed on the classifiers having Behavioral Sequences. This filtering removes the classifiers that have not been sufficiently tested by ALCS. However, the most appropriate

Behavioral Sequences may have not been used enough during the exploration of the environments and are then deleted. Their deletion would then promote less efficient decision policies compared to populations where they would still be present.

Although these discrepancies would require further analysis to determine their origin more precisely, they remain marginal compared to the overall performance of CRACS. Indeed, CRACS preserves the learned environmental representations upon which the decision policies are built. Because there is no environmental transition lost, CRACS does not impair the ability of these different ALCS to exploit all these transitions to set up decision policies fitted to the learning tasks.

6 CONCLUSION

Anticipatory Learning Classifier Systems can be used for reinforcement learning problems in uncertain environments. However, as with any rule-based approach, environmental uncertainties increase the number of classifiers produced by ALCS, reducing *de facto* their interpretability.

To reduce the size of ALCS populations of classifiers and thus extract knowledge from them, CRACS (Compaction of Rules in Anticipatory Classifier Systems) is introduced in this paper. CRACS solely relies on the knowledge contained within the populations and has been designed to be used with any kind of ALCS. Based on a comprehensive review of knowledge extraction methods dedicated to Learning Classifier Systems, CRACS relies on a subsumption mechanism to avoid knowledge losses and classifier filtering.

The results of a thorough experimental protocol using maze environments with various levels of uncertainty and different ALCS show that CRACS (1) efficiently shrinks the populations of classifiers (2) without impairing the built environmental representations, (3) contributes to the generalization of these representations and (4) does not make ALCS's ability to solve their tasks less effective. CRACS is thus able to reinforce their interpretability without degrading their learning capacities.

The review of different knowledge extraction techniques for Learning Classifier Systems paves the way for future work on ALCS populations. This is particularly the case for techniques based on the building of new rules. For example, they could be used in conjunction with ALCS generalization mechanisms. They could also provide new explanatory elements about the perceptual changes observed in the ALCS classifiers. CRACS could also be improved to facilitate the implementation of decision policies by ALCS. Our algorithm does not look at the reward predictions of the classifiers that are deleted: these could for example be transferred under certain conditions between the classifiers.

REFERENCES

[1] Alejandro Barredo Arrieta, Natalia Diaz-Rodriguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion* 58 (2020), 82–115.

[2] Anthony J Bagnall and Zhanna V Zatuchna. 2005. On the classification of maze problems. In *Foundations of Learning Classifier Systems*. Springer, 305–316.

[3] A Martin V Butz, B David E Goldberg, and C Wolfgang Stolzmann. 2002. The anticipatory classifier system and genetic generalization. *Natural Computing* 1, 4 (2002), 427–467.

[4] Martin V Butz, Pier Luca Lanzi, and Stewart W Wilson. 2008. Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation* 12, 3 (2008), 355–376.

[5] Martin V Butz and Wolfgang Stolzmann. 2001. An algorithmic description of ACS2. In *International Workshop on Learning Classifier Systems*. Springer, 211–229.

[6] Phillip William Dixon, Dawid Wolfe Corne, and Martin John Oates. 2002. A ruleset reduction algorithm for the XCS learning classifier system. In *International workshop on learning classifier systems*. Springer, 20–29.

[7] Chunsheng Fu and Lawrence Davis. 2002. A modified classifier system compaction algorithm. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Citeseer, 920–925.

[8] Yang Gao, Joshua Zhexue Huang, and Lei Wu. 2007. Learning classifier system ensemble and compact rule set. *Connection Science* 19, 4 (2007), 321–337.

[9] Joachim Hoffmann. 2003. Anticipatory behavioral control. In *Anticipatory behavior in adaptive learning systems*. Springer, 44–65.

[10] John H Holland, Lashon B Booker, Marco Colombetti, Marco Dorigo, David E Goldberg, Stephanie Forrest, Rick L Riolo, Robert E Smith, Pier Luca Lanzi, Wolfgang Stolzmann, et al. 1999. What is a learning classifier system?. In *International Workshop on Learning Classifier Systems*. Springer, 3–32.

[11] Faten Kharbat, Mohammed Odeh, and Larry Bull. 2007. New approach for extracting knowledge from the XCS learning classifier system. *International Journal of Hybrid Intelligent Systems* 4, 2 (2007), 49–62.

[12] Tim Kovacs. 1998. XCS classifier system reliably evolves accurate, complete, and minimal representations for Boolean functions. In *Soft computing in engineering design and manufacturing*. Springer, 59–68.

[13] Pier Luca Lanzi et al. 2001. Mining interesting knowledge from data with the XCS classifier system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann San Francisco, CA 94104, USA, 958–965.

[14] Yi Liu, Will N Browne, and Bing Xue. 2018. Adapting bagging and boosting to learning classifier systems. In *International conference on the applications of evolutionary computation*. Springer, 405–420.

[15] Yi Liu, Will N Browne, and Bing Xue. 2021. A comparison of learning classifier systems' rule compaction algorithms for knowledge visualization. *ACM Transactions on Evolutionary Learning and Optimization* 1, 3 (2021), 1–38.

[16] Yi Liu, Will N Browne, and Bing Xue. 2021. Visualizations for rule-based machine learning. *Natural Computing* (2021), 1–22.

[17] Romain Orhand, Anne Jeannin-Girardon, Pierre Parrend, and Pierre Collet. 2020. BACS: A Thorough Study of Using Behavioral Sequences in ACS2. In *International Conference on Parallel Problem Solving from Nature*. Springer, 524–538.

[18] Romain Orhand, Anne Jeannin-Girardon, Pierre Parrend, and Pierre Collet. 2020. PEPACS: Integrating Probability-Enhanced Predictions to ACS2. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. 1774–1781.

[19] Romain Orhand, Anne Jeannin-Girardon, Pierre Parrend, and Pierre Collet. 2022. Accurate and Interpretable Representations of Environments with Anticipatory Learning Classifier Systems. In *Genetic Programming: 25th European Conference, EuroGP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings*. Springer, 245–261.

[20] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[21] Olivier Sigaud and Stewart W Wilson. 2007. Learning classifier systems: a survey. *Soft Computing* 11, 11 (2007), 1065–1078.

[22] Jie Tan, Jason Moore, and Ryan Urbanowicz. 2013. Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In *ECAL 2013: The Twelfth European Conference on Artificial Life*. MIT Press, 110–117.

[23] Ryan J Urbanowicz, Ambrose Granizo-Mackenzie, and Jason H Moore. 2012. An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. *IEEE computational intelligence magazine* 7, 4 (2012), 35–45.

[24] Eva AM van Dis, Johan Bollen, Willem Zuidema, Robert van Rooij, and Claudi L Bockting. 2023. ChatGPT: five priorities for research. *Nature* 614, 7947 (2023), 224–226.

[25] Stewart W Wilson. 1995. Classifier fitness based on accuracy. *Evolutionary computation* 3, 2 (1995), 149–175.

[26] Stewart W Wilson. 2001. Compact rulesets from XCSI. In *International Workshop on Learning Classifier Systems*. Springer, 197–208.

[27] Robert Zhang, Rachael Stolzenberg-Solomon, Shannon M Lynch, and Ryan J Urbanowicz. 2021. LCS-DIVE: An automated rule-based machine learning visualization pipeline for characterizing complex associations in classification. *arXiv preprint arXiv:2104.12844* (2021).