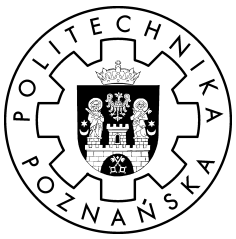
Poznań University of Technology
Institute of Computing Science



Paweł Liskowski

Co-Evolution Versus Evolution with Random Sampling for Acquiring Othello Position Evaluation

Master's Thesis

Supervisor: Dr inż. Wojciech Jaśkowski

Poznań, 2012

Abstract

This thesis focuses on comparison and analysis of methods that learn the Othello Position function. This test-based problem can be naturally embedded in the common conceptual framework of coevolution. However, in coevolutionary learning, a proper navigation through the problem search space requires the fitness function to assess the utility of an individual objectively, which in the coevolutionary environment can only be attained by calculating the true generalization performance, i.e., by playing against all possible opponent strategies and taking average game outcomes. In most cases this is computationally infeasible. Recently, Chong et al. proposed an approach to find the required number of opponents (test cases) for a robust generalization performance estimation and subsequently use it to guide the coevolutionary search. These principles were applied to the Improved Coevolutionary Learning (ICL), an evolutionary algorithm which uses a random sample of opponents to obtain a fitness value for evolving individuals. The method of estimating true generalization performance of any candidate solution in order to use it directly as a fitness measure has raised questions whether such an approach indeed yields solutions which are more robust and generalize better. Along with some premises suggesting that when generalization framework is used, coevolutionary learning leads to the search of strategies with increasingly higher utility, these issues constitute the main motivation behind this work.The results of extensive computational experiments prove that guiding coevolutionary search on the basis of games against a sample of random opponents employed by ICL has indeed a great potential when applied to the problem of Othello. Nevertheless, we show that it is possible to design a coevolutionary algorithm, using the Hall of Fame archive and competitive fitness sharing, of better performance than ICL. Additionally, we have found that in some situations ICL exhibits less behavioral diversity and loses its supremacy when compared to modern coevolutionary algorithms.

Final conclusions point to the need of further investigation of the generalization performance in the coevolutionary environment. We propose a few potential directions for the future work.

Acknowledgments

I would like to express my gratitude to my supervisor Wojciech Jaśkowski for the advice, support and a large interest in my work. I deeply thank him especially for insightful discussions, useful criticism and for what I learned from him. His enthusiasm and inspiration were invaluable to me.

I wish to thank my family and my girlfriend for their love, patience and support they showed me throughout writing of this thesis. Finally, I deeply thank my father for numerous discussions and many constructive comments. His knowledge and experience will always inspire me.

Contents

Chapter 1

Introduction

1.1 Problem Setting and Motivation

Evolutionary computation is an area of research deeply embedded within computing science and is a subfield of artificial intelligence or, more specifically computational intelligence [26]. Natural processes have always served as a source of inspiration for human kind, thus it is not surprising that the scientists pursued the idea of adopting biological principles into computation grounds. The use of Darwinian principles for automated problem solving dates back to the 1950s. A few years later, in early 1960s, three independent interpretations of these ideas started to emerge, giving birth to the mainstream of evolutionary computation [37, 40, 69]. In the last few decades, the continuous advancement of technology encouraged the application of evolutionary computation to virtually every area of problem solving. As astonishing as it may seem, evolutionary computation has proven to be an immensely powerful problem-solving strategy, demonstrating the applicability of evolutionary principles. Evolutionary computation has been successfully used in a wide variety of fields to evolve solutions to difficult problems such as complex engineering problems[69], cancer detection [35, 36], automatic evolution of computer programs [49], modeling adaptive systems by means of genetic programming [40] or learning in games [75, 32, 33]. Crucially, as larger problems are considered, the solutions evolutionary methods discover are often more efficient and robust than those designed explicitly by a human.

Computational intelligence addresses a broad range of complex problems adopting for this purpose various nature-inspired computational methodologies and approaches. These problems often share the concept of an elementary interaction between a candidate solution and a test. Depending on the environment and particular properties of an analyzed problem, a candidate solution might be an abstract design, a machine learning hypothesis or in case of game-playing agents a strategy. Accordingly, a test may take the form of an environmental challenge, a training example, or in case of games, a opponent strategy. Intuitively, the performance of a candidate solution is determined by the outcomes of such tests. However, the number of tests in a problem is typically large or even infinite in case of problems with real-valued variables, effectively precluding the possibility of confrontation between candidate solutions and all tests [41]. Consequently, on the basis of limited interactions and their outcomes, final conclusions about particular underlying phenomenon

are substantially limited and, in extreme cases, simply misleading. In the literature, such problems are known as *test-based problems* [8, 23, 41]. Since determining outcomes of interactions between a candidate solution and all tests is computationally infeasible, accurate and efficient (in terms of computational effort) evaluation function cannot be computed for non-trivial problems. Thus, to solve a test-based problem, an efficient algorithm is expected to determine a particularly interesting set of interactions worth computing from the point of view of the ultimate goal. Upon outcomes of these interaction, a superior, in some sense, candidate solution with respect to other candidate solutions can be emerged. In the light of these concepts, *competitive coevolutionary algorithm* [4, 39, 73] seems to constitute an intuitive method for solving test-based problems. Interactions between candidate solutions and tests are performed in an adaptive and dynamic manner, allowing to identify valuable tests by the search process, at the same time driving the evolution in the assumed direction. Consequently, evolving candidate solutions are challenged with increasingly harder tests, encouraging the *arms race* [60] among coevolving individuals. Thus, coevolution is capable of responding to individual's performance by constructing adequate learnable gradient [83].

Coevolution have been successfully used in the fields of optimization and machine learning for many years, dating back as far as the 1960s [70]. More recently coevolutionary algorithms have gained favor in creating solutions in the area of *interactive domains* and were applied to many test-based problem such as board and strategic games [32, 33, 46, 47, 54]. Another famous work in the area of competitive environments was provided by Hillis [39], which involved coevolution of sorting networks and number sequences to be sorted in competing populations. It was followed by evolving closed-loop controllers of simulated medical activities on patients [76]. Another interesting example of application of coevolutionary algorithms to a test-based problem is evolution of rules for cellular automata [45].

However, despite the initial success of coevolutionary learning in solving test-based problems, it turned out that in some situations they exhibit certain *pathologies,* ultimately failing to obtain high-quality solutions. Common variants of those pathological behaviours include disengagement, cyclic dynamics, overspecialization and forgetting [62, 31, 84]. Another difficulties encountered during modeling coevolutionary algorithms are hard to understand dynamics [45, 66], which hinder overall progress of the search procedure. Jointly, those harmful phenomenons constitute a major challenge in coevolutionary algorithms design.

Although significant effort has been undertaken to overcome these pathologies, apart from minor remedies slightly reducing their negative impact, no real solution to the problem has been found so far. Eventually, this has led to devoting much of the research effort to understand the nature and origin of coevolutionary pathologies. While certain studies imply that coevolutionary pathologies are deeply associated with the use of a relative fitness measure in the selection process [67, 27] other go even further, and show that the generic design of the coevolutionary learning framework negatively influences the performance by imposing to much dynamics on search operators [12, 16, 15].

The adoption of generalization framework originating from the machine learning to coevolutionary ground has recently become particularly interesting research area [87, 13, 14, 11, 10]. The role of a generalization framework is to predict a global performance

of a learning system. Despite many significant differences between machine learning and evolutionary computation, it is believed that generalization framework can be applied to cope with test-based problems, providing insight into absolute quality of a solution. Theoretical framework which addresses the problem of determining the generalization performance of coevolutionary learning was presented in [11] and then further improved in [10]. The notion of estimating true generalization performance of any candidate solution and subsequently using it directly as a fitness measure has raised questions whether such an approach indeed yields solutions which are more robust and generalize better . Along with some premises suggesting that when generalization framework is used, coevolutionary learning leads to the search of strategies with increasingly higher utility, these issues constitute the main motivation behind this work.

1.2 Scope and Objectives

In the context of the discussion outlined in the previous section, this study fits into the disciplines of evolutionary computation and machine learning and focuses on analysis and comparison of different approaches for solving test-based problems: Improved Coevolutionary Learning (ICL) [10] and enhanced coevolutionary learning in particular. Both methods are considered as attractive approaches for problem solving in cases where an absolute quality measure to guide the search for solutions with increasingly higher utility is not available or difficult to acquire. Canonical example of such a problem is learning game-playing strategies. For the purpose of testing studied learning algorithms, we concentrate on the particular test-based problem: acquisition of Othello position evaluation function.

The main objective of this thesis is to investigate the generalization performance in a competitive coevolutionary learning setting. We would like to verify experimentally whether using generalization estimates as the fitness measure during the evolution improves the coevolutionary search and yields solutions which are more robust and generalize better. Furthermore, we aim at inspecting the notion stating that coevolutionary learning leads to the search of strategies with increasingly higher utility provided the generalization framework is used. We employ not only various quality measures but also make extensive use of different experimental settings to thoroughly examine the influence of generalization framework on coevolution. Moreover, we consider an enhanced version of coevolutionary learning and compare it to the improved approach which relies directly on the generalization estimates. Apart from these goals, we have several additional objectives to fulfill:

- Design an efficient coevolutionary algorithm capable of competing with ICL in terms of the generalization performance.
- Employ different player architectures and examine performance of the studied algorithms with reference to the generalization performance.
- Compare various coevolutionary algorithms on the generalization performance criterion.
- Develop a software framework to conduct computational experiments.

Chapter 2

Background & Methods

2.1 Test-Based Problems

Development of automated problem solvers is one of central themes of artificial intelligence. However, designing algorithms that are expected to successfully realize this idea, requires in-depth knowledge about a considered problem's domain. A number of problems investigated in the field of computational intelligence can be formulated as an optimization process. In an optimization problem both model of solution (knowledge about an underlying system) and desired output (answer of the system) are known a priori, providing the ability to describe the task as finding an input leading to the expected output with respect to the function defined on the domain. Alternatively, this can be seen as maximizing or minimizing a given objective function which from the problem-solving perspective represents finding a solution to the task.

However, there exists a broad range of compound problems for which the objective function does not exist or is so complex that evaluating it becomes computationally intractable. For instance, in the context of learning game-playing strategies an objective function could be defined as the expected result over games with *all* possible strategies from the problem's search space. Unfortunately, even for simple games the number of different strategies is so computationally intractable [65].

Problems that are dependent upon interactions with usually a large set of some abstract *tests* are called test-based problems [8]. The most prevalent goal set before these interactions is estimation of quality of the candidate solution. Other terms commonly used for similar types of problems are coevolutionary domain [27], competitive fitness environment [2] or interactive domain [29].

More formally, we define a test-based problem as a formal object $\mathcal{G} = (S, T, G)$ which consist of:

- a set S of candidate solutions
- a set T of tests
- an interaction function $G : S \times T \to \mathbb{R}$.

In this thesis we restrict our attention to the case where the codomain of G consists of number of points possible to score in the game of Othello, that is a set $\{0, 1, 3\}$. If $G(s, t) = 0$, we say that solution s loses the interaction (game in this case) with t; if

$G(s,t) = 1$, we say that s draws the game with t; finally, if$G(s,t) = 3$, solution s wins the interaction. Since in this thesis we focus on *symmetrical test-based problems*, we assume that $S = T$. Importantly, there are still two roles: candidate solutions and tests, but they are played by the same entities.

Commonly used solution concept for the test-based problems is known as *maximization of expected utility* [29]. This concept involves seeking a candidate solution maximizing the expected score against a randomly selected opponent. Adoption of this framing is equivalent to formulating learning task as an optimization problem in which the objective function is the expected performance achieved in the course of competition against any possible opponents from the set of tests T. Even though such definition of the best scoring strategy is natural and intuitive, it is infeasible to compute the exact performance of an individual due to the vast number of potential opponent strategies. This issue can be dealt with by dropping the expensive objective function in favor of approximate quality measures which limit the number of opponents resulting in computationally feasible function.

2.2 Generalization performance

2.2.1 Origins

Generalization is the ability of a machine learning algorithm to perform well on previously unseen data after being trained on a training examples. In the literature it is often referred to in terms of input-output pairs — the core objective is to find a solution that best predicts the required output for any input that has not been seen during the training process. Generally, distribution of training examples used to train the algorithm is unknown, thus forcing the learner to extract and capture core characteristic of the data and subsequently apply the discovered patterns in new cases to accurately predict the output. Thus, the ultimate goal set before the learner is to generalize from its experience [5].

Even though the notion of generalization is well documented and understood in the field of machine learning, it is not clear how to apply generalization paradigms to coevolutionary learning ground due to a lack of theoretical framework to justify how the generalization performance is determined with respect to the particular problem. Nevertheless, first attempts to adopt the generalization framework and investigate the coevolutionary learning through the prism of generalization performance were carried out in [87, 13]. Those studies have used a large sample of random test cases obtained from the search space to estimate the true generalization performance. Since the true generalization performance is calculated using the entire space, it is impossible to determine how accurate such an estimation is. This issue was later addressed by the theoretical framework intended for more precise, quantitative performance analysis of coevolutionary learning proposed in [11]. The generalization performance of a solution was estimated using a set of random test cases by aggregating interaction results and taking their average outcome with confidence bounds provided by Chebyshev's theorem [43]. Recently, this approach was further improved in [10] by exploiting the near-Gaussian nature of generalization estimates. Consequently, tighter bounds based on parametric testing could be provided, eventually allowing to find the required number of test cases for a robust generalization performance estimation.

We give two reasons why applying the concept of measuring generalization performance to coevolutionary learning is worth venturing. First, it could be used to provide an absolute quality measure on how well a coevolutionary learning system is performing in the context of a particular problem. In such case, one could answer the question of how well the coevolutionary learning generalizes. Second, it can be used as a means to compare the overall performance of different coevolutionary learning systems with respect to the particular problem.

2.2.2 Definition

We define true generalization performance of a strategy $s \in S$ as its expected performance over test strategies from set T. Such a definition naturally defines the outcome of the search as a strategy which maximizes expected score against a randomly selected opponent, or more generally, maximizes the average outcome against all opponents when the selection is based on an uniform distribution. This framing approximates the solution concept of maximization of expected utility introduced in [29]. For simplicity, we sometimes refer to the true generalization performance simply as generalization performance without loss of generality.

Although this definition is simple and intuitive, the generalization performance can be difficult to determine for at least two reasons. Firstly, the analytical function for game outcomes can be unknown, thus in such cases we cannot determine the generalization performance by solving analytically a closed-form formula. Secondly, the strategy space is often large or even infinite, thus making it computationally intractable. To address this issue, following [10] we estimate the generalization performance by calculating the average performance of the evolved strategy against a sample $T_N \subseteq T$ of test strategies randomly drawn from the strategy space.

More formally, the estimated generalization performance of a strategy$s \in S$ is given as follows:

$$\hat{G}_s(T_N) = \frac{1}{N} \sum_{t \in T_N} G(s, t).$$

2.2.3 Application of Generalization Performance to Coevolutionary Learning

In coevolutionary learning, proper navigation through the problem search space requires the fitness function to assess the utility of an individual objectively which in the coevolutionary environment can only be attained by calculating the true generalization performance, i.e., by playing against all possible opponent strategies and taking average game outcomes. However, in most cases this is computationally infeasible. Previous research [13, 87] is this field has shown that reducing the computational effort and biasing the search by limiting the opponents sample may not necessarily lead to strategies with increasingly higher utility. Approach presented by Chong et al. in [10] allows to find the required number of opponents (test-cases) for a robust generalization performance estimation and subsequently use it to guide coevolutionary search. These principles were successfully

applied to the improved coevolutionary learning algorithm proposed for the first time in [10].

2.3 Coevolution

2.3.1 Coevolutionary Algorithms

Coevolutionary algorithms are inspired by a biological evolutionary process and focus on exploiting phenomenon of a arms race observed in natural environments. In nature, fitness of a coevolving species is almost exclusively dependent on interactions with other organisms and the environment they are living in. Dynamically changing environment requires species to adapt and struggle for both survival and reproduction. When combined, these principles formulate the *natural selection* law described by Darwin [17]. Since natural selection rewards the fittest individuals, each species evolution moves towards development of features which aim at either outperforming the competition or cooperating to achieve the common goal. In this way, species exert selective pressure on each other, thus influencing mutual evolutionary development. Ultimately, this allows certain species to evolve traits which give them an edge over other coevolving organisms. All these principles influenced development of coevolutionary algorithms and constitute a major source of inspiration for them.

In computing science, coevolutionary algorithms have been introduced into the field of artificial intelligence as an extension to evolutionary algorithms. They intend to model interactions between individuals as observed in nature to accurately imitate fitness evaluation function. Specifically, individual's fitness depends on other individuals, thereby is *subjective* [84] in the sense that it is a function of its interactions with other individuals. The nature of those interaction might be either cooperative or competitive. In the former case, individuals work together to achieve a common objective which can be naturally viewed as decomposition of a difficult problem P, into sub-problems $\{p_1, p_2, \ldots, p_n\}$. Thus, such algorithms are well suited for *compositional problems* [68, 85]. In case of the competitive form, interactions usually take the form of an encounter between two or more competing individuals, and consequently, a gain for one means a loss for the others. Thus, a competitive coevolution is particularly useful when evaluation function is unknown or difficult to define, perfectly fitting test-based problems. Since in this thesis we focus on evolving game strategies, in the following, by "coevolutionary algorithm" we will mean its competitive form.

Coevolutionary learning and evolutionary computing techniques are both search algorithms based on the mechanisms of natural selection and genetics. That is, they apply Darwinian principle of the survival of the fittest [17] among computational structures with the stochastic processes of selection and variation that mimic the natural evolution such as gene mutation and recombination. Since coevolutionary algorithms are derived from evolutionary ones [34], they share core mechanics and features like for instance maintaining a population of individuals. Central to all evolutionary computing techniques is the idea of searching a problem space by randomly changing an initially random population of solutions in such a manner that the better (fitter) solutions have more chances to survive and pass their genes to subsequent generations.

Contrary to evolutionary algorithms, the driving force behind coevolution is the continuous arms race taking place between competing individuals [60]. Differences between coevolutionary and evolutionary approaches lie primarily in the evaluation phase, and are related to the fitness assessment. Evolutionary algorithms designed to solve optimization problems have direct access to the objective function of a problem, thus allowing to easily compute individual's objective fitness. In such case, it is possible to objectively assess how good a particular individual (solution) is. On the other hand, in coevolutionary algorithms, individual's fitness is typically calculated by aggregating the outcomes of interactions it participated in. As mentioned earlier, the fitness assigned to the individual is subject to the state of constantly changing environment. Thus, in coevolutionary algorithms, the relative fitness substitutes objective fitness and guides the selection procedure, but the correlation between the subjective and objective fitness of a solution is generally unknown. Thus, we can only determine how good the particular individual fares in the context of other individuals of (usually) the same generation. Ensuring objective progress continues to be a major challenge in coevolutionary ground. This constitutes well-motivated area of research, however there are several methods which attempt to address this issue. One of them are coevolutionary archives discussed in Section 2.3.3.

2.3.2 Interaction Patterns

In this study, we rely on a single, homogeneous population of players. In the literature such a setup is known as one-population coevolution [57] or is also referred to as competitive fitness environment [2, 55]. In coevolutionary algorithms individuals are evaluated on the basis of interactions with other individuals. However, prior to evaluation opponents for each individual must be specified. The abstract pattern responsible for pairing individuals is known as *competition topology* and has been subject to previous research in [2, 61, 78]. Among many popular topologies, one commonly used in practice is a *Round-Robin Tournament*. In this pattern each member of population interacts with every other individual. For instance, in one-population coevolution all members of population serve as opponents, except for the one being evaluated. Such an approach results in the most accurate evaluation.

2.3.3 Coevolutionary Archives

A method in coevolutionary learning aimed at improving reliability and sustaining overall progress during search is an coevolutionary archive by counteracting, or at least neutralizing, the negative impact of coevolutionary pathologies. Typically, an archive is defined as a set of top-performing individuals encountered during the evolutionary process. The most common archives employed in the literature can be described as *best-of-generation* models as they consist of the fittest individuals of the m most recent generations. Such an archive is also known as Hall of Fame [73]. Keeping a sample of individuals in the archive allows to achieve a sense of historical progress by testing current solutions against those already present in the archive [28]. Additionally, an archive is a source of genetic material for future generations, thereby preventing extinction of certain valuable traits.

Apart from classical Hall of Fame, modern examples of archives include Incremental

Pareto Coevolution Archive [18], Layered Pareto Coevolution Archive [20], MaxSolve [19], DECA [22], EPCA [86] and Nash Memory [28].

2.3.4 Diversity Maintenance

A tendency towards reduced population diversity along with the problem of premature convergence is a major concern of the evolutionary computation. To address this issue, several attempts to maintain genetic diversity have been proposed. All of these methods rely on the assumption that the loss of diversity in the population of individuals can be harmful to the optimization process as it may restrict the search to local optima. On the other hand, the bigger diversity of the population, the better exploration of the problem's search space.

The most popular diversity maintenance techniques include deterministic crowding [24], fitness sharing [38], competitive fitness sharing [73, 72] and spatial embedding [39].

Among them, *competitive fitness sharing* proposed by Rosin and Belew is particularly interesting because it maintains genetic diversity in a population by encouraging niching — individuals are rewarded for being able to beat opponents that few others can. In coevolutionary scenarios opponents are often treated as a resource shared among individuals. As opposed to the usual way to assign fitness based on the outcomes of interactions across all competitions [2, 39, 4], in competitive fitness sharing the fitness assigned to a candidate solution (individual) s defeating tests (other individuals) from T is given as $\sum_{t \in T} \frac{1}{N_t}$, where N_t is the total number of candidate solutions in the population defeating test t. Consequently, individuals are rewarded less for how many opponents they overcome and more for who they outmatch, rewarding genetic novelty and maintaining diversity. Thus, competitive fitness sharing attempts to secure survival of important individuals throughout the evolution process.

Diversity maintenance was found to counterpart to some extend *disengagement,* since it is directly associated with a loss of diversity [7, 9].

2.3.5 Advantages of Coevolution

There are several benefits of applying coevolutionary algorithms to test-based problems. Firstly, coevolution it is able to autonomously acquire knowledge about the problem domain, efficiently imitating the natural human learning process. Consequently, similarly to other unsupervised learning approaches, coevolution is capable of developing solutions to the problem without any human expertise. Secondly, contrary to the evolutionary approach, they do not require an objective fitness function to evaluate individuals — interactions among competing population members are sufficient to guide the search. Consequently, by engaging players in the mutual pressure to outperform each other, coevolution provides an adaptive gradient that might otherwise be hard to obtain [64]. Interesting feature of coevolutionary learning is *focusing* which refers to the ability of adaptation to those aspects of task which have not yet been optimized. This can be achieved by using coevolving individuals as opponents [39, 44]. Finally, coevolutionary algorithms constitute generic framework naturally suited to solve problems that cannot be framed in the context of optimization.

Coevolution is also a method capable of solving *open-ended problems* [31]. This is possible since it does not require the objective fitness function to exist.

2.4 Othello

2.4.1 Brief History

Othello was originally invented with the name Reversi in England around 1883 by Lewis Waterman, who in fact may have copied most of the ideas of the game from John W. Mollett. "The Game of Annexation" published in 1870 by Mollet was a very similar game with nearly identical rules but used a cross shaped board instead of a standard square one.

The modern rule set of Othello used during the international tournaments originated in the 1970s in Japan and was proposed by Goro Hasegawa. The name of the game was chosen with reference to the play by William Shakespeare "Othello, the Moor of Venice", most likely referring to the conflict between the Moor Othello and Iago — the main character of the play. Nevertheless, as the game was dynamic and full of dramatic reversals, it quickly gained notable popularity in Japan followed by Europe and North America.

Today, Othello is perceived as a simple classic board game, however it is still one of the most popular games with numerous tournaments and regular world championships.

2.4.2 Game Rules

The game of Othello is a deterministic, perfect information, zero-sum board game played by two players on an 8×8 board. There are 64 identical pieces typically called disks, which are white on one side and black on the other. Each of the two sides corresponds to one player which are referred to as light and dark after the sides of Othello pieces. The game starts with the center four squares of the board occupied with two black and two white pieces arranged diagonally. Players make moves alternately by placing their pieces of on the board until it is completely filled or until neither of them is able to make a subsequent, legal move. The ultimate objective of the game is to have the majority of own pieces on the board at the end of the game. The game might end in a draw, provided both players have placed the same number of disks on the board.

In Othello, the black player starts the game by making a first move. A legal move consists of placing a new piece on an empty square adjacent horizontally, vertically, or diagonally to an opponent's existing piece. However, the move must be made in such a manner that at least one of opponent's pieces lies between the player's new piece and existing pieces. After placing the piece, all opponent's pieces lying in the line are flipped over to become the player's pieces. If multiple lines exist, flipping affects them all. This is the main reason why the game is so dynamic — in a single move a player may gain significant advantage over the other player, effectively turning the tide. If a player has no valid moves, he forfeits by passing the turn to the opponent who is rewarded by playing a second turn in a row. The game ends when all the squares of the board are either filled with pieces or when neither player is able to make a legal move.

2.4.3 Strategy Representation

Important issue to be considered when learning game-playing strategies is the architecture of a player which is greatly influenced by the strategy representation. There are many different strategy representations, however for the purpose of this study, we focus on weighted piece counters and n-tuple networks.

2.4.3.1 WPC

The position-weighted piece counter (WPC) is a linear weighted board evaluation function which implements a state evaluator concept [80, 82]. Therefore, it is explicitly used to evaluate how beneficial a given state is for the particular player. The WPC evaluation function of an Othello game strategy takes the form of a vector of 64 weights. It assigns a weight w_i to a board location i and uses scalar product to calculate the utility f of a board state $\mathbf{b}$:

$$f(\mathbf{b}) = \sum_{i=1}^{8\times 8} w_i b_i.$$

The input value b_i depends on the occupation of the particular board location. More specifically, it is equal to 0 in case of an empty location, +1 if a black piece is present or -1 in case of a white piece. The output of this function reveals how favorable a given board state $\mathbf{b}$ is for a particular player. The players interpret this value in a complementary manner: the black player prefers moves leading towards states with a higher value, whereas lower values are favored by the white player.

A strategy represented by WPC can be easily understood by inspecting the weight values. For instance, Fig. 2.1 shows the weights matrix of a standard heuristic player (SWH). As it can be easily seen, high values placed in the corners imply that the player focuses on obtaining those squares at the same time avoiding adjacent locations since they contain the lowest values from the whole matrix.

Simplicity of WPC is considered to be its primary advantage. Such representation results in fast board evaluation, thus allowing to conduct many games in a relatively short time.

Position-weighted piece counter representation has been commonly used in previous research [54, 47, 74].

1.00	-0.25	0.10	0.05	0.05	0.10	-0.25	1.00
-0.25	-0.25	0.01	0.01	0.01	0.01	-0.25	-0.25
0.10	0.01	0.05	0.02	0.02	0.05	0.01	0.10
0.05	0.01	0.02	0.01	0.01	0.02	0.01	0.05
0.05	0.01	0.02	0.01	0.01	0.02	0.01	0.05
0.10	0.01	0.05	0.02	0.02	0.05	0.01	0.10
-0.25	-0.25	0.01	0.01	0.01	0.01	-0.25	-0.25
1.00	-0.25	0.10	0.05	0.05	0.10	-0.25	1.00

Figure 2.1: The standard heuristic player's strategy represented by WPC.

2.4.3.2 n-tuple Network Architecture

An n-tuple network is a specific type of artificial neural network [1] and was originally developed for use in optical character recognition by Bledsoe and Browning [6]. Is was also successfully applied to classification [71] and function approximation tasks [48]. In the context of this thesis, particularly influential articles have been recently published which employed the n-tuple architecture for game-playing purposes [52, 50].

An n-tuple system operates on some complex entity X, which elements can be retrieved using some form of coordinates. For the purpose of processing any input object consistent with assumptions of the considered system, an n-tuple network begins with sampling it with m n-tuples. Such an n-tuple is defined as a sequence of n variables $a_{ij}, j = 0 \ldots n-1, i = 0 \ldots m-1$, each corresponding to predetermined coordinates in the input. Furthermore, it might be viewed as a template for an n-digit number in base-v numeral system, assuming that each variable takes one of v possible values. The number represented by the n-tuple t_i is used internally as an index in the associated look-up table LUT_i which contains parameters equivalent to weights in a typical neural network. Thus, for a given input X, the output of the n-tuple network can be calculated as follows:

$$f(X) = \sum_{i=0}^{m-1} LUT_i \left[\sum_{j=0}^{n-1} x(a_{ij}) v^j \right],$$

where $x(a_{ij})$ denotes retrieving from X the element located at position pointed by a_{ij}.

When applied to the problem of Othello, an n-tuple network is used as a state evaluation function, taking board state as an input and returning its utility. A set of inputs is taken from specific board locations each encoded as 0,1, 2 provided it is occupied by a white piece, black piece, or is empty respectively. Therefore, an single n-tuple represents a ternary number used as an index for the associated look-up table containing 3^n entries. Additionally, symmetric sampling proposed in [52] is utilized. Consequently, every n-tuple is employed eight times, once for each of possible board reflection and rotation . To obtain output of the particular n-tuple values from its look-up table indexed by all such equivalents must be summed together.

A n-tuple network has several advantages making application to the game of Othello particularly noteworthy. Firstly, it is conceptually simple and built upon well-known artificial neural networks. Secondly, it is computationally efficient, as the computation time is linear in the number of n-tuples in the network. Recent studies have shown that it may be even 15 times faster than the best neural network [59]. Finally, its ability of realizing non-linear mappings to spaces of higher dimensionality seems to be especially promising as it allows traversing a search space more accurately.

Chapter 3

Experiments and Results

3.1 Experimental setup

3.1.1 Methods

In this thesis we compare several algorithms for solving test-based problems, which will be described in the following subsections.

3.1.1.1 Coevolutionary Learning

In this thesis by Coevolutionary Learning (CCL) we denote a $(\mu + \lambda)$ evolutionary strategy [34] with competitive fitness. In order to learn game-playing strategy, the algorithm begins with a population of μ randomly generated players, which are then evolving for m generations. Population members compete by playing games with each other. The results of these confrontations determine the fitness which is assigned to each player. In each generation a new population is created from the previous one. To accomplish this the parents are deterministically selected from the set of both the parents and offspring. Selection is based on the ranking of the individuals' fitness taking the μ best individuals (often referred to as truncation selection [34]). Each of the μ fittest individuals produce λ/μ offspring through a mutation operator. Thus, the next generation consists of the μ parents and the λ new offspring. No recombination operator is used. This method will be used as a baseline for most of the forthcoming experiments. In our experiments we also consider variation of this algorithm (denoted as CEL-ST1) which the presented setup with the Hall of Fame archive and the competitive fitness sharing leaving the core of algorithm intact.

CCL was used by Chong et al. for Iterated Prisoner Dillema and Othello in [10].

3.1.1.2 Coevolutionary Learning with Hall of Fame and Fitness Sharing

Coevolutionary Learning with Hall of Fame (CEL-ST2) follows the idea of competitive coevolution [4, 2] and is based upon typical, one population approach where the fitness of individuals depends on the outcomes of interactions between them. To learn a game-playing strategy, CEL starts with generating a random initial population of players. Prior

to forming next generation, the best individuals are selected according to the relative, internal fitness assessment and are subject to genetic variation operators such as recombination and mutation. In this case, to evaluate the players we rely on the round-robin tournament (see Section 2.3.2). Next, the evaluated players are selected by means of the tournament selection, followed by the uniform crossover combined with Gaussian mutation. Obtained in this way offspring replace the former individuals. Here we extend this method with the Hall of Fame archive (see Section 2.3.3), which serves as a source of the best-of-generation individuals discovered throughout the course of evolution. According to Rosin and Belew [73], submission of the genetic material for future generations is essential for the performance. Continuous evaluation against those individuals might ensure sense of progression across the evolution. In order to further improve performance of evolved strategies, simple sum fitness was abandoned in favour of the competitive fitness sharing proposed by Rosin and Belew [73].

3.1.1.3 Improved Coevolutionary Learning

Improved Coevolutionary Learning (ICL) is an algorithm proposed for the first time by Chong et al. in [10]. ICL directly relies on the generalization performance estimate as the fitness measure during the evolution process. The learning procedure is exactly the same as in the case of CCL (see Section 3.1.1.1) except for the evaluation step; instead of using the competitive fitness (a round robin tournament) each individual is evaluated against a set of random opponents. Throughout our experiments, we mostly rely on the sample size of 50,000. However, to provide more accurate estimation, we sometimes use the sample size of 1,000,000. Since the nature of coevolution requires that the individual's evaluation takes place in the context of at least one other individual (competitive fitness), such formulating raises the question whether this approach can be called coevolution. Typically in coevolution individual's fitness is estimated by aggregating results of multiple interactions between other members of population. Thus, in our opinion, ICL resembles more an evolutionary algorithm rather than a coevolutionary one.

3.1.2 Players Architecture and Search Operators

Adopted strategy representation explicitly implies the nature of search space and indirectly affects selection of the search operators. The considered search heuristics operate in a continuous weight space in case of both WPC and n-tuple representation. The performance of an evolutionary strategy depends on the design of the search operators such as mutation, recombination and finally selection which will be further discussed in 3.1.4. Ideally, we would like to design these operators in such a manner that they guarantee the evolvability, i.e, searching a problem space in such a manner that the fitter solutions are obtained over time, of the system throughout the evolution process. It is vital that any point in the search space is reachable in a finite number of steps through a mutation because otherwise the search procedure might not be able to escape a local optima. The main goal of the recombination operator (also referred to as crossover) is to exchange of certain traits between the individuals and to transfer of valuable components to the next generation.

In the first part of our experiments we follow the previous research in the field of learning Othello strategies, and rely on the position-weighted piece counter (WPC) rep-

resentation (see Section 2.4.3.1). Regarding the game of Othello, it is sufficient to learn only 64 weights. In the following we give more details on employed breeding procedure and adopted search operators. To begin with breeding, several individuals must be selected. In order to accomplish this, the evaluated candidate solutions participate in the typical tournament selection with tournament size of 5. Subsequently, chosen individuals mate using uniform crossover, and the resulting offspring undergoes Gaussian mutation ($\sigma = 0.25$) with probability $p_{wm} = 0.05$. These operators are used in all of the studied coevolutionary algorithms.

Promising potential unveiled in the recent studies [52, 59] have inspired us to apply the n-tuple system (see Section 2.4.3.2) to the problem of Othello. Therefore, in the second part of our experiments, we transform investigated algorithms to directly rely on on the n-tuple system. We begin immediately with the largest 12×6 architecture (8748 weights) which has been recently successfully applied to Othello by Manning [59]. Following [52] we decided to employ the input assignment procedure that results in randomly placed snake-shaped tuples. Regarding the look-up table weights, their initial values are randomly scattered in the search space. For instance, in case of our algorithms, we start from weights initialized randomly from the [-10, 10] range.

Due to differences between WPC and n-tuple strategy representations, different genetic operators had to be employed in order to properly navigate the algorithm through the problem search space defined by the n-tuple player's architecture. The operators work on lookup table weights and n-tuples positions, which together form individual's genome. The following genetic operators are used:

- **weight mutation** — each weight from the LUT table undergoes a Gaussian mutation ($\sigma = 0.25$) with probability $p_{wm} = 0.05$.
- **topology crossover** — defines reproduction with probability $p_{xover} = 1$ — two individuals exchange entire tuples along with look-up tables. During recombination entire tuples are exchanged along with their lookup tables. Thus, each offspring inherits $m/2$ randomly selected n-tuples from each parent, where m is the number of n-tuples.

3.1.3 Measuring Quality of Evolved Strategies

Although in this thesis we concentrate on the maximization of expected utility solution concept and we employ the generalization performance quality measure of evolved individuals, we also use two other quality measures, which are spotted in articles concerning Othello. Used quality measure employed follow:

- **Playing against random strategies** — this method involves generating a set of randomized strategies from the test set T. An individual is evaluated on the basis of games with previously sampled opponents. This method approximates the true generalization performance of the particular individual. We note that this method was previously used in [10, 87, 13] and will be also referred to as *random sampling*.
- **Playing against the random player** — this method tests how well candidate solution performs against a wide variety of opponents. However, contrary to competing against a completely random strategy, the evaluation is based on playing

with a player who chooses a random move for each board state. We emphasize, that playing against a random strategy is different from playing against a random player. This quality measure was used first by Lucas and Runnarson [54] and more recently in [50, 80, 81].

- **Playing against the standard heuristic player** — the primary goal of this quality measure is to study how well a player copes with relatively strong hand-crafted WPC-based strategy, which is shown in Fig. 2.1. In order to make the measure meaningful in a context of a deterministic game of Othello, we enforce strategies to make random moves with probability $\epsilon = 0.1$ on both players to directly affect their behavior diversity. Such formulation results in changed game definition, however following [54], we assume that the ability to play such a randomized game is positively correlated with the ability of playing the original deterministic Othello. This method has been used in previous studies of Othello [80, 81, 59, 53] and it is used to rank the strategies in the Othello League [53].

3.1.4 Selecting the Best Individual

Any algorithms solving a test-based problem must eventually select an individual which is its final result. In case of coevolutionary algorithms the final individual is usually an individual from the last generation, however it is not obvious which one should be returned, since the true objective function is not known to the algorithm and the correlation between a subjective fitness and the objective fitness is generally unknown [21].

We would like to emphasize that the selection of the final individual is important, since it affects the result of comparison of studied algorithms. Here we use two methods of selecting the final individuals:

- **Subjective selection** — an individual with the highest fitness is selected. Importantly, this method does not involve any extra computation, since it directly employs the fitness measure used during the evaluation of an individual. However, assuming a correlation between subjective and objective fitness may be misleading. Consequently, finding the best individual in terms of objective fitness is not guaranteed. Despite this drawback, this approach is extensively used during our experiments and for a certain generation an individual selected in this way is denoted as *subjectively best-of-generation* individual, while *subjectively best-of-run individual* is the subjectively best-of-generation individual for the last generation.
- **Objective selection** — defining objective measure of the individual's quality for the test-based problems is often not a simple task due to inaccurate evaluation of the candidate solutions. More importantly, fitness value obtained by an individual is subject to a relative function which outcome depends on the context of individuals in the same evolutionary process. Therefore, subjective fitness cannot be directly used to asset the objective quality of an individual. Intuitively, one of the possible ways to approximate individual's quality is to simulate interactions and mutual influences with either random or expert individual, i.e. game player or strategy. Following this concept and extending Chong's research, we decided to estimate objective fitness by testing each candidate against a sample of randomly generated strategies. Such

measure is capable of closely reflecting strategy's true performance. The obvious downside of this method is substantial computational effort required to compute objective fitness value. We will refer to the individual selected in such a manner as *objectively best-of-generation* individual.

3.2 Rerun of Chong's Experiments

3.2.1 Experiment Setup and Objective

In order to set the baseline for our investigation in the first experiment we repeat some of the experiments performed by Chong et al. [10] on Othello. More specifically, we focus on measuring generalization performance of Othello strategies. To achieve this goal, we estimate the generalization performance of a black player through games against a random test sample of white players. We considered four algorithms: CCL and three versions of ICL: ICL-N50000, ICL-N500 and ICL-N125, which was not considered by Chong et al. All strategies were encoded by WPC and were learning to play black (see Section 2.4.3.1). Following Chong et al. [10], CCL maintained 50 and ICL 20 individuals in the population. As the number of generations was set to 200 for all algorithms, the computational effort expressed in the number of games played during learning, is different for each method (see Table 3.1). That is why for a direct comparison we included algorithm ICL-N125, which has the same computational effort as CCL. For statistical significance all experiments were repeated 30 times.

3.2.2 Results

Results of the experiment are summarized in Table 3.1. Average, Min and Max columns denote the results obtained by the subjectively best-of-run individual. As a quality measure we used an estimate of generalization performance obtained using 50,000 random white player WPC strategies. Average results are followed by the confidence interval value after the $\pm$ sign. The column denoted by t-test presents the t value obtained using a two-tailed test with 29 degrees of freedom which is statistically significant at level $\alpha = 0.05$.

The results obtained in our experiment confirm the results reported by Chong et al. Average results obtained by us differ by at most 0.017 from results reported in [10]. All versions of ICL produce much better results than CCL and the differences are statistically significant. The more computational effort, the better results obtained by ICL, but even when the computational effort is the same, ICL-N125 surpasses CCL by a large margin.

Figures 3.1-3.4 present the generalization performance of 30 individual runs on a single plot. We can also confirm the observation of Chong et. al that among the considered methods CCL is characterized by the largest variance. We emphasize that due to the large sample size used in case of ICL-N50000, estimated generalization performance is close to the true generalization performance. This can

Figure 3.5 compares subjective average results of four considered methods on a single axis (computational effort). Notice that CCL is far worse than any ICL.

Table 3.1: Comparison of CCL and ICL algorithms learning to play Othello with **black** pieces. Strategies are represented by **WPC**. Results for **subjective** best-of-run individuals.

Algorithm	#Gen.	Pop.Size	Comp. Effort	Average (50k)	Min	Max	t-test
CCL	200	50	500,000	0.7518 ± 0.0191	0.6406	0.8253	-
ICL-N125	200	20	500,000	0.858 ± 0.0061	0.8234	0.8891	-10.8877
ICL-N500	200	20	2,000,000	0.8801 ± 0.0055	0.8433	0.9089	-13.2823
ICL-N50000	200	20	200,000,000	0.907 ± 0.003	0.8923	0.9197	-16.5051

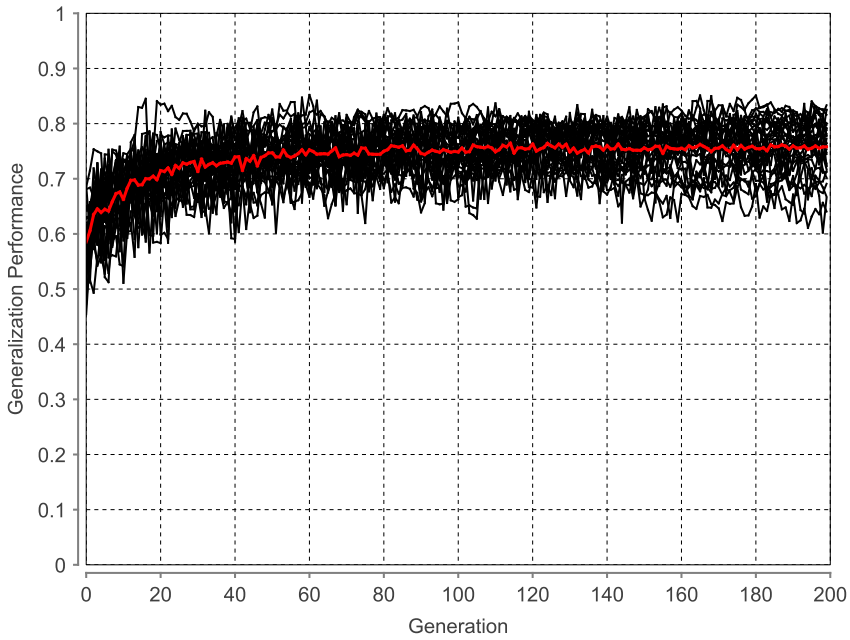


Figure 3.1: Estimated generalization performance of the **subjectively** best-of-generation strategy from the population measured during the evolutionary process. **CCL** algorithm learning to play **black**. Based on **WPC** architecture.

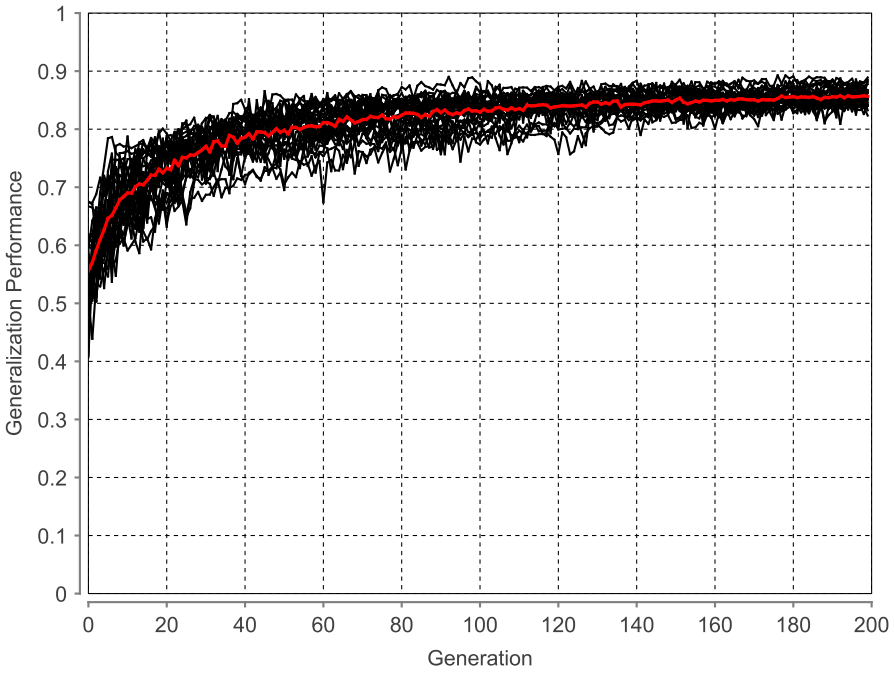


Figure 3.2: Estimated generalization performance of the **subjectively** best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N125** algorithm learning to play **black**. Based on **WPC** architecture.

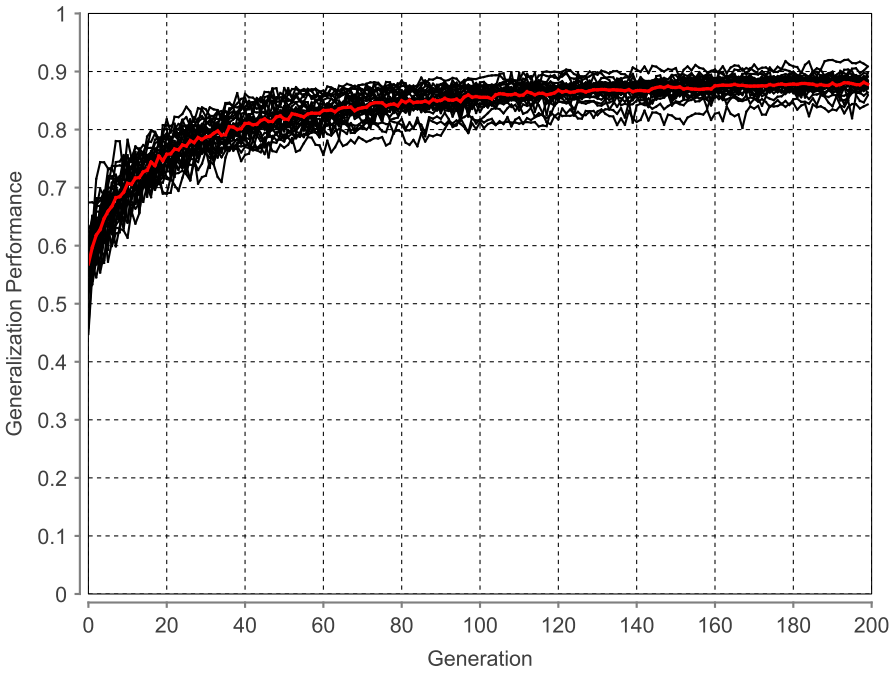


Figure 3.3: Estimated generalization performance of the **subjectively** best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N500** algorithm learning to play **black**. Based on **WPC** architecture.

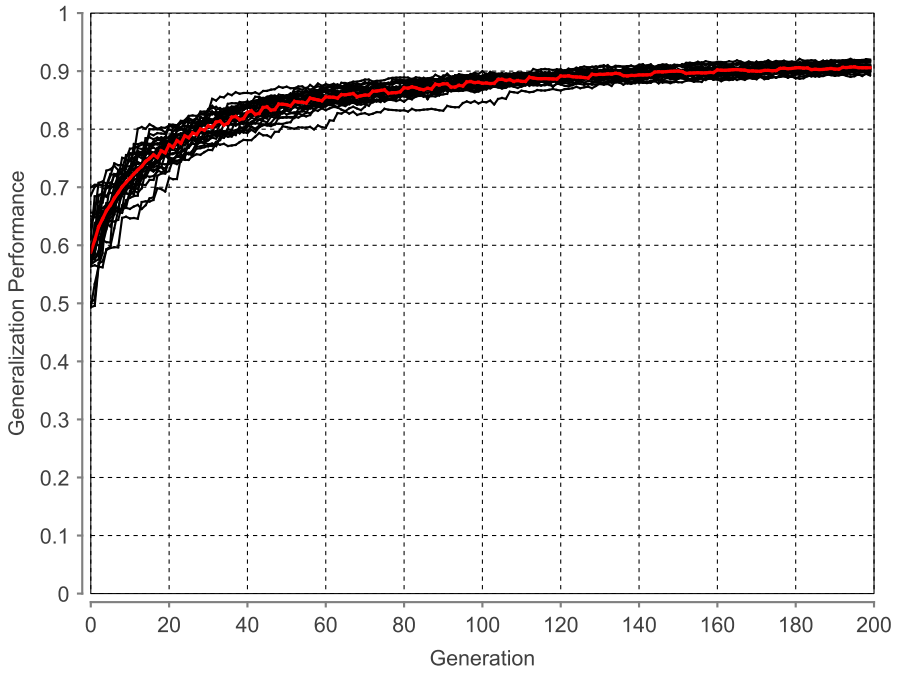


Figure 3.4: Estimated generalization performance of the **subjectively** best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N50000** algorithm learning to play **black**. Based on **WPC** architecture.

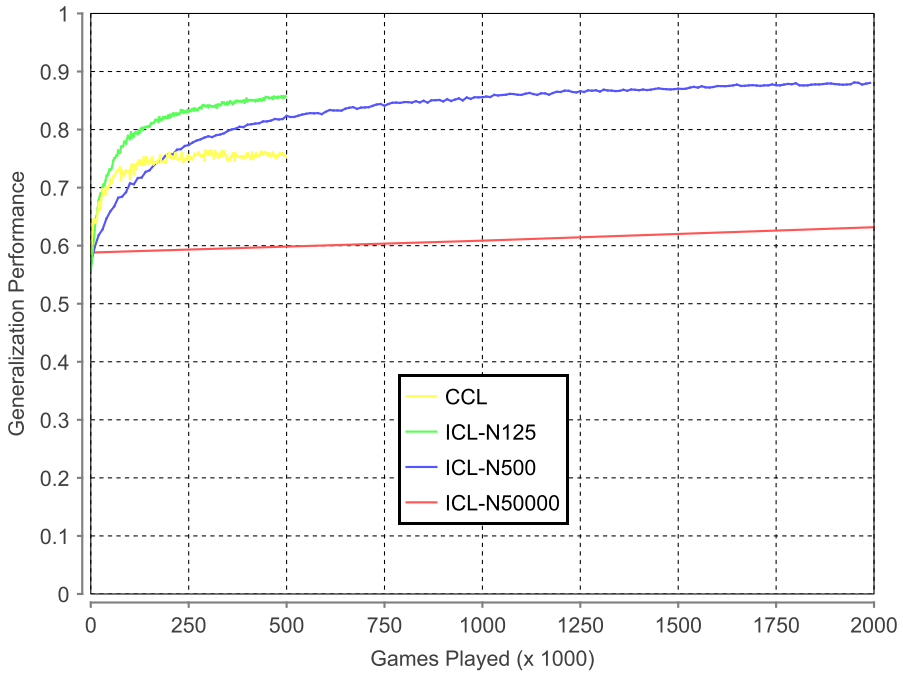


Figure 3.5: Comparison of estimated generalization performance of the **subjectively** best-of-generation strategies measured throughout the evolutionary process for CCL and ICL algorithms. **WPC** architecture. Algorithms learning to play **black**.

3.2.3 Discussion

The results obtained in this experiment may be surprising for the practitioners using coevolutionary methods. A straightforward evolutionary strategy with a random sampling (ICL) can be better than a competitive fitness coevolution (ICL). This raises the question whether there are any conditions for which coevolution is better. We will investigate this issue in the following experiments.

3.3 Analysis of CEL Performance

3.3.1 Experiment Setup and Objective

Since the coevolutionary algorithm (CCL) used by Chong et al. [10] is simple (see Section 3.1.1.1), it may not adequately represent what can be achieved using modern coevolutionary methods. In this section, we extend the analysis performed in the previous one by evaluating the coevolutionary learning algorithm enhanced with the Hall of Fame archive and the competitive fitness sharing method (CEL, see Section 3.1.1.2). We consider two different setups of CEL: CEL-ST1 and CEL-ST2. The objective of CEL-ST1 is solely to verify the impact of using an archive and a diversity maintenance method on the generalization performance as it only extends the CCL mechanics with i) the Hall of Fame archive and ii) the competitive fitness sharing. On the other hand, CEL-ST2 is built from scratch and is expected to answer the question of how well modern coevolutionary algorithm performs when compared to ICL. Both CEL-ST1 and CEL-ST2 maintain 50 individuals in the population. To allow for a comparison between methods, in case of CEL-ST1 we set the number of generations to 100 while for CEL-ST2 the number of generations was set to 400.

3.3.2 Results and Discussion

Results of this experiments and the previous one were merged and summarized in Table 3.2. Figures 3.6 and 3.7 present the generalization performance of the subjectively best-of-generation strategy for each generation of the evolutionary process for 30 independent runs. Similarly to the previous experiment, we used an estimate of generalization performance obtained using 50,000 random white player WPC strategies as the quality measure.

The results obtained in this experiment prove that it is possible to design much better coevolutionary algorithm than the simple CCL. Notice that while computational efforts of CEL-ST1 and CCL are even, the increase in performance exceeds 4%. Moreover, the extremes have also significantly changed in favor of CEL-ST1. Interestingly, in case of CEL-ST2 we notice further improvement in the generalization performance, however at the expense of increased computational effort. On the other hand even CEL-ST2 does not match ICL algorithm of the same computational effort (ICL-N500).

Therefore we conclude that upgrading a simple coevolutionary algorithm with the Hall of Fame archive and the competitive fitness sharing leads to significant increase in the generalization performance. Nevertheless, results achieved by both versions of CEL are still worse than any variant of ICL, unquestionably confirming their supremacy.

Figure 3.8 compares subjective average results of all considered methods on a single axis (computational effort). Notice that while CEL-ST2 learns more rapidly than any other method, in the long run it achieves the same level of performance as ICL-N125. CEL-ST1 proved to be better than CCL, however the difference is rather small.

Our experiment has shown, that in the field of playing games with a random strategy, ICL has no worthy opponent. Intrigued by these results, we will analyze the nature of this success deeper in subsequent experiments.

Table 3.2: Results for different algorithms. Black. WPC. **Subjectively** best-of-last-generation players.

Method	#Gen.	PopSize	Comp. Effort	Avg	Min	Max	t-test
CCL	200	50	500,000	0.7518 ± 0.0191	0.6406	0.8253	-
ICL-N125	200	20	500,000	0.858 ± 0.0061	0.8234	0.8891	-10.8877
ICL-N500	200	20	2,000,000	0.8801 ± 0.0055	0.8433	0.9089	-13.2823
ICL-N50000	200	20	200,000,000	0.907 ± 0.003	0.8923	0.9197	-16.5051
CEL-ST1	100	50	500,000	0.7923 ± 0.0127	0.7046	0.8592	-3.6309
CEL-ST2	400	50	2,000,000	0.8277 ± 0.013	0.7271	0.8885	-5.778

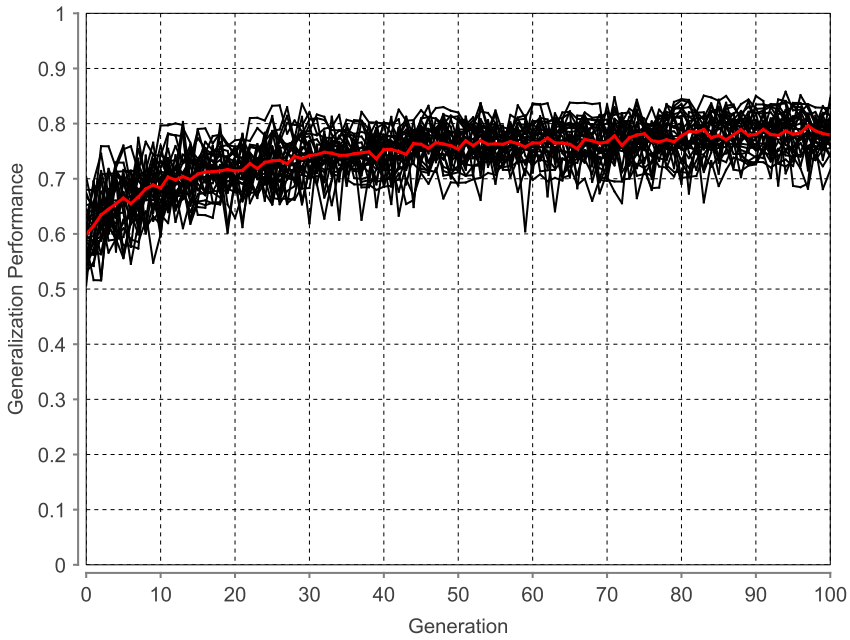


Figure 3.6: Estimated generalization performance of the **subjectively** best-of-generation strategy from the population measured throughout the evolutionary process. **CEL-ST1** algorithm learning to play **black**. Based on **WPC** architecture.

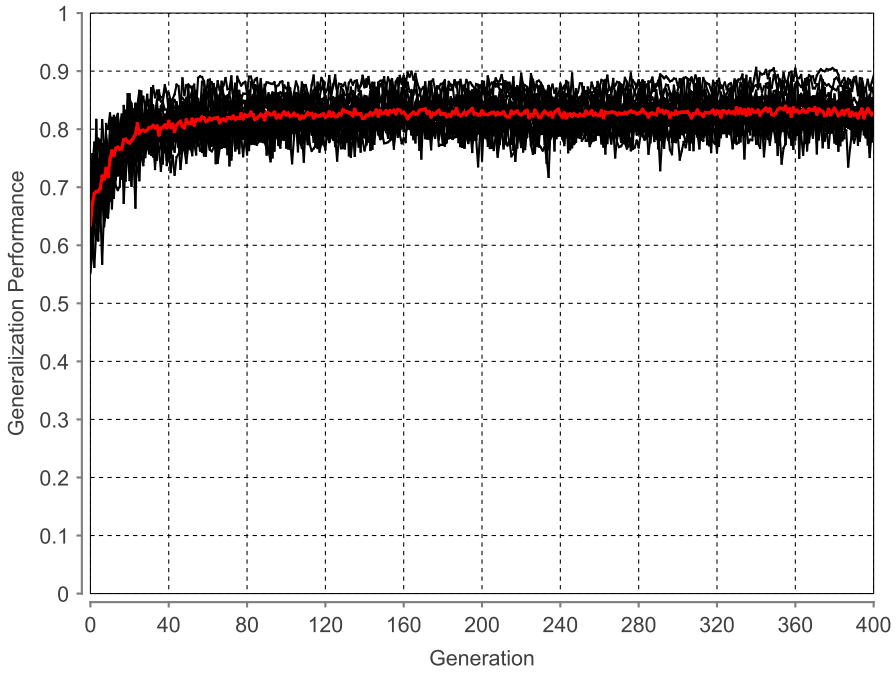


Figure 3.7: Estimated generalization performance of the **subjectively** best-of-generation strategy from the population measured throughout the evolutionary process. **CEL-ST2** algorithm learning to play **black**. Based on **WPC** architecture.

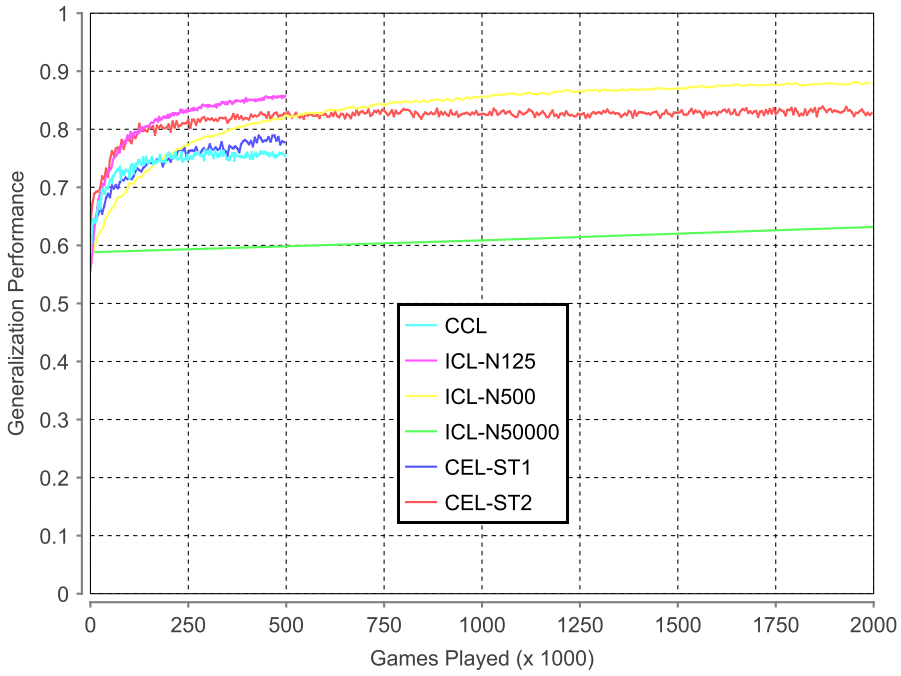


Figure 3.8: Comparison of estimated generalization performance of the **subjectively** best-of-generation strategies measured throughout the evolutionary process of CCL, CEL and ICL. **WPC** architecture. Algorithms learning to play **black**.

3.4 Subjective vs. Objective Selection

3.4.1 Experiment Setup and Objective

Measuring generalization performance involves selecting the best-of-generation individual at the end of every generation throughout the run. Following Chong et al, in the previous experiments we used to this aim subjective best-of-generation selection procedure (see Section 3.1.4). However, simply choosing the individual with the highest fitness may not necessarily yield the best individual due to the relative nature of fitness evaluation in the coevolution. The purpose of this experiment is to investigate whether employing an objective selection improves the overall generalization performance of the studied algorithms. Since any form of objective selection requires additional computational effort, we would like to determine whether such an approach is profitable in the long term. Objective selection method employed in this experiment is based on estimating generalization performance of every candidate solution in the population and subsequently choosing one with the highest value. Thus, prior to determining the best-of-generation individual, we estimate the generalization performance of each individual by playing 50,000 games with strategies randomly drawn from the test set T. We will always refer to the subjective and objective selection in the context of best-of-generation selection, unless stated otherwise.

3.4.2 Results and Discussion

Results of the experiment are shown in Table 3.3. Notice that the objective selection leads to noticeable improvements in the generalization performance in case of every studied method. Objective selection turned out to be especially beneficial for CEL and CCL. The increase in the generalization performance reaches 0.0435 in case of CEL-ST1. However, this is not the case when ICL algorithms are considered since they already use the generalization performance estimates directly as the fitness measure. Consequently, the observed impact of objective selection is much weaker. This is especially visible in case of the ICL-N50000 where the difference between average results obtained using subjective and objective selection is just 0.0009. This can be explained by the fact that both ICL-N50000 and the objective selection procedure use equal sample sizes to estimate the generalization performance.

Observe that, comparing the algorithms using objectively best-of-generation individuals, the gap between ICL algorithms and coevolutionary algorithms, although still non negligible, is further reduced.

Table 3.3: Impact of **objective** and **subjective** selection methods on the generalization performance.

Method	Subjective			Objective		
	Avg	Min	Max	Avg	Min	Max
CCL	0.7518 ± 0.0191	0.6406	0.8253	0.7883 ± 0.0169	0.6641	0.8448
ICL-N125	0.858 ± 0.0061	0.8234	0.8891	0.8678 ± 0.0051	0.8488	0.8927
ICL-N500	0.8801 ± 0.0055	0.8433	0.9089	0.886 ± 0.0054	0.8467	0.9178
ICL-N50000	0.907 ± 0.003	0.8923	0.9197	0.9079 ± 0.0028	0.8946	0.9206
CEL-ST1	0.7923 ± 0.0127	0.7046	0.8592	0.8358 ± 0.0079	0.8054	0.8793
CEL-ST2	0.8277 ± 0.013	0.7271	0.8885	0.8588 ± 0.0077	0.8022	0.9019

3.5 ICL Performance Against a Heuristic Player

3.5.1 Experiment Setup and Objective

The purpose of this experiment is to examine the average performance of ICL on a different quality measure than generalized performance used by Chong et al. This time, we measure the quality of 30 subjectively best-of-run players evolved by each of the studied method in 50,000 games against a heuristic player (see Section 3.1.3). Additionally, to complement the comparison, we decided to study the generalization performance of a heuristic player

3.5.2 Results and Discussion

The results of this experiment are summarized in Table 3.4. For brevity, we note that the average performance refers to the objective quality achieved by best-of-run players in games with a heuristic player, averaged over 30 runs. The analysis of the results leads to the following conclusions. Firstly, it can be observed that the average performance obtained by CEL-ST2 surpasses ICL-N125 by a little margin (recall that the methods share the same computational effort). Notice also that the Hall of Fame archive and the competitive fitness sharing help both CEL-ST1 and CEL-ST2 to achieve a higher level of player when compared to the CCL. Secondly, ICL-N500 which has the same computational effort as CEL-ST2 is still slightly better, although the difference is very small and reaches only 0.0087. Interestingly, despite the fact that ICL-N50000 uses much more computational effort than any other studied method, we observe that beyond the limit set by ICL-N500 there is no significant increase in the average performance since the difference between them is only 0.0016. We note that the results achieved by CEL-ST1 and CEL-ST2 are consistent with those reported in [50].

In terms of the objective quality obtained by best-of-run players, superiority of ICL over CEL is noticeable, especially when the bigger learning samples are used throughout the evolution, as it is in the case of ICL-N500 and ICL-N50000. However, this leads to a significant raise in the computational effort (cf. Table on page 26). We notice that CEL-ST2 is close to ICL-N500 as difference between them is only 0.0087. The same conclusion applies to CEL-ST1 and ICL-N125.

Table 3.4: Average performance of the **subjectively** best-of-run players against a **heuristic** player. Algorithms were learning to play **black**.

Method	Avg	Min	Max
CCL	0.2228 ± 0.0186	0.135	0.3344
CEL-ST1	0.2641 ± 0.0155	0.1839	0.36
CEL-ST2	0.2841 ± 0.0155	0.1939	0.3725
ICL-N125	0.2728 ± 0.0185	0.1677	0.3663
ICL-N500	0.2928 ± 0.0166	0.208	0.3755
ICL-N50000	0.2964 ± 0.0172	0.2103	0.3917

3.5.3 Generalization Performance of a Heuristic Player

Table 3.5 shows the estimated generalization performance of a heuristic player using both 50,000 and 1,000,000 random white player WPC strategies. It is surprising that heuristic player is so weak compared to best WPC players, because in direct (randomized as explained in 3.1.3) game with CTDL+HoF it wins in ca. 55% of times [81]. Similarly, it wins in ca. 55% of times with TDL+CEL+HoF [80]. Thus, we conclude that SWH is not a good player on average but the question is why it does so well against coevolved WPC players. Although interesting, we did try to explain this phenomena through other experiments; it requires further research.

Table 3.5: Estimated generalization performance of a heuristic player. Different sample sizes.

Player	Encoding	50k	1mln
SWH	WPC	0.74456	0.745776

3.6 ICL Performance Against a Random Player

3.6.1 Experiment Setup and Objective

Similarly to the previous experiment, we would like to assess the average performance of the improved coevolutionary learning algorithm using another popular objective quality measure, that is games with a random player. To pursue this goal, we measure the quality of 30 subjectively best-of-run players evolved by each of the studied method in 50,000 games against a random player (see Section 3.1.3).

3.6.2 Results and Discussion

The results of this experiment are summarized in Table 3.6. For brevity, we note that the average performance refers to the subjective quality achieved by best-of-run players in

games with a random player, averaged over 30 runs.

Basing on the results, we draw several conclusions. First, note that the order of methods is the same as in the previous section (cf. Table 3.4). Furthermore, we observe that CEL-ST2 is once again close to ICL-N500 as difference between them is only 0.0146. Significant superiority of the improved coevolutionary learning is manifested only in case of ICL-N50000. However, we stress that this variant of ICL uses substantially larger learning sample throughout the evolution, and consequently employs much more computational effort.

We note that the results achieved by CEL-ST1 and CEL-ST2 are consistent with those reported in [81, 82]. Nevertheless, we observed that regardless of the quality measure, classical coevolutionary learning represented by CCL does not yield satisfactory results. Thus, on the basis of conducted experiments, we agree with Chong et al. that simple coevolutionary learning does not necessarily lead to evolving strategies with increasingly higher generalization performance when a relative fitness measure is used to guide the search. Finally, we emphasize that it is possible to design a much better algorithm than CCL which is comparable to ICL-N125 and ICL-N500 in terms of objective quality obtained by best-of-run players.

Table 3.6: Average performance of the **subjectively** best-of-run players against a **random** player. Algorithms were learning to play **black**.

Method	Avg	Min	Max
CCL	0.7088 ± 0.0153	0.6254	0.7847
CEL-ST1	0.7935 ± 0.0102	0.7335	0.8423
CEL-ST2	0.8216 ± 0.092	0.7524	0.8679
ICL-N125	0.8171 ± 0.0075	0.7745	0.853
ICL-N500	0.8362 ± 0.0066	0.8094	0.8759
ICL-N50000	0.8641 ± 0.0045	0.8442	0.8883

3.7 ICL Performance Against Random n-tuple Strategies

3.7.1 Experiment Setup and Objective

The objective of this experiment is to compare the studied algorithms using another criterion. In this experiment as a quality measure we used an estimate of generalization performance obtained using 50,000 random white player n-tuple strategies. Prior to the forthcoming experiments which employ the n-tuple system in the role of player's architecture, we would to see how the WPC-based and n-tuple-based quality measures correlate. More details on measuring objective quality of a player by means of random sampling can be found in Section 3.1.3.

3.7.2 Results and Discussion

The objective quality achieved by best-of-run players in games with a random n-tuple player, averaged over 30 runs, is presented in Table 3.7. Not surprisingly, the results of this experiment repeat the pattern observed in the previous experiments. We observe that the more computational effort, the better results obtained by ICL, but even when the computational efforts are the same, ICL still beats CEL, as in the case of CEL-ST2 and ICL-N500. More interestingly, our results imply that it is fairly easier to win with a random n-tuple strategy than with a random WPC strategy (cf. Table 3.2). The key to properly understand these results lies in the probabilistic nature of the adopted measure. We hypothesize that the probability of sampling a well performing strategy is smaller in case of the n-tuple architecture due to its increased complexity when compared to the WPC. Consequently, when sampling n-tuple strategies, their average performance is accordingly lower. Upon this experiment, we hypothesize that random WPC strategies pose a greater challenge, thus guiding coevolutionary search on the basis of games against them might yield better results. We will use these premises in the subsequent experiment.

Table 3.7: Average performance of the **subjectively** best-of-generation individuals against a set of randomly generated **n-tuple** strategy. Algorithms were learning to play **black**.

Method	Avg	Min	Max
CCL	0.75 ± 0.0172	0.6545	0.8197
CEL-ST1	0.8148 ± 0.013	0.7219	0.8638
CEL-ST2	0.8589 ± 0.041	0.8336	0.8918
ICL-N125	0.8514 ± 0.0063	0.8224	0.8839
ICL-N500	0.8697 ± 0.0054	0.8455	0.9063
ICL-N50000	0.8955 ± 0.0036	0.8759	0.9133

3.8 ICL vs. CEL on the n-tuple Architecture

3.8.1 Experiment Setup and Objective

In Section 3.2.1 we confirmed the results obtained by Chong et al. and we admitted that guiding the coevolutionary search on the basis of games against a sample of random opponents employed by ICL has a great potential when applied to the problem of Othello. Nevertheless, we would like to point out that the previous research in this field is limited to the study based on the WPC representation [10]. To further investigate the performance of ICL, we use considerably more complex, non-linear architecture to encode player's strategy — a symmetric n-tuple network discussed in Section 2.4.3.2. To provide a fair comparison, we decided to assume the same settings which were used in our previous experiment described in Section 3.2.1. We consider three algorithms: CEL-ST2 and two versions of ICL: ICL-N500 and ICL-N125. ICL-N50000 was not included in our comparison since with n-tuple architecture it demands too much computational power. As one generation requires over 1 hour of computation, the whole experiment consisting of

30 runs of 200 generations would eventually take at around 250 days. Provided we could use computational laboratory equipped with 15 4-core 3.09GHz computers, we would still need over 16 days to finish the computation. Unfortunately, for the purpose of this study, we could not afford such expenses. Thus, we will focus on the ICL with smaller samples.

Our ultimate goal is to find a coevolutionary algorithm of better performance than ICL, thus in this experiment we do not consider CEL-ST1, which has been found worse than CEL-ST2 in the previous experiments (see Fig. 3.8). Instead, we consider a version of CEL-ST2 using the population size of 50 and 100.

ICL maintains 50 individuals in the population. Additionally, each experiment was held twice to test the performance when learning to play black only and black and white at the same time. In case of learning to play as a black player, as a quality measure we used an estimate of generalization performance obtained using 50,000 random white player WPC strategies. Conversely, when learning to play both black and white, we estimate the generalization performance on the basis of 25,000 games with random black and 25,000 games with random white players. We present obtained results for both subjective and objective selection of the best-of-generation individual.

To provide a fair comparison we decided to stop all runs when the number of games played reaches 2,000,000. In the following, we present the obtained results.

3.8.2 Results When Learning to Play Black

Figures 3.10, 3.11 and 3.12 present the generalization performance of the subjectively best-of-generation strategy for all generations of the evolutionary process. The results were averaged over 30 independent runs obtained by CEL-ST-2, ICL-N125 and ICL-N500 respectively.

The results of the experiment are summarized in Tables 3.8 and 3.9. Both CEL and ICL gained on the generalization performance after we applied the n-tuple system to the algorithms. Comparing these results to the ones obtained in previous experiments and presented in Table 3.2 implies that the gain reaches over 12% in case of ICL-N500 and almost 18% in case of CEL-ST2 maintaining 100 individuals in the population.

Based on the experiment, we draw several conclusions. First, we observe that after leveling computational efforts CEL-ST2 takes the lead and is slightly better than any other algorithm. Second, note that reducing number of generations for ICL-N500 adversely affected the generalization performance which is slightly worse than in case of ICL-N125. Figure 3.9 compares subjective average generalization performance of three considered methods on a single axis. Notice that in the beginning, the generalization performance of CEL-ST2 is slightly higher than the performance of ICL. However, in the long run ICL-N500 catches up and performs similarly to the CEL-ST2. Another important remark is that CEL-ST2 learns more rapidly than ICL which can be observed on the plot. This is especially visible when comparing CEL-ST2 to ICL-N125.

Table 3.8: Summary of results for ICL and CEL based upon **n-tuple** architecture. Equal computational efforts. Learning to play as a **black** player. **Subjectively** best-of-last-generation players.

Method	#Gen.	PopSize	Comp. Effort	Avg	Max	Min
CEL-ST2	400	50	2,000,000	0.9117 ± 0.0115	0.9633	0.8712
CEL-ST2	133	100	≈2,000,000	0.9653 ± 0.0094	0.9884	0.9442
ICL-N125	320	50	2,000,000	0.9252 ± 0.0054	0.9754	0.881
ICL-N500	80	50	2,000,000	0.9651 ± 0.0053	0.9875	0.9309

Table 3.9: Summary of results for ICL and CEL based on **n-tuple** architecture. Equal computational efforts. Learning to play as a **black** player. **Objectively** best-of-last-generation players.

Method	#Gen.	PopSize	Comp. Effort	Avg	Max	Min
CEL-ST2	400	50	2,000,000	0.9316 ± 0.099	0.9776	0.8934
CEL-ST2	133	100	≈2,000,000	0.9813 ± 0.0075	0.9927	0.9537
ICL-N125	320	50	2,000,000	0.9294 ± 0.0052	0.9769	0.9091
ICL-N500	80	50	2,000,000	0.969 ± 0.0049	0.9917	0.9422

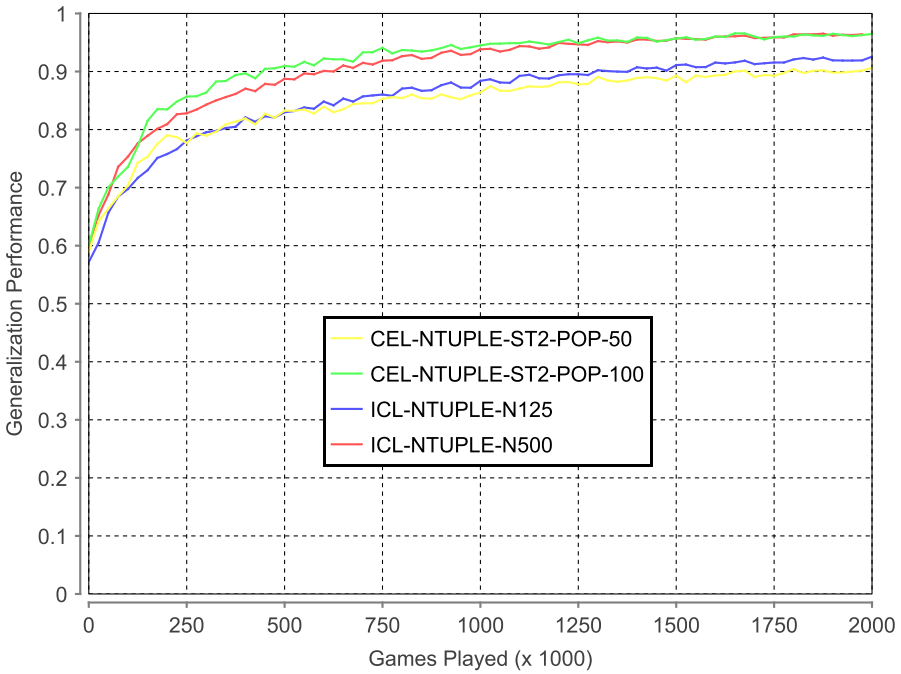


Figure 3.9: Comparison of estimated generalization performance of the **subjectively** best-of-generation strategies measured throughout the evolutionary process for different **n-tuple** methods. **A**lgorithms learning to play **black**.

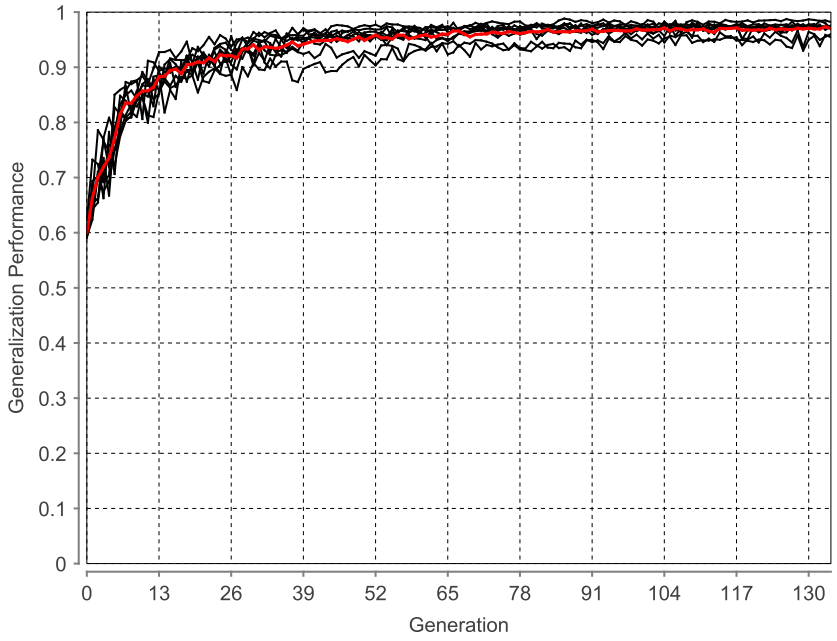


Figure 3.10: Estimated generalization performance of the best-of-generation strategy from the population measured throughout the evolutionary process. **CEL-ST2** algorithm **with population size 100** learning to play **black**. Based on **n-tuple** architecture.

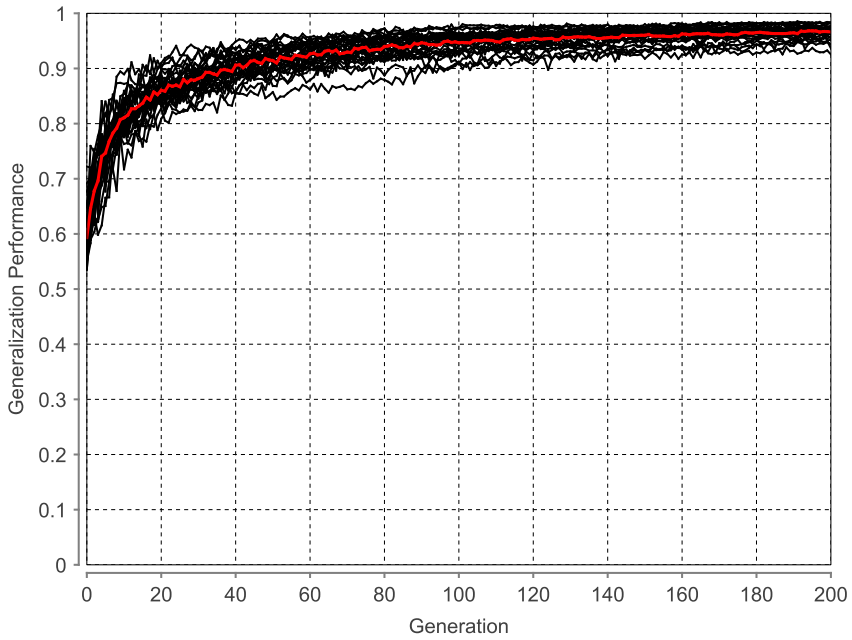


Figure 3.11: Estimated generalization performance of the best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N125** algorithm learning to play **black**. Based on **n-tuple** architecture.

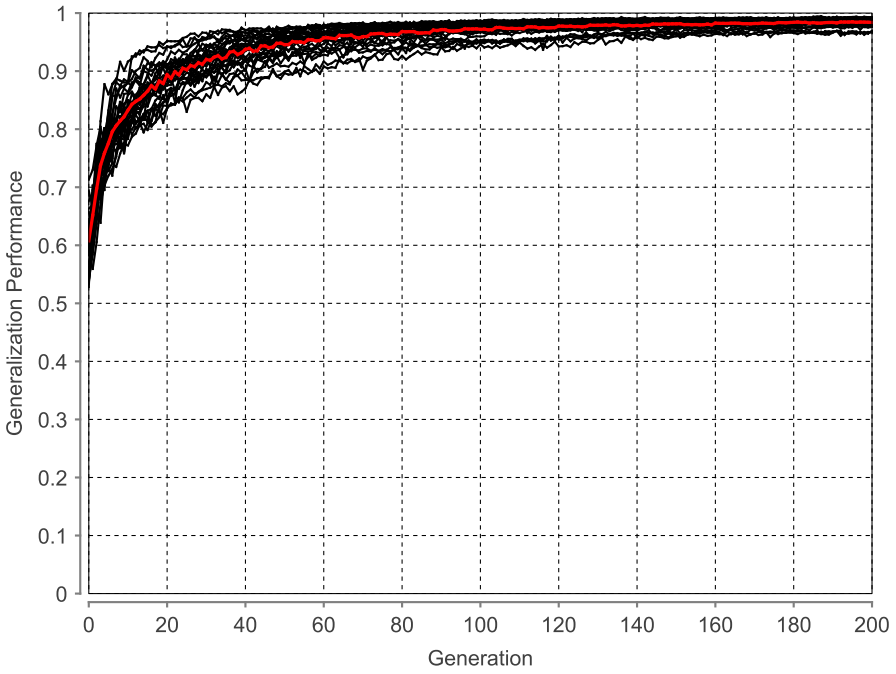


Figure 3.12: Estimated generalization performance of the best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N500** algorithm learning to play **black**. Based on **n-tuple** architecture.

3.8.3 Results When Learning to Play Black and White

Figures 3.14, 3.15 and 3.16 present the generalization performance of the top performing strategy of the population across the evolutionary process for 30 independent runs obtained by CEL-ST-2, ICL-N125 and ICL-N500 respectively.

The results of the experiment are summarized in Tables 3.10 and 3.11. The obtained results confirm that leveling computational efforts works in favor of CEL-ST2. Even though it still outperforms both ICL-N125 and ICL-N500, it does so by a little margin. Interestingly, we observe even bigger gap in the average generalization performance between ICL-N125 and ICL-N500 in favor of the former. However, Fig. 3.13 suggests that ICL-N500 learns the fastest of all. CEL-ST2 is right behind ICL-N500 and well ahead of ICL-N125.

Moreover, the experimental results clearly indicate that learning to play as both black and white player is is significantly more difficult than learning to play only as black. As a result, the generalization performance has dropped for each of the studied algorithms by approximately 10% when compared to the results from Table 3.8. ICL is also characterized by noticeably increased variance as observed on Figs. 3.15-3.16 when compared to Figs. 3.11 and 3.12.

Table 3.10: Summary of results for ICL and CEL based on **n-tuple** architecture. Equal computational efforts. Learning to play as a **black and white** player. **Subjectively** best-of-last-generation players.

Method	#Gen.	PopSize	Comp. Effort	Avg	Max	Min
CEL-ST2	400	50	2,000,000	0.7446 ± 0.0132	0.8298	0.7089
CEL-ST2	133	100	≈2,000,000	0.8406 ± 0.0107	0.9013	0.8018
ICL-N125	320	50	2,000,000	0.8152 ± 0.0126	0.9371	0.8241
ICL-N500	80	50	2,000,000	0.8643 ± 0.0151	0.9055	0.7924

Table 3.11: Summary of results for ICL and CEL based on **n-tuple** architecture. Equal computational efforts. Learning to play as a **black and white** player. **Objectively** best-of-last-generation players.

Method	#Gen.	PopSize	Comp. Effort	Avg	Max	Min
CEL-ST2	400	50	2,000,000	0.7648 ± 0.0101	0.8332	0.7116
CEL-ST2	133	100	≈2,000,000	0.8655 ± 0.0096	0.9175	0.8237
ICL-N125	320	50	2,000,000	0.8189 ± 0.0119	0.9406	0.8318
ICL-N500	80	50	2,000,000	0.8729 ± 0.0153	0.9136	0.8039

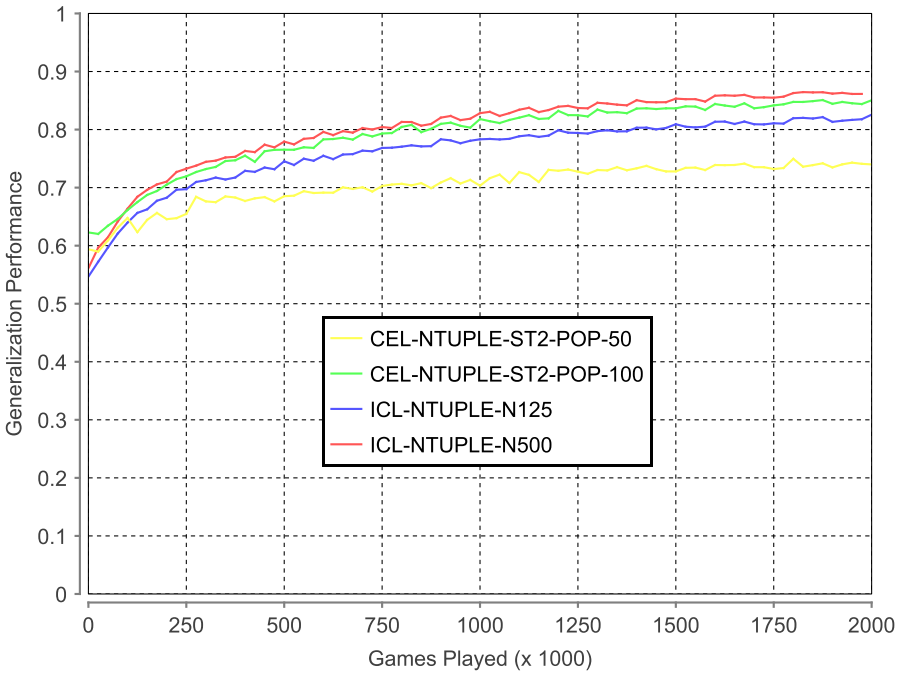


Figure 3.13: Comparison of estimated generalization performance of the subjectively best-of-generation strategies measured throughout the evolutionary process for different **n-tuple** methods. **A**lgorithms learning to play **black and white**.

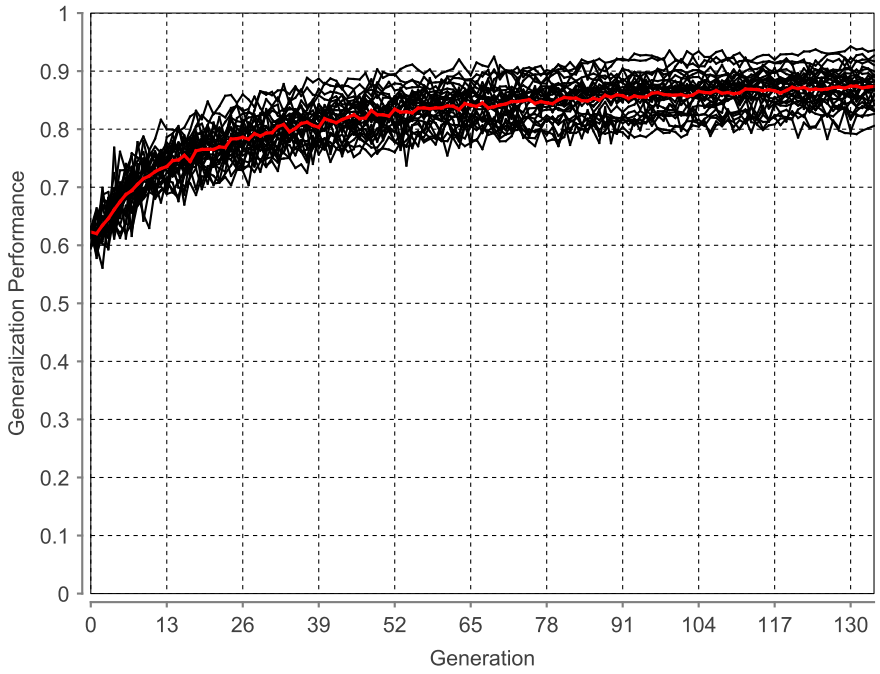


Figure 3.14: Estimated generalization performance of the best-of-generation strategy from the population measured throughout the evolutionary process. **CEL-ST2** algorithm **with population size 100** learning to play **black and white**. Based on **n-tuple** architecture.

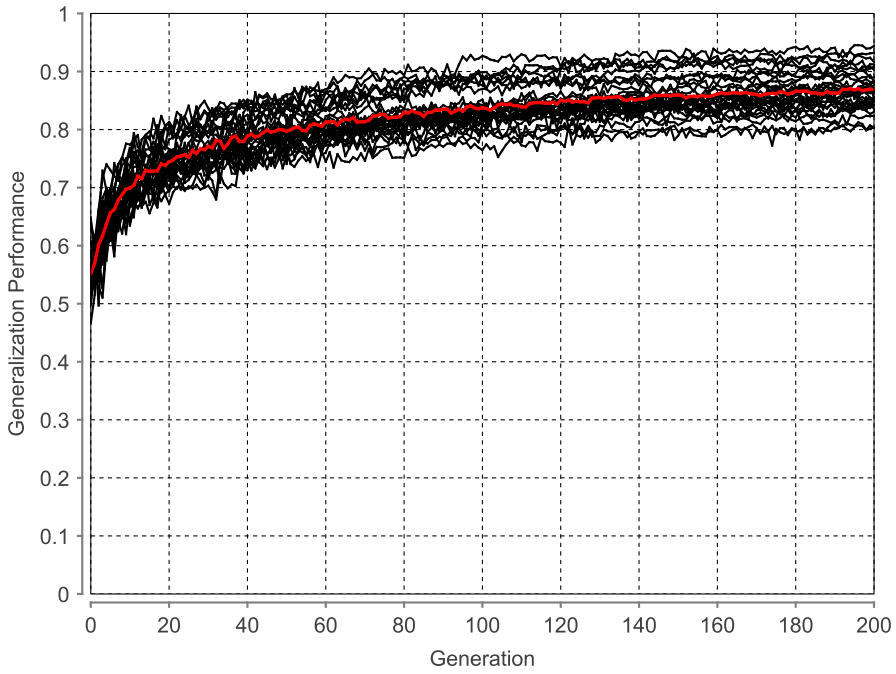


Figure 3.15: Estimated generalization performance of the best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N125** algorithm learning to play **black and white**. Based on **n-tuple** architecture.

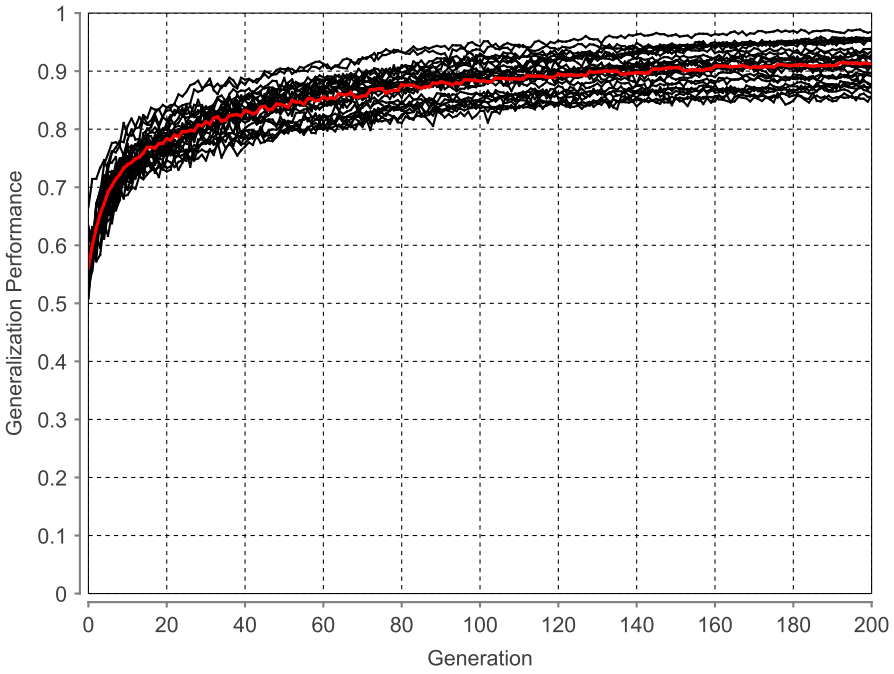


Figure 3.16: Estimated generalization performance of the best-of-generation strategy from the population measured throughout the evolutionary process. **ICL-N500** algorithm learning to play **black and white**. Based on **n-tuple** architecture.

3.8.4 Discussion

On the basis of performed experiments we conclude that combining the studied algorithms with the n-tuple system was indeed a wise step. The overall performance of evolved players scales very well and comparing to previous experiments, the obtained results are significantly better. We hypothesize that limited dimensionality of the search space offered by the WPC might in certain cases contribute to some of well-known coevolutionary pathologies, namely forgetting [30] and over-specialization [84]. Since coevolution is extremely prone to any of these undesired phenomenons, overall performance deteriorates whenever any of them occur. On the other hand, based on this experiment we observe that the studied algorithms effectively utilize the possibilities offered by non-linear n-tuple networks. The objective selection yields improved candidate solutions, especially in case of the coevolutionary learning. This phenomenon was previously observed in Section 3.4.2 and is explained therein. Interestingly, we observe substantially better performance when a bigger population is used. The exact reason why this happens remains so far unknown, however we hypothesize that due to the enhanced player's architecture, the number of certain niches and traits among individuals grows proportionally to the search space. In such case, bigger genetic diversity delivered by the considerably greater population would certainly be favorable and could result in the performance improvement. On the other hand, we suspect that the increased number of games played between the individuals could also positively influence the final results. Further investigation of this issue is beyond the scope of this work, however explaining this phenomenon might be crucial to further improve the proposed algorithms.

Our results show that it is possible to achieve performance superior to ICL using a coevolutionary algorithm. Especially in the early stage of evolution, we observe that CEL is superior to ICL. We suspect, that due to the increased dimensionality of the n-tuple search space, random sampling may not work as intended, ultimately leading to higher requirements in terms of learning time. Conversely, as observed in our previous experiments, in case of the simpler WPC architecture, the coevolutionary search converges to the final solution much faster. Our research has also shown that regardless of the architecture, it is easier to learn playing only one color. Learning to play both black and white is indeed significantly more demanding which is reflected in the inferior results. Note that when the objective selection is used, coevolutionary algorithms are capable of finding not only better solutions but also reduce their overall variance.

Through our experiments we definitely learned that there is no explicit answer to the question of which approach is better. In reality, crucial factor which must be considered before employing any of these procedures is a tradeoff between available time for learning and the performance one is aiming for. We realize that in the long term, particularly when the greater computational effort is admissible, ICL might provide better solutions. On the other hand, when shorter computation time is vital, the classical coevolutionary learning with an archive and a diversity maintenance method seems to be a better choice.

3.9 Generalization Performance of External Methods

3.9.1 Experiment Setup and Objective

In this experiment we analyze and compare our results with results obtained in other studies on the generalization performance criterion. Discussed methods originate from [50, 81, 10] and can be divided into two groups with respect to the player's architecture. Thus, to provide a fair comparison, we consider WPC and n-tuple based methods independently. For the purpose of this experiment, we obtained 30 individuals evolved by each of the examined methods. However, due to the high computational cost associated with processing n-tuples, the number of individuals was reduced to 24 for the methods based on the n-tuples architecture. To obtain the average performance we estimated the generalization performance of each individual and then aggregated the results into a single value. We decided to execute the experiment twice, each time engaging different sample size to estimate the generalization performance. Therefore, following our previous experiments, we begin with a sample size of 50,000, while in the second run we set the sample size to 1,000,000 which is expected to provide even better estimation accuracy. Such a comparison allows us to review recently developed algorithms in the context of our study and draw conclusions about the improved coevolutionary learning in a broader sense.

3.9.2 Results and Discussion

Clearly, most of the methods cannot be directly compared due to the different computational efforts. However, we observe that CEL+HoF from [81] is quite near ICL-N500 as the

difference between them is only 0,0265. Furthermore, we noticed the order of CEL+HoF, CTDL+HoF, CTDL and CEL imposed by the generalization performance is consistent with the order obtained in [81] (Fig. 2), where methods were compared on the basis of games with a random player. These results suggest, that both random player and a random strategy are similarly challenging. We observed that when the generalization performance criterion is used to compare various methods based on the WPC architecture, hardly any algorithm is able to compete with ICL. Interestingly, obtained results show that 50,000 games is enough to accurately approximate the generalization performance as we observe almost no difference in estimation when sample size is 1,000,000.

As discussed in Section 3.8, using the n-tuple network to represent a player's strategy allows to significantly improve learning algorithms and evolve considerably better players. The positive influence of such a setup is also reflected in this experiment, as we study methods originating from [50]. All of the examined algorithms perform extremely well and are almost unbeatable by any of the randomly sampled strategies. Having said that, we note that even 1,000,000 games does not allow to clearly differentiate those methods. We suspect, that in a direct comparison odds of a WPC player winning to a n-tuple player are very low. We will further investigate this issue in Section 3.12.2.

Table 3.12: Comparison of various methods from different experiments on the generalization performance criterion. Subjective selection. Sample size $N = 50,000$.

Player	Source	Encoding	Effort	Avg	Max	Min
CCL	Chong[10]	WPC	500,000	0.7518 ± 0.0191	0.8253	0.6406
ICL-N125	Chong	WPC	500,000	0.858 ± 0.0061	0.8891	0.8234
ICL-N500	Chong	WPC	2,000,000	0.8801 ± 0.0055	0.9089	0.8433
ICL-N500	Chong	WPC	10,000,000	0.897 ± 0.0038	0.9093	0.8747
ICL-N50000	Chong	WPC	200,000,000	0.907 ± 0.003	0.9197	0.8923
CEL-ST1	PL	WPC	500,000	0.7923 ± 0.0127	0.8592	0.7046
CEL-ST2	PL	WPC	2,000,000	0.8283 ± 0.0112	0.8744	0.7457
CEL+HoF	CC[81]	WPC	10,000,000	0.8705 ± 0.0022	0.8874	0.848
CTDL	CC	WPC	10,000,000	0.8503 ± 0.0034	0.8751	0.8184
CTDL+HoF	CC	WPC	10,000,000	0.866 ± 0.0026	0.8909	0.8404
CEL	CC	WPC	10,000,000	0.7799 ± 0.0079	0.8484	0.6637
CTDLpx	TCIAG[50]	NTuples	2,000,000	0.9906 ± 0.0015	0.9957	0.9814
ETDLsxmt	TCIAG	NTuples	2,000,000	0.9895 ± 0.0027	0.9959	0.9724
CTDLpxmt	TCIAG	NTuples	2,000,000	0.9868 ± 0.0034	0.9971	0.9633
CTDLsxmt+HoF	TCIAG	NTuples	2,000,000	0.9888 ± 0.0018	0.9965	0.9813
CTDLsxmt	TCIAG	NTuples	2,000,000	0.988 ± 0.0032	0.9957	0.9569

Table 3.13: Comparison of various methods from different experiments on the generalization performance criterion. Subjective selection. Sample size $N = 1,000,000$.

Player	Source	Encoding	Effort	Avg	Max	Min
CCL	Chong	WPC	500,000	0.7533 ± 0.0186	0.831	0.6394
ICL-N125	Chong	WPC	500,000	0.8573 ± 0.0059	0.8904	0.8215
ICL-N500	Chong	WPC	2,000,000	0.88 ± 0.0051	0.9079	0.8425
ICL-N50000	Chong	WPC	200,000,000	0.9068 ± 0.0027	0.9192	0.8939
CEL-ST1	PL	WPC	500,000	0.8006 ± 0.0119	0.8511	0.7197
CEL+HoF	CC	WPC	10,000,000	0.8717 ± 0.0029	0.8847	0.8566
CTDL	CC	WPC	10,000,000	0.8519 ± 0.0044	0.8757	0.824
CTDL+HoF	CC	WPC	10,000,000	0.8668 ± 0.0044	0.8924	0.839
CEL	CC	WPC	10,000,000	0.7827 ± 0.0088	0.8173	0.7226
CTDLpx	TCIAG	NTuples	2,000,000	0.9906 ± 0.0015	0.9956	0.9816
ETDLsxmt	TCIAG	NTuples	2,000,000	0.9894 ± 0.0026	0.9957	0.9729
CTDLpxmt	TCIAG	NTuples	2,000,000	0.9867 ± 0.0034	0.997	0.9637
CTDLsxmt+HoF	TCIAG	NTuples	2,000,000	0.989 ± 0.0018	0.9967	0.9817
CTDLsxmt	TCIAG	NTuples	2,000,000	0.988 ± 0.0031	0.9958	0.9582

3.10 Generalization Performance of Othello League Players

3.10.1 Experiment Setup and Objective

An overview of different Othello players and their estimated performance is provided by the Othello Position Evaluation Function League [53]. The biggest advantage of the league is diversity of the participating players. They have been submitted to the league by different researches from around the world and often implement various approaches and represent a wide range of behaviors. The on-line trial league shows the performance over 100 randomized games against the SWH player with probability of a random move $\epsilon = 0.1$. The primary objective of this experiment is to investigate the generalization performance of the best league players. Among several hundreds strategies submitted to the league by anonymous contestants we selected the top ten strategies. To achieve this, players were initially divided into two groups with regard to the architecture. Subsequently, the best players from each group were selected according to the league score, allowing to check whether league score would be confirmed by the generalization performance. Following our previous experiments, we estimated the generalization performance using 50,000 and 1,000,000 random white player WPC strategies.

3.10.2 Results and Discussion

Based on results in Table 3.14 we conclude that the n-tuple players are indeed excellently performing players. Note that the generalization performance of the league players is almost exactly the same as the performance obtained by the best evolved n-tuple players in [50, 80]. Interestingly, we observed that the current league champion denoted by epTDLmpx_12x6 who is a ETDL-evolved player proposed in [50] copes with randomly generated strategies a bit worse when compared to other league players. This phenomenon can be easily explained since ETDL uses the static evaluation function based on the SWH player, consequently optimizing the player's behaviour against this specific opponent [50].

Regarding the best WPC league players shown in Table 3.15, we conclude that they are still no match for ICL-N50000 since their average generalization performance is noticeably worse. Note that certain league players achieved result similar to ICL-N500.

Table 3.14: Estimated generalization performance of top ten Othello League **n-tuple** players.

Player Name	Network Size	50k	1mln
epTDLmpx_12x6	12×6	0.9881	0.988291
prb_nt30_001	30×6	0.9967	0.996792
prb_nt15_001	15×6	0.99464	0.994885
epTDLxover	12×6	0.99526	0.994939
t15x6x8	15×6	0.9699	0.969055
SelfPlay15	12×6	0.98224	0.982423
tz278_2	278×2	0.97536	0.975436
Nash70	12×6	0.98348	0.983696
x30x6x8	30×6	0.9457	0.946923
pruned-pairs-56t	56×2	0.96718	0.967368

Table 3.15: Estimated generalization performance of top ten Othello League **WPC** players.

Player Name	50k	1mln
r0nan-1227736739614	0.82046	0.818697
bssWPC_es9e-1240573442753	0.88818	0.887152
r0nan-1227736742208	0.8184	0.819345
r0nan-1227737008988	0.85288	0.855098
tog1-1237736078509	0.88272	0.883625
r0nan-1227736322289	0.79866	0.798103
r0nan-1227737932122	0.84206	0.842713
srs8-1237733668414	0.88768	0.885233
asd-1227195030580	0.74394	0.74642
r0nan-1227738106121	0.83238	0.833234

3.11 Tournament Between ICL and Other Methods

3.11.0.1 Experiment Setup and Objective

In order to measure the relative performance between methods originating from [81] and [10], we employ a round-robin tournament involving diverse set of best-of-generation individuals obtained from the studied learning algorithms. The best-of-generation strategies which were subject to the generalization performance assessment in the previous experiments are now gathered into teams, each representing method they originate from. Such an approach provides a better insight into the overall performance of a particular method as none of the quality measures exploited before can be anticipated to represent diversification and a rich repertoire of behaviours typical for moderately strong Othello players. Regarding the tournament organization, we rely on a common formula, where each team member plays against all members from the other team and the final score is the overall sum of points obtained by all members of the team. Finally, to clarify the presentation we apply the tournament to two groups of methods, starting with the WPC-based and ICL and ending with the encounter of the n-tuple and ICL methods. The former approach results in 30×30 games, while the latter limits the number of games to 24×24 due to the heavy computational effort associated with processing n-tuples.

3.11.1 Results and Discussion

The results of the experiment are show in Table 3.16. Methods in rows originate from [81]. In each cell of the table we can find the number of games won by the team in the row and the and computed probability of winning in percents. Surprisingly, ICL methods proposed in [10] are outperformed by most of the algorithms in the rows. Explaining this phenomenon is not a trivial task, however we may hypothesize that due to the fact that in this tournament an individual faces exclusively well-performing strategies, what truly matters is the ability to play against a strong opponent and not against a random one. The individuals trained by ICL do not have a chance to compete with other individuals from the same population and learn from such an experience. Ultimately, this leads to optimization towards playing with a specific group of random opponents which could have contributed to such results of the experiment. Clearly, guiding optimization process on the basis of games played against a sample of random opponents, although raises overall performance, it may not necessarily imply increased behavioral diversity of the individual.

On the basis of these experiments, we conclude that the improved coevolutionary learning may not be necessarily as robust as claimed in [10]. Additionally, these results give us some premises, that coevolved players are capable of beating moderately strong Othello players, as opposed to players who have been obtained differently.

Table 3.16: Round-robin tournament between **WPC**-based and ICL methods.

	CCL	CEL-ST1	ICL-N125	ICL-N500	ICL-N50000
CEL	625 / 900 = 69%	404 / 900 = 44%	557 / 900 = 61%	516 / 900 = 57%	509 / 900 = 56%
CEL+HoF	741 / 900 = 82%	700 / 900 = 77%	753 / 900 = 83%	713 / 900 = 79%	721 / 900 = 80%
CTDL	772 / 900 = 85%	676 / 900 = 75%	700 / 900 = 77%	699 / 900 = 77%	716 / 900 = 79%
CTDL+HoF	775 / 900 = 86%	714 / 900 = 79%	763 / 900 = 84%	745 / 900 = 82%	739 / 900 = 82%

Table 3.17: Round-robin tournament between **n-tuple**-based and ICL methods.

	CCL	CEL-ST1	ICL-N125	ICL-N500	ICL-N50000
CTDLpx	570 / 576 = 98%	563 / 576 = 97%	573 / 576 = 99%	571 / 576 = 99%	565 / 576 = 98%
CTDLpxmt	569 / 576 = 98%	559 / 576 = 97%	568 / 576 = 98%	564 / 576 = 97%	565 / 576 = 98%
CTDLsxmt	569 / 576 = 98%	557 / 576 = 96%	558 / 576 = 96%	565 / 576 = 98%	569 / 576 = 98%
CTDLsxmt+HoF	568 / 576 = 98%	557 / 576 = 96%	569 / 576 = 98%	571 / 576 = 99%	565 / 576 = 98%
ETDLsxmt	571 / 576 = 99%	556 / 576 = 96%	566 / 576 = 98%	565 / 576 = 98%	567 / 576 = 98%

3.12 Two-individual test

3.12.1 Experiment Setup and Objective

In our final experiment, we employ the statistical framework proposed in [10] to compare relative performance of the best strategies obtained by each of the studied methods. For the purpose of this experiment we assume, that both strategies to be compared compete against the same set of N randomly generated strategies $T_N = \{t_1, t_2, \dots t_n\}$. Statistical test regarding the relation between the Generalization Performances of the analysed strategies can be performed using paired t-test.

Let us denote the game outcome of strategy $s \in S$ playing against strategy $t \in T$ by $G(s,t)$.

We begin with computing a series of performance differences on T_N:

$$D(n) = G(s_1, t_n) - G(s_2, t_n) \quad n = 1, 2, \dots, N,$$

where $s_1, s_2 \in S$.

Let us denote the average performance differences by

$$\hat{D}(T_N) = \frac{1}{N} \sum_{n=1}^{N} D(n).$$

In order to test whether strategy s_1 outperforms strategy s_2 at significance level α we

need to check if $Z_i(T_N) \geq z_\alpha$, where

$$Z_i(T_N) = \frac{\hat{D}(T_N)}{\sqrt{\frac{\sum_{n=1}^{N}\left(D(n)-\hat{D}(T_N)\right)^2}{N(N-1)}}}.$$

In order to select a representative strategy for each method, we analyzed the best-of-generation players from all 30 runs. To avoid extensive complication, we decided to compare their relative fitness values, obtained throughout the course of evolution, and choose the player with the highest one. Furthermore, we assume the size of sample $N = 50,000$ and significance level $\alpha = 0.05$.

3.12.2 Results and Discussion

The results of the experiment are presented in table 3.18. Methods in rows originate from [81]. Each cell of the table contains a t-value obtained by testing the hypothesis that a strategy obtained by method in the column is better than the strategy obtained by the method in the row. Bolded values mean statistical significance at level α. Not surprisingly, ICL-N50000 has proved to evolve the best strategy among WPC methods. Nevertheless, although the average results of ICL-N500 and ICL-N125 are higher than those of CEL+HoF methods this difference is too low to be statistically important.

Table 3.18: t-test values for a relative performance between best individuals of each method. **Bolded** values mean statistical significance at level $\alpha = 0.05$.

	ICL-N125	ICL-N500	ICL-N50000
CEL	**54.77**	**56.49**	**76.61**
CEL+HoF	1.87	1.48	**23.11**
CTDL	**5.24**	**6.56**	**27.34**
CTDL+HoF	**3.27**	**2.64**	**21.95**

Chapter 4

Discussion, Conclusions and Future Work

4.1 Conclusions and Discussion

In this paper we extended the work of Chong et al. [10] and focused on the particular test-based problem: acquisition of Othello position evaluation function.

On the basis of conducted experiments, we admit that guiding coevolutionary search on the basis of games against a sample of random opponents employed by Chong's ICL has a great potential when applied to the problem of Othello. Nevertheless, we were able to point some areas in which ICL superiority over coevolutionary-based methods no longer holds. These areas include games with both heuristic player (Section 3.5.2) and a random player (Section 3.6.2). We showed that in some cases the modern approach to the coevolutionary learning is capable of obtaining slightly better candidate solutions than the ICL (Section 3.1.1.3).

The experimental results (Section 3.3) also demonstrate the positive effect of enhancing a coevolutionary algorithm with the Hall of Fame archive and the competitive fitness sharing on the generalization performance. Players evolved in such a manner achieve a higher level of play when compared to those obtained using simpler coevolutionary methods (Section 3.2). Additionally, we have also demonstrated that employing an objective best-of-generation selection of individuals leads to noticeable improvements in the generalization performance in case of every studied method (Section 3.4). This stays in contrast with subjective best-of-generation selection which may not necessarily yield the best individual due to the relative nature of fitness evaluation in the coevolution. Moreover, using the objective selection of best-of-run individual helps more coevolutionary methods than ICL methods. This should be kept in mind when comparing ICL with coevolutionary methods.

In this study we also investigated in detail the impact of two different player architectures, namely WPC and n-tuple network, on the generalization performance. While experimental results proved that using more complex, non-linear architecture in a form of n-tuple network is beneficial in terms of the generalization performance, they also showed that random sampling employed by ICL is not as efficient as in case of a simpler WPC architecture compared to coevolutionary methods. As a result, the coevolutionary algorithm

enhanced with the Hall of Fame archive and the competitive fitness sharing (CEL-ST2) turned out to perform better that ICL when using the same computational effort (Section 3.8). Especially in the early stage of evolution, we observed that CEL is more robust when compared to ICL.

Moreover, we investigated the relative performance between all best-of-run individuals obtained by ICL and different coevolutionary methods by performing a round-robin tournament (Section 3.11). The experimental results showed that ICL methods are outperformed by those based on coevolutionary learning which implies that in some situations ICL exhibits less behavioral diversity comparing to CEL.

We considered more algorithms and settings than the previous work on the subject [10], but more analysis is still required. Although, we have finally found a coevolutionary algorithm of better performance than ICL, we never tried to play with parameters of ICL, which could also improve its performance. Thus, despite some results in favor of coevolution, we are far to state that coevolutionary algorithms are generally more promising that random sampling-based methods.

For the purpose of this thesis we developed a software framework built upon cECJ [79] and ECJ [56]. It was designed to allow flexible experiment definition as well as easy deployment and collection of results. As the software integrates with both cECJ and ECJ, it may prove to be especially useful for users who are already familiar with those systems.

4.2 Future work

During our research, many interesting hypotheses arose, some of which have been already presented and shortly discussed earlier in this work. Let us point out a few possible directions of future work:

- As discussed in Section 3.4.2 selection of the appropriate individual which becomes the outcome of either specific generation or, more generally, of the evolutionary run is not a trivial task. This issue has been previously addressed in [42] and we would like to continue this research and propose some novel methods solving this problem.
- Adaptation of classical local search metaheuristics such as Simulated Annealing, Tabu Search or Iterated Local Search to solve test-based problems seems to constitute well-motivated area of research since it has been shown that $(1 + 1)$ coevolutionary algorithm is not worse than coevolutionary learning for certain problems [63].
- Using coevolutionary learning with two-population coevolution in which candidate solutions and tests are bred separately would allow us to not only use more advanced archives such as LAPCA or IPCA (see Section 2.3.3), but also to verify the impact of using more evaluations to assess subjective fitness of an individual. This idea seems particularly appealing since it would allow to separately tune the population sizes of candidate solutions and tests.
- During our research an interesting hypothesis has emerged according to which coevolved players compete on the satisfactory level with moderately strong Othello players whereas they sometimes lose to much worse players. A premise supporting

this claim could be find in Section 3.11. To verify this, more analysis is required.

- As noted in Section 2.4.3.1, a strategy represented by WPC can be easily understood by inspecting the weight values. Therefore, we analyzed strategies evolved by ICL and noticed that, contrary to our expectations, they were asymmetric. We were not able to determine why such an asymmetry occurred, nor what are implications of such representation. This issue requires further research.

Bibliography

[1] J.A. Anderson and J. Davis. *An introduction to neural networks*, volume 1. MIT Press, 1995.

[2] P.J. Angeline and J.B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270. San Mateo, California, 1993.

[3] D. Ashlock and B. Powers. The effect of tag recognition on non-local adaptation. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 2045–2051. IEEE, 2004.

[4] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. *The dynamics of norms*, pages 199–220, 1987.

[5] C.M. Bishop and SpringerLink (Service en ligne). *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[6] W.W. Bledsoe and I. Browning. Pattern recognition and reading by machine. page 225. IEEE Computer Society, 1899.

[7] J.C. Bongard and H. Lipson. 'managed challenge'alleviates disengagement in co-evolutionary system identification. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 531–538. ACM, 2005.

[8] A. Bucci, J. Pollack, and E. De Jong. Automated extraction of problem structure. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 501–512. Springer, 2004.

[9] J.P. Cartlidge. *Rules of engagement: competitive coevolutionary dynamics in computational systems*. PhD thesis, University of Leeds, 2004.

[10] S.Y. Chong, P. Tiňo, D.C. Ku, and X. Yao. Improving generalization performance in co-evolutionary learning. *Evolutionary Computation, IEEE Transactions on*, (99):1–1, 2012.

[11] S.Y. Chong, P. Tiňo, and X. Yao. Measuring generalization performance in coevolutionary learning. *Evolutionary Computation, IEEE Transactions on*, 12(4):479–505, 2008.

[12] S.Y. Chong and X. Yao. Behavioral diversity, choices and noise in the iterated prisoner's dilemma. *Evolutionary Computation, IEEE Transactions on*, 9(6):540–551, 2005.

[13] P. Darwen and X. Yao. On evolving robust strategies for iterated prisoner's dilemma. *Progress in Evolutionary Computation*, pages 276–292, 1995.

[14] P. Darwen and X. Yao. Automatic modularization by speciation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 88–93. IEEE, 1996.

[15] P.J. Darwen and X. Yao. Why more choices cause less cooperation in iterated prisoner's dilemma. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 987–994. IEEE, 2001.

[16] P.J. Darwen and X. Yao. Co-evolution in iterated prisoner's dilemma with intermediate levels of cooperation: Application to missile defense. *International Journal of Computational Intelligence and Applications*, 2:83–108, 2002.

[17] C. Darwin. On the origin of species by means of natural selection. 1859. *Leipzig: Verlag Philipp Reclam*, 1984.

[18] E. de Jong. The incremental pareto-coevolution archive. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 525–536. Springer, 2004.

[19] E. De Jong. The maxsolve algorithm for coevolution. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 483–489. ACM, 2005.

[20] E.D. de Jong. Towards a bounded pareto-coevolution archive. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 2341–2348. IEEE, 2004.

[21] E.D. De Jong. Objective fitness correlation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 440–447. ACM, 2007.

[22] E.D. De Jong and A. Bucci. Deca: Dimension extracting coevolutionary algorithm. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 313–320. ACM, 2006.

[23] E.D. De Jong and J.B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192, 2004.

[24] K.A. De Jong. Analysis of the behavior of a class of genetic adaptive systems. 1975.

[25] K.A. De Jong. Genetic algorithms are not function optimizers. *Foundations of genetic algorithms*, 2:5–17, 1993.

[26] A.P. Engelbrecht. *Computational intelligence: an introduction*. Wiley, 2007.

[27] S. Ficici and J. Pollack. Pareto optimality in coevolutionary learning. *Advances in Artificial Life*, pages 316–325, 2001.

[28] S. Ficici and J. Pollack. A game-theoretic memory mechanism for coevolution. In *Genetic and Evolutionary Computation GECCO 2003*, pages 203–203. Springer, 2003.

[29] S.G. Ficici. *Solution concepts in coevolutionary algorithms*. PhD thesis, Brandeis University, 2004.

[30] S.G. Ficici. Multiobjective optimization and coevolution. *Multiobjective Problem Solving from Nature*, pages 31–52, 2008.

[31] S.G. Ficici and J.B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the sixth international conference on Artificial life*, pages 238–247. MIT Press, 1998.

[32] D. Fogel and L. Fogel. An introduction to evolutionary programming. In *Artificial Evolution*, pages 21–33. Springer, 1996.

[33] D.B. Fogel. *Blondie24: Playing at the Edge of AI*. Morgan Kaufmann, 2002.

[34] D.B. Fogel and Z. Michalewicz. *Handbook of evolutionary computation*. Taylor & Francis, 1997.

[35] D.B. Fogel, E.C. Wasson, and E.M. Boughton. Evolving neural networks for detecting breast cancer. *Cancer letters*, 96(1):49–53, 1995.

[36] D.B. Fogel, E.C. Wasson III, E.M. Boughton, and V.W. Porto. Evolving artificial neural networks for screening features from mammograms. *Artificial Intelligence in Medicine*, 14(3):317–326, 1998.

[37] L.J. Fogel. *On the organization of intellect*. PhD thesis, 1964.

[38] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 41–49. L. Erlbaum Associates Inc., 1987.

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

[40] John H. Holland. Adaptation in natural and artificial systems. 1975.

[41] W. Jaśkowski. *Algorithms for Test-Based Problems*. PhD thesis, University of Technology, 2011.

[42] W. Jaśkowski and W. Kotlowski. On selecting the best individual in noisy environments. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 961–968. ACM, 2008.

[43] H. Jeffreys. *Theory of probability*. Oxford University Press, USA, 1998.

[44] H. Juille and J.B. Pollack. Co-evolving intertwined spirals. In *in Proceedings of the Fifth Annual Conference on Evolutionary Programming*. Citeseer, 1996.

[45] H. Juille and J.B. Pollack. Coevolving the" ideal" trainer: Application to the discovery of cellular automata rules. In *University of Wisconsin*. Citeseer, 1998.

[46] K.J. Kim and S.B. Cho. Evolutionary algorithms for board game players with domain knowledge. *Advanced Intelligent Paradigms in Computer Games*, pages 71–89, 2007.

[47] K.J. Kim, H. Choi, and S.B. Cho. Hybrid of evolution and reinforcement learning for othello players. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 203–209. IEEE, 2007.

[48] A. Kolcz and N.M. Allinson. N-tuple regression network. *Neural networks*, 9(5):855–869, 1996.

[49] John R. Koza. Genetic programming: On the programming of computers by means of natural selection. 1992.

[50] K. Krawiec and M.G. Szubert. Learning n-tuple networks for othello by coevolutionary gradient search. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 355–362. ACM, 2011.

[51] Krzysztof Krawiec and Marcin Szubert. Coevolutionary temporal difference learning for small-board go. In *IEEE Congress on Evolutionary Computation, CEC 2010*, 2010.

[52] S.M. Lucas. Learning to play othello with n-tuple systems. *Australian Journal of Intelligent Information Processing*, 4:1–20, 2008.

[53] S.M. Lucas. Othello competition. http://algoval.essex.ac.uk:8080/othello/League.jsp, 2012. [Online; accessed 29-June-2012].

[54] S.M. Lucas and T.P. Runarsson. Temporal difference learning versus co-evolution for acquiring othello position evaluation. In *Computational Intelligence and Games, 2006 IEEE Symposium on*, pages 52–59. IEEE, 2006.

[55] S. Luke et al. Genetic programming produced competitive soccer softbot teams for robocup97. *Genetic Programming*, 1998:214–222, 1998.

[56] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, and A. Chircop. Ecj: A java-based evolutionary computation research system. http://cs.gmu.edu/~eclab/projects/ecj/, 2007.

[57] S. Luke and R.P. Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 236–241, 2002.

[58] Sean Luke. *Essentials of Metaheuristics*. Lulu, 2009. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[59] E.P. Manning. Using resource-limited nash memory to improve an othello evaluation function. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):40–53, 2010.

[60] S. Nolfi and D. Floreano. Coevolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.

[61] L. Panait and S. Luke. A comparative study of two competitive fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 503–511. Citeseer, 2002.

[62] J. Paredis. Coevolving cellular automata: Be aware of the red queen. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 393–400, 1997.

[63] J.B. Pollack and A.D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240, 1998.

[64] J.B. Pollack, A.D. Blair, and M. Land. Coevolution of a backgammon player. In *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 92–98. Cambridge, MA: The MIT Press, 1997.

[65] E. Popovici, A. Bucci, R.P. Wiegand, and E.D. De Jong. Coevolutionary principles. *Handbook of Natural Computing*, 2010.

[66] E. Popovici and K. De Jong. Understanding competitive co-evolutionary dynamics via fitness landscapes. In *Artificial Multiagent Symposium. Part of the 2004 AAAI Fall Symposium on Artificial Intelligence*, 2004.

[67] E. Popovici and K. De Jong. Relationships between internal and external metrics in co-evolution. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2800–2807. IEEE, 2005.

[68] M.A. Potter and K.A.D. Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29, 2000.

[69] I. Rechenberg. Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution. 1973.

[70] J. Reed, R. Toombs, and N.A. Barricelli. Simulation of biological evolution and machine learning. i. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology*, 17(3):319, 1967.

[71] R. Rohwer and M. Morciniec. A theoretical and experimental account of n-tuple classifier performance. *Neural Computation*, 8(3):629–642, 1996.

[72] C.D. Rosin. *Coevolutionary search among adversaries*. PhD thesis, Citeseer, 1997.

[73] C.D. Rosin and R.K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[74] T.P. Runarsson and E.O. Jonsson. Effect of look-ahead search depth in learning position evaluation functions for othello using-greedy exploration. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 210–215. IEEE, 2007.

[75] A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 11(6):601–617, 1967.

[76] A.V. Sebald and J. Schlenzig. Minimax design of neural net controllers for highly uncertain plants. *Neural Networks, IEEE Transactions on*, 5(1):73–82, 1994.

[77] D. Shilane, J. Martikainen, S. Dudoit, and S.J. Ovaska. A general framework for statistical performance comparison of evolutionary computation algorithms. *Information Sciences*, 178(14):2870–2879, 2008.

[78] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

[79] M. Szubert. cecj: Coevolutionary computation in java. http://www.cs.put.poznan.pl/mszubert/projects/cecj.html, 2009.

[80] M. Szubert, W. Jaśkowski, and K. Krawiec. Coevolutionary temporal difference learning for othello. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 104–111. Ieee, 2009.

[81] M. Szubert, W. Jaśkowski, and K. Krawiec. Learning board evaluation function for othello by hybridizing coevolution with temporal difference learning. *Control and Cybernetics*, 40(3), 2011.

[82] Marcin Szubert. Coevolutionary reinforcement learning and its application to othello. Master's thesis, Poznan University of Technology, 2009.

[83] S. Viswanathan and J.B. Pollack. On the coevolutionary construction of learnable gradients. In *Proceedings of the 2005 AAAI Fall Symposium on Coevolutionary and Coadaptive Systems. AAAI Press*, 2005.

[84] R.A. Watson and J.B. Pollack. Coevolutionary dynamics in a minimal substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 702–709. Morgan Kaufmann, 2001.

[85] R.P. Wiegand, W.C. Liles, and K.A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, volume 2611, pages 1235–1245, 2001.

[86] L. Yang, H. Huang, and X. Yang. An efficient pareto-coevolution archive. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 484–488. IEEE, 2007.

[87] X. Yao, Y. Liu, and P. Darwen. How to make best use of evolutionary learning. *Complex Systems: From Local Interactions to Global Phenomena*, pages 229–242, 1996.