Accelerating Coevolution with Adaptive Matrix Factorization

Paweł Liskowski
Poznan University of Technology
Poznan, Poland
pliskowski@cs.put.poznan.pl

Wojciech Jaśkowski
IDSIA Dalle Molle Institute for Artificial Intelligence Research
Manno TI, Switzerland
wojciech@idsia.ch

ABSTRACT

Among many interaction schemes in coevolutionary settings for interactive domains, the round-robin tournament provides the most precise evaluation of candidate solutions at the expense of computational effort. In order to improve the coevolutionary learning speed, we propose an interaction scheme that computes only a fraction of interactions outcomes between the pairs of coevolving individuals. The missing outcomes in the interaction matrix are predicted using matrix factorization. The algorithm adaptively decides how much of the interaction matrix to compute based on the learning speed statistics. We evaluate our method in the context of coevolutionary covariance matrix adaptation strategy (CoCMAES) for the problem of learning position evaluation in the game of Othello. We show that our adaptive interaction scheme allows to match the state-of-the-art results obtained by the standard round-robin CoCMAES while, at the same time, considerably improves the learning speed.

CCS CONCEPTS

•Theory of computation → Evolutionary algorithms; •Computing methodologies → Non-negative matrix factorization;

KEYWORDS

CMA-ES; Nonnegative Matrix Factorization; Machine Learning

ACM Reference format:
Paweł Liskowski and Wojciech Jaśkowski. 2017. Accelerating Coevolution with Adaptive Matrix Factorization. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071320

1 INTRODUCTION

In some search and optimization problems, the objective function is infeasible to compute since it would require evaluating a candidate solution on a huge (or even infinite) set of tests. Such test-based problems [10] often involve interactive domains, in which there exists a natural, and often adversarial interaction between entities. A single *interaction* between a candidate solution and a test produces a scalar outcome that reflects the capability of the former to *pass* the latter (expressed in the simplest case as a binary value). Canonical examples of such domains include searching for sorting programs that are supposed to sort integer sequences, evolving genetic programming trees that are evaluated on a number of tests, or learning game playing agents that play with each other.

Due to the interactive nature of test-based problems, coevolutionary algorithms constitute a natural way of solving them. Coevolutionary algorithms differ from the evolutionary ones in the way the fitness in the evolving population is computed. In contrast to the evolutionary algorithms, the coevolutionary ones do not have access to the objective evaluation function and, in order to compute the fitness, they need to employ a surrogate function instead. The surrogate function is most often in the form of an interaction scheme, which involves the evolving individuals.

Many interactions schemes for coevolution have been proposed in the past. These methods include k-random opponents, single-elimination tournament [31], or fitnessless selection [19]. However, it is the round-robin tournament that provides the most precise evaluation of candidate solutions. The major drawback of the round-robin tournament is its computational cost resulting from the number of interactions that have to be performed in order to evaluate an individual.

In this paper, we propose a novel interaction scheme that computes only some of the interactions between the coevolving individuals and predicts the rest using non-negative matrix factorization. Crucially, the proposed approach adaptively decides how many interactions to compute based on the learning speed statistics. In the experimental part of the paper, we evaluate our method in the context of coevolutionary covariance matrix adaptation strategy (CoCMAES) for the problem of learning the position evaluation function in the game of Othello. We show that our adaptive interaction scheme allows to match the state-of-the-art results obtained by the standard round-robin CoCMAES and considerably improves the learning speed.

2 BACKGROUND

2.1 Test-based Problems

A test-based problem (a.k.a. an interactive domain) involves a set of candidate solutions S, a set of tests T, and an interaction function $g : S \times T \to \mathbb{R}$ that defines how well a given solution solves a given test. The solution to the problem is defined by a solution concept [8]. In the most popular case, it is the maximization of expected utility, i.e.:

$$s^* = \arg\max_{s \in S} \mathbb{E}_{t \in T}[g(s, t)].$$

In this paper, we limit our attention to symmetric domains such as two player games, in which $S = T$. Note, however, that our method can be easily generalized to the $S \neq T$ case.

GECCO '17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071320

Algorithm 1 A general outline of a single-population coevolutionary algorithm

Require: population size λ

1: $P \leftarrow$ Initialize(λ)
2: **repeat**
3: $F \leftarrow ComputeFitness(P)$ ▷ Interaction scheme
4: $P \leftarrow$ UpdatePopulation(F) ▷ Evolutionary operators
5: **until** Termination Condition

2.2 Coevolutionary Algorithms

The general scheme of the single-population coevolutionary algorithm for symmetric domains is shown in Algorithm 1. The algorithm maintains a population P of candidate solutions. In each step, it begins by computing the fitness of each candidate solution using a given interaction scheme. Then, the population is updated by any evolutionary algorithm according to the computed fitness values.

In the round-robin tournament interaction scheme [1, 33], each candidate solution interacts with every other member of the population, and the outcomes of these interactions are stored in an *interaction matrix* G. The fitness of an individual $s \in P$ is then determined as

$$f(s) = \frac{1}{|P|} \sum_{t \in P \setminus \{s\}} g(s,t),$$

which estimates the expected utility of an individual. Notice that, this estimation is biased and non-stationary since it depends on the current population P. Evaluating the fitness for the whole population requires computing $|P|^2 - |P|$ interaction outcomes.

2.3 Matrix Factorization

Our approach to speed-up coevolution involves using non-negative matrix factorization on the round-robin interactions outcomes matrix. Here we introduce this concept in more detail.

Given a non-negative $m \times n$ matrix G and the desired rank k (typically $k \ll min(m,n)$), non-negative matrix factorization (NMF) [2] searches for non-negative matrices W and H that together form a lower rank approximation of G, i.e.:

$$G \approx WH \;\; s.t. \;\; W, H \geq 0, \tag{1}$$

where $W \in \mathbb{R}^{m \times k}$ is called *weights matrix* and $H \in \mathbb{R}^{k \times n}$ is a *feature matrix*.

In general case, G is in an interaction matrix between m candidate solutions in S and n tests in T, and each candidate solution $s \in S$ is associated with a row in W (a vector $w_s \in \mathbb{R}^k$), and each test $t \in T$ corresponds to a column in H (a vector $h_t \in \mathbb{R}^k$). In the specific case considered in this paper, G is square $m \times m$ matrix, and both W and H describe the same entities.

In order to solve the NMF problem, equation (1) is commonly reformulated as the following optimization problem:

$$\min_{W,H} f(W,H) \equiv \frac{1}{2}||G - WH||_F^2 \;\; s.t. \;\; W, H \geq 0, \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm. In the simplest scenario, NMF model is trained by fitting to the observed interaction outcomes in G.

As it follows from (1), an estimate of an interaction outcome of a candidate solution s with a test t can be found by calculating the dot product of two vectors corresponding to s and t:

$$\hat{g}_{pt} = w_p^T h_t = \sum_{j=1}^{k} w_{pj} h_{jt}. \tag{3}$$

In practice, the optimization problem given by Eq. 2 is often enhanced with a regularization term that forces the factors to be *sparse*. Sparsity can be easily enforced by adding a penalty term, such as a L1 and/or L2-norm penalty:

$$\min_{W,H} f(W,H) \equiv \frac{1}{2}||G - WH||_F^2 + \lambda(||W||_F^2 + ||H||_F^2), \tag{4}$$

Sparse representations are advantageous as they tend to encode the data using just a few active components that allow for easier interpretation of the factors. Sparseness in both W and H is crucial to learn parts-based and intuitive features of data [23].

Perhaps the simplest way to minimize expression (4) is stochastic gradient descent that employs the following update rules:

$$w'_{sj} = w_{sj} + \alpha \frac{\partial}{\partial w_{sj}} e_{st}^2 = p_{sj} + \gamma(2e_{st}h_{jt} - \lambda w_{sj}) \tag{5}$$

$$h'_{jt} = h_{jt} + \alpha \frac{\partial}{\partial q_{jt}} e_{st}^2 = q_{jt} + \gamma(2e_{st}w_{sj} - \lambda h_{jt}), \tag{6}$$

where $j = 1, \ldots, k$, γ is the learning rate, and $e_{st}^2 = (g_{st} - \hat{g}_{st})^2 = (g_{st} - w_s^T h_t)^2$ is the error between the known and predicted outcome of interaction for given s and t. The new values of W and H are found in each iteration by multiplying the current one by a factor that depends on the quality of approximation in (1). The update rules are applied for a fixed number of iterations, or until the error given by the left-hand side of (4) is sufficiently small.

What truly makes MF a powerful tool in machine learning is the ability to factorize G even when some of its elements are missing, i.e., when G is *sparse*. This property of MF has been extensively used in recommender systems to fill in the gaps in a large matrices (of, e.g., users' recommendations [21]) based only on small fraction of known elements. In such a scenario, the objective of NMF is to minimize the regularized squared error on known interaction outcomes. To learn the factors W and H, the update rules given by (5) and (6) are applied only to the elements of G that are not missing. The missing outcomes in G are then modeled as dot products between the corresponding vectors in W and H (Eq. 3).

Much of the appeal of NMF comes from its ability to extract *underlying features* of G as basis vectors in W. The non-negativity constraint helps the model to learn parts-like representations by additively combining features that attempt to 'reproduce' the original input. NMF became a popular tool in pattern recognition or classification, where it shows its strengths in learning meaningful features from real-life datasets such as collections of face images or text documents [23, 32]. In the following, we employ it as the core component of our algorithm.

3 METHODS

In this section, we describe the main contribution of this paper, namely the adaptive matrix factorization-based interaction scheme for coevolution. We begin by introducing its simpler variant, and later extend it to automatically adjust its parameters during learning.

3.1 Constant-α Matrix Factorization-based Interaction Scheme

Based on the observations made in previous sections, we propose MFIS, a Matrix Factorization-based Interaction Scheme that employs matrix factorization to speed-up coevolution by computing only a fraction of interactions $\alpha \in (0, 1]$ between individuals in the population instead of the full round-robin matrix. The method exploits the ability of NMF to model *any* element of G as the inner product in the joint latent space of factors (Eq. 3) in order to predict interaction outcomes that were not computed and fill the missing values in G. The proposed coevolutionary MF-based interaction scheme calculates the *sparse* interaction matrix G between the candidate solutions from the current population P in the following way (see Algorithm 2):

(1) For each candidate solution $s \in P$, draw a nonempty random subset of opponents $T_s \subset P \backslash \{s_i\}$ of size $\lfloor \alpha |P| \rfloor$ to interact with.
   (a) Perform the interactions between s and $t \in T_s$, updating the appropriate cells in the interaction the interaction matrix $G = (g_{ij})$.
   (b) Fill in the remaining entries in G with zeros, treating them as unknowns.
(2) Factorize G in non-negative components W and H (see Section 2.3).
(3) Use the matrices W and H to reconstruct the interaction outcomes in G by calculating $\hat{G} = (\hat{g}_{ij}) = WH$.
(4) Compute the fitness of each candidate solution $s \in P$ as $f(s_i) = \sum_{j=1}^{n} g_{ij}$, by substituting the *missing* g_{ij}s with their estimates $\hat{g}_{ij}$s.

The predictions made by the method are based on the behavioral similarity between individuals in the population P. The similarity of two individuals is computed based on the similarity of their interaction outcome vectors. The evaluation in the MF-based interaction scheme is, therefore, *contextual*: prediction $\hat{g}_{ij}$ made for a missing outcome depends not only on corresponding individuals s_i and s_j but also on other individuals in P. All available outcomes of interactions together determine the factorization model and influence how the predictions for missing outcomes are made. To account for the changes in P stemming from the evolution of candidate solutions, we perform independent NMF in each generation.

3.2 Adaptive Matrix Factorization-based Interaction Scheme

The most important parameter of MFIS is α that controls the fraction of interactions to be computed. It trades-off the computation performance and the fitness evaluation precision. Here we propose an Adaptive MFIS (AMFIS) to automatically adapt α based on the current learning performance statistics.

As the progress made by CMA-ES stalls, α is automatically increased to improve the evaluation accuracy and provide the search

Algorithm 2 Matrix Factorization-based Interaction Scheme (MFIS) with constant $\alpha \in (0, 1]$ which trades-off fitness evaluation precision and computational performance.

Require: factorization rank k.

```
function ComputeFitness(P, α)
    for s ∈ P do
        T_s ← Sample(T, α)
        for t ∈ T_s do
            G_ij ← interact(s, t)
    W, H ← NMF(G, k)
    Ĝ ← WH                                  ▷ predicts missing G_ij s
    G ← input-missing(Ĝ, G)
    for s_i ∈ P do
        F(s_i) ← (1/|P|) Σ_{j=1}^{n} g_ij
    return F
end function
```

Algorithm 3 Adaptation Matrix Factorization-based Interaction Scheme (AMFIS).

Require: window size w, minimum observations φ, step size γ, history of learning speed H, exploration rate ϵ, exploration step γ

```
level ← 1                                   ▷ A global variable

function ComputeFitnessAdaptive(P)
    curr ← level
    if Rand(0, 1) < ε then
        curr ← testlevel + 1
    α ← curr × γ
    F ← ComputeFitness(P, α)
    UpdateLevel(curr)
    return F
end function

function UpdateLevel(l)
    learn_speed = Δp/Δe
    H[curr] ← Concat(H[curr], learn_speed)
    V_level ← w most recent values from H[level]
    V_level+1 ← w most recent values from H[level + 1]
    if len(V_level) < φ or len(V_level+1) < φ then
        return
    if mean(V_level) < mean(V_level+1) then      ▷ switch condition
        if α_i < 1 then
            level ← level + 1
            H[level] ← ∅
end function
```

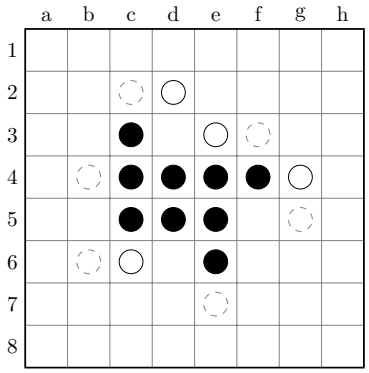

Figure 1: An exemplary Othello board position. White is the player to play and it has 6 possible actions (dashed gray circles). Playing c2 will make c3, c4, and c5 white.

process with richer and more detailed fitness. The α is discretized into $1/\gamma$ *levels*. We start with *level*=1 (corresponding to $\alpha = \gamma = 0.1$). The adaptation mechanism is inspired by the algorithms solving multi-armed bandit problems: with probability $1 - \epsilon = 0.9$, we use the *current* level and with probability $\epsilon = 0.1$, we perform *exploration* for the next level. For each level, we track the change of the learning speed defined as:

$$learn\_speed = \frac{\Delta p}{\Delta e},$$

where Δp is the the most recent increase in the objective performance of the best individual from the current generation and Δe denotes the computational effort (the number of interactions) that caused this increase. By observing the learning speeds for two consecutive levels, the algorithm eventually makes a decision to switch from *level* to *level* + 1. Such a switch is permanent and increases the fitness precision at the expense of the number of interactions to compute. Note also that, in order to make the decision on the most current data, we take into account only w most recent learning speeds. See also Algorithm 3 for more details.

The decision to make a switch is by default based on the comparison of learning speed averages (line 19 in Algorithm 3). We also consider a variant, in which we use the Student's t-test to judge whether it is time to increase the *level*. We refer to this variant of the proposed approach as AMFIS-TT.

4 POSITION EVALUATION FOR OTHELLO

In this section, we describe the problem of learning position evaluation functions for Othello, which will be later used in the experimental section. We encode our position evaluation functions using systematic n-tuple networks, which are also introduced in the section. In this settings, we strictly follow [16], which has recently established the state of the art in this domain.

4.1 Othello

The game of Othello (a.k.a Reversi) is a two-player, deterministic, sequential, zero-sum game played on an 8×8 board with double-sided pieces with white and black face, each face assigned to one player. The players take turns by placing pieces on the board, with their colors face up, one at a time. A legal move consists of placing a piece on an empty square, which forces flipping some of the opponent's pieces. The location to place a new piece must: 1) be adjacent to at least one of the board's pieces, and 2) be one end of a vertical, horizontal, or diagonal line segment that starts in another player's piece and it contains only the opponent's pieces (which are surrounded on both sides); the segment does not contain empty spaces. All surrounded opponent's pieces are then flipped to the other color; if multiple line segments exist, flipping affects all of them. Figure 1 shows an example of an Othello position with six possible moves of the white player. The game ends when no player has a legal move (usually when there are no empty board positions left). The player that has more pieces at the end of the game is the winner. If both players have the same number of pieces, the game ends with a draw.

Othello is an asymmetrical game. Thus, in this study, for convenience, we consider *double game*s, in which each of the two players plays two games in a row: one game as black and the second one as white.

4.2 Position Evaluation with n-Tuple Networks

Due to a huge number of possible states in Othello, the position evaluation function has to be approximated. A powerful and computationally efficient function approximators are *n-tuple networks*, which were originally proposed by Bledsoe and Browning [3] for optical character recognition.

An n-tuple network consists of m tuples of different board locations. For a given board state $\mathbf{b}$, the network outputs the sum of values returned by the individual tuples. The ith tuple, where $i = 1, \ldots, m$, contains a sequence of n_i board locations $(loc_{ij})_{j=1,\ldots,n_i}$, and an associated look-up table LUT_i. The table contains weights for each possible pattern for the sequence of locations in the tuple. The result of an n-tuple network can be thus interpreted as a position evaluation function p:

$$\begin{aligned} p(\mathbf{b}) = \sum_{i=1}^{m} p_i(\mathbf{b}) &= \sum_{i=1}^{m} \mathrm{LUT}_i \left[idx\left(\mathbf{b}_{loc_{i1}}, \ldots, \mathbf{b}_{loc_{in_i}}\right)\right] \\ idx\left(\mathbf{v}\right) &= \sum_{j=1}^{|\mathbf{v}|} v_j c^{j-1}, \end{aligned}$$

where $\mathbf{b}_{loc_{ij}}$ is a board value at location loc_{ij}, $\mathbf{v}$ is a sequence of board values ($0 \leq v_k < c$, for $k = 1, \ldots, |\mathbf{v}|$), and c denotes the number of possible board values ($c = 3$ for Othello). As a result, one look-up table contains 3^{n_i} weights.

To improve the effectiveness of n-tuple networks, we also exploit the inherent symmetries of a game board [28] in the method called symmetric sampling. In symmetric sampling, a single tuple is employed 8 times, returning one value for each possible board rotation and reflection (see Fig. 3).

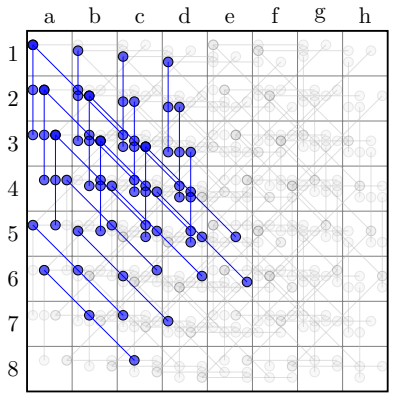

Figure 2: All 24 straight 3-tuples (648 weights). Their symmetric expansions have been shown in light gray.

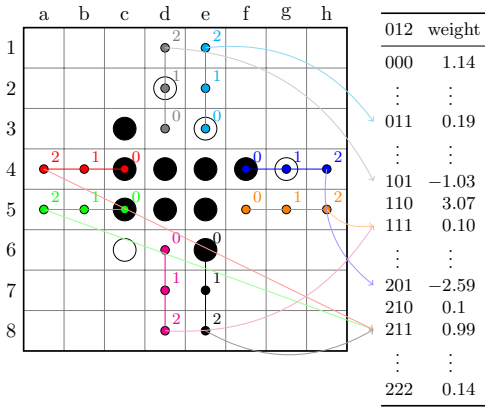

Figure 3: A straight 3-tuple employed eight times for the given board position (symmetric sampling). The eight symmetric expansions of the 3-tuple return $0.19 - 1.03 + 2 \times 0.1 - 2.59 + 3 \times 0.99 = -0.26$.

4.2.1 Board Inversion of Selecting the moves. A game-playing agent selects its moves based on the position evaluation function. It selects the move that leads to a position of the highest value. Since we expect the agent to play both white and black, when playing white, before making its decision the agent first inverts the board pieces (playing as if it were the black player). This method is called board inversion and has been found more efficient [11] than its alternatives such as output negation.

4.2.2 Systematic n-Tuple Networks.

How to choose the tuples? Lucas [28] proposed to randomly generate a small number of long snake-shaped sequences. However, recently it has been found, that a large number of short systematically selected tuples lead to better results [12]. The *systematic n-tuple network* consists of all possible vertical, horizontal, and diagonal n-tuples of the same length (see Fig. 2). Its smallest representative is a network of 1-tuples. Thanks to symmetric sampling, only 10 of them are required to cover an 8×8 Othello board, and such a 10×1-tuple network contains $10 \times 3^1 = 30$ weights.

The comparison of different n-tuple architectures has been performed by Jaśkowski and Szubert [16]. The authors report that the combination of straight 4-tuples and (square) 2×2 tuples worked the best.

5 RELATED WORK

A number of different interaction schemes have been proposed for coevolution. A round-robin tournament (a.k.a complete mixing [29]) involves computing interaction outcomes for all pairs in the population, which requires $n^2 - n$ interactions. The k-random opponents method [34] lets an individual play with k opponents drawn at random from the current population (nk interactions). Angeline and Pollack proposed the single-elimination tournament [1], which requires only $n-1$ games to play but it precisely computes the fitness only for the tournament winner. Finally, fitnessless coevolution [13] uses the outcomes of the games to directly drive tournament selection. This method involves $(k-1)n$ interactions, where k is the tournament size.

The fitness computed based on reconstructing the interaction outcomes matrix by matrix factorization can be treated as a *surrogate fitness*. In evolutionary computation, a surrogate fitness function provides a computationally cheaper approximation of the original objective function. Surrogates are particularly helpful in domains where evaluation is computationally expensive, e.g., when it involves simulation. They usually rely on simplified models of the process being simulated, hence their alternative name: *surrogate models* [20]. In continuous optimization, such models are typically implemented using low-order polynomials, Gaussian processes, or artificial neural networks. Surrogates can be applied to virtually every area of evolutionary computation, including population initialization, mutation operators and fitness evaluation.

Several other studies attempted to reduce the number of evaluations in used in evolutionary algorithms (EAs) for test-based problems. An arguably simplest approach is to draw a subset of tests $T' \subset T$ and allow the candidate solution interact only with them. This approach was studied in the context of EAs, in which it is known as Random Sampling Evolutionary Learning [5]. Apart from speeding up the evolution, the motivation is that candidate solutions that perform well on various different subsets might have captured essential knowledge to generalize to all tests in T.

The idea of reconstructing interaction outcomes via factorization of sparse interaction matrix has been proposed for the first time in the context of genetic programming (GP) [27]. The proposed method, dubbed Surrogate Fitness via Factorization of Interaction Matrix (SFIMX), reduces the number of required interactions between programs and tests in GP. To this end, it factorizes of the matrix of outcomes vectors resulting from applying programs in a population to the tests that define the program synthesis task. In SFIMX, the spared evaluation cycles are spent on additional programs in extended population.

Matrix factorization has also been used as a means to 'multi-objectivize' GP. Discovery of Objectives via Factorization (DOF) proposed in [25], employs NMF to heuristically derive a low number

of search objectives from an interaction matrix, and uses these objectives to drive the search. The central observation that motivates DOF is that NMF can be used to explain the interaction outcomes in G by characterizing both programs and tests in terms of factors inferred from the patterns observed in their interactions. These factors are used to recast the GP problem as a multi- rather than a single-optimization problem. In every generation, DOF feeds the factors from W directly into NSGAII [7] selection procedure in order to select the parent programs and generate candidate solutions for the next generation.

Similarly to DOF, Discovery of Search Objectives by Clustering (DOC) also derives new search objectives that characterize the candidate solutions in P and form the basis for selecting the most promising individuals for the next generation [22]. DOC applies clustering to n columns of G that are treated as points in m-dimensional space. DOC builds upon the approach designed for coevolutionary algorithms in [24, 26].

The position evaluation in Othello has been frequently employed for evaluating both evolutionary [6, 14, 15, 30, 35, 39, 41] and temporal difference learning methods [36], and for comparing their empirical results [17, 35, 38]. The best evaluation function for Othello to date has been obtained using the Coevolutionary CMA-ES algorithm [16]. We refer to this work for a review and computational comparison of all past approaches to this problem.

In the context of Othello, n-tuple networks were employed for the first time under the name of *tabular value functions* by Buro [4] in his famous Logistello program. More recently, they were popularized by Lucas [28] and successfully applied to other games such as Connect 4 [40], the puzzle game of *2048* [37] and Tetris [18].

6 EXPERIMENTS

In the following experiments, we evaluate our matrix factorization-based interaction scheme on the problem of position evaluation in the game of Othello (see Section 4). We evolve individuals which are real-valued vectors interpreted as the weights of a systematic n-tuple network consisting of all straight 4-tuples and all square 2×2-tuples. As the optimization method, we employ Coevolutionary CMA-ES (CoCMAES) [16], which uses the covariance matrix adaptation evolutionary strategy [9], a state-of-the-art continuous black-box optimization method. The step-size σ of CMA-ES is initialized to 1. The initial starting point for CMA-ES is generated by sampling the weights uniformly from the range $[-0.1, 0.1]$. We use the population size $\lambda = 400$. All the algorithms were run 5 times.

To objectively measure the progress of coevolutionary learning algorithms, we strictly followed the protocol from [16], to which we refer for details. We employed an external performance measure consisting in playing (double) Othello games against 11 previously published position evaluation functions on 1000 opening positions. Every 10 generations, we report the average of 22000 games.

The AMFIS setups, which automatically adapt α during a run (cf. Section 3), employ the same objective performance measure for computing the learning speed but estimate it only from 100 positions to minimize the computational overhead. In this way, the extra computational effort needed by the adaptive interaction scheme amounts to about 1.3% of the total effort.

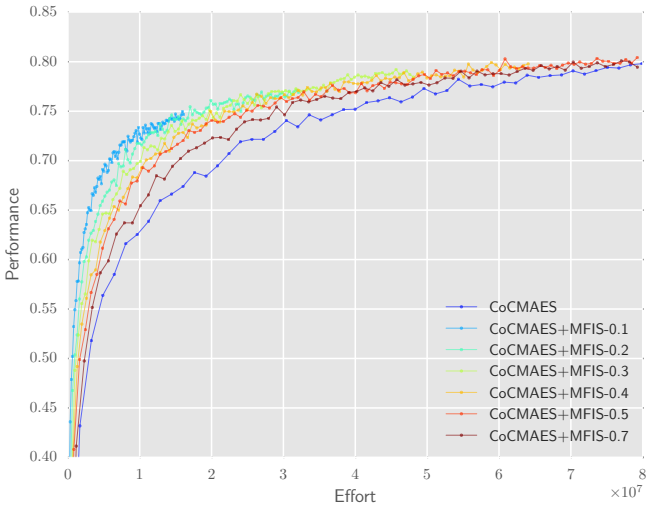

Figure 4: Average performance of the best-of-generation individuals obtained using CoCMAES and MFIS with constant α. The runs were stopped after 2000 generations.

The methods studied in this paper were implemented[1] in Java. We used the Hansen's CMA-ES implementation [2]. By using factorization rank $k = \log(\lambda) \approx 10$, the computational cost of the factorization was approximately $2 - 10\%$ of the total cost of the fitness evaluation.

6.1 CoCMAES+MFIS (constant α)

In the first experiment, we were interested in verifying whether the proposed MFIS is a viable method for accelerating the vanilla round-robin-based CoCMAES. For this aim, we control the fraction of interactions to be calculated by $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.7\}$ in Algorithm 2. Since the number of interactions is reduced by a factor of $1 - \alpha$, in each generation we spare $(1 - \alpha)|P|^2$ interactions.

Figure 4 plots the objective performance of the best-of-generation individual as a function of computational effort (the number of interactions). The results clearly demonstrate that MFIS provides significant speed-up while maintaining the overall performance. The curve that belongs to the baseline method CoCMAES is dominated by the other methods for most of the time, indicating that the same performance can be achieved much faster. For instance, CoCMAES+MFIS with $\alpha = 0.1$ achieves the performance level of 0.75 2.2 times faster than the baseline Co-CMAES. Despite the fact we stopped the run after 2000 generations, it can already be observed that it would not achieve the same performance level as the baseline method, since the higher the performance level, the more precise fitness is required.

6.2 CoCMAES+AMFIS (adaptive α)

The observations made in Section 6.1 led us to design the adaptive variant of the proposed interaction scheme that automatically adjusts α during evolution (cf. Section 3 and Algorithm 3). The method dubbed AMFIS-20 uses step size $\gamma = 0.1$, leading to 10 α

[1] https://github.com/pliskowski/cma-mf
[2] https://code.google.com/p/cma-es

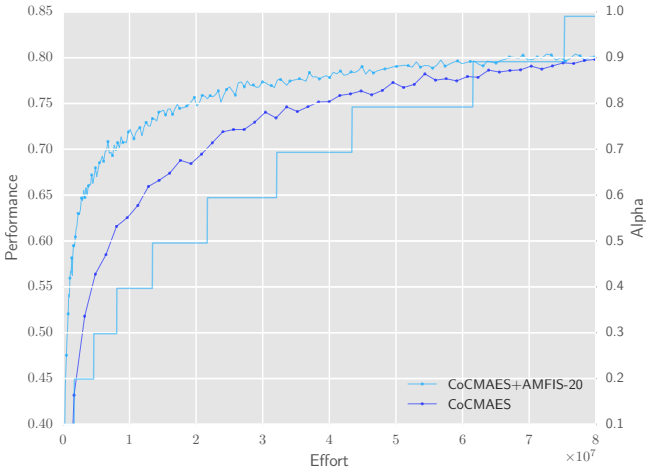

Figure 5: Average performance of the best-of-generation individuals obtained using CoCMAES and CoCMAES augmented by AMFIS-20.

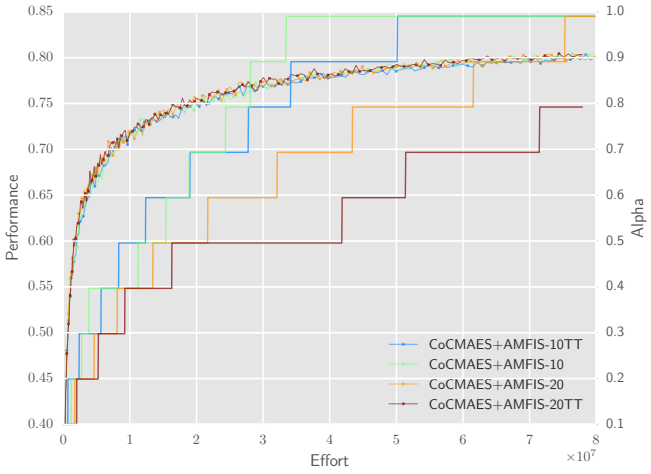

Figure 6: Average performance of the best-of-generation individuals obtained using AMFIS, window size $w \in \{10, 20\}$ and the switch conditions utilizing either 'greater than' or the Student's t-test.

levels of $\{0.1, 0.2, \ldots, 1.0\}$. We also set the window size $w = 20$ so that the mean learning speed is computed from the 20 most recent observations and $\varphi = 50$ to make sure that the algorithm gathers enough samples prior to making any decisions regarding the level of α.

In Fig. 5, we compare the objective performance of the best-of-generation individuals obtained using AMFIS-20 and the baseline CoCMAES. We also plot α to visualize how it changes during the learning. The proposed adaptive MF-based interaction scheme learns faster throughout the evolutionary run and ultimately achieves a similar level of performance as the baseline CoCMAES. This is not surprising since AMFIS converges eventually to the round-robin CoCMAES. As demonstrated by the plot, AMFIS-20 starts computing the complete interaction matrix G at the end of evolution, shortly after reaching the milestone of 7.5×10^7 interactions. From that point onward, it progresses just as a regular CoCMAES.

6.3 Other AMFIS variants

In the final experiment, we investigate the robustness of AMFIS to the condition on which the α level is increased (c.f. Algorithm 3) and the window size w. Apart from AMFIS-20 shown earlier, we consider three additional setups: AMFIS-10 that uses $w = 10$, AMFIS-10TT that uses $w = 10$ and the Student's t-test, and AMFIS-20TT employing $w = 20$ and the t-test with $p = 0.25$. By decreasing the window size, we expected the algorithm to become more susceptible to outliers and noise when evaluating the performance gains from different levels of α. Also, we expected that the statistically-sound Student's t-test should be more robust than a simple 'greater than' condition.

The results shown in Fig. 6 surprised us since both the learning speed and the final performance of the compared methods turned out to be similar. Despite this, we observe that the variants employing $w = 20$ are less eager to increase α, especially in the later stages of evolution. This suggests that the higher values of w should be in principle more economic in terms of the computational effort, allowing for potentially bigger savings during the run. A similar observation can be made for the impact of the t-test as it requires more evidence to allow the algorithm to increase α.

Overall, these results allow us to conclude that the proposed method is robust to the choice of the parameters.

7 CONCLUSIONS

In this paper, we proposed an adaptive coevolutionary interaction scheme that computes only a fraction of interactions needed for the fitness evaluation, thereby allowing to considerably improve the learning speed of coevolutionary algorithms. Our method, adaptive matrix factorization-based interaction scheme (AMFIS) uses computationally efficient non-negative matrix factorization to predict missing interaction outcomes. What is important, AMFIS can automatically adapt its main parameter α, which controls the fitness precision vs. learning speed trade-off. In the limit, it converges to the (precise) round-robin interaction scheme.

We demonstrated how the proposed interaction scheme can be applied to significantly speed-up CoCMAES, the state-of-the-art coevolutionary method, for the problem of learning position evaluation in the game of Othello. Although we limited our considerations to the symmetric single-population case, nothing precludes other designs such as extending our interaction scheme to the two-population coevolution.

The method is founded on a solid mathematical ground, it is robust to its parameters, and it is computationally efficient, which makes it particularly appealing for handling large and computationally expensive problems. In machine learning, it is common to apply NMF to matrices with tens of thousands of rows an columns [21]. This capability may come in handy for problems requiring large populations, or large numbers of tests.

This preliminary study can be extended in multiple ways. In addition to already mentioned two-population coevolution, we

would be interested to see whether it is possible to further extend the method by computing only the interaction outcomes that are difficult to estimate and predicting the others.

One weakness of the proposed method consists in using the objective performance measure in order to adapt the α parameter. This may not be always available. Thus, in the future work, we would like to drop this requirement by using a coevolutionary archive consisting of best-of-generation individuals.

ACKNOWLEDGMENTS

P. Liskowski acknowledges the support from grant 2014/15/N/ST6/04572 funded by the National Science Centre, Poland. W. Jaśkowski was supported by Ministry of Science and Higher Education grant "Mobility Plus" no 1296/MOB/IV/2015/0. The computations were performed in Poznan Supercomputing and Networking Center.

REFERENCES

[1] P. J. Angeline and J. B. Pollack. Competitive Environments Evolve Better Solutions for Complex Tasks. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 264–270, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[2] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[3] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Proc. Eastern Joint Comput. Conf.*, pages 225–232, 1959.

[4] M. Buro. Experiments with Multi-ProbCut and a new high-quality evaluation function for Othello. In H. J. van den Herik et al., editor, *Games in AI Research*, pages 77–96. Univ. Maastricht, 2000.

[5] S. Y. Chong, P. Tino, D. C. Ku, and Y. Xin. Improving Generalization Performance in Co-Evolutionary Learning. *IEEE Transactions on Evolutionary Computation*, 16(1):70–85, 2012.

[6] S. Y. Chong, P. Tino, D. C. Ku, and X. Yao. Improving Generalization Performance in Co-Evolutionary Learning. *IEEE Transactions on Evolutionary Computation*, 16(1):70–85, 2012.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[8] S. G. Ficici. *Solution concepts in coevolutionary algorithms*. PhD thesis, Brandeis University, Waltham, MA, USA, 2004. Adviser-Pollack, Jordan B.

[9] N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[10] W. Jaśkowski. *Algorithms for Test-Based Problems*. PhD thesis, Institute of Computing Science, Poznan University of Technology, Poznań, Poland, 2011. Adviser: Krzysztof Krawiec.

[11] W. Jaśkowski. Systematic n-tuple networks for othello position evaluation. *ICGA Journal*, 37(2):85–96, June 2014.

[12] W. Jaśkowski. Systematic n-tuple networks for position evaluation: Exceeding 90% in the othello league. Technical Report RA-06/2014, arXiv:1406.1509, Institute of Computing Science, Poznan University of Technology, Poznań, Poland, 2014.

[13] W. Jaśkowski, K. Krawiec, and B. Wieloch. Evolving strategy for a probabilistic game of imperfect information using genetic programming. *Genetic Programming and Evolvable Machines*, 9(4):281–294, 2008.

[14] W. Jaśkowski, P. Liskowski, M. Szubert, and K. Krawiec. Performance profile: a multi-criteria performance evaluation method for test-based problems. *International Journal of Applied Mathematics and Computer Science*, 26(1):215–229, 2016.

[15] W. Jaśkowski, P. Liskowski, M. G. Szubert, and K. Krawiec. Improving Coevolution by Random Sampling. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 1141–1148, New York, NY, USA, 2013. ACM.

[16] W. Jaśkowski and M. Szubert. Coevolutionary CMA-ES for knowledge-free learning of game position evaluation. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(4):389–401, 2016.

[17] W. Jaśkowski, M. Szubert, and P. Liskowski. Multi-criteria comparison of coevolution and temporal difference learning on othello. In A. I. Esparcia-Alcazar and A. M. Mora, editors, *EvoApplications 2014*, volume 8602 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2014.

[18] W. Jaśkowski, M. Szubert, P. Liskowski, and K. Krawiec. High-dimensional function approximation for knowledge-free reinforcement learning: a case study in SZ-Tetris. In *GECCO'15: Proceedings of the 17th annual conference on Genetic and Evolutionary Computation*, pages 567–574, Mardid, Spain, July 2015. ACM, ACM Press.

[19] W. Jaśkowski, B. Wieloch, and K. Krawiec. Fitnessless coevolution. In M. Keijzer, editor, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 355–362, Atlanta, GA, USA, jul 2008. Association for Computing Machinery, Association for Computing Machinery.

[20] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6:481–494, 2002.

[21] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[22] K. Krawiec and P. Liskowski. Automatic derivation of search objectives for test-based genetic programming. In *European Conference on Genetic Programming*, pages 53–65. Springer, 2015.

[23] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[24] P. Liskowski and K. Krawiec. Discovery of implicit objectives by compression of interaction matrix in test-based problems. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 611–620. Springer, 2014.

[25] P. Liskowski and K. Krawiec. Non-negative matrix factorization for unsupervised derivation of search objectives in genetic programming. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 749–756. ACM, 2016.

[26] P. Liskowski and K. Krawiec. Online Discovery of Search Objectives for Test-based Problems. *Evolutionary Computation*, mar 2016.

[27] P. Liskowski and K. Krawiec. Surrogate fitness via factorization of interaction matrix. In *European Conference on Genetic Programming*, pages 68–82. Springer, 2016.

[28] S. M. Lucas. Learning to play Othello with N-tuple systems. *Australian Journal of Intelligent Information Processing Systems, Special Issue on Game Technology*, 9(4):01–20, 2007.

[29] S. Luke and R. P. Wiegand. Guaranteeing coevolutionary objective measures. In K. A. de Jong, R. Poli, and J. E. Rowe, editors, *Foundations of Genetic Algorithms VII*, pages 237–251, Torremolinos, Spain, 2002. Morgan Kaufman.

[30] E. P. Manning. Using Resource-Limited Nash Memory to Improve an Othello Evaluation Function. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):40–53, 2010.

[31] L. Panait and S. Luke. A comparison of two competitive fitness functions. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 503–511, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[32] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons. Text mining using non-negative matrix factorizations. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 452–456. SIAM, 2004.

[33] E. Popovici, A. Bucci, R. P. Wiegand, and E. D. de Jong. Coevolutionary Principles. In G. Rozenberg, T. Bäck, and J. N. Kok, editors, *Handbook of Natural Computing*, pages 987–1033. Springer, 2012.

[34] C. Reynolds. Competition, coevolution and the game of tag. In R. A. Brooks and P. Maes, editors, *Artificial Life IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 59–69, MIT, Cambridge, MA, USA, 1994. MIT Press.

[35] T. Runarsson and S. Lucas. Preference Learning for Move Prediction and Evaluation Function Approximation in Othello. *Computational Intelligence and AI in Games, IEEE Transactions on*, 6(3):300–313, 2014.

[36] I. E. Skoulakis and M. G. Lagoudakis. Efficient Reinforcement Learning in Adversarial Games. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, pages 704–711. IEEE, Nov. 2012.

[37] M. Szubert and W. Jaśkowski. Temporal difference learning of n-tuple networks for the game 2048. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[38] M. Szubert, W. Jaśkowski, and K. Krawiec. On scalability, generalization, and hybridization of coevolutionary learning: a case study for othello. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(3):214–226, 2013.

[39] M. Szubert, W. Jaśkowski, P. Liskowski, and K. Krawiec. Shaping Fitness Function for Evolutionary Learning of Game Strategies. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1149–1156, New York, NY, USA, 2013. ACM.

[40] M. Thill, P. Koch, and W. Konen. Reinforcement Learning with N-tuples on the Game Connect-4. In *Proc. of the 12th International Conference on Parallel Problem Solving from Nature*, pages 184–194, Berlin, Heidelberg, 2012. Springer.

[41] S. van den Dries and M. A. Wiering. Neural-Fitted TD-Leaf Learning for Playing Othello With Structured Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 23(11):1701–1713, Nov. 2012.