Coevolutionary Heuristics for Maximization of Expected Utility

Thomas Willkens and Jordan Pollack

DEMO Lab
Brandeis University, Waltham, MA, USA
twillkens@brandeis.edu, pollack@brandeis.edu

Abstract

Coevolutionary learning is a flexible paradigm with many applications, but its practice is hindered by various subtle pathologies. The Discovery of Objectives via Clustering (DOC) algorithm is a heuristic approach for learner selection that employs the maximization of expected utility (MEU) solution concept. DOC shows some potential for addressing certain issues; however, modification is found necessary to prevent the pathology of overspecialization. We propose QueMEU, a novel test generator and memory mechanism that uses a sampling policy over a queue to maintain diversity and provide an effective learning gradient. Incorporation of QueMEU improves the performance of DOC on abstract numbers game problems as well as the challenging density classification task. QueMEU outperforms the standard test generator as well as a more established alternative that employs fitness sharing.

Introduction

The field of *coevolutionary learning* (CEL) has contributed a rich body of theory and work with relevance to various disciplines, including artificial life, reinforcement learning (RL), and black-box optimization.

In artificial life, notable early results with coevolution include the sorting networks of Hillis (1990), the virtual creatures of Sims (1994), and puncuated equilibria among iterated Prisoner's Dilemma strategies in Lindgren (1992). CEL has connections to *novelty search* (Lehman and Stanley, 2011; Popovici et al., 2012) and been used to investigate *open-endedness* and related dynamics such as *complexity growth* (Brant and Stanley, 2017; Stanley, 2019; Wang et al., 2019; Moran and Pollack, 2019; Willkens and Pollack, 2022, 2023).

CEL has produced effective game-playing agents for domains often studied in RL, such as checkers, backgammon, and small-board Go (Fogel, 2001; Pollack et al., 1970; Krawiec et al., 2011). CEL has also been applied to video games and real-time strategy games (Monroy et al., 2006; Elfeky et al., 2021). Arulkumaran et al. (2019) argues that population-based RL methods such as AlphaStar can be interpreted from a coevolutionary perspective.

In optimization, Wolpert and Macready (2005) show that some coevolutionary algorithms offer *free lunches*, in contrast to the well-known results of Wolpert and Macready (1997). Further research in connection with coevolution led to the genesis of *co-optimization* as an independent field of study (Service and Tauritz, 2008; Popovici and Jong, 2009; Popovici, 2017). Coevolution has since been applied in diverse areas such as cybersecurity, educational technology, and healthcare recommender systems (Garcia et al., 2017; Wiegand et al., 2022; Alcaraz-Herrera and Cartlidge, 2023).

However, CEL can lead to unpredictable results due to unique challenges known as *coevolutionary pathologies*. These include *disengagement*, *overspecialization*, *intransitivity*, and *Red Queen* dynamics. If unaddressed, these pathologies may result in *loss of gradient*, *cycling*, and collapse to *mediocre stable states* (Ficici and Pollack, 1998; Cliff and Miller, 1995).

We focus here on heuristics associated with the *Pareto coevolution* learning paradigm (Ficici and Pollack, 2001). These approaches are driven by *solution concepts* for test-based problems; a solution concept refers to the set of learner entities that the search procedure is intended to find (Ficici, 2005; Popovici et al., 2012). Also important is the use of *informativeness* as a criterion for test selection to establish more effective learning gradients. Formalized using order theory in Bucci (2007), these concepts inspired a family of *coevolutionary archive* algorithms with *monotonicity* guarantees intended to address pathologies (de Jong, 2007).

In particular, we examine the Discovery of Objectives via Clustering (DOC) algorithm (Liskowski and Krawiec, 2017). DOC is a coevolutionary heuristic for learner selection intended to approximate the *maximization of expected utility* (MEU) solution concept. MEU seeks the set of learners with the highest expected utility when presented with a test selected at random (Popovici et al., 2012). We identify a possible issue with DOC as presented that could lead to *overspecialization* and explain reported suboptimal results.

We propose a novel test generator, QueMEU (Queue-Based Generator for Maximization of Expected Utility), which has some features in common with both classic co-

evolution as well as more recent work in computer vision and neural architecture search (Juille, 1998; Real et al., 2018). We show that using DOC for learners in combination with QueMEU for tests leads to improved performance on *numbers game* benchmark problems (de Jong and Pollack, 2004) as well as the *density classification task* (DCT) with $N = 149$ (Pagie and Mitchell, 2002). QueMEU is also shown to outperform a more established method, the *Advanced* generator of de Jong and Bucci (2006).

Test-Based Coevolution

A test-based problem may be defined by the tuple $\langle \mathcal{S}, \mathcal{T}, g, Q \rangle$, where:

- $\mathcal{S}$ is the set of *candidate solutions* to the problem, also referred to as *learners*.
- $\mathcal{T}$ is the set of *tests*, defined as possible problem instances that learners must solve.
- $g : \mathcal{S} \times \mathcal{T} \to \mathbb{R}$ is the *interaction function*, which yields a real-valued outcome for a learner on a test. We concern ourselves here with binary outcomes, where $g(s,t) = 1$ if the learner s solves the test t and 0 otherwise.
- $S^* \subseteq \mathcal{S}$ represents the goal of the search: the set of learners with properties captured by the appropriate *solution concept* for the problem (Ficici, 2004).
- $Q : \mathcal{S} \to \mathbb{R}$ is the *quality function*, which ranks learners globally with respect to the solution concept.

We focus here on the *maximization of expected utility* (MEU) solution concept, which seeks the set of learners that maximize the expected outcome against a test selected from $\mathcal{T}$ at random:

$$Q_{MEU}(s) = \mathbb{E}_{t \in \mathcal{T}}[g(s,t)]. \tag{1}$$

MEU is a common solution concept that can be applied to a wide range of problems, such as finding generalist game-playing agents or designs for resilient physical structures (Ficici, 2004; de Jong, 2005; Popovici et al., 2012; Popovici, 2017).

Learner Evaluation

As the number of learners and tests is typically far too large for exhaustive fitness calculation, coevolution employs populations of learners $S \subset \mathcal{S}$ and tests $T \subset \mathcal{T}$. Interaction outcomes between members of S and T are calculated using g and stored in an *interaction matrix* named G. Conventionally, a scalar *fitness function* is used to rank learners for the purpose of selection:

$$f_T(s) = \frac{|\{t \in T : g(s,t) = 1\}|}{|T|} \tag{2}$$

Scalar fitness evaluation is often suboptimal for coevolution as it discards information about which learners pass which tests. Variations of *Pareto coevolution* (Noble and Watson, 2001; Ficici and Pollack, 2001) exploit the connection between test-based coevolution and multiobjective optimization, using the *dominance relation* to identify learners with unique abilities:

$$s_1 \succ_G s_2 \Leftrightarrow \forall t \in T : g(s_1,t) \geq g(s_2,t) \wedge \exists t \in T : g(s_1,t) > g(s_2,t). \tag{3}$$

Ranking learners using dominance helps address the *intransitivity* pathology (Bucci and Pollack, 2003); however, Ficici and Pollack (2001) note that the nondominated front may eventually grow to encompass nearly the whole population. To maintain progress and diversity in such conditions, they propose a form of *implicit fitness sharing* (IFS) as an intralayer ranking mechanism (Juillé and Pollack, 1996; Smith et al., 1993). IFS assigns greater reward to a learner for solving a test that few others can solve, and gives less credit for solving a test that many others can solve:

$$f_T^{IFS}(s) = \sum_{t \in T : g(s,t)=1} \frac{1}{|\{s' \in S : g(s',t) = 1\}|} \tag{4}$$

Test Evaluation

For evaluation of tests, the simplest method is to reward the more "challenging" tests that fail the most learners:

$$f_S(t) = \frac{|\{s \in S : g(s,t) = 0\}|}{|S|} \tag{5}$$

Diversity maintenance may likewise be achieved using fitness sharing:

$$f_S^{IFS}(t) = \sum_{s \in S : g(s,t)=0} \frac{1}{|\{t' \in T : g(s,t') = 0\}|}. \tag{6}$$

However, this method may result in *disengagement*, such that the entire test population is too difficult for any learner to beat, leading to loss of gradient. The notion of *distinctions* (Ficici and Pollack, 2001) is proposed to help address this issue; when using distinctions, we instead reward tests that reveal one learner to perform better than the other. This is related to the concept of *test informativeness* and theory regarding the geometrical organization of test-based problems (Bucci, 2007).

$$d_S(t) = \sum_{(s_1,s_2) \in S \times S} [g(s_1,t) > g(s_2,t)]. \tag{7}$$

We likewise may perform IFS over the distinctions made by the test population, assigning greater reward to tests that

make distinctions between learners that few other tests can make:

$$d_S^{IFS}(t) = \sum_{(s_1,s_2)\in S\times S} \frac{[g(s_1,t) > g(s_2,t)]}{|\{t' \in T : g(s_1,t') > g(s_2,t')\}|}. \tag{8}$$

The *Advanced* generator of de Jong and Bucci (2006) uses a linear combination of shared fitness and shared distinctions. This rewards more challenging tests that still make distinctions between learners:

$$f_S^{ADV}(t) = \alpha f_S^{IFS}(t) + \beta d_S^{IFS}(t). \tag{9}$$

In that work, $\alpha = 3.0$ and $\beta = 1.0$ were used.

Archive Methods

While sometimes effective when used in a typical two-population setup, the evaluation methods described above have been shown to suffer from coevolutionary pathologies such as overspecialization (Watson and Pollack, 2002; de Jong and Pollack, 2004; de Jong and Bucci, 2006). For this reason, a family of *coevolutionary archive* algorithms have been proposed to ensure *monotonic progress* toward the desired solution concept (Ficici, 2005). These include Nash Memory for the Nash equilibrium solution concept (Ficici and Pollack, 2003); the Incremental Pareto Coevolution Archive (IPCA), the Layered Pareto Coevolution Archive (LAPCA), and the Dimension Extracting Coevolutionary Archive (DECA) for the Pareto optimal set solution concept (de Jong, 2004, 2007; de Jong and Bucci, 2006); and Max-Solve for the MEU solution concept (de Jong, 2005).

Many of these archive methods are paired with *generators* akin to $(\mu + \lambda)$ evolution strategies (Beyer and Schwefel, 2002). For example, the approach recommended in de Jong and Bucci (2006) uses $f_T^{ADV}(s)$ and $f_S^{ADV}(t)$ symmetrically to rank both learners and tests, discarding the bottom half of each population and choosing parents from each at random (de Jong, 2004, 2005). These explorative generators provide input to the coevolutionary memory, which is intended to provide stability and counteract pathologies such as cycling.

However, the cost of maintaining an archive grows with the number of learners, and new candidates must interact with archive members to determine fitness for entry. This can be restrictive depending on the expense of the fitness function and the desired resolution of the archive. Archives often must be bounded in practice, and an archive that is too small cannot approximate monotonic progress. Thus the practical utility of archive methods is not guaranteed, and their relative efficiency remains an open research area (Popovici et al., 2012).

For this reason, we will focus our attention on DOC and related heuristics that do not require additional evaluations per generation. Future work could include a principled comparison with archive methods as well as the family of *Population-Based Pareto Hill Climber* (P-PHC) variants (Bari et al., 2018), which present a promising alternative approach.

Discovery of Objectives via Clustering

The Discovery of Objectives via Clustering (DOC) algorithm offers mitigation for the problems of the *scalar evaluation bottleneck* and the expanding nondominated Pareto front (Liskowski and Krawiec, 2017). By clustering on the columns of the interaction matrix, DOC produces a compressed set of *derived objectives* that are loosely analogous to particular *skills* that a learner may possess:

1. Obtain the $m \times n$ interaction matrix G using the interaction function g with the set of learners S and tests T.
2. Perform clustering on the columns of G to produce k clusters, partitioning the set of tests into $\{T_1, \ldots, T_k\}$ where $T_j \subset T, j \in \{1, \ldots, k\}, 1 \leq k \leq n$, and $T_j \neq \emptyset$.
3. Create the *derived interaction matrix*, named G', by averaging row-wise the columns of G corresponding to each cluster as follows:

$$g'_{i,j} = \frac{1}{|T_j|} \sum_{t\in T_j} g(s_i, t). \tag{10}$$

Techniques from traditional multiobjective optimization, such as the well-known Nondominated Sorting Genetic Algorithm II (NSGA-II) of Deb et al. (2002) may then be employed on G' to select a diverse and performant set of learners.

One point of concern is that DOC may introduce *false positive distortions*, where one learner dominates the other with respect to G' while being incomparable in G. Liskowski and Krawiec (2017) argue that given the limited information presented by G at a moment in time, complete preservation of the original dominance relationships is unecessary, and false positives could even be beneficial from a heuristic point of view. Moreover, the more damaging errors of *false negative distortions* and *dominance inversions* are shown in Liskowski and Krawiec (2017) to be impossible with DOC.

One might assume that the use of NSGA-II implies that DOC instantiates one of the Pareto Optimal Set solution concepts (Ficici, 2004). However, Liskowski and Krawiec (2017) argue that the loss of information regarding full dominance relationships relates DOC more closely to the MEU solution concept (Popovici et al., 2012).

Risk of Overspecialization

While DOC is shown to provide statistically significant improvement over a base model, closer inspection reveals potential room for enhancement. Liskowski and Krawiec

(2017) justifiably focus their attention on the learner population given the nature of their contribution, but the test population also plays an important role in driving coevolutionary progress. We note that the test population in Liskowski and Krawiec (2017) is evaluated using a simple sum of distinctions, equivalent to $d_S(t)$, paired with a straightforward $(\mu + \lambda)$ evolution strategy. In the absence of effective diversity maintenance, tests are likely to coalesce around a subset of problem dimensions, leading to overspecialization (de Jong and Pollack, 2004; Bucci, 2007).

The choice of test generator may explain the acknowledged underperformance of DOC on *numbers game* tasks such as Compare-on-All (Liskowski and Krawiec, 2017). Learners produced by a stable coevolutionary algorithm would be expected to have high values in all dimensions and thus pass all initial tests by the end of the run. Despite this potential issue, DOC has some promising properties. The MEU solution concept applies to many important domains (Popovici, 2017), and advancements in this area could prove to be of general interest. We thus begin our search for a more effective test generator by considering earlier coevolutionary literature as well as contemporary alternatives.

QueMEU: A Queue-Based Test Generator

Asymmetrical Generation Gaps

Prior to the advent of archive methods, coevolutionary populations using asymmetrical rates of reproduction were found to be effective: One population may produce rapid turnover, while the other behaves in more of a steady-state fashion. For example, Juille (1998) used a 5% generation gap for the test population and an 80% generation gap for the learner population. As this configuration produced excellent results on the *density classification task* (DCT) (Pagie and Mitchell, 2002), it is worth considering the benefits of a similar approach. However, Juille (1998) uses a substantial amount of domain knowledge in the form of hand-crafted fitness functions. We desire a test generator that can be applied to a wider range of problems, using only information about whether a learner passed or failed a test.

Aging Evolution

An interesting evolutionary approach was proposed in Real et al. (2018), which helped bring neural architecture search (NAS) to mainstream attention after achieving state-of-the-art performance on the ImageNet dataset. The *aging evolution* method, also referred to as *regularized evolution*, was argued to have unique benefits for finding effective and trainable network architectures. Aging evolution represents the population as a steady-state queue. On each generation, the best-performing network spawns a child that is placed at the front of the queue, and the model at the end is dequeued. This method was found to be especially effective for NAS; since each model has a limited lifespan and must be retrained before evaluation, trainable models are effectively selected for.

While employing a queue-based population, our use case is different. We wish our test population to avoid overspecialization and falling into a single niche, while also maintaining a steady gradient for the learner population to improve in performance on over time. Put loosely, we intend to approximate the practical effects of archive-based coevolutionary methods but in a stochastic fashion that unfolds more gradually over time.

QueMEU Algorithm Description

In its most general form, QueMEU can be defined as a sampling policy over a queue paired with some measure of test informativeness to select winners from the resulting interaction pool. Rather than place specific tests in an archive in particular relation to learners, members pass down the queue with some chance of being sampled or creating children each generation. If chosen and found to provide useful information about the learner population, the test is re-enqueued. This provides more gradual and targeted exploration while giving learners the chance to rediscover old skills that have been forgotten. Once a test has been moved back into the active population, learners are given focused exposure for a guaranteed duration to solidify progress on the lesson presented by the test.

For our experiments, we use a particular QueMEU variant in which the sampling policy can be represented as pair of queues: the *population* queue, initialized with random tests, which feeds into the *archive* queue, which is initially empty. Parents are sampled from the population to generate a set of *children*. The test *interaction pool* remains a fixed size each generation, comprising the population, the children, and a set of tests sampled at random from the archive. Following interaction with learners, each member of the active pool is scored using $d_S^{IFS}(t)$; i.e., the sum of distinctions made between members of the learner population, each discounted by the number of other tests that make the same distinction.

After sorting by score, the set of n_{win} highest-scoring tests are added to the front of the population queue; if a test is already present in the population or the archive, it is displaced and moved to the front. (If an older test is found to be a fruitful source of distinctions, that implies the test has not yet been fully mastered, so we wish to give the learner population more time to focus on it.)

To maintain a constant population size, the oldest tests are removed and added to the archive. If the size of the archive exceeds the limit, the oldest tests are permanently removed. To produce the interaction pool for the next generation, we sample n_{arc} members from the archive along with $n_{par} = n_{pop} - n_{arc}$ parents from the population to generate children. This aims for a compromise between stability and targeted exploration that does not rely upon domain knowledge.

Algorithm 1 QueMEU: Test Generator

Initialize the *test queue*.
Define a *policy* for sampling operations.
Use the policy to sample *representatives* and *parents* from the queue.
Generate *children* from the *parents*.
Combine the *representatives* and the *children* to form the *interaction pool*.
while not terminated **do**
    Perform interactions between the *interaction pool* and the current set of *learners*.
    for each test in the *interaction pool* **do**
        Score the test using an *informativeness* measure.
    end for
    Select the *winners* with the highest scores.
    Filter out *winners* from the *test queue*.
    Enqueue the *winners* to the front of the *test queue*.
    if *test queue* size exceeds limit **then**
        Remove oldest tests from *test queue*.
    end if
    Optionally, update the *policy* using information gained from the search.
    Sample from the queue using the *policy* to generate a fresh *interaction pool*.
end while

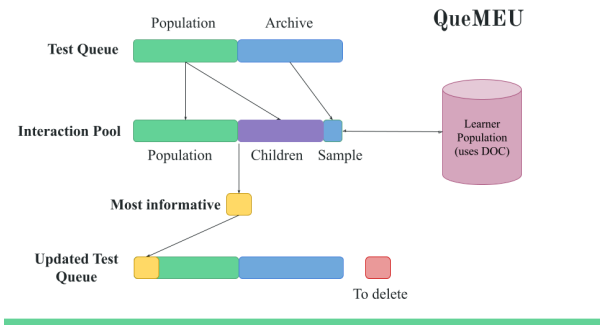

Figure 1: Visual overview of one generation for the QueMEU test generator variant used in the Experiments.

Experiments

Repository

Data from the experiments as well as a link to the code repository can be found at `https://zenodo.org/records/11626077`.

Configuration

In all experiments, we use DOC for the learner population as described in (Liskowski and Krawiec, 2017). The population size is 100 and the number of children selected is also 100. To select the number of clusters, we choose $k \in \{2, 3, 4, 5\}$, picking the clustering with the highest quality according to the *silhouttes metric* of Rousseeuw (1987). Following interaction with tests, we use NSGA-II on the derived matrix G', sorting the combined population of 200 learners by rank and crowding distance. We perform 50% truncation on the sorted learners to obtain the new population of size 100, before selecting 100 parents using tournament selection with tournament size 5 to generate children.

For simplicity, all experiments use mutation without recombination. We perform 60 trials for each configuration. Liskowski and Krawiec (2017) also describe *DOC-BIN* and *DOC-AVG*, two enhancements to DOC that improve performance on some problems. However, this increase appears dependent on the domain, and so for simplicity we use the base DOC algorithm.

For the test generator, we consider three alternatives:

1. **Standard**: The test population is generated using the sum of distinctions, equivalent to $d_S(t)$, paired with the $(\mu+\lambda)$ evolution strategy using $\mu = 100$ and $\lambda = 100$. This follows Liskowski and Krawiec (2017) and serves as our control.

2. **Advanced**: The test population is generated using the linear combination of shared fitness and shared distinctions, equivalent to $f_S^{ADV}(t)$, with $\alpha = 3.0$ and $\beta = 1.0$. This follows de Jong and Bucci (2006) and is a more established alternative. We use this to test the efficacy of fitness sharing in countering coevolutionary pathologies for DOC.

3. **QueMEU**: The test population is generated using Algorithm 1 following the description above. We use $n_{pop} = 100$ for the size of the population and $n_{arc} = 10$ for generating the interaction pool. Thus, on a typical generation the interaction pool comprises 100 population members, 90 of their children, and 10 random samples from the archive. We select $n_{win} = 2$ using the top performers in the pool according to shared distinctions $d_S^{IFS}(t)$. The maximum archive size is 1,000, which is effectively unbounded given the number of generations per experiment. Future work could investigate the effects of different parameter values on algorithm performance.

Statistical Methods

To assess statistical significance, we use Kruskal-Wallis tests followed by post-hoc Wilcoxon tests. A Bonferroni correction is used to correct for multiple comparisons. Glass's Δ is used to measure effect size; while context-dependent, a value of 0.2 is generally thought to be low, while a value of 0.8 is generally considered high. As we wish to see if the base DOC algorithm can be improved, the Standard configuration is used as the control, with Advanced and QueMEU as the experimental conditions. We use a significance level of 0.05 for all tests.

Numbers Game

Our first experiments employ the *numbers game* (NG) variants *Compare-on-All* (COA) and *Compare-on-One* (COO). These abstract benchmark problems are designed to probe a CEL algorithm's resilience to coevolutionary pathologies (Watson and Pollack, 2002; de Jong and Pollack, 2004). NG learners and tests both are represented as j-dimensional real-valued vectors. Interactions are defined as comparisons between the two vectors.

The COA game compares all dimensions of a learner s and a test t. If $s_i \geq t_i$ for all i in $1 \ldots j$, the learner passes the test:

$$g_{\text{all}}(s,t) = \begin{cases} 1 & \text{if } \forall i = 1, \ldots, j : s_i \geq t_i, \\ 0 & \text{otherwise.} \end{cases}$$

The COO game compares s and t only on the dimension in which t has the highest value. This is defined as $k = \arg\max_{i=1,\ldots j} t_i$. COO is considered to be more difficult, as it strongly incentivizes overspecialization on a subset of underlying dimensions:

$$g_{\text{one}}(s,t) = \begin{cases} 1 & \text{if } s_k \geq t_k \\ 0 & \text{otherwise.} \end{cases}$$

We use $j = 5$ dimensions for both games and pick two dimensions at random for mutation. We note that Liskowski and Krawiec (2017) use an "easier" NG variant by adding random noise sampled uniformly from $[-0.1, 0.1]$ to the test vectors. We use a "harder" variant which is more common in the coevolutionary literature, sampling from $[-1.5, 1.0]$ instead; the negatively biased mutation rate mimics the higher likelihood of a mutation being deleterious in real-world domains (de Jong and Pollack, 2004). We measure performance as the value of the *minimum dimension* of the learner with the highest fitness in G, as the goal of the NG is to increase learner values across all dimensions.

Results The results for COA are shown in Figure 2. All generators manage to increase their minimium dimension value over time. We find that on the final generation, the Advanced configuration significantly outperforms the Standard configuration with moderate effect size ($p = 0.0054$, Wilcoxon test; Glass's $\Delta = 0.631$). However, QueMEU also significantly outperforms Standard on the final generation with substantially greater effect size ($p < 0.0001$, Wilcoxon test; Glass's $\Delta = 2.698$). We note that the steady, low-variance performance increase exhibited by the QueMEU condition resembles that of stricter archive methods (de Jong, 2007). This provides some evidence that QueMEU maintains a steadier gradient for the learner population and enables consistent improvement over time.

Figure 3 gives results for COO. We note that here the Advanced configuration does not differ significantly from the Standard configuration ($p = 0.2093$, Wilcoxon test; Glass's $\Delta = 0.9527$). Meanwhile, QueMEU provides significantly greater performance than Standard with very high effect size ($p < 0.0001$, Wilcoxon test; Glass's $\Delta = 99.1142$). We observe that the initial rise in minimum dimension value for Standard and Advanced, followed by decrease towards the origin, exhibits classic symptoms of overspecialization. As in the COA experiment, QueMEU improves steadily over time, showing that the method overcomes the overspecialization pathology induced by COO.

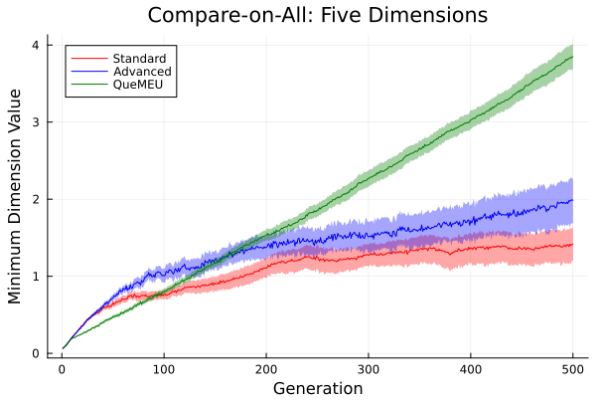

Figure 2: Minimum dimension value of the learner with highest fitness, averaged over 60 trials. Shaded regions reflect bootstrapped 95% confidence intervals.

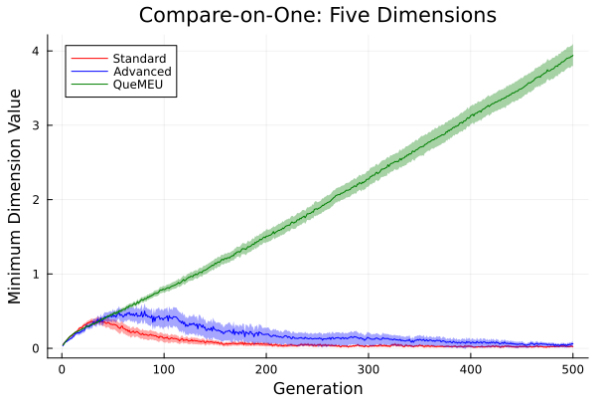

Figure 3: Minimum dimension value of the learner with highest fitness, averaged over 60 trials. Shaded regions reflect bootstrapped 95% confidence intervals.

Density Classification Task

The NG results are promising and give some justification to the theoretical motivations behind QueMEU. However, it has been argued that performance on NG domains is not necessarily predictive of performance on more complex tasks with higher problem dimension (Jaśkowski and Krawiec, 2011). To test QueMEU further, we use another classic coevolutionary benchmark problem: the density classification

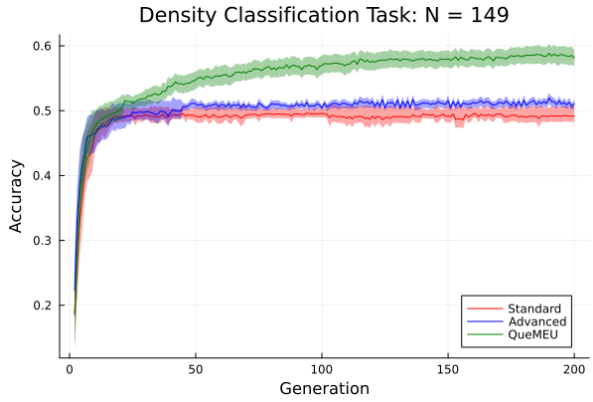

Figure 4: Classification accuracy of the learner with the highest fitness for the DCT with $N = 149$, averaged over 60 trials. Shaded regions reflect bootstrapped 95% confidence intervals.

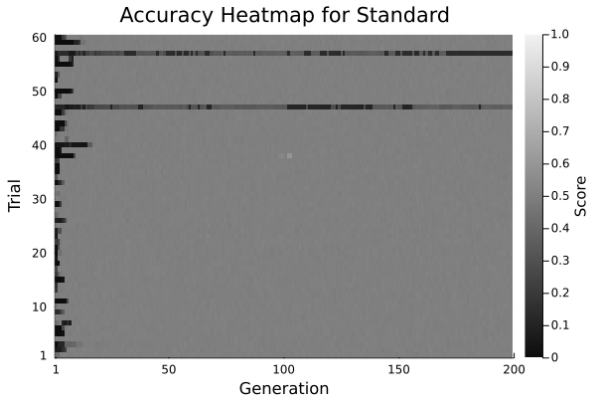

Figure 5: Heatmap for the Standard configuration demonstrating scores over 10k unbiased IC samples over time among trials on the $N = 149$ DCT. Darker regions represent zero scores.

task (DCT) with $N = 149$ (Pagie and Mitchell, 2002). This task is also performed in Liskowski and Krawiec (2017), which reports an average accuracy over sixty trials for DOC of **51.45%**, as well as an average accuracy of **53.26%** for the advanced DOC-AVG method. Thus, our results may be compared with those reported in the previous work in terms of performance.

The DCT represents learners as *elementary cellular automata* (ECA) rules and tests as binary string *initial conditions* (ICs). It is known that initializing ICs uniformly leads to tests that are too difficult for early rules to classify. We follow Liskowski and Krawiec (2017) in initializing ECA rules uniformly at random; IC tests are initialized by sampling a number d from $\{0, \ldots, N\}$, where d is the number of zeros in the IC, and then shuffling the resulting string. A classification score is obtained by allowing the IC to evolve for 320 timesteps. If the CA relaxes to a state of all ones and the *density* value (percentage of ones) $\rho \geq 0.5$, or if it relaxes to all zeros and $\rho < 0.5$, the learner passes the test. If the CA makes an incorrect prediction or does not relax to all ones or zeros, the learner fails the test. We mutate learner ECAs with a per-bit mutation rate of 2% and test ICs with a per-bit mutation rate of 5%.

Results To estimate MEU, quality scores are obtained by identifying the learner ECA in the population with the highest fitness and averaging its classification accuracy over $10,000$ unbiased IC samples as in Liskowski and Krawiec (2017). Performance over time is plotted in Figure 4. We find that the Advanced method exhibits moderate effect size, but it does not statistically significantly outperform Standard following application of the Bonferroni correction ($p = 0.0662$, Wilcoxon test; Glass's $\Delta = 0.3809$). QueMEU is again found to significantly outperform Standard with strong effect ($p < 0.0001$, Wilcoxon test; Glass's $\Delta = 1.7581$).

Despite the result of statistical nonsignificance, we note that the performance curve of Advanced appears to visually exceed Standard, reaching a somewhat higher plateau. To investigate this and better understand the learning dynamics of CEL heuristics on the DCT, we generate performance heatmaps for each condition. For interpreting the figures, the x-axis corresponds to the generation, and the y-axis to one of the sixty trials. Each cell corresponds to the classification accuracy of the learner for that trial and generation with the highest fitness in the population over 10,000 unbiased ICs. An accuracy of zero is shaded black, an accuracy of 50% is grey, while 100% accuracy would be shaded white. We then can track learner progression over all trials, expecting accuracy to begin at 0%, denoted by black cells at the left of the heatmap. As the learner population improves, we expect to see a gradual shift from black to lighter shades of grey.

Examining Figure 5, we see that the Standard generator appears unable to advance beyond the *default* strategies of emitting all 1s or all 0s no matter the IC, guaranteeing a performance of roughly 50% (Pagie and Mitchell, 2002). The blocks of light grey in Figure 6 indicate that the Advanced generator indeed performs better, and that the additional diversity permits the occasional discovery *block-expanding strategies* (Pagie and Mitchell, 2002). However, the scattered shading is indicative of either cycling or coexistence of suboptimal default strategies within the population. Figure 7 demonstrates the greater reliablity and stability of QueMEU in achieving block-expanding strategies, as indicated by the preponderance of light grey stripes. We note that at the final generation, our average QueMEU accuracy of **58.12%** exceeds the best reported average accuracy of **53.26%** found using DOC-AVG in Liskowski and Krawiec (2017) under similar conditions.

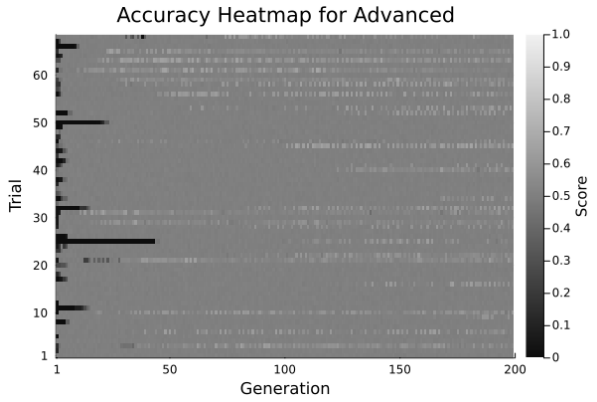

Figure 6: Heatmap for the Advanced configuration demonstrating scores over 10k unbiased IC samples over time among trials on the $N = 149$ DCT. Darker regions represent zero scores.

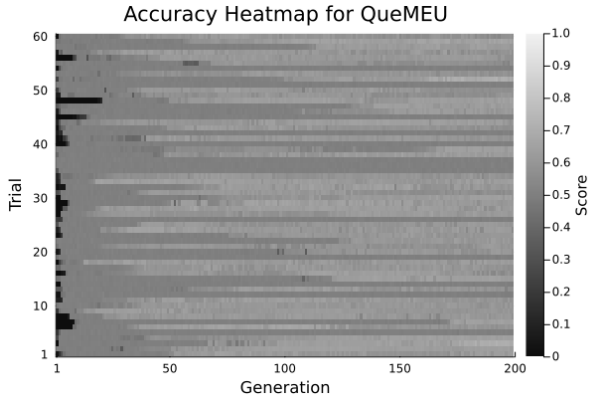

Figure 7: Heatmap for the QueMEU configuration demonstrating scores over 10k unbiased IC samples over time among trials on the $N = 149$ DCT. Darker regions represent zero scores.

Conclusion

The DOC algorithm has shown some promise in addressing known issues in coevolutionary learning. However, the standard version of the algorithm presented in Liskowski and Krawiec (2017) is found vulnerable to the coevolutionary pathology of overspecialization. An alternative test generator, QueMEU, is proposed with the intention of capturing some of the stabilizing properties of coevolutionary archive methods without their additional evaluation overhead.

It is shown that standard DOC falls prey to coevolutionary pathologies on two NG tasks, while QueMEU exhibits steady growth in performance. The practical capabilities of QueMEU are tested on the DCT task, a challenging coevolutionary benchmark. QueMEU again outperforms the standard method as well as the established Advanced generator on the DCT.

While these are encouraging results, more experiments are needed to understand the strengths and limitations of QueMEU and how it performs across different domains. Alternative selection or sampling schemes could yield superior results. More advanced follow-up work could include the incorporation of reinforcment learning or metaevolution to update the sampling and queue maintenance policy. DOC could be compared with other learner generators, and modifications tested for improved performance. It would be interesting to compare QueMEU directly against a strict archive method such as MaxSolve, which also incorporates the MEU solution concept (de Jong, 2005). It should also be compared with variants of the P-PHC family of heuristics (Bari et al., 2018).

Nonetheless, QueMEU in combination with DOC demonstrates strong performance across a challenging set of coevolutionary benchmarks, showing its potential worth as a tool for coevolutionary learning.

References

Alcaraz-Herrera, H. and Cartlidge, J. (2023). Using coevolution and substitution of the fittest for health and well-being recommender systems. *SN Computer Science*, 4(3).

Arulkumaran, K., Cully, A., and Togelius, J. (2019). Alphastar: an evolutionary computation perspective. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*.

Bari, A. T. M. G., Gaspar, A., Wiegand, R. P., and Bucci, A. (2018). Selection methods to relax strict acceptance condition in test-based coevolution. *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, 1:3–52.

Brant, J. C. and Stanley, K. O. (2017). Minimal criterion coevolution: a new approach to open-ended search. *Proceedings of the Genetic and Evolutionary Computation Conference*.

Bucci, A. (2007). Emergent geometric organization and informative dimensions in coevolutionary algorithms.

Bucci, A. and Pollack, J. B. (2003). Focusing versus intransitivity geometrical aspects of co-evolution.

Cliff, D. and Miller, G. F. (1995). Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *European Conference on Artificial Life*.

de Jong, E. D. (2004). Towards a bounded pareto-coevolution archive. *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, 2:2341–2348 Vol.2.

de Jong, E. D. (2005). The maxsolve algorithm for coevolution. In *Annual Conference on Genetic and Evolutionary Computation*.

de Jong, E. D. (2007). A monotonic archive for pareto-coevolution. *Evolutionary computation*, 15 1:61–93.

de Jong, E. D. and Bucci, A. (2006). Deca: dimension extracting coevolutionary algorithm. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*.

de Jong, E. D. and Pollack, J. B. (2004). Ideal evaluation from coevolution. *Evolutionary Computation*, 12:159–192.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Elfeky, E. Z., Elsayed, S. M., Marsh, L., Essam, D. L., Cochrane, M., Sims, B., and Sarker, R. A. (2021). A systematic review of coevolution in real-time strategy games. *IEEE Access*, 9:136647–136665.

Ficici, S. G. (2004). Solution concepts in coevolutionary algorithms.

Ficici, S. G. (2005). Monotonic solution concepts in coevolution. In *Annual Conference on Genetic and Evolutionary Computation*.

Ficici, S. G. and Pollack, J. B. (1998). Challenges in coevolutionary learning: arms-race dynamics, open-endedness, and medicocre stable states.

Ficici, S. G. and Pollack, J. B. (2001). Pareto optimality in coevolutionary learning. In *European Conference on Artificial Life*.

Ficici, S. G. and Pollack, J. B. (2003). A game-theoretic memory mechanism for coevolution. In *Annual Conference on Genetic and Evolutionary Computation*.

Fogel, D. B. (2001). Blondie24: Playing at the edge of ai.

Garcia, D., Lugo, A. E., Hemberg, E., and Reilly, U.-M. (2017). Investigating coevolutionary archive based genetic algorithms on cyber defense networks. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*.

Hillis, W. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1):228–234.

Jaśkowski, W. and Krawiec, K. (2011). How many dimensions in co-optimization. *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*.

Juille, H. (1998). Coevolving the ideal trainer : Application to the discovery of cellular automata rules. *Proc. Third Annual Genetic Programming Conference, 1998*, pages 519–527.

Juillé, H. and Pollack, J. B. (1996). Co-evolving intertwined spirals. In *Evolutionary Programming*.

Krawiec, K., Jaśkowski, W., and Szubert, M. G. (2011). Evolving small-board go players using coevolutionary temporal difference learning with archives. In *International Journal of Applied Mathematics and Computer Sciences*.

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19:189–223.

Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. *Artificial Life*, pages 295–312.

Liskowski, P. and Krawiec, K. (2017). Online discovery of search objectives for test-based problems. *Evolutionary Computation*, 25:375–406.

Monroy, G. A., Stanley, K. O., and Miikkulainen, R. (2006). Coevolution of neural networks using a layered pareto archive. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*.

Moran, N. and Pollack, J. B. (2019). Evolving complexity in prediction games. *Artificial Life*, 25:74–91.

Noble, J. and Watson, R. A. (2001). Pareto coevolution: using performance against coevolved opponents in a game as dimensions for pareto selection.

Pagie, L. and Mitchell, M. (2002). A comparison of evolutionary and coevolutionary search. *Int. J. Comput. Intell. Appl.*, pages 53–69.

Pollack, J., Blair, A., and Land, M. (1970). Coevolution of a backgammon player.

Popovici, E. (2017). Bridging supervised learning and test-based co-optimization. *Journal of Machine Learning Research*, 18(38):1–39.

Popovici, E., Bucci, A., Wiegand, R. P., and de Jong, E. D. (2012). Coevolutionary principles. In *Handbook of Natural Computing*.

Popovici, E. and Jong, K. A. D. (2009). Monotonicity versus performance in co-optimization. In *Foundations of Genetic Algorithms*.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018). Regularized evolution for image classifier architecture search. *ArXiv*, abs/1802.01548.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Service, T. C. and Tauritz, D. R. (2008). Co-optimization algorithms. In *Annual Conference on Genetic and Evolutionary Computation*.

Sims, K. (1994). Evolving virtual creatures. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*.

Smith, R. E., Forrest, S., and Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1:127–149.

Stanley, K. O. (2019). Why open-endedness matters. *Artificial Life*, 25:232–235.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019). Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *ArXiv*, abs/1901.01753.

Watson, R. and Pollack, J. (2002). Coevolutionary dynamics in a minimal substrate. *Morgan Kaufmann*.

Wiegand, R. P., Bucci, A., Kumar, A. N., Albert, J. L., and Gaspar, A. (2022). Identifying informatively easy and informatively hard concepts. *ACM Trans. Comput. Educ.*, 22:7:1–7:28.

Willkens, T. and Pollack, J. (2023). Modes analysis of prediction games. *The 2023 Conference on Artificial Life*.

Willkens, T. and Pollack, J. B. (2022). Evolving unbounded neural complexity in pursuit-evasion games. *The 2022 Conference on Artificial Life*.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.*, 1:67–82.

Wolpert, D. H. and Macready, W. G. (2005). Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation*, 9:721–735.