DIVERSITY AND NOVELTY AS OBJECTIVES IN POKER

by

Jessica Pauli de Castro Bonson

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2016

Table of Contents

List of Tables

List of Figures

Abstract

Evolutionary algorithms are capable to lead to efficient solutions without a predefined design and few human bias. However, they can be prone to early convergence and may be deceived by a non-informative or deceptive fitness function, and thus the agents may end up as suboptimal solutions or not be able to solve the task. Diversity maintenance and novelty search are methods developed to deal with these drawbacks. The first method modifies the evolution so agents are selected based on their fitness and their diversity. Novelty search builds on top of it, and evolve individuals that are not only diverse, but that also possess novel behaviors.

Currently that are no studies on diversity and novelty for tasks that possess both deceptive properties and large amounts of ambiguity. In this work Heads-up Texas Hold'em Poker is used to provide a domain exhibiting both properties simultaneously. Specifically, Poker contains ambiguity due to imperfect information, stochasticity, and intransitivity. It is also deceptive, due to the complex strategies necessary to perform well in the game, such as bluffing. Finally, this poker variant also contains a behavior space that is extremely large, due to its many game states and decision points. This thesis investigates if diversity maintenance and novelty search are still beneficial under a task that posses these features. These techniques are compared between themselves and between the classic evolutionary method. The goal is to analyze if these methods improve the diversity in the population, and if it leads to an improved performance. This work does not aim to develop a world-class poker player, but to assess the significance of diversity and novelty search.

The results show that diversity maintenance methods were not only able to produce a diverse range of strategies for poker, but also to produce statistically better strategies than in a scenario with no diversity.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Malcolm I. Heywood, for his guidance, support and patience, not only during this Master degree but also during the AI project in my interchange year. I am very glad that I had both these opportunities. I was able to learn a lot and I am sure this knowledge will be important in my next plans in my career and life. I am also glad to the guidance given by Dr. Andrew R. McIntyre, that helped me during my research and interactions with the company.

I must thank my husband and parents for their help, incentive, and patience throughout these almost two years. Everything would have been much harder for me without their support.

I am also thankful to my readers Dr. Nur Zincir-Heywood and Dr. Dirk Arnold, for their interest and feedback on my work, to Dr. Raza Abidi for his good work on moderating my thesis defense, and to the company IGT for funding this research.

Chapter 1

Introduction

Evolutionary methods have been used to solve many reinforcement learning problems [19, 17, 22]. These methods are valued because they are capable of leading to efficient solutions without a predefined design, and due to their innovative potential and less human bias [38]. However, their high-level fitness function makes them prone to early convergence, that may end the evolution with suboptimal solutions [37]. Another problem occurs when the fitness function is not informative or is deceptive, thus the agents evolved based on it do not learn all the structures in the domain, and perform poorly [29]. A fitness function is deceptive when improvements in fitness move the agents away from the better quality solutions [53]. Thus, it also may end up penalizing strategies that are essential to develop complex agents in the long run.

A few techniques have been proposed to deal with these drawbacks: behavioral diversity maintenance [38] and novelty search [29]. Behavioral diversity maintenance defines evolution as a multi-objective process: agents are valued both for being fit and for having behaviors that are diverse from their peers. Novelty search is a subtype of behavioral diversity, where individuals are rewarded not only for being diverse, but for being novel. In other words, they must evolve behaviors previously unseen in the population. This is defined in terms of a comparison between an individual's behavior and that of an archive of 'novel' behaviors. Novelty search has been used successfully both when used alone and when coupled with fitness [44, 25, 38].

The current works on behavioral diversity maintenance and novelty search provided empirical results on tasks that are deceptive, and sometimes contain imperfect information. However, no study has been performed on deceptive tasks with high ambiguity and huge behavior space (further discussion on Section 3.6). This work applies behavioral diversity maintenance and novelty search to a task that contains both these characteristics: Heads-up Texas Hold'em Poker.

Texas Hold'em Poker is one of the most studied and popular variants of poker

[9, 47]. It contains ambiguity due to imperfect information, stochasticity, and intransitivity[1]. It is also deceptive, since strategies such as bluffing and double-bluffing are counter-intuitive to be learned initially, but are essential to develop a strong poker player. Finally, Limit Heads-up Texas Hold'em Poker has 3.16×10^{17} possible game states and 3.19×10^{14} decision points [11], thus it contains a behavior space extremely large.

The goal of this research is to analyze if behavioral diversity maintenance and novelty search are still beneficial under a task that posses a lot of ambiguity and a huge behavior space. The agents evolved with these methods are compared against themselves and against agents evolved on the classic evolutionary method, i.e. fitness-based. The comparisons analyze if these methods indeed are able to improve the diversity in the population, and if it translates into an improved performance. This work also analyzes details of the strategies employed by the evolved agents, such as aggressiveness and bluffing. It is important to point out that *the goal of this work is* ***not*** *to develop a world-class poker player.* However, the agents were trained against opponents that employed the classical strategies for poker [4], and were able to get a good performance against them as well as against an unseen adaptive Bayesian opponent [2].

The poker agents were developed using the Symbiotic Bid-Based (SBB) [32, 33, 31] framework. This Genetic Programming (GP) framework coevolves teams of programs in order to deal with complex tasks via task decomposition. The team population competes with the point population to obtain a fitness value. The points represent the environment, where each point is a poker hand. The SBB framework was adapted for poker with reinforcement learning, by adding a game environment, opponents and inputs based on previous research on poker. General-purpose versions of the code developed in this work are available on Github [2] [3].

The remainder of this thesis is organized as follows. Chapter 2 presents background on Texas Hold'em Poker and discusses previous work on the development of intelligent poker agents using machine learning. Chapter 3 focuses on behavioral diversity and novelty search. It presents background on these methods and compares

[1] As in the game of 'rock-paper-scissors'

[2] https://github.com/jpbonson/SBBReinforcementLearner

[3] https://github.com/jpbonson/SBBClassifier

the characteristics of the tasks where they have been employed. Chapter 4 explains how SBB works, how it was adapted to poker, and presents details of how behavioral diversity maintenance and novelty search were implemented in this work. Chapter 5 describes the experiments and discusses the results obtained and the agents evolved. Finally, Chapter 6 presents the conclusion and outlines future work.

Chapter 2

Poker

Poker is a challenging game to test machine learning techniques due to its imperfect information, stochasticity and complex strategies [9]. Imperfect information due to private cards, so the agents have to deduce their opponent's strength and strategy by its behavior. Stochasticity is due to the random cards that can be drawn, both the private and the public ones. Poker also enables diverse options of strategies, from more defensive to more aggressive ones, including deceiving strategies such as bluffing and double-bluffing (respectively, pretending that one has a weak hand, and pretending to bluff). Another characteristic of these strategies is the intransitivity, where A beats B, B beats C, but A not necessarily beats C.

Texas Hold'em Poker is the variant of the game employed in this research. It is the most researched variant of poker game [47], and one of the most popular versions played in casinos [11].

2.1 Texas Hold'em Poker

2.1.1 Rules

Texas Hold'em Poker is composed of four rounds: preflop, flop, turn, and river. Each player starts with two private cards (hole cards), and at each round public cards are added to the board, up to five cards (community cards). The strength of a player's hand is defined by the five strongest combination of cards, considering both the private and public ones. At each turn the players can either fold, call, or raise. During the match the players bet chips, i.e. add them to the pot, according to how they assess theirs and their opponent strength and strategy. A match ends when one of the players surrender or if it was played until no more actions can be performed (i.e. a showdown). In case of a showdown, the player who has the strongest hand gains the chips in the pot. The goal of the game can be either to maximize the chips

won across the hands, or win hands in order to win a tournament, and this can be achieved by various types of strategies.

The three available actions are described below.

Fold When a player gives up the hand, losing any right to contest the current chips in the pot. The chips are then acquired by the opponent. If a player does not fold up until the flop, it is considered they *played* the hand.

Raise When a player adds chips to the pot besides the minimum amount required. In the limit variant, the amount of chips added to the pot is equal to the Small Blind Bet for the preflop and flop, and equal to the Big Blind Bet to the turn and river. There is a maximum amount of raises that can be performed per round, usually 4 or 5, and after it the players can either call or fold. This action is referred as *bet* when it is the first raise in the round.

Call When a player matches the value raised by the opponent (i.e. add the same quantity of chips to the pot). After the opponent raises, the player can either call or raise more, but they must at least add the chips raised by the opponent, or fold their hand. When both players call, the round ends. A call is also called a *check* when there is no value to match to.

The main components of the game are as follows.

Hole Cards The cards that the players receive at the beginning of a hand. Each player gets 2 random cards. These cards are private and only the player can see them. The hole cards frequently are used to analyze if a hand is worth playing or not based on their potential strength.

Community Cards Public cards that are accessible to all players and can be used by them to calculate their hand strength. Only the five best cards, from the private and public ones, are used to calculate the hand strength. Three community cards are drawn in the flop, one in the turn, and another one in the river. Due to this, the flop is an important turn to define one's strategy for that hand.

Chips It represents the players' money. Usually, players have a limited amount of chips available, and they can not play more hands if they are out of them.

Pot The total amount of chips that the players wager during a single hand.

Blind Bets Forced bets that the players must do when starting a hand. For the 2-players game, the player who is the dealer must bet a Small Blind, while the other player bets the Big Blind. Usually the Small Blind is half the value of the Big Blind. For more players, the Small Blind is posted by the player to the left of the dealer, and the Big Blind is posted by the player to the left of the Small Blind.

Finally, the rounds and the showdown are detailed as follows.

Preflop The players do the blind bets, and receive the hole cards. The dealer is the first to act. In the limit Hold'em Poker, for this and the next round the raises must be equal to the Big Blind (also called *small bet*).

Flop The first three community cards are drawn. For this and the next rounds, the dealer acts last.

Turn The fourth community card is drawn. In the limit Hold'em Poker, for this and the next round the raises must be equal to twice the Big Blind (also called *big bet*).

River The last community card is drawn. If there is more than one player still in the hand by the end of this round, a showdown occurs.

Showdown The players show their hands, and the one with the strongest hand using 5 cards out of the 7 cards available wins the chips on the pot. In case of a draw, the chips are distributed among the players. Cards are stronger depending on how high are their ranks (ace, king, queen, etc) and what combinations of cards are formed in the 5 cards. From weakest to strongest, the combinations are: high card, a pair, two pairs, three of a kind, a straight, a flush, a full house, four of a kind, and a straight flush[1].

In this work the variant of Texas Hold'em Poker used is the $heads-up$ (2 players), with limit (the values the players can raise are defined by the *small bet* and *big bet*,

[1] http://www.pokerlistings.com/poker-hand-ranking

instead of being able to bet as much chips as the players have). More information on poker in general is available in [48].

2.1.2 Examples of Strategies and Bluffing

According to [4] there are four classical styles of playing poker: Loose Aggressive, Loose Passive, Tight Aggressive, and Tight Passive. Loose/Tight refers to how many hands the player plays, where looser players play more hands. Aggressive/Passive is related with a player's preference for raising over calling. Experts [43] argue that the Tight Aggressive is the most profitable out of these four strategies, since it waits for the best hands and then aggressively tries to maximize their gains in these hands. A disadvantage is that a player that uses this strategy constantly may end up being predictable, and so they can be exploited by other agents.

There is another important factor regarding poker strategies: bluffing. A player can bluff by pretending they have a strong hand when they do not (or vice-versa). This strategy is specially effective to scare tight players out of hands, and thus exploit them. Nonetheless, it must be carefully used since players can infer that another player is bluffing (simply because one can not have so many good hands), and because the bluffing may end up appearing in a showdown.

2.2 Machine Learning for Poker

2.2.1 University of Alberta's Computer Poker Research Group

The most prominent researches on developing intelligent agents for poker have been performed by University of Alberta's Computer Poker Research Group (CPRG). Their focus is on developing individual best agents [9, 7, 14]. The first agents developed were Loki, and Poki, both knowledge-based systems. These agents had access to heuristics (such as rules and formula-based functions drawn from expert knowledge), opponent modeling, and hand simulations that help them decide their next action. The simulations use selective-sampling and generate the possible ways the current hand may play out. Selective-sampling in this case means that the opponent's actions are stored and used to bias which hands are relevant to be simulated.

More poker agents were developed by Alberta's group, using game theory principles and abstractions [6]. The agents worked for heads-up limit Texas Hold'em and used defensive strategies. So the goal was to never lose, but they did not exploit the opponent. The abstractions simplified the game tree and eliminated betting rounds. Improved versions of these agents, called Hyperborean-Eqm and Polaris [12], used the CFR algorithm [54]. This algorithm was able to produce and solve game models with reduced loss of information due to abstraction, and thus improved the agents' performances.

On the opposite side of the previous agents, CPRG also developed agents that focused on adaptation and exploitation [8], instead of playing safe. These agents relied on a complex opponent modeling based on the betting of their opponents to predict and exploit their next actions.

In one of their most recent works, it was announced that heads-up limit Texas Hold'em was weakly solved [11]. 'Weakly solved' can be briefly summarized as a strategy that is not necessarily the best one, but that is very hard to be exploited. To do this they used the algorithm CFR+, an improved version of the previous CFR that is able to deal with much larger extensive-form games. The algorithm required significant amounts of computational effort (900 core-years of computation and nearly 11 TB of disk space).

The main differences between the work developed by Alberta's group and the research conducted here is that here the goal is to evolve a diverse group of capable agents, instead of creating a single best player. Also, the agents evolve their strategies from scratch instead of using knowledge-based heuristics. Another difference is that the agents work as teams of programs instead of a single program. Finally, due to the evolution of multiple agents across generations, it is not possible to use simulations. Such an approach would not be computationally tractable due to too many hands to simulate and analyze.

In short, we are not attempting to 'solve' the game as such, but employ Poker as a task domain which exhibits multiple sources of uncertainty and deception. The specific question that we are interested in answering is whether diversity and novelty preservation mechanisms are still useful under such a setting when building strategies with genetic programming.

2.2.2 Evolutionary Poker Agents

The works reviewed in this section are more similar to this research in the sense that they also aim to identify poker agents through some learning mechanism. Typically some form of Evolutionary Computation is assumed, in which each individual plays multiple hands at each generation. The best ones are selected to reproduce, and new agents are generated based on them. This process occurs during multiple generations, and the agents in the last generation are the final result of the algorithm. In a few works coevolution is used, adding a new population that can either compete or cooperate with the previous population. The agents can be represented in many ways, e.g.: artificial neural networks (ANN), parametrized functions, and programs.

Function-based Approach for Adaptive Agents

One of the earliest works on evolving agents for poker was done by [4, 5]. [5] is an improved version of [4], using a more complex model. The authors evolved agents for a simplified version of 10-agents Texas Hold'em, with only one betting round where all cards are already distributed. Their goal was to evolve agents that could adapt to their opponents' strategies. The game was divided into risk versus position. There were 4 possible risks, based on the current bets, and 3 possible positions, for a total of 12 game states. Each state had a population associated with it. The authors used three functions that together defined the agent's strategy. In total these functions had 7 parameters, that represent an agent and evolve along generations. Each agent plays 500 hands per generation, in tables with 5 other individuals and 5 either loose aggressive or tight passive fixed opponents. At the end of each generation the best half of the population generates offspring while the other half is discarded. The evolved agents were able to exploit the strategies of the loose aggressive and tight passive opponents.

A similar approach for the poker task was taken in [24]. In their work they compared evolving function-based agents against rule-based static agents, for the same simplified version of poker. The tables had 5 agents, one evolutionary and four static. The evolving agents were able to outperform the rule-based agents along the generations.

Pareto Dominance and Coevolution

In [42]'s work the authors used Pareto Dominance and coevolution to evolve agents with robust strategies. The poker variant used was limit Texas Hold'em with up to 10 agents per table. The goal was to prove that Pareto Coevolution (PC) can be used to find a set of agents that performs well and is able to generalize against various types of opponents. Pareto Dominance is a multi-objective optimization technique. In their work this method was used so that an agent A is considered better than an agent B only if A performs at least as well as B against all opponents and is better than B against at least one opponent. More information on this method is available on Section 3.4. In their work coevolution means that evolving agents are competing against each other. The authors compared self-play using coevolution, for a scenario with PC versus one with traditional Genetic Algorithm (GA). A traditional fitness function is used in the GA. Their results were that the Pareto technique seems promising, since the strategies developed with it performed better (i.e. won more chips per hand) against groups of predefined opponents with varying strategies.

These results were reinforced in [41], where the authors built from the previous work by modeling the agents with artificial neural networks (ANN). Each player was composed by two ANN: one for the preflop, and another for the rounds after the flop. The ANN's inputs were related with hand evaluation, opponent modeling, and game states. The outputs were the three basic actions (fold, call, raise). They also used deterministic crowding along with PC to obtain diversity maintenance, and thus avoid early convergence. Deterministic crowding means that a child replaces its parent if it its better than it. The populations consist of 20 agents that evolve over 5000 generations, competing in tables that included themselves and a reference group of strategies for a total of 200 games with 100 hands each. A few scenarios were considered: PC vs GA, with and without a head-start (agents evolve from heuristics instead of from scratch), and with and without fixed reference opponents. The agents evolved with PC steadily improved against the reference group, while the agents with GA had a messier training curve. The authors' conclusion was that knowledge was maintained over successive generations using PC, while GA was not able to maintain it. The agents evolved with PC also won more chips per hand than the ones with GA. However, for the scenarios without a head-start, the PC agents lose

more chips than win against the reference opponents. The authors state that diversity maintenance helped improve the agents' performance, but they did not provide an analysis regarding how the diversity maintenance affected their results.

Coevolution and Hall of Fame

The work of [39] evolved poker agents for 9-agents tournaments of no-limit Texas Hold'em. They were represented as ANNs, and the selection of the agents would occur through the mean rank obtained in multiple tournaments per generation. The authors tested the agents against predefined baseline opponents and previously evolved agents. Their results showed that adding coevolution and hall of fame improved the agents' results, in comparison with a scenario that did not use these methods. Hall of fame is a method that stores current agents to use them as opponents in future generations, more information is available in Section 4.2.4.

In a later work [40], the authors further explored the benefits of coevolution and hall of fame, mainly in relation to evolutionary forgetting. Evolutionary forgetting can lead to selecting agents with strategies that were not considered to be good in previous generations. The agents had access to inputs that provided information about the pot, bets, chips, number of opponents, hand strength, position, and opponent modeling (their aggressiveness). The output of the ANNs was stochastic, not deterministic, so all actions had a chance to be chosen, to avoid predictability. The agents were benchmarked against simple static opponents and previously evolved agents. The authors found out that using a large hall of fame had a positive impact in the agents' performance, while coevolution was mostly insignificant when used along with hall of fame. They also compared agents evolved using three different fitness functions (the agent's rank in the tournament, the chips won, and the hands won), and considered that the agent's rank was the best one.

Another work related with coevolution, but without a hall of fame, was performed by [51]. In their work the agents evolved to play a simplified 10-agents Texas Hold'em, with just the preflop. Their results showed that the agents evolved with coevolution ended up in a cycle of counter-strategies, where new strategies where able to beat the current one, but they did not evolve to a general strategy. A problem that could potentially be solved using a hall of fame.

Bayesian Opponent Modeling

The work in [3] is an improved version of the previous work at [2], adding ANNs that evolve over time. In their first work, the authors developed an opponent modeling technique using Bayesian probabilities. The agents were modeled using a rule-based approach, where two parameters defined the player's looseness and aggressiveness. The authors implemented four agents for each classical playing style, and four anti-agents that were tuned so they would get the best performance against each of the classical agents. The final player used Bayesian probabilities, along with precomputed probabilities of how each style plays, to define what was the opponent's style. After that, the Bayesian player used the appropriate anti-player to select actions against that opponent. This model is explained in more details in Section 4.2.4. The player was tested for a simplified version of poker, for 4-agents tournaments, and performed well against a deterministic opponent.

In the next work [3], they compared ANNs that evolved over time. A group had access to the Bayesian opponent modeling, while the other had not. The other inputs were the hand strength, the chips, and the last opponent action. The agents were tested against various dynamic agents, and the authors found out that the player using the Bayesian opponent model obtained a better result when facing both known and unknown tables of opponent combinations.

2.3 Discussion

The main difference from the previous works and the one reported in this research is that we are interested in the evolution of poker agents to ***answer the following question: whether behavioral diversity maintenance and novelty search are still beneficial under tasks that posses a lot of uncertainty, hidden information, and stochasticity***. Although behavioral diversity and novelty search are claimed to be effective under 'deceptive tasks', this is stated based on empirical observation. Hence the basis for such an observation is limited to the specifics of the task under which the original evaluation was performed. In this work, we assume the Poker domain on account of the wider sources of uncertainty, hidden information, and stochasticity. Hence, potentially a more thorough assessment of the utility of these

diversity maintenance schemes under a 'deceptive task'. As far as we are aware there has not been previous work related to assess the importance of behavioral diversity and novelty in such a domain. To provide context to the research on behavioral diversity and novelty, the next chapter reviews previous works in other domains.

Additionally, it is important to emphasize that while we intended on evolving intelligent agents capable of playing the poker game at a reasonable level, and provided a few analysis on their playing style, ***we are not attempting to build a world's best poker playing agent.*** Attempts to do this to date emphasize large play databases, expert knowledge, and roll outs to provide 'optimal' evaluation of strategy (e.g.: [11]). In this work we focus on evolutionary methods where agents learn to play from scratch.

Chapter 3

Behavioral Diversity and Novelty as Objectives

3.1 Deceptive Fitness Functions

Most previous works on behavioral diversity and novelty have been done in the domain of Evolutionary Robotics [38, 29]. Evolutionary algorithms are good candidates to solve robotics tasks because they are capable to lead to efficient solutions without a predefined design, and thus are less affected by human bias, that could restrain the solutions [38]. However, as per any (non-exhaustive) search process, premature convergence may occur [37].

Behavioral diversity and novelty can be used to avoid early convergence, mainly on *deceptive* tasks. A task is considered deceptive when optimizing the fitness function leads to suboptimal solutions (i.e. a fitness gradient that leads away from the global optima). Additionally, the steps that could indeed lead to the optimal solution may actually be penalized by the fitness function [53]. Two examples in the robotics field are maze navigation and biped locomotion [29]. In the case of a maze, being in a point near to the end of it does not necessarily means the robot is in the correct path to actually arrive to it. In the biped locomotion task, where the goal is to go further away from the starting point, robots may end up lunging as far as they can, instead of actually learning to walk.

Deception also occurs in other complex tasks, such as poker, that is the focus of this research. In poker if an agent tries to learn how to play focusing only on the fitness function (i.e. win more chips per hand), it would evolve into a tight aggressive player, that is the best strategy among the classical strategies. In this strategy the player bets only on strong hands, and tries to maximize its winnings by playing aggressively in these hands. However, such a player by no means would be the best poker playing agent. The reason is that it is easy to predict and then exploit a tight aggressive strategy. For example, by pretending to have stronger hands to make it fold, i.e. bluffing. However, the process of learning how to bluff is counter-intuitive,

since firstly the agent must play a match 'wrongly' with a weak hand, to just then realize that this behavior can be useful to scare away other players if used in the correct context. Another way to exploit such an agent would be to simply fold every time it plays aggressively and you do not have a strong hand, since it is obvious that it has a stronger hand. In short, a player that only tries to maximize the fitness function would not learn how to bluff and how to hide its hand strengths, and thus would be a very suboptimal solution for the task of playing poker.

3.2 Behavioral Diversity and Novelty Search

To avoid problems with deceptive tasks, many authors used diversity maintenance techniques [16, 23, 38, 17, 22]. In these methods the agents are rewarded not only for their fitness, but also for how diverse they are from their peers. The goal of these techniques is to increase exploration, and thus enable the agents to learn strategies that may not immediately increase their performance, but help them find more robust strategies. Diversity maintenance can be categorized mainly between genotypic and behavioral. Diversity based on genotype rewards agents for having different underlying structures, while based on behavior rewards them for having different strategies.

A significant difference between these diversity maintenance methods is that diversity based on genotypes will not necessarily create agents with diverse functionality, and thus may still fall to a deceptive local optima [27, 29]. Another difference is that, depending on the representation, genotypic diversity can be hard to compute (e.g. distance between graphs). This problem is partially solved when computing behavioral diversity: the individual's structure can be ignored, and instead it is only necessary to calculate distances between behaviors. These distances can be task-specific [38] or generic, such as based in the agent's actions and states [19].

A type of behavioral diversity is novelty search, where agents are incentivated to produce novel behavior. The main difference between novelty search and other behavioral diversity methods is that this technique uses an archive of behaviors. This archive is updated over generations with the most novel individuals, that are selected using a minimum threshold. For each generation the diversity of the individuals in the population is calculated considering both the population and the archive. This way there is a steady improvement of novelty along generations and backtracking is

avoided. While there is an increased computational cost due to the use of an archive, the archive size can be limited to mitigate this problem [27].

Unlike general behavioral diversity methods, novelty search can be used without a fitness function to produce solutions [44, 25], and in some cases even produces better solutions when used this way [44, 28, 27, 25]. This occurs since the progress happens by discovering innovations that leads to further innovations, without dependence on the fitness function being informative. In order to discover innovations, usually is necessary that the agents learn the structure of the domain.

To implement novelty search is necessary a behavioral distance metric, and a way to compute how similar an individual is to the rest of the population in the behavior space. Areas in the behavior space with too many individuals have less novelty, while individuals that are sparsely in the space have greater novelty. A way to reward novelty based in the sparseness of a point is using the average distance to the k-nearest neighbors of that point, as recommended by [29]. If the distance of a individual to its nearest neighbors is big, then it is in a sparse area. If it is small, then it is a dense area. The formula for the sparseness of an individual p at point x is given in Equation 3.1. u_i represents the ith-nearest neighbor of x, and $dist(x, u_i)$ is the behavioral distance between x and u_i in the search space. Thus, if $p(x)$ is big the individual is in a sparse space and has high novelty.

$$p(x) = \frac{1}{k} \sum_{i=1}^{k} dist(x, u_i) \tag{3.1}$$

Naturally, such a formulation assumes that the start state (initial configuration) of the task is sufficiently similar between solutions for the measure of diversity or novelty to be informative.

3.3 Diversity Maintenance

Diversity maintenance methods are used to evolve diverse agents in the population and thus encourage exploration over exploitation in evolutionary algorithms. It has been demonstrated to improve the performance of the solutions in many domains [38, 10]. Diversity maintenance techniques can be applied to various dimensions, such as genotype and behavior. In both cases a typical configuration consists of a fitness

modification method coupled with a distance metric. A few of the modifications commonly used are fitness sharing [18], crowding [1] and multi-objective optimization [15, 36, 37]. These modifications can be coupled with speciation [18], where similar individuals are assigned to a specie, and evolution occurs separately for each specie. Two of the most used behavioral distance metrics are Hamming distance [19, 16] and Normalized Compression Distance (NCD) [30, 19]. Both these metrics are generic, and thus task-independent, so they decrease the human bias added to the evolutionary process. They are explained in more detail below.

3.3.1 Behavioral Distance Metrics

Hamming Distance

According to [19, 16], the Hamming distance is regarded as powerful, computationally cheap, and easy to implement. The main idea is that it counts the number of bits that differ between two binary sequences. The formula is shown in Equation 3.2.

$$d(x, y) = \sum_{i=0}^{T} \delta(\beta_x[i], \beta_y[i]) \tag{3.2}$$

δ represents the Kronecker delta, and β are a string of concatenated behaviors (action histories), where i is the i-th entry in the concatenated sequence. So the distance of two behaviors is the number of positions i where the individuals chose different actions.

Normalized Compression Distance (NCD)

As shows in the results obtained by [19], NCD seems to produce better results, is more robust, and is not restricted to strings of equal length and order. The downside is that it is more computationally expensive. The main idea is that NCD exploits similarities between compressed sequences of strings. The formula is shown in Equation 3.3.

$$d(\beta_x, \beta_y) = \frac{C(\beta_x\beta_y) - min(C(\beta_x), C(\beta_y))}{max(C(\beta_x), C(\beta_y))} \tag{3.3}$$

$C(\beta)$ is the compressed length of behavior β, and $\beta_x\beta_y$ is the concatenation of the sequences β_x and β_y. If β_x and β_y are equal, a compressor should be able to detect that

$\beta_x\beta_y$ is just two copies of the same sequence, and thus $C(\beta_x\beta_y) \simeq C(\beta_x) = C(\beta_y)$, then NCD $\simeq 0$. Similarly, if β_x and β_y have little in common, $C(\beta_x\beta_y) - C(\beta_x) \simeq C(\beta_y)$ and NCD $\simeq 1$.

3.4 Multi-Objective Optimization and Pareto Dominance

Multi-objective optimization techniques can be used to evolve the solutions in order to achieve two or more objectives at the same time. A common use is to balance fitness and diversity. This way the agents still have access to the information given by the fitness function, but are not fully dependent on it, since there is also a pressure to explore rather than exploit, that favors diverse agents.

Most evolutionary algorithms that simultaneously optimize several objectives use the Pareto dominance method [38]. In Pareto dominance, a solution x dominates another solution y if x is not worse than y regarding all objectives, and x is better than y in at least one objective. A Pareto front is the set of all non-dominated solutions in the search space. This front is considered Pareto-optimal, in the sense that the solutions can not improve in one objective without decreasing their score in another objective.

An intuitive way to explain the Pareto dominance method is that when comparing the performance of two individuals over multiple objectives, if these individuals are better on different subsets of the objectives (i.e. one does not dominate the other), then you can not meaningfully rank them. At the end, the Pareto front will contain individuals that specialized in different strategies and trade-offs to optimize the set of all objectives (e.g.: by prioritizing an objective, by balancing them, etc). The agents in the Pareto front then are used as candidates that are selected to produce offspring and/or are kept for the next generation.

3.5 Previous Work

The authors of [38] reviewed various works on evolutionary robotics. Their focus was on behavioral diversity as a way to deal with early convergence. These researches were compared in a common framework, with three different mobile robotic tasks and two different ANNs genotypes. The framework was the NSGA-2 [15], a multi-objective

evolutionary algorithm. They were also compared with results produced by the NEAT framework. The diversity mechanisms compared were multi-objective and fitness sharing, separately and together, both encouraging behavioral diversity. They also compared with evolution with no diversity maintenance. The distance metrics were Ad-hoc behavioral distances, Hamming distance on the agents' states and actions, and genotype-based distances. The authors' goal was to find which combinations of distance and diversity mechanism consistently outperformed the other combinations, independently of genotype or task.

Their results are as follows:

- the 'no diversity' configuration (control group) was not able to dominate any other configuration, and obtains a near-zero success rates and a zero median fitness in two out of the three tasks;
- while genotypic diversity improved the results in most of the experiments in relation to the control group, it obtained worse results than most behavioral diversity combinations;
- the task-specific behavioral distances obtained both some of the better and worse results, depending on the configuration and task. This shows that they can be powerful when appropriately designed, but terrible otherwise;
- while NEAT was more efficient than the control group, its results were worse than the ones using behavioral diversity;
- compared to the improvement obtained via behavioral diversity, the differences between the genotypes encoding were negligible;
- the best configurations were multi-objective with Hamming distance and multi-objective with fitness sharing and Ad-hoc distances. They dominated the other configurations in all experiments, regardless of genotype or task. But it is noteworthy that the first one is more robust, since there is no need to set a parameter or handcraft a distance metric;
- in short, the results show that explicitly encouraging behavioral diversity improved the performance in all the three tasks, regarding both fitness and success.

Additionally, in the works reviewed by the authors they found the following results regarding novelty search and behavioral diversity, summarized below:

- novelty search significantly improved evolution in at least three relevant tasks (a deceptive maze [26, 27, 35], fast biped walking [27], and search in a T-maze [45]);
- in the case of the deceptive maze [35], novelty search alone found solutions faster than a multi-objective approach, while the multi-objective approach developed more efficient solutions. A approach using only fitness as the objective was never able to find a solution;
- for a task of finding a specific sequence of circuit activation (light-seeking) [36], a multi-objective behavioral diversity approach was able to find solutions, while a standard algorithm was not;
- in [37] the authors compared multi-objective optimization with behavioral diversity and with novelty search against NEAT. The task was to compute a Boolean function with a deceptive fitness. They found that all three approaches were equally efficient and better than fitness-based evolution. But behavioral diversity and novelty search were also simpler to implement than NEAT;
- in [52] the authors tried to improve NEAT by replacing its genotypic distance by a behavioral distance. The algorithm was able to evolve more diverse behaviors, but it did not improve its performance. The conclusion was that it probably occurred because the task was too simple (obstacle avoidance);
- the authors in [34] also tried to improve NEAT by adding behavioral diversity. They tested it in a generic and complex suite of problems for reinforcement learning. They found that the modification improved the algorithm's performance, particularly in tasks with a high noise level and large state space;
- in [19] the author compared three distances for behavioral diversity, along with genotypic and fitness distance. The task was the Tartarus problem. The behavioral distances outperformed the other distances, with NCD being the best one.

The next researches reviewed are works focused on novelty search that were not included in [38]'s review.

In [44] the authors tested novelty search on three reward-based learning tasks: T-maze, double T-maze, and foraging bee. All the tasks were defined so the fitness function would be deceptive. The individuals were ANNs. They compared agents that evolved based solely on fitness with agents that evolved only with novelty. Novelty search outperformed fitness for the T-maze tasks, showing that it can deal well with deceptive problems. However, it obtained the same success rate for the bee foraging task as the fitness-based evolution. But even with the same result, the solutions evolved with novelty were faster than those evolved with fitness. The authors considered that this occurred due to a noisy environment, that offered a vast space of exploitable behavioral strategies.

Novelty search was applied to a deceptive version of the Tartarus problem in [13]. The individuals were represented as ANNs. The authors compared scenarios with only novelty, only fitness, and combinations of them. Their results show that using novelty as the sole criterion for selection encourages behavioral diversity, but it does not necessarily lead to high average fitness. Novelty search ended up performing worse than using only fitness. The best results were obtained by balancing fitness and novelty. The authors considered that the bad performance of novelty in this task was due to a large solution space, containing few good solutions.

In [25] the authors used novelty search to deal with two robot locomotion tasks: obstacle avoidance and swimming. The individuals were represented by a robot topology and an ANN that worked as a controller. Both the topology and the controller were evolved (body-brain co-evolution). Their results showed that novelty search outperformed fitness-based search in the obstacle avoidance task, that was very deceptive. Fitness-based solutions were not able to solve this task. There was no significant difference between combining fitness and novelty and using only novelty. For the swimming task, novelty search performed worse than fitness. The authors argue that this occur both because the task was not very deceptive and due to the large unconstrained behavior space. However, it is worth noting that even in this large space novelty search still outperformed random search.

In [29] the authors explored novelty search for deceptive robot locomotion tasks.

The tasks were biped locomotion and maze navigation. Both were configured so they could be executed with varying degrees of deception. The individuals were represented as ANNs and evolved using the NEAT algorithm. The authors compared combinations of three diversity maintenance techniques: genotypic speciation, an age objective, and a novelty objective. These techniques were combined using Pareto Dominance, and were tested with and without a fitness objective. Their results differed for each task. For the biped domain, the most effective technique was to combine fitness and novelty, and there was also a slight benefit in maintaining genotypic diversity. For the maze domain, the greatest impact was if a novelty objective was included or not, with little benefit from adding objectives for fitness and genotypic diversity. Their conclusion was that novelty search performance is task-dependent, and may have a better scalability for deceptive and complex problems than the traditional approach based on fitness.

3.6 Discussion

A comparison between the characteristics of the tasks where behavioral diversity and novelty have been employed is shown in Table 3.1. Tasks that employed novelty are marked with (N), and tasks with behavioral diversity with (B). The table summarizes the characteristics of the tasks that the authors of previous works analyzed, and how they compare with the Texas Hold'em Poker task. The following characteristics were compared:

Deceptive (De) A task is deceptive when optimizing the fitness function leads to suboptimal solutions.

Imperfect information (II) Imperfect information occurs when the agents do not have access to all useful information currently in the environment.

Stochasticity (St) When the task suffers influence of random variables.

Intransitivity (IT) A task contains intransitivity when a strategy A beats B, B beats C, but A not necessarily beats C.

Another comparison between these works is provided in Table 3.2. This table summarizes the results for each task regarding the performance for agents evolved

only with fitness (F), only with diversity (D), and with a multi-objective optimization of the two (F+D). Here 'performance' means the metric the original authors used to compare the agents, usually success rate or fitness. 'Diversity' is the novelty or behavioral diversity metric used by the authors.

As shown in Table 3.1, many previous works on behavioral diversity and novelty worked with tasks that involved deception or low-informative fitness functions. This occurs since these are the types of tasks where these diversity methods can be more impactful, since they diminish the importance of the fitness function.

The researches with imperfect information were on tasks where the agents had limited vision of a environment they could walk on, or where some environment's properties were hidden (e.g.: the reward of different positions). Approaches that used behavioral diversity or novelty were better than fitness only in all scenarios with imperfect information, but the results are mixed regarding F+D versus D approaches.

Just two previous works dealt with stochasticity: the deceptive and the discrete Tartarus task. While F+D was better than F in both cases, it is worth noting that for the deceptive task novelty alone was worse than fitness alone. Additionally, none of the previous works had a task with intransitivity, so it was easier to define when a strategy was better than other during the evolution process.

Finally, there is also a difference regarding the possible size of the behavior space. Heads-Up Limit Hold'em Poker has 3.16×10^{17} possible game states and 3.19×10^{14} decision points where a player is required to make a decision [11]. All the other tasks analyzed in previous works on Table 3.1 had much smaller search spaces. Additionally, swimming [25], the task with a higher behavior space due to unconstrained movement and evolution of both the robot's morphology and controller, was one of the few tasks where fitness only outperformed novelty only. Some authors [44, 28, 25] consider that if the behavior space is too large and unconstrained, novelty search may spend most of the time exploring behaviors that are irrelevant to the objective and never find any useful solutions.

In conclusion, to date only very specific claims have been made regarding the performance and robustness of behavioral diversity and novelty as objectives. Including scenarios with multi-objective optimization and with novelty search alone. The main differences of the poker task from the previous works are that it contains stochasticity,

Table 3.1: Summary of the characteristics of the tasks researched on previous works on novelty and behavioral diversity. The comparison is between deception (De), imperfect information (II), stochasticity (St), and intransitivity (IT). The tasks are tagged with N for novelty and B for behavioral diversity.

Tasks	**De**	**II**	**St**	**IT**
obstacle avoidance (N) [25]	yes	no	no	no
swimming (N) [25]	no	no	no	no
T-maze (N) [45, 44]	yes	yes	no	no
foraging bee (N) [44]	yes	yes	no	no
Tartarus (deceptive) (N) [13]	yes	yes	yes	no
humanoid gait (N) [27, 29]	yes	no	no	no
light-seeking (B) [36, 38]	no	yes	no	no
ball collecting (B) [16, 38]	no	yes	no	no
Tartarus (discrete) (B) [19]	no	yes	yes	no
deceptive maze (N,B) [26, 35, 27, 38, 29]	yes	yes	no	no
xor and xor (N,B) [37]	yes	no	no	no
Texas Hold'em Poker	yes	yes	yes	yes

intransitivity, and a much larger behavioral space. None of these characteristics were present at the same time in previous works. And even individually, few works have dealt with them when analyzing behavioral diversity and novelty search.

Table 3.2: Summary of the results from the previous researches on novelty and behavioral diversity. The comparison is between fitness alone (F), diversity alone (D), and multi-objective optimization of the two (F+D). The tasks are tagged with N for novelty and B for behavioral diversity.

Tasks	**Result**
obstacle avoidance (N) [25]	D = F+D > F
swimming (N) [25]	F > D
T-maze (N) [45, 44]	D > F
foraging bee (N) [44]	D = F
Tartarus (deceptive) (N) [13]	F+D > F > D
humanoid gait (N) [27, 29]	F+D > D > F
light-seeking (B) [36, 38]	F+D > F
ball collecting (B) [16, 38]	F+D > F
Tartarus (discrete) (B) [19]	F+D > F
deceptive maze (N,B) [26, 35, 27, 38, 29]	F+D > D > F
xor and xor (N,B) [37]	F+D > F

Chapter 4

Genetic Programming Applied to Poker

This chapter will focus on the technologies used to evolve agents for the game of poker. The main algorithm used was Symbiotic Bid-Based (SBB) [32, 33, 31, 17], a framework to coevolve teams of programs in order to deal with complex tasks. This framework was adapted to work for poker using reinforcement learning. General-purpose versions of the code developed in this work are available on Github [1] [2]. The next sections will talk about SBB and reinforcement learning in more details, specifically on what was used in this work.

4.1 Symbiotic Bid-based (SBB) framework

4.1.1 Architecture

Symbiotic Bid-based (SBB) is a genetic programming (GP) framework where a symbiont and a host population interact in order to generate complex individuals that are able to deal with task decomposition. The symbiont population contains programs, while the host population contains teams of programs. Besides the cooperative coevolution that occurs between programs and teams, there is also a third population: the points. The points represent the environment, and contains inputs associated with it. For example, attributes for a classification task or game states in a reinforcement learning task.

Figure 4.1 shows these three populations and how they interact with each other. The teams are composed of two or more programs. The same program can exist in multiple teams. Each program has an action, that can be a label in a classification task, or an actual action (e.g.: 'jump', 'kick', 'walk'...) that may or may not modify the environment in a reinforcement learning task.

A *bidding* is used to decide which program's action will be selected to be executed

[1] https://github.com/jpbonson/SBBReinforcementLearner
[2] https://github.com/jpbonson/SBBClassifier

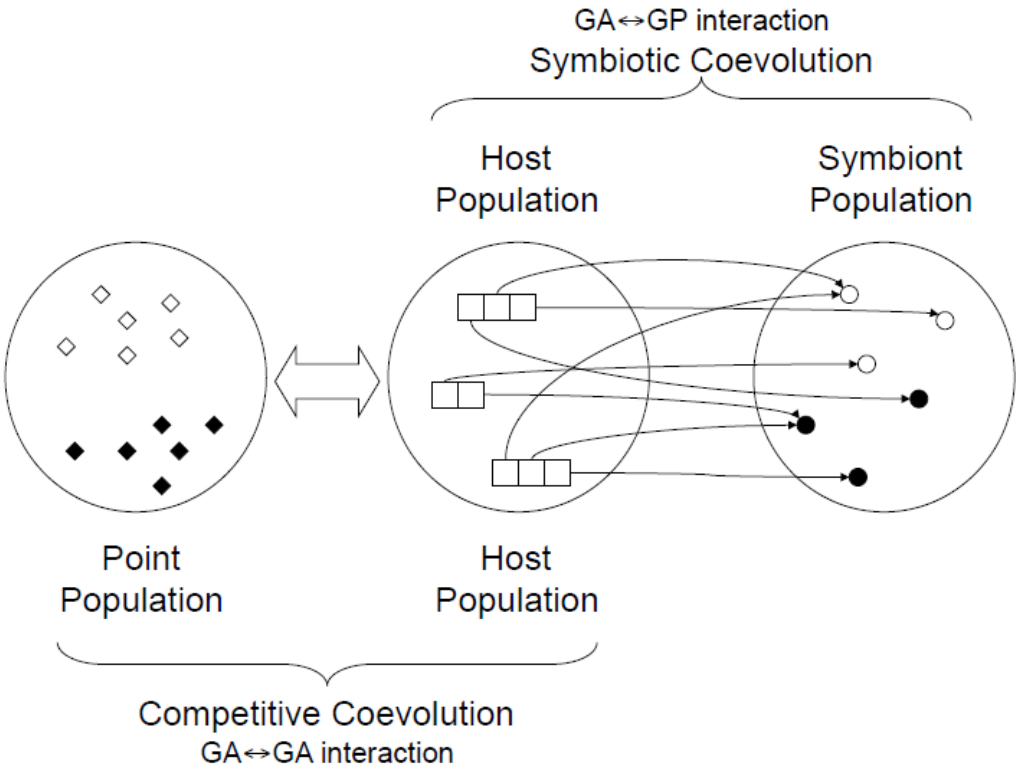


Figure 4.1: Overall SBB architecture. Diamonds are the points, they represent the environment. The circles are the symbionts (programs). Each program has a list of instructions and an action. The host population is composed of teams of programs. The host performance is evaluated based on its interaction with the point population. (Figure from [33])

by the team. In this process each program in the team produces an output for the current inputs. The one with the higher value wins, and its action is chosen to be the team's action for that point (or for the current state in the point, in cases such as a game match). The goal of the teams is to achieve high fitness scores against the individuals in the point population, and thus be selected to generate offspring for further generations.

The SBB framework is based on the classic Genetic Algorithm (GA) framework shown on Figure 4.2. The teams and programs populations are randomly initialized. Teams must have at least two programs with different actions. Programs are initialized as a random list of valid instructions. These populations are modified during a predefined number of generations. In each generation evaluation, selection, and generation of offspring occurs. These processes work together to select and produce

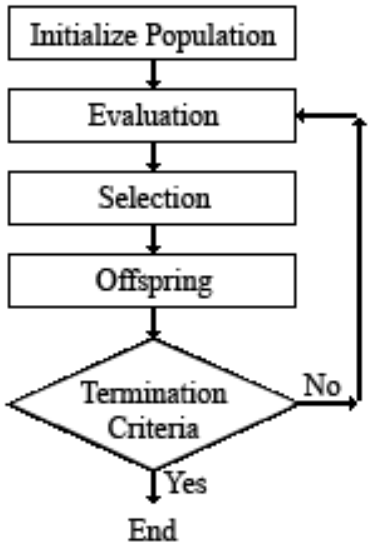


Figure 4.2: Overall flow of the classic Genetic Algorithm.

individuals to the next generations. Each of these steps will be explained in more details in the following sections. The teams appearing in the last generation are the final outcome of the algorithm.

SBB has been used in various domains. [33, 31] applied it to classification tasks. This framework also was used for various reinforcement learning tasks, such as the Acrobot handstand task [17], Pinball [23], Keepaway Soccer [22, 21], Truck Reversal [31], TicTacToe [10] and Rubik's Cube [31, 49].

4.1.2 Program Representation

Programs are composed of a list of instructions and an action. The list is initialized between a minimum and maximum size with random valid instructions. The action is randomly selected from the valid actions. Instructions have the general form $r[x] = r[x]\langle op\rangle r[y]$ or $r[x] = \langle op\rangle r[x]$, depending on the arity of the operator. x and y represent registers, where x is the target register and y is the source register. A source register may refer to an internal state or an input. There are two types of relational instructions: IF $r[x]\langle op\rangle r[y]$ THEN $\langle instruction\rangle$] and IF $r[x]\langle op\rangle r[y]$ THEN $-r[x]$]. Both these relational instructions were used in order to allow more complex solutions. The operations available to the programs are shown in Table 4.1. The final output of a program is the value that it contains in its first register ($r[0]$). All registers are

initialized with zero.

Table 4.1: Operations used in the instructions.

Type of Operator	Operators
Simple	$+, -, /, *$
Complex	$\ln, \exp, \cos$
Relational	$<, \geq$

Not all instructions modify the final result of a program. These instructions are called *introns*. Nonetheless, introns still are useful to the evolution, since they provide the basis for investigating neutral networks or combining multiple possibly negative modifications into a positive change (escaping local minima). However, since they do not provide useful information to the final program output, they are removed before the program's execution to save computational time.

Two sample programs are shown below, both after introns were removed. The first one is a simple program with minimal size, that adds the content from the the third input to the content of $r[0]$, and then divides it by input 9. The second one uses the inputs registers 3 and 13 and the program registers 0 and 3 along with more complex instructions.

```
r[0] = r[0] + i[2]
r[0] = r[0] / i[9]
```

```
r[3] = r[3] + i[13]
if r[3] >= r[0]:
    if r[3] < i[3]:
        r[0] = ln(r[0])
```

4.1.3 Evaluation and Selection

Teams are evaluated at the beginning of each generation. Each team runs against each point in the point population. The mean of the results is their fitness. The results are values between 0.0 and 1.0 that can be, for example, the accuracy in a classification task, or the percentage of matches won in a reinforcement learning task. The teams with the highest fitness values are then selected to reproduce, while

the rest of the population is discarded. To avoid premature convergence, i.e. the population be dominated by individuals that are all essentially equal and thus stop evolving, a multi-objective approach can be used. In this approach the diversity of the teams is considered along with their fitness during the selection, so teams must have both a good performance and a diverse strategy in order to be kept. The diversity maintenance implemented in this work will be explained in more details in Section 4.3. Background on diversity and multi-objective optimization is available on Chapter 3.

Evaluation also occurs during validation. After a few generations teams run against unseen points in order to track their performance without bias from their previous knowledge. In a first step all teams are evaluated against all validation points. Then, the best one (the champion) is selected to execute against a higher number of new points, in order to obtain an accurate performance of the best trained team.

4.1.4 Mutation

The teams kept by the selection step are mutated to generate new teams, i.e. offspring. These new teams will replace the ones removed in the previous step. Teams are selected to reproduce via uniform probability. Mutations can occur in the level of teams and of programs, based on probabilities. New teams receive copies of the programs of its parent, not references. So the mutations affect only the offspring, not the whole population. The available mutations are explained in the following list.

- Team Level

  Add Program Add a random program from the program population to the team, as long as the size of the team is not bigger than the maximum.

  Delete Program Remove a random program from the team. Only occurs if the team is still able to have at least two programs with different actions.

  Mutate Program Randomly select programs from the team to apply mutations.

- Program Level

Add Instruction Add a random valid instruction, as long as the total size is not bigger than the maximum.

Delete Instruction Remove an instruction. Only occurs if the program will not be shorter than the minimum size.

Random Change Instruction Modifies either the operation or the registers used by the instruction.

Swap Instruction Change instruction to the place of another instruction.

Random Change Action Modify the program's action to another valid action.

4.2 Adaptations with Reinforcement Learning for the Poker Task

This section focuses on the modifications made on the SBB framework so it would support reinforcement learning for the poker domain. For example: fitness function, inputs, and opponents. More information on reinforcement learning for poker can be found on Section 2.2, where previous works were reviewed.

4.2.1 Fitness and Score

For the poker task the performance of a player can be measured by the average chips won per hand. It provides more details on how strong it played and what was its strategy besides a simple [lose, draw, win] [47]. So, the fitness function for the teams is based on how many chips they obtain per match, considering the maximum number of chips that they can theoretically win or lose. According to this, 0.0 fitness means they lost all the possible chips (ie.: raised in all their opportunities but lost at the showdown), 0.5 means they neither lost or won chips, and 1.0 means they raised every time and won the showdown.

The version of the Texas Hold'em Poker implemented is the full game with 4 betting rounds, small bet of 10 and big bet of 20. So the maximum amount of chips a player can either win or lose per match is 240. This way, the fitness [0.0, 1.0] is a mapping from [-240, +240]. As an example, a player that obtained a mean score of 0.6 across its matches won an average of 48 chips per match. The formula for the

chips won per hand is provided in Equation 4.1, where f is the fitness value. The same idea is used to calculate the agents' performances in the test matches, where it is called score.

$$chips\_per\_hand(f) = ((f - 0.5) * 2.0) * 240 \tag{4.1}$$

4.2.2 Inputs

Environment inputs were implemented in order to enable the evolved player to see and act on poker matches. They are normalized between 0.0 and 10.0. The inputs are divided in two groups: game state and opponent model. Game state inputs contain information about the current hand being played. Opponent model inputs have attributes that characterize the opponent's behavior across hands. The opponent model can be used by the agents to analyze and counter-attack the opponent's strategy. The implemented inputs are described in more details below.

- Game State Inputs

  Hand Strength This input calculates the current hand strength of the player's hand by comparing it with all possible hands that the opponent could have for the current round [14]. Equation 4.2 shows the formula for this input, where *ahead* counts how many times the player's hand would beat the opponent's hand in the possible scenarios, *tied* counts the amount of times they would draw, and *behind* counts how many times it would be weaker than the opponent's hand.

$$hand\_strength = \frac{ahead + tied/2.0}{ahead + tied + behind} \tag{4.2}$$

  Effective Potential It is composed of three hand metrics that are used in different rounds. This occurs in order to provide the most information about the current and future strength of the hand. For the preflop is used hand equity (estimations of chances of winning a hand generated by Monte Carlo simulation [3]. It considers future strength). For flop and turn, Hand Potential is used, which can be described as a version of Hand

[3] http://oscar6echo.blogspot.ca/2012/09/poker-4-monte-carlo-analysis.html

Strength that calculates the hand rank using the present information plus simulations with all possible future cards to come. Finally, for the last round, the river, Hand Strength is used since all necessary information is already available.

The formula for Hand Potential is provided in Equation 4.3. It was adapted from the work by [14]. p means 'partial hand potential'. For example, $p[behind][ahead]$ is a counter of the times the player's hand was *behind* the opponent's hand for the current round, but was *ahead* of it in the next round (i.e. a favorable community card was added to the board in that round). *total* is the total of times the player's hand was behind, tied, or ahead in the current round, and is used to normalize the final value.

$$hand\_potential = \frac{sum\_of\_partial\_hand\_potentials}{sum\_of\_totals}$$

$$sum\_of\_partial\_hand\_potentials = p[ahead][ahead] + p[behind][ahead] + p[behind][tied]/2.0 + p[tied][ahead]$$

$$sum\_of\_totals = total[behind] * 1.5 + total[tied] + total[ahead] \tag{4.3}$$

Pot Odds This input calculates how much the current pot is worth considering: a) the bet that the player must do in order to keep playing the hand, and b) the current pot value. The formula is shown in Equation 4.4 [14].

$$pot\_odds = \frac{bet}{pot + bet} \tag{4.4}$$

Betting Position If the player is the first or the second to play in the current round.

Round The current round (preflop, flop, turn, or river).

- Opponent Model Inputs

Last Action The last opponent's action (fold, call, or raise) in the current hand. Default to 'call' at the start of a new hand.

Overall Long-term Aggressiveness For each hand an aggressiveness value is calculated. This value is the mean of the actions' values executed in that hand (where fold is 0.0, call is 0.5, and raise is 1.0). The formula is summarized in Equation 4.5. The long-term aggressiveness is the mean aggressiveness across all hands played against this opponent for the current generation. This is an adaptation for fixed bets from the description in [40]'s work.

$$aggressiveness = \frac{\#calls * 0.5 + \#raises}{\#folds + \#calls + \#raises} \tag{4.5}$$

Overall Short-term Aggressiveness The same as the previous input, only that is just considers the last ten hands. The goal of this input is to be able to deal with opponents that change and adapt their behavior over time.

Hand Aggressiveness The aggressiveness considering only the current hand. Since folds are not computed (it would mean the hand ended), calls are worth 0.0 and raises are 1.0, as in Equation 4.6.

$$hand\_aggressiveness = \frac{\#raises}{\#calls + \#raises} \tag{4.6}$$

Tight/Loose How many hands the opponent played, i.e. the agents played at least until the flop round.

Passive/Aggressive The ratio between calls and raises in the opponent's actions per hand.

Bluffing The percentage of the time a showdown occurred but the opponent had a weak hand instead of a strong one.

Chips The current score of chips won or lost against the opponent. So the player can adapt its strategy when it is losing or winning a lot (e.g.: play more defensively or exploit more the opponent).

Self Overall Short-term Aggressiveness As aggressiveness, but for the player itself, so the player is able to notice if it is playing in a specific pattern against this opponent and thus change it to be more unpredictable and avoid being exploited.

4.2.3 Point Population and Hand Balance

For the poker task each point corresponds to a game match. Each possible match was preprocessed before the training step, so each point stores metrics that characterize the match, such as the agents' positions, hand strength and effective potential of the hands of each player for each round. This was implemented to improve the training time and to ensure the teams would also see the exact same match for a given point. A total of 100.000 random hands were preprocessed this way.

The training points are balanced in nine categories. The categories represent the combinations between the hand strength of the player and the opponent (e.g.: both with weak hands, one with a strong hand while the other has a weak one, etc). The hand strengths were categorized between weak, intermediate, and strong. Where weak has the 60% weakest hands, strong has the 10% strongest ones, and the other 30% are in the intermediate category. Since the points are balanced, each category has the same weight during training. This hand balance was performed in order to ensure the teams would have equal opportunities to learn to use and go against each type of hand. Otherwise, there was a risk that they would just learn to deal with the weaker hands or fold all the time. These balanced hands are equally distributed between the opponent pool, so that if there 4 opponents in the pool and 72 points, each opponent will participate in 2 points for each of the 9 types of hand balance.

For the test scenarios, teams performed against both balanced and unbalanced points. So it would be possible to analyze their performances against a scenario correspondent to their training, and one to the real-world task.

4.2.4 Opponents

The opponents available to go against the agents are: static opponents (Loose Aggressive, Loose Passive, Tight Aggressive, and Tight Passive), Bayesian opponents, and Hall of Fame opponents. The static opponents follow the strategies of the four general playing styles in poker described in [4]. They are simple, but provide a diverse and classical baseline. The Bayesian opponent is smarter and able to adapt to its opponent. It was based on the work by [2, 3]. The Hall of Fame [46] is available to enable self-play (so the teams always have opponents with a similar difficult level to play against) and to prevent evolutionary forgetting [40]. In evolutionary forgetting

Table 4.2: Training configuration of each opponent pool.

Opponent Pool	**Total**
Static Opponents (LA, LP, TA, TP)	4 (one of each)
Bayesian Opponent	4
Hall of Fame	0-2

agents may end up developing strategies in cycles and thus stop evolving. The quantity of opponents that each player go against per training generation is summarized in Table 4.2. Each type of opponent is explained in more details in the sections below.

Static Opponents

The static opponents work as basic starting opponents. The goal of this pool of opponents is to provide diverse opponents, both easy and hard ones, to improve the learning curve and the diversity of the agents' behaviors during training, as described in [50]'s work. According to [4] there are four main play styles in poker: Loose Aggressive, Loose Passive, Tight Aggressive, and Tight Passive. Loose agents are characterized by overvaluing their hand strength, and thus playing many hands and dealing with higher risks. Tight agents are more careful and only play hands when they are sure they are very strong, but they safer strategy can be exploited by an opponent with bluffing (when the opponent pretends to have a stronger hand than it actually has, to scare away the other player). Aggressive agents raise more than call when they play a hand, so they risk both losing and winning more chips. Passive agents prefer to call rather than raise, and then they waste opportunities to obtain more chips. Each of of these combinations of strategies have different weaknesses, and a diverse range of strategies is necessary to maximize the chips won against all of them.

The static opponents were defined using the model from [2]. In their work the authors used a parametrized rule-based model to define each style, shown in Figure 4.3. The parameters are α and β. α defines if a hand will be played given its winning probability (WP). Hand strength is used as the WP in this work. The lower the α, the lower is the minimum strength a hand must have in order to be played, so the player's strategy is looser. β defines how many chips the player is willing to bet in the hands that it plays. Lower β values create more aggressive teams. Even if a hand

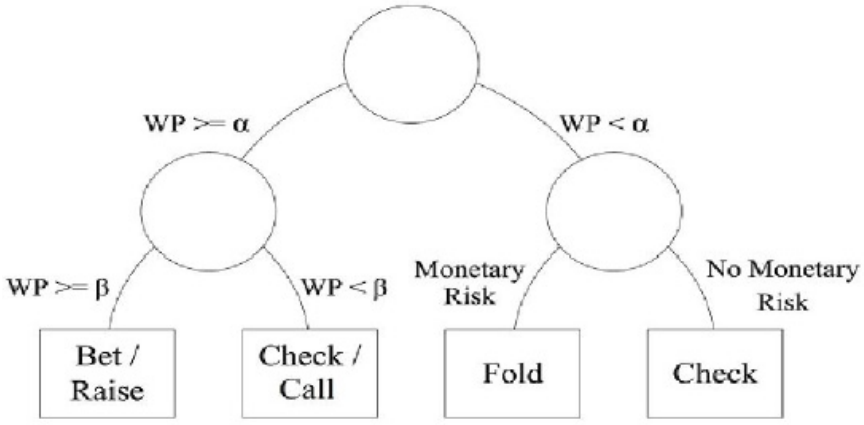


Figure 4.3: Rule-based model for basic poker agents. WP means 'winning probability', and in this work it is the hand strength. α defines how many hands will be played, and β how much will be risked in these hands. (Figure from [2])

Table 4.3: The α and β values chosen to define the static opponents. α impacts how many hands are played, and β defines the opponents' aggressiveness.

Opponent	α	β
Loose Aggressive	0.2	0.4
Loose Passive	0.2	0.8
Tight Aggressive	0.8	0.85
Tight Passive	0.8	0.95

is not strong enough to surpass the α value, the player still will play it if there is no monetary risk, i.e. there is nothing to lose by keep playing it.

A few adaptations were necessary from the original work, since it does not define what the WP was, and it considers One-Card Poker instead of the full game. New α and β values using hand strength as WP for the complete game are provided in Table 4.3. These values can be summarized as follows: Loose agents will play 80% of the stronger hands, while Tight ones will play 20% of them. Aggressive agents will raise for 75% of the strongest hands they play, and Passive ones will raise for 25%, calling the other hands.

Bayesian Opponent

The Bayesian opponent is the most complex opponent available in the pool. It is able to adapt against its opponent's strategies and to use the best available counter-strategy against it. As described in the work by [2], this player classifies the opponent

Table 4.4: Probabilities of each action being used by each player, obtained running 1000 matches for each combinations of agents.

Player Style	**Action Probability**		
	fold	**call**	**raise**
Loose Aggressive	0.05	0.43	0.52
Loose Passive	0.05	0.73	0.22
Tight Aggressive	0.37	0.36	0.27
Tight Passive	0.35	0.56	0.09

playing style using Bayesian probabilities. The classification is between LA, LP, TA, and TP. These probabilities are updated after each opponent's action, to try to model the opponent's overall behavior. The Bayesian player contains four trained anti-agents, modeled using the α and β parameters as explained in the previous opponent type. Each one was parametrized in order to obtain the maximum possible performance against each of the four types of classical playing styles. When it is the Bayesian player turn to act, it uses the updated Bayesian probabilities and predefined probabilities of actions per style to define the opponent's playing style. Then it calls the correspondent anti-player, that generates the action that will be executed by the Bayesian player.

The original model worked for a simpler version of poker, One-Card poker, and had only one round. Due to this, in this work the anti-agents' α and β parameters and the action probabilities per style were generated from scratch, so they would work effectively for the full poker game. Table 4.4 shows the action probabilities per style. They are the ratio of the use of each action against the four static opponents. Table 4.5 contains the α and β values for each anti-player, calculated by tuning the parameters in intervals of 0.1 across 1000 matches each time against the target player, where $0.0 \leq \alpha \leq \beta \leq 1.0$. This table also contains the score obtained against each player by the anti-agents. Against the tight opponents in balanced hands the anti-agents are only able to barely draw, but in the other configurations it obtains significant results above the break-even point.

Hall of Fame

Hall of Fame (HoF) is a method developed by [46], where agents evolved from previous generations are stored and used as opponents for agents in the current generation.

Table 4.5: α and β parameters obtained for the complete game of poker for each anti-player, along with the score obtained using these parameters against the target player.

Anti-Player	**Balanced Hands**			**Unbalanced Hands**		
	alpha	**beta**	**score**	**alpha**	**beta**	**score**
Anti-LA	0.5	0.5	0.525	0.4	0.4	0.528
Anti-LP	0.6	0.6	0.546	0.4	0.5	0.54
Anti-TA	0.2	0.2	0.499	0.0	0.0	0.515
Anti-TP	0.2	0.2	0.499	0.0	0.0	0.515

It encourages a steady arms race, since the evolved agents must defeat both current and old individuals. Also, since the agents must be able to perform well against a more diverse range of opponents, it helps to avoid overspecialization. Additionally, as shown in [40]'s work, HoF is essential to avoid a phenomenon called evolutionary forgetting. In this phenomenon the agents may end up developing strategies in cycles instead of continually, since without a HoF they may lose track of what was previously evolved. Another potential benefit of the HoF is to enable agents to go against another trained and complex player.

In this work, at the end of each generation the fittest team is selected to be placed in the HoF. Its maximum size is 20, so after this maximum is achieved a team is removed for each team added. The removal criteria is a Pareto Dominance comparing all teams in the HoF regarding their fitness and diversity, then the worst one is removed. The goal is not only to have the best teams in the HoF, but also a diverse pool of opponents with different strategies. After each 100 generations a slot for a HoF opponent is added to the pool of opponents, starting with 0 and up to 2 opponents at the end of the training. This is performed because evolved teams in the initial generations lack enough complexity to be good opponents, and thus their potential benefits do not outperform the computational cost of executing more matches.

4.3 Diversity Maintenance

To ensure diversity is incentivated, four types of diversity maintenance techniques are applied to the teams across various configurations: bid diversity, genotypic diversity

maintenance, behavioral diversity maintenance, and novelty search. All these methods were chosen so they would be task independent. The diversity maintenance methods were implemented using Pareto Dominance (explained in Section 3.4), so there would be an multi-objective optimization approach between diversity and fitness. For the configurations with both genotypic and behavioral diversity maintenance, just one method was selected in each generation, alternately. More information on the background of these methods are available on Section 3.3.

4.3.1 Bid Diversity

Bid diversity [33] is a behavioral diversity maintenance method specific for the SBB framework, and thus task independent. Here, a child is continually mutated until its bid profile is different from the ones of already existing teams in the population. A bid profile is a list of actions generated by making the team run against a sample of inputs from training scenarios. The sample of inputs is created in runtime, by randomly adding inputs that occurred during matches. The goal of this procedure is to obtain sample inputs that represent a diverse range of situations in a poker match. There is a maximum size to the sample, so inputs are added in a FIFO order.

4.3.2 Genotypic Diversity Maintenance

The genotype diversity used in this work is specific for the SBB framework [21]. It is based on program membership, and thus is task independent. In this metric the distance between two teams tm_i and tm_j is the ratio of active programs common for both teams. An active program is one that has at least once being chosen to provide an action (i.e. won a bid) during training across multiple matches. The formula is shown on Equation 4.7, where $Tm_{active}(tm_x)$ is the set of active programs in team x.

$$dist(tm_i, tm_j) = 1 - \frac{Tm_{active}(tm_i) \cap Tm_{active}(tm_j)}{Tm_{active}(tm_i) \cup Tm_{active}(tm_j)} \tag{4.7}$$

4.3.3 Behavioral Diversity Maintenance

Hamming distance

Hamming distance is a popular distance for behavioral diversity maintenance because its is considered powerful and computationally cheap [19, 16]. However, one of its limitations is that the sequence of inputs must have the same size. Due to this, the encoding used for Hamming distance does not consider each action executed by the players, but what was their overall aggressiveness per hand played. The aggressiveness is computed as the mean value of the actions, where fold is 0.0, call is 0.5, and raise is 1.0. It was categorized as 0 when below 0.33, 1 when between 0.33 and 0.66, and 2 when above 0.66. Below is an example of this encoding for five poker hands.

```
02101
```

Normalized Compressed Distance

Normalized Compression Distance (NCD) was chosen due to its versatility regarding the input size and good previous results [19, 21]. Due to this, NCD is capable to deal with an encoding with more information than the one used for Hamming, since it works for sequences of actions of varying length per hand. The encoding was a string with the game state per match (match id, for the first five characters, and player position, for the sixth one), the game state per round (hand strength and effective potential, also categorized between 0, 1 and 2), and the player's action per round (f, c, or r) of all the poker matches played in the current generation. An example of this encoding for five hands is shown below, one hand per line (the same hands as the ones encoded for Hamming):

```
13643121f
40412110c11r11r11c21r21r21c00r00r00r00r00c
10458011r11f
15872122f
19489021r21f
```

4.3.4 Novelty Search

Novelty search was implemented using NCD and an archive. Each generation the diversity of the agents was calculated comparing them against the current population along with an archive of previous novel behaviors. The archive was the same size of the team population. At the end of a generation the ten individuals with highest novelty were added to the archive, while the ones with lowest novelty in the archive where removed from it.

4.4 Training Parameters

Table 4.6 shows the parametrization used to train the SBB teams. Tables 4.7 and 4.8 shows the parameters used to control the mutations. Appendix A contains more details on the training performance.

Table 4.6: Parameters used during training.

Parameter	Value
Runs	25
Generations	300
Training Points	600
Validation Points	1200
Champion Points	2400
Sample of Inputs for Bid Profile	1200
Teams	100
Team Replacement Rate	0.5
Hand Replacement Rate	0.2
Selection Type	Uniform
Team Size	Min: 2, Max: 16
Program Size	Min: 5, Max: 40
Total Registers	5
Reproduction	Mutation

Table 4.7: Mutation parameters for teams.

Team Mutation	Value
Add/Del Program	0.7
Mutate Program	0.2

Table 4.8: Mutation parameters for programs.

Program Mutation	Value
Add/Del Instruction	0.5
Random Change Instruction	1.0
Swap Instruction	1.0
Random Change Action	0.1

4.5 Summary

In this work poker agents are evolved as teams of programs using the Symbiotic Bid-based (SBB) genetic programming (GP) framework and reinforcement learning. In this framework teams coevolve with programs in order to perform task decomposition and deal with complex tasks. At each generation the team population play against the point population (i.e. game matches), and is evaluated in terms of chips won, i.e. their score. A 0.0 score means a team lost all chips it would be possible to lose, 0.5 means a break-even, and 1.0 means they won everything they could win. At the end of each generation the best teams are selected to the next generation and to generate offspring via mutation. The best teams are defined using Pareto Dominance, so both fitness and diversity are taken in consideration.

The SBB framework was adapted to the poker task using reinforcement learning and a few modifications, including: environmental inputs, opponents, and hand balancing. There were implemented 5 inputs about the game state, and 9 modeling the opponent. Three types of opponents were implemented. The first group, the static opponents, are rule-based models that represent the four classical poker playing styles (Loose Aggressive, Loose Passive, Tight Aggressive, and Tight Passive). The second type is the Bayesian opponent, a more complex player that is able to adapt to and counter-attack its opponent's strategy. It is implemented using Bayesian probabilities to infer the opponent's strategy, and acts using anti-players trained against each of the classical playing styles. The last group, the Hall of Fame, is available in order to avoid evolutionary forgetting and to enable the teams to go against opponents as complex as themselves.

Chapter 5

Experiments and Results

5.1 Experiments

The following case studies will be performed and compared. There are three groups, the first one's goal is to compare the performance of teams that were and were not trained against a complex opponent, i.e. the Bayesian opponent. The second group analyzes how diversity impacts the individual and collective performance of the evolved teams for four configurations that compared various degrees of diversity. The third group is a more detailed take on diversity, and compares nine models regarding how they modified the agents' diversity and performance. The configurations in the opponent complexity group were trained using maximum diversity (NCD and genotype). All the configurations in the last two groups were trained against static opponents and HoF (not against the Bayesian opponent). More information about the diversity measures can be found in Section 4.3 and about the opponents in Section 4.2.4.

1. Opponent Complexity Group
    - trained on static opponent pool and HoF;
    - trained on static opponent pool, HoF and Bayesian opponents.
2. Degrees of Diversity Group
    - no diversity;
    - bid diversity only;
    - diversity maintenance (genotypic and behavioral (NCD)) only;
    - bid diversity and diversity maintenance (genotypic and behavioral (NCD)).
3. Diversity Models Group

- Agents trained only with fitness as an objective;
- Agents trained with fitness as an objective, and bid diversity;
- Agents trained with fitness and novelty (NCD) as objectives;
- Agents trained only with novelty (NCD) as an objective;
- Agents trained with fitness and behavioral diversity (NCD) as objectives;
- Agents trained with fitness, behavioral diversity (NCD) and genotypic diversity as objectives;
- Agents trained with fitness and behavioral diversity (Hamming) as objectives;
- Agents trained with fitness, behavioral diversity (Hamming) and genotypic diversity as objectives;
- Agents trained with fitness and genotypic diversity as objectives.

The test scenarios consisted of 1260 balanced and 1260 unbalanced hands, against the static opponents (LA, LP, TA, TP) and the Bayesian opponent (more details on Sections 4.2.4 for opponents and 4.2.3 for hand balance). The comparisons between the models are illustrated via the concept of cumulative population wide performance [10], i.e. cumulative plots. Cumulative plots contain two curves, the lower curve is the distribution of the scores per individual agents. The upper curve contains the 'cumulative' scores, so only the best scores per point are considered to calculate it. In both curves the agents are ranked by their individual score. Extra lines are added to these charts to provide contextual information: draw (the break-even point), random player score (a player that just randomly selects actions), and Bayesian opponent score.

As explained in Section 4.2.1, the performance of the teams per hand is evaluated regarding their score, i.e. the average chips won considering the best (1.0) and the worst (0.0) possible outcomes, where 0.5 means that neither side won or lost chips. Besides the results per group, the behavior and properties of trained teams are analyzed regarding their aggressiveness, strategies, and inputs used at the end of this chapter.

5.2 Results for Opponent Complexity Group

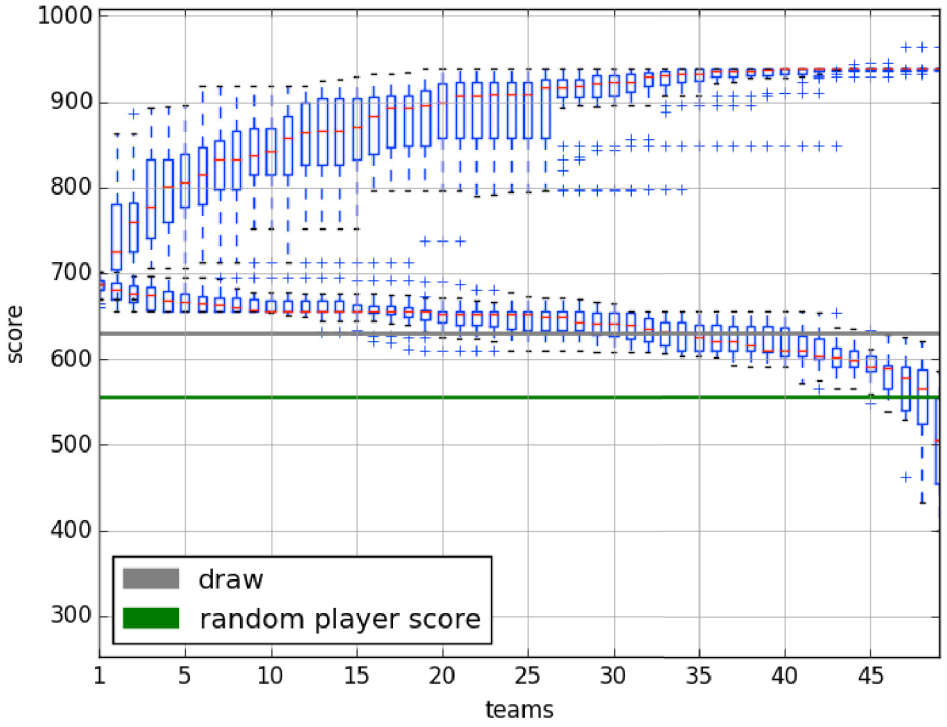
The first experiments compared the cumulative curves between teams trained and not-trained against the Bayesian opponent. The expectation was that teams trained against a more complex opponent would be more complex themselves, and thus would be able to outperform the non-Bayesian-trained teams.

A (non-parametric) Mann-Whitney U test with $p \leq 0.05$ was used to compare the distributions of the best individual performances and the cumulative performances for the 25 runs in all the test scenarios. There was no statistical difference between any of the scenarios (a sample against the Bayesian opponent for unbalanced hands is shown in Figure 5.1). It means that even teams trained against less dynamic opponents were able to perform just as well against the smartest opponent as the teams that trained against it. While the initial hypothesis was not proved correct, these results suggest that the evolved teams have a good ability to generalize their strategies, and thus beat even opponents that they have not seen before. Since these agents groups are equivalent, for the next sections the focus will be only on the results of the teams that were not trained against the Bayesian opponent.

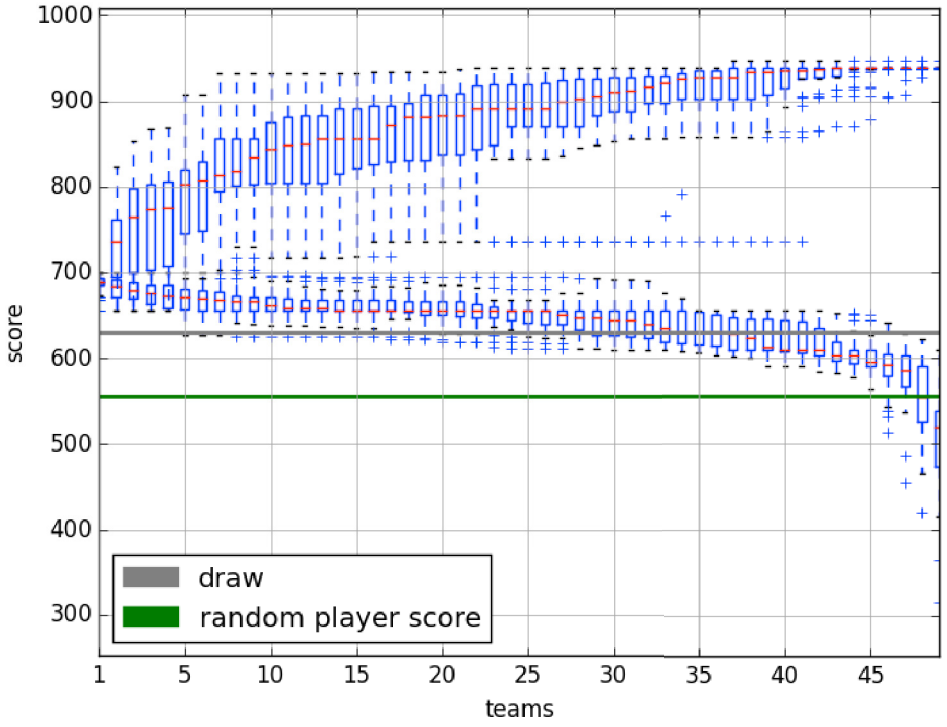
5.3 Results for Degrees of Diversity Group

The first configuration involves the teams trained in a setting with no diversity, neither diversity maintenance nor bid diversity. Samples of the results for this configuration are shown in Figure 5.2, against the Bayesian and the LA opponent with unbalanced hands. The cumulative performance distributions are nearly flat, indicating a clear lack of diverse strategies amongst the teams (otherwise they would have been able to perform better in different types of hands).

Figure 5.3 shows the cumulative curves for teams that were trained in the exact same configuration as the previous one, but with the addition of bid diversity (i.e. offspring is mutated until its pattern of actions is different from the other teams in the population). There is a slight improvement in the best individual performance and a very perceptible increase in the cumulative curves against both opponents. This improvement in both individual and cumulative curves is even greater after adding genotype and behavioral diversity metrics, as shown in Figure 5.4. The charts for the

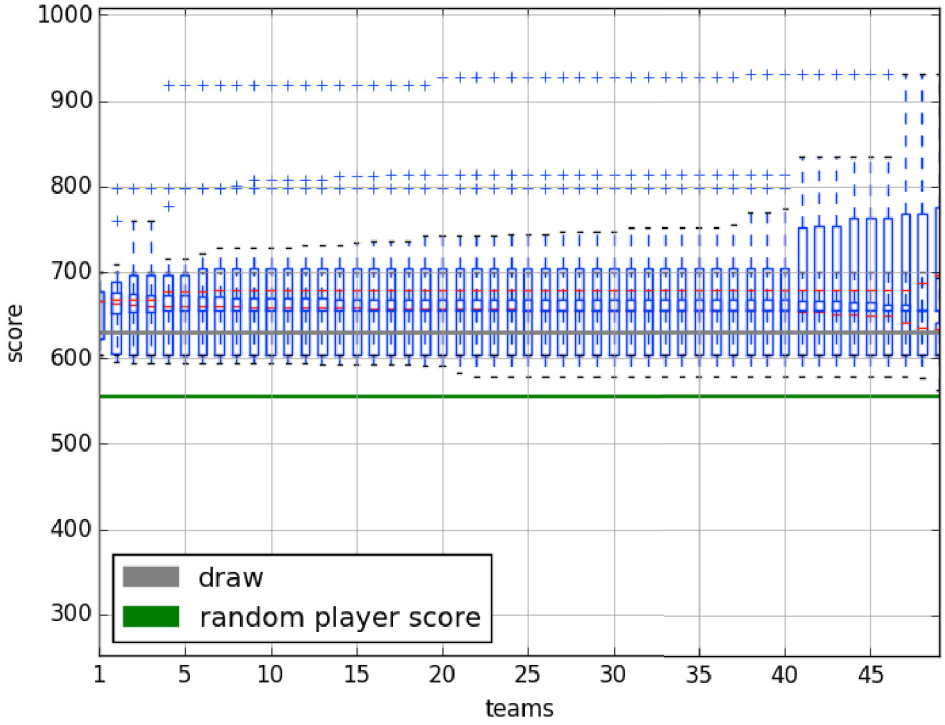


(a) Teams trained against Bayesian opponent

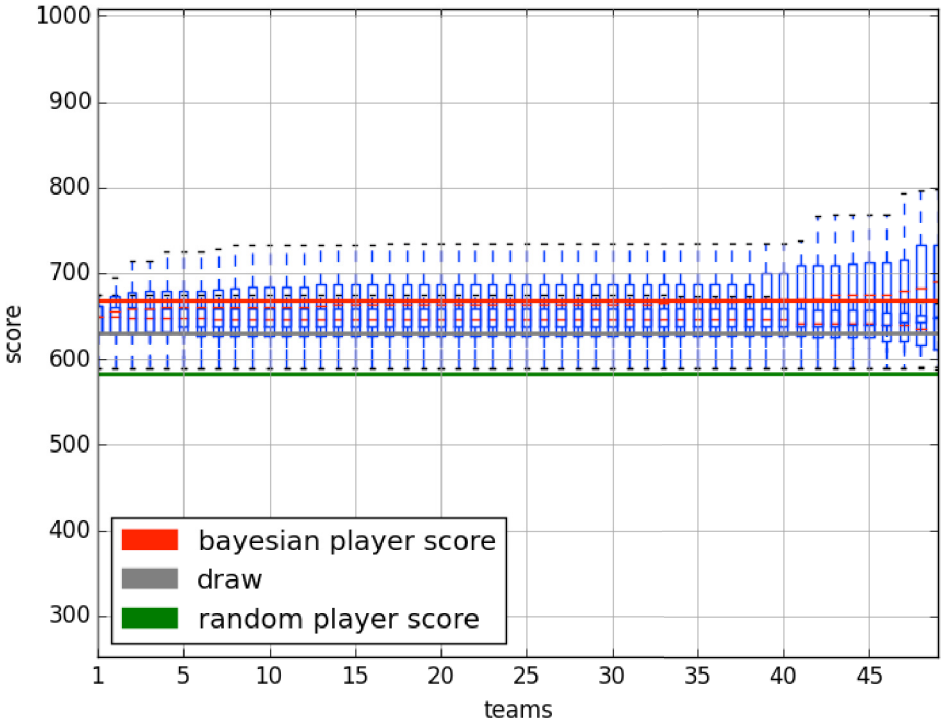


(b) Teams not trained against Bayesian opponent

Figure 5.1: Comparison between teams trained and not trained against the Bayesian opponent for 1260 unbalanced hands against the Bayesian opponent.



(a) Against Bayesian opponent



(b) Against LA opponent

Figure 5.2: Cumulative curves of SBB teams trained with no diversity, for 1260 unbalanced hands.

Table 5.1: Means of best individual and cumulative scores against the Bayesian opponent in unbalanced hands, for three scenario cases: no diversity, bid diversity only, and diversity maintenance (genotypic and behavioral).

Diversity Type	**Mean Best Individual**	**Mean Cumulative**
no diversity	655.145	710.797
bid diversity only	671.652	790.226
diversity maintenance	687.00	938.578

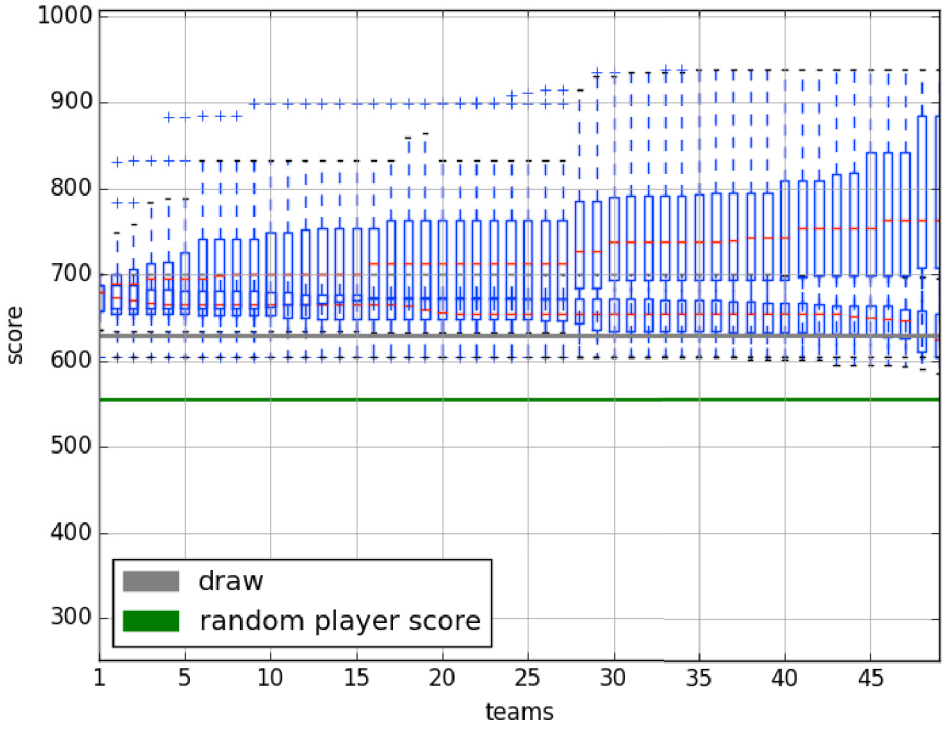
configuration with diversity maintenance but not bid diversity are not shown since these results ended up being statistically equivalent to the ones without bid diversity (for $p \leq 0.05$, Mann-Whitney U Test). It seems that while bid diversity improves diversity, its benefits are overshadowed by the use of diversity metrics.

It is clear that teams trained using diversity measures perform better not only individually, but even more when they work collectively, increasing their ability to obtain more chips from a variety of hands. Table 5.1 contains the means of best individual performances and cumulative performances across the 25 runs for the scenario against the Bayesian opponent for unbalanced hands (significant for $p \leq 0.05$, Mann-Whitney U Test). For both metrics, an increase in diversity measures led to an increase in best individual and cumulative scores. The next section explores diversity maintenance for more models and scenarios.

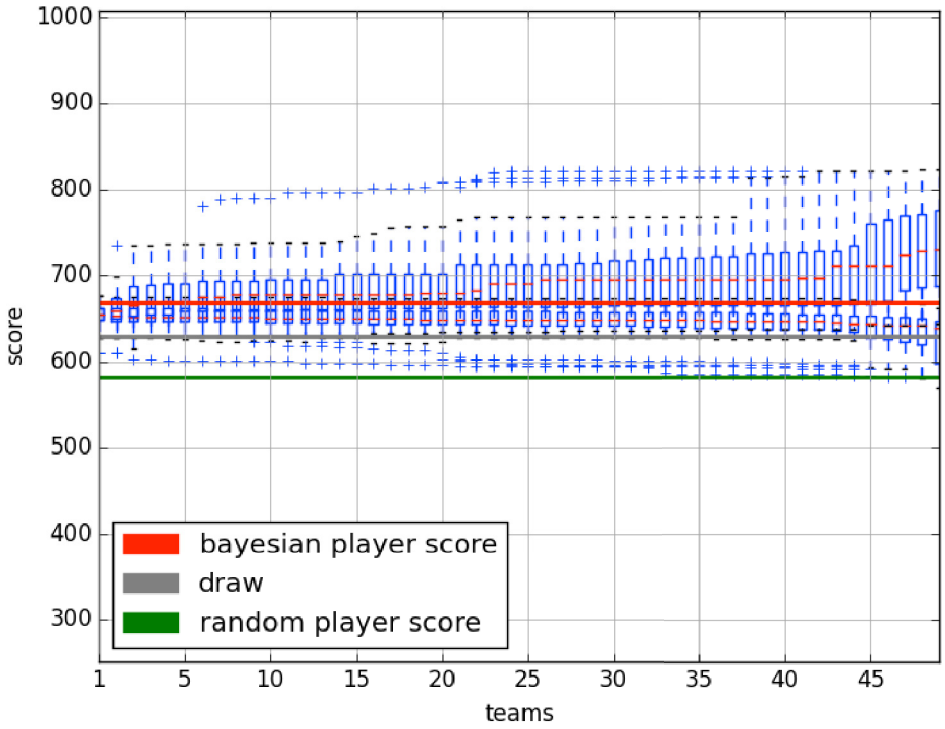
5.4 Results for Diversity Models Group

This experiment compares models with various degrees of diversity and diversity maintenance implementation. The goal is to answer the main question of this thesis: analyze if behavioral diversity maintenance and novelty search are still beneficial under a task that posses a lot of ambiguity and a huge behavior space. And more specifically, which models increase diversity and if this increased diversity leads to a better performance. 'Diversity' is compared using the distributions for the cumulative scores in the cumulative plots. 'Performance' is analyzed in terms of the distributions of the best individual scores in the cumulative plots. The models are executed in ten scenarios, for the combination of the five opponents and two hand balances. The following models are compared:

no_div Agents trained only with fitness as an objective;

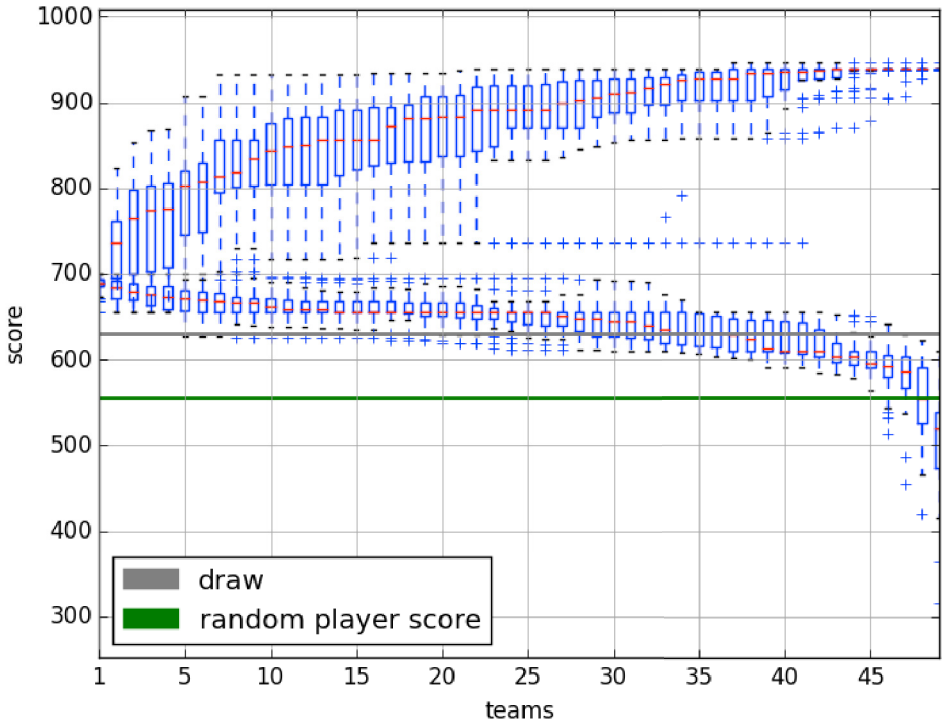


(a) Against Bayesian opponent

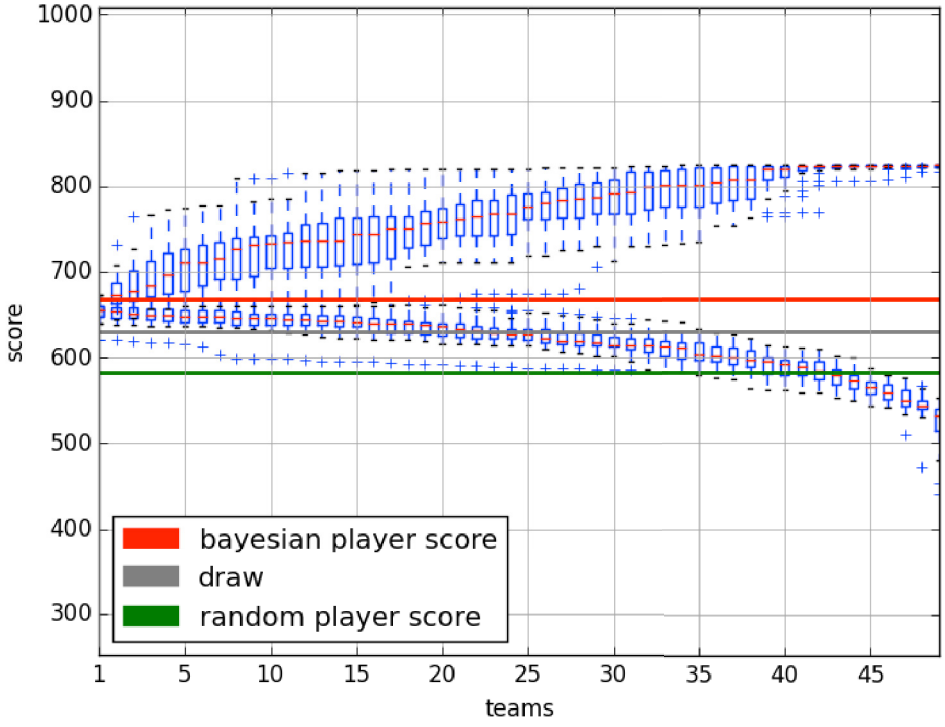


(b) Against LA opponent

Figure 5.3: Cumulative curves of SBB teams trained with bid diversity, for 1260 unbalanced hands.



(a) Against Bayesian opponent



(b) Against LA opponent

Figure 5.4: Cumulative curves of SBB teams trained with diversity maintenance (genotypic and behavioral (NCD)), for 1260 unbalanced hands.

bid Agents trained with fitness as an objective, and bid diversity;

nov Agents trained with fitness and novelty (NCD) as objectives;

only_nov Agents trained only with novelty (NCD) as an objective;

ncd Agents trained with fitness and behavioral diversity (NCD) as objectives;

ncd_gen Agents trained with fitness, behavioral diversity (NCD) and genotypic diversity as objectives;

ham Agents trained with fitness and behavioral diversity (Hamming) as objectives;

ham_gen Agents trained with fitness, behavioral diversity (Hamming) and genotypic diversity as objectives;

gen Agents trained with fitness and genotypic diversity as objectives.

The abbreviations used to describe the ten scenarios are xY, where x represents the hand balance (b: balanced, u: unbalanced) and Y represents the opponent (B: Bayesian, LA: Loose Aggressive, LP: Loose Passive, TA: Tight Aggressive, TP: Tight Passive). The results for all the models against these scenarios are summarized in Table 5.2, for best individual score, and Table 5.3, for cumulative score. The scores were normalized between the minimum and maximum possible values. The last line of the tables contains the models' rank, where in this case a lower rank implies that the model consistently performs better over multiple types of opponent and hand balance. A non-parametric Friedman test [20] was applied to both tables, and resulted that the models are significantly different for $p \leq 0.01$. Additional information is available in Appendix B, with the rank version of Table 5.2 and Table 5.3, and samples of the cumulative charts of all models against the Bayesian opponent for balanced and unbalanced hands.

Since the null-hypothesis was rejected, we proceeded with two post-hoc tests: The Bonferroni-Dunn test was used to compare the models against the one with no diversity (no_div), and the Nemenyi test was used to compared all the models to each other.

For the two-tailed Bonferroni-Dunn test ($p \leq 0.05$, critical distance: 3.336) ncd_gen, nov, ham, ncd, gen, and ham_gen obtained a better diversity (cumulative score) than

Table 5.2: Comparison of the median normalized best individual scores for models with no diversity and with various types of diversity maintenance. Nine models are compared in ten scenarios (b and u marks the type of hand balance, and B, LA, LP, TA, and TP are types of opponents.

	no_div	bid	nov	only_nov	ncd	ncd_gen	ham	ham_gen	gen
bB	0.5	0.503	0.498	0.484	0.495	0.51	0.498	0.511	0.507
bLA	0.505	0.508	0.514	0.471	0.51	0.511	0.515	0.513	0.508
bLP	0.522	0.521	0.529	0.472	0.524	0.527	0.528	0.529	0.518
bTA	0.513	0.516	0.517	0.511	0.517	0.513	0.518	0.52	0.521
bTP	0.516	0.521	0.519	0.509	0.518	0.521	0.52	0.521	0.521
uB	0.529	0.538	0.503	0.477	0.502	0.544	0.539	0.545	0.549
uLA	0.515	0.519	0.519	0.475	0.516	0.519	0.518	0.52	0.521
uLP	0.532	0.533	0.532	0.481	0.529	0.532	0.537	0.533	0.527
uTA	0.51	0.512	0.512	0.509	0.512	0.514	0.513	0.514	0.514
uTP	0.513	0.515	0.514	0.509	0.513	0.514	0.535	0.516	0.515
rank	6.9	4.75	5	9	6.5	3.15	3.9	2.15	3.65

Table 5.3: Comparison of the median normalized cumulative scores for models with no diversity and with various types of diversity maintenance. Nine models are compared in ten scenarios (b and u marks the type of hand balance, and B, LA, LP, TA, and TP are types of opponents.

	no_div	bid	nov	only_nov	ncd	ncd_gen	ham	ham_gen	gen
bB	0.526	0.567	0.626	0.603	0.602	0.691	0.626	0.691	0.691
bLA	0.542	0.579	0.688	0.687	0.689	0.69	0.689	0.69	0.689
bLP	0.559	0.594	0.687	0.676	0.686	0.688	0.687	0.686	0.687
bTA	0.524	0.535	0.567	0.567	0.567	0.567	0.566	0.565	0.563
bTP	0.527	0.541	0.566	0.563	0.565	0.565	0.564	0.563	0.561
uB	0.552	0.606	0.601	0.599	0.6	0.744	0.744	0.744	0.744
uLA	0.548	0.579	0.653	0.65	0.652	0.654	0.652	0.653	0.653
uLP	0.566	0.605	0.651	0.641	0.651	0.652	0.651	0.651	0.651
uTA	0.518	0.525	0.542	0.542	0.542	0.542	0.541	0.54	0.539
uTP	0.52	0.53	0.541	0.54	0.541	0.541	0.54	0.54	0.539
rank	9	7.7	3.3	5.9	4.2	2.2	3.95	4.45	4.3

no_div. And for this same test on performance (best individual score), only ham_gen and ncd_gen were better than no_div. Not all models that significantly improved diversity improved the performance, but the only ones that significantly improved the performance were the ones that improved the diversity. While these results present that it is not guaranteed that an increase in diversity will lead to an increase in performance, diversity is shown to be essential to obtain a better performance. Additionally, the only configurations were diversity lead to improved performance were the ones combining fitness, behavioral diversity and genotypic diversity. These three objectives seem to be essential to evolve diverse and strong agents for the Hold'em Texas Poker task.

The two-tailed Nemenyi test ($p \leq 0.05$, critical distance: 3.8) was used to rank the models according to which models they outperform. These results are summarized in Table 5.4, for best individual score, and Table 5.5, for cumulative score. The goal of this test was to see how novelty (only_nov and nov) compared against the other models and against themselves. The results indicate that novelty search alone (only_nov) is not a good metric for Texas Hold'em Poker. Six out of eight models outperform it for best individual performance, and the other two are equivalent. Even for diversity, where it was expected that it would obtain its best results, only_nov was not significantly better than any of the other models. The model with novelty and fitness (nov) was significantly better than only_nov. For best individual performance it was better in a direct comparison (Table 5.4). And, in an indirect comparison on the previous Bonferroni-Dunn test, it obtained a better cumulative score than no_div, while only_nov did not. These results imply that even for a deceptive and ambiguous task such as poker, fitness is a necessary objective in order to tackle its complexity. However, even with nov having better results than only_nov, the only models were diversity lead to improved performance in relation to no_div did not use novelty (ham_gen and ncd_gen).

5.5 Properties of the Trained Teams

This section analyzes properties of the trained teams, such as their strategies regarding aggressiveness, bluffing, and hands played, and how the teams used the inputs available to them. Overall, teams bluffed much more in unbalanced scenarios than

Table 5.4: Model comparisons for best individual score, with Nemenyi test for $p \leq 0.05$.

Model (Best Individual Score)	**Is significant better than...**
ham_gen	ncd, no_div, only_nov
ncd_gen, gen, ham, bid, nov	only_nov
ncd, no_div, only_nov	none

Table 5.5: Model comparisons for cumulative score, with Nemenyi test for $p \leq 0.05$.

Model (Cumulative Score)	**Is significant better than...**
ncd_gen, nov	bid, no_div
ham, ncd, gen, ham_gen	no_div
no_div, bid, only_nov	none

in balanced ones, which is expected given the higher amount of weak hands. This behavior enabled them to exploit the opponent's weaker hands. Additionally, in these scenarios the teams played slightly tighter strategies, to avoid being exploited themselves. Both these properties enabled them to obtain higher scores in the unbalanced, real-world, scenario. Unless stated otherwise, the agents analyzed in this section were trained against the classical opponents and the HoF, using fitness, and genotypic and behavioral diversity (NCD) as objectives. Appendix C contains, as a matter of curiosity, details on sampled trained teams as, for example, their specific performance, and strategy.

5.5.1 Novelty and Bluffing

Chapter 3 formulated the hypothesis that novelty would incentive bluffing. This would occur since novelty allows strategies that are not immediately beneficial (or that are temporally detrimental) to evolve, leading to more complex strategies. Bluffing is an example of such a case, since before learning how to effectively bluff, it may result in the player being exploited in weak hands. This was partially confirmed in our results, as demonstrated in the sample shown in Figure 5.5. These charts show the distribution of the use of bluffing across the models for balanced and unbalanced hands against the Bayesian opponent. Table 5.6 contains the data for bluffing across the ten scenarios. The data was analyzed via Friedman test, and the results were shown to be significantly different for $p \leq 0.01$. Two-tailed Bonferroni-Dunn was used as a post-hoc test, to compare the model only_nov against the others. For a

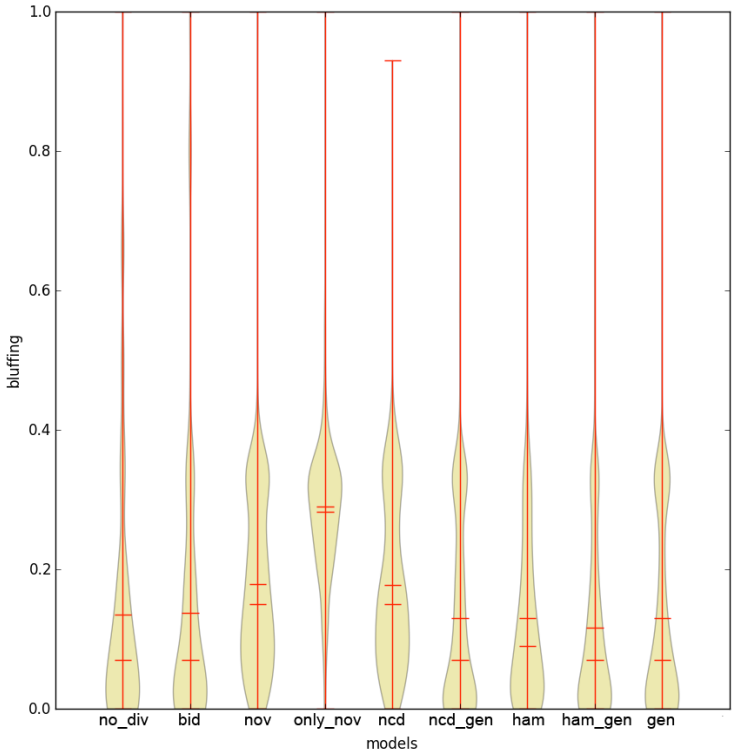
Table 5.6: Comparison of the median rate of bluffing for the models. Nine models are compared in ten scenarios (b and u marks the type of hand balance, and B, LA, LP, TA, and TP are types of opponents.

	no_div	bid	nov	only_nov	ncd	ncd_gen	ham	ham_gen	gen
bB	0.07	0.07	0.15	0.29	0.15	0.07	0.09	0.07	0.07
bLA	0.129	0.075	0.154	0.291	0.146	0.101	0.1	0.101	0.101
bLP	0.215	0.162	0.204	0.3	0.188	0.157	0.161	0.172	0.17
bTA	0.25	0.132	0.221	0.3	0.186	0.172	0.14	0.162	0.146
bTP	0.296	0.234	0.26	0.303	0.23	0.235	0.195	0.225	0.22
uB	0.25	0.35	0.43	0.54	0.45	0.36	0.34	0.33	0.36
uLA	0.418	0.401	0.423	0.524	0.44	0.427	0.399	0.424	0.417
uLP	0.531	0.509	0.494	0.546	0.492	0.504	0.478	0.508	0.507
uTA	0.567	0.423	0.5	0.537	0.478	0.491	0.444	0.461	0.45
uTP	0.587	0.522	0.528	0.547	0.511	0.535	0.495	0.518	0.537
Rank	3.6	6.9	3.65	1.2	4.45	5.25	7.9	6	6.05

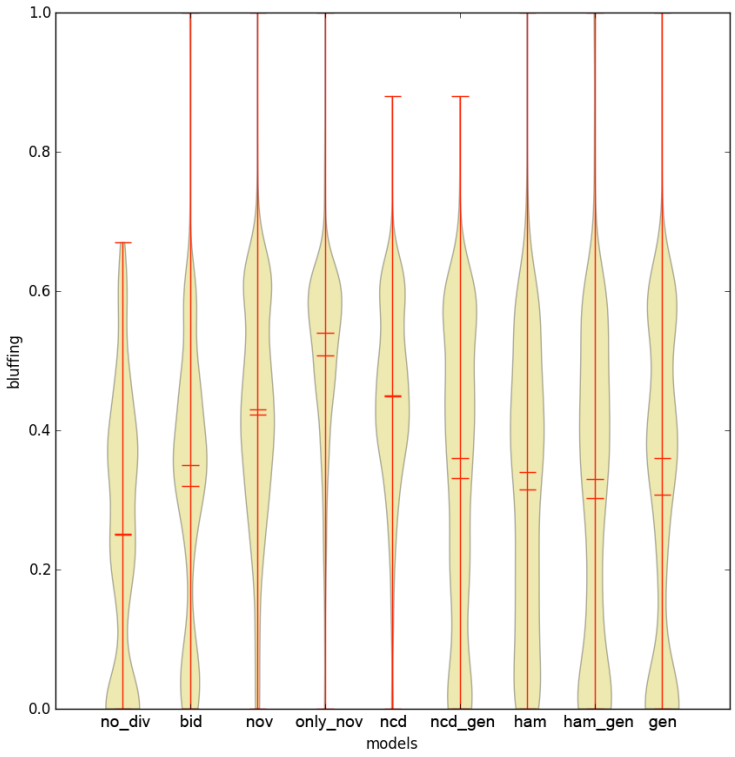
significance of $p \leq 0.05$, only_nov bluff was as effective as no_div, nov, and ncd, and bluff significantly more than the other five models (ham, bid, gen, ham_gen, and ncd_gen).

5.5.2 Playing Styles

The evolved agents played with diverse strategies. Figure 5.6 shows the playing styles of all teams evolved in the 25 runs on the axis of passive/aggressive and tight/loose, where 0 indicates a more passive or tight behavior, and 1 a more aggressive or loose one. The charts show the teams for the scenarios with balanced and with unbalanced hands. The lines mark the mean of the values in an axis. There is a preference towards slightly more passive strategies (ie. more calls than raises) in both scenarios. Also, the teams switch to a tighter strategy (ie. fold more hands before the flop round) in the unbalanced points scenario, in order to deal with their own weaker hands. Figure 5.7 demonstrates the relationship between theses strategies and the scores obtained, i.e. the chips gained, where 0.5 means no one won chips. While in the balanced scenario the teams performed near the break-even point, in the unbalanced, real-world scenario, the majority of the teams performed significantly better than the Bayesian opponent. While there are tendencies for the types of behaviors for the majority of the teams, they are diverse enough to be plotted in the whole extension



(a) Balanced hands



(b) Unbalanced hands

Figure 5.5: Distribution of bluffing for 1250 evolved teams across 25 runs, for 1260 hands against the Bayesian opponent.

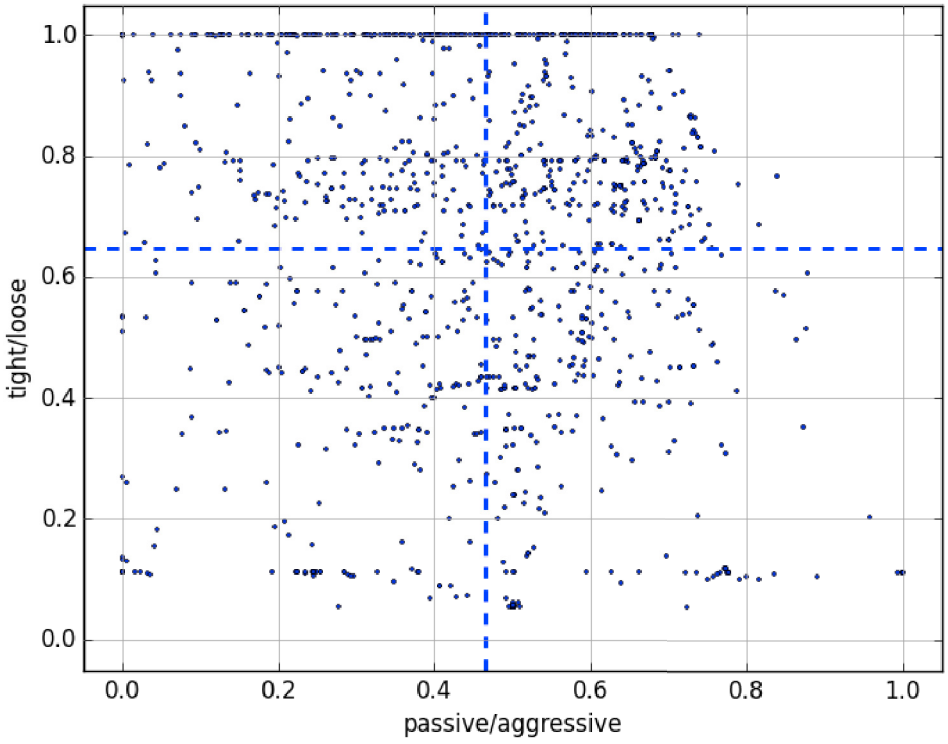
of the axis for passive/aggressive and tight/loose strategies.

Besides changing to tighter strategies, Figure 5.8 shows that in the unbalanced scenario the teams bluff more, and win more chips from bluffing. Here a team is considered to be bluffing if they are at the river (the last round, so all public cards are known) with weak cards, but keep playing the round until the end (showdown or opponent fold). So, while the teams play less hands to avoid losing chips due to weaker hands, they also increase their bluffing, to exploit the opponent's weaker hands. It can be speculated that the teams are using their opponent modeling inputs to find out when the opponent seems to have weaker hands, and then bluff.

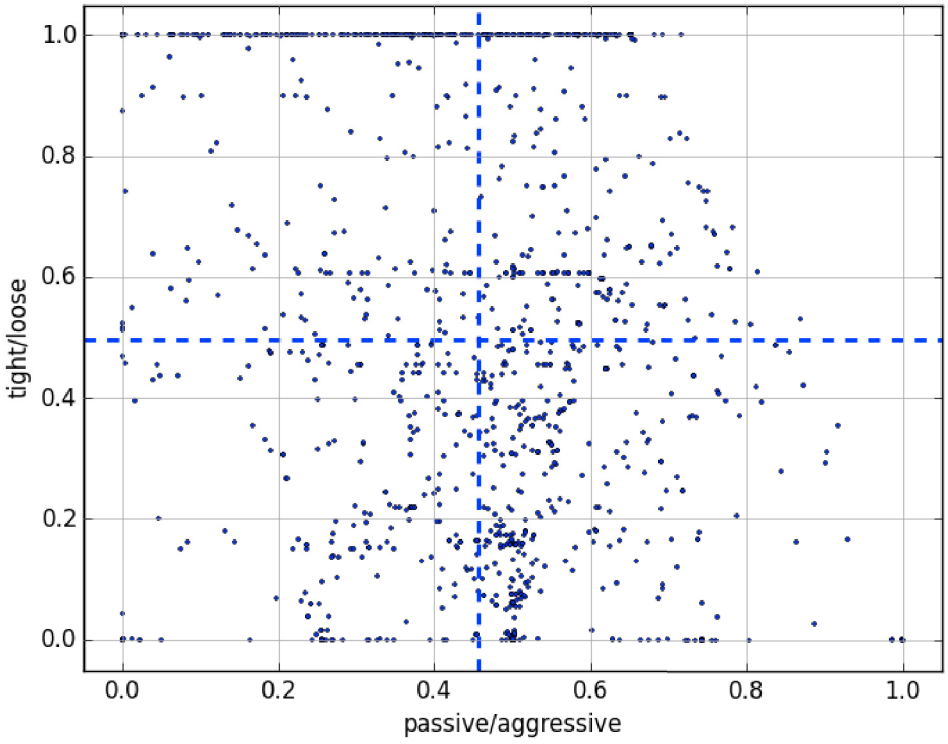
Finally, in terms of score the agents in the unbalanced scenario (real-world), obtained higher best individual and cumulative scores than in the balanced scenario (for a (non-parametric) Mann-Whitney U test with $p \leq 0.01$). However, as shown in Figure 5.9, the teams won more hands in the balanced scenario (i.e. obtained a positive outcome of chips in more hands). This indicates that the higher score in the unbalanced scenario is being obtained by exploiting more chips of the fewer hands won.

5.5.3 Cumulative Wins

Besides cumulative curves for scores, cumulative curves for wins were generated. By 'win' we mean hands where the player obtained a positive outcome, i.e. a score higher than 0.5. A won hand is worth 1 point, while a draw hand is 0.5. Interestingly, in the same scenarios where there is diversity in terms of score, there is no diversity in terms of hands won, as shown in Figure 5.10. This means that the best performing individual already wins the maximum amount of hands that could be won (Figure 5.10 subplots (b), (d), (f)), and adding new teams does not increase the capacity to analyze the hand's values and risks (if it did, the new teams would be able to play and win more 'winnable' hands). So the main gain from diversity is not winning more hands, but the better exploitation of hands in order to be able to gain more chips per hand (Figure 5.10 subplots (a), (c), (e)).

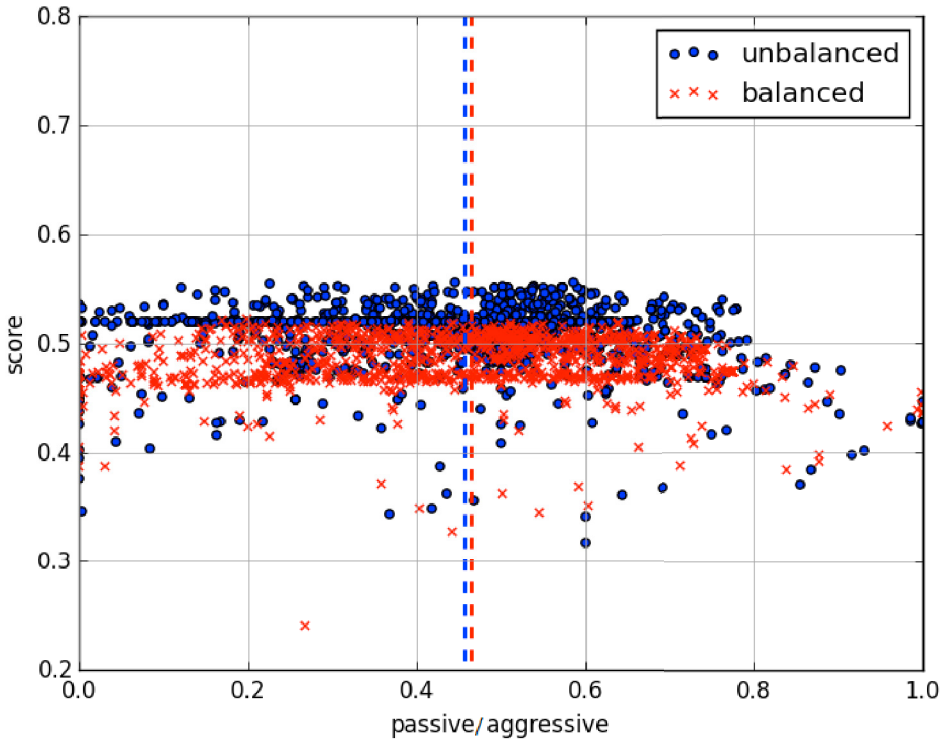


(a) Balanced scenario

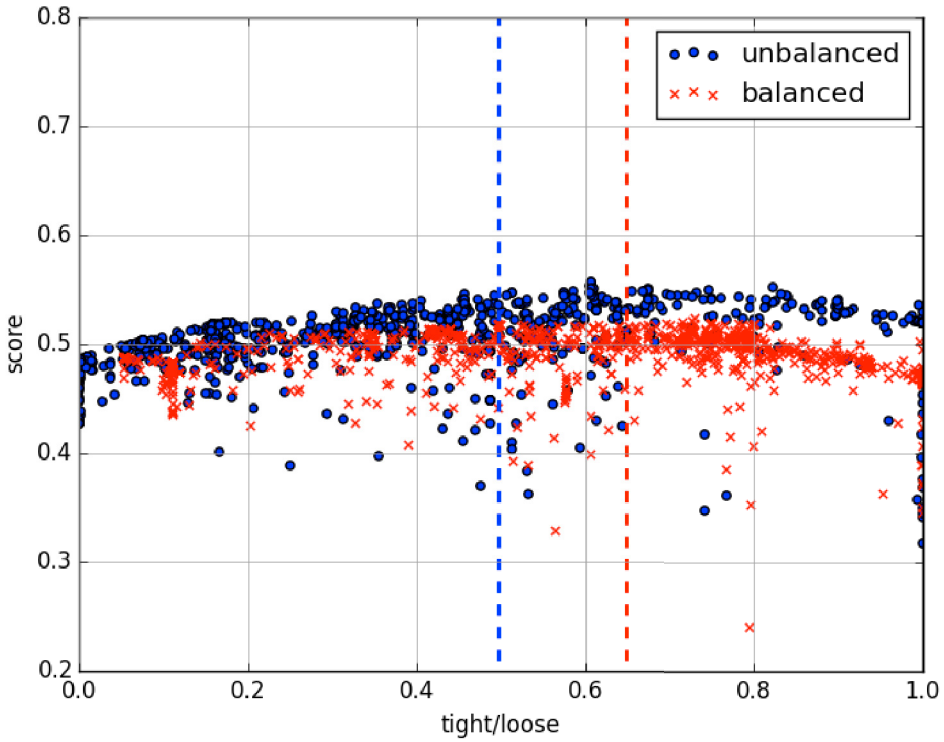


(b) Unbalanced scenario

Figure 5.6: Behaviors of the evolved teams against the Bayesian opponent in 1260 hands across 25 runs, for a total of 1250 teams. The lines mark the means of the values.



(a) Strategies on Passive/Aggressive axis



(b) Strategies on Tight/Loose axis

Figure 5.7: Strategies vs Score (chips gained) against the Bayesian opponent in 1260 hands across 25 runs, for a total of 1250 teams. The lines mark the means of the values.

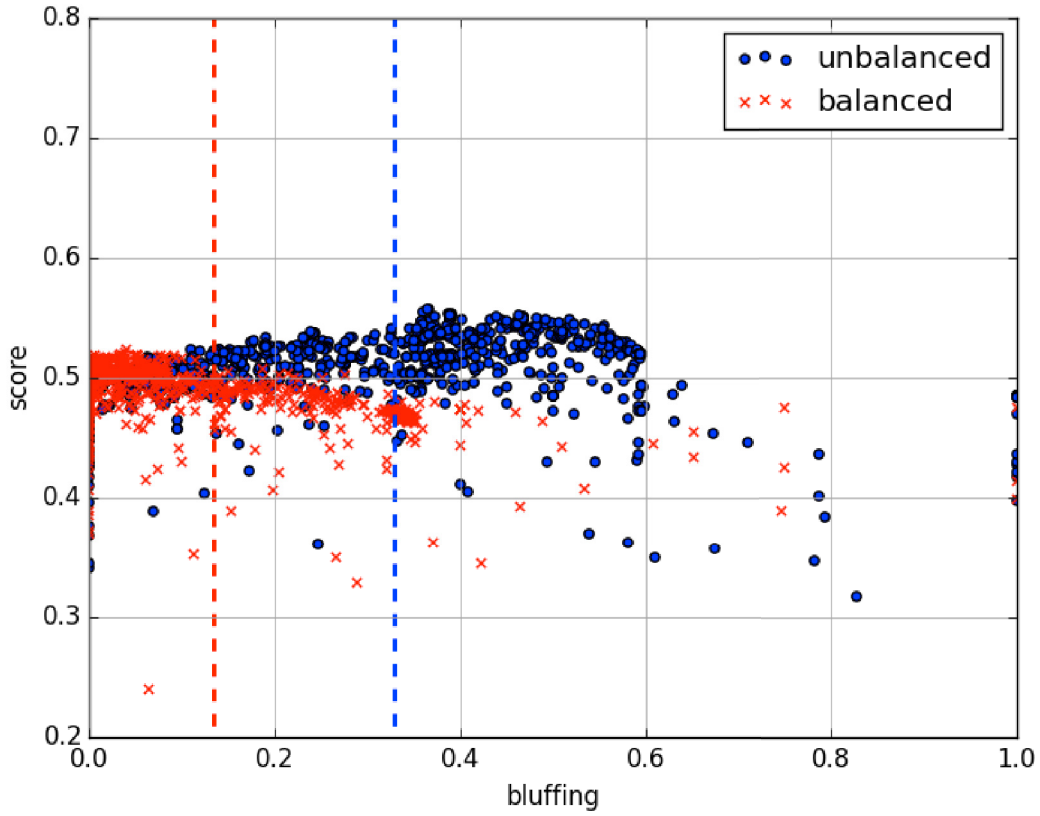


Figure 5.8: Bluffing rates vs Score (chips gained) against the Bayesian opponent in 1260 hands across 25 runs, for a total of 1250 teams. The lines mark the means of the values.

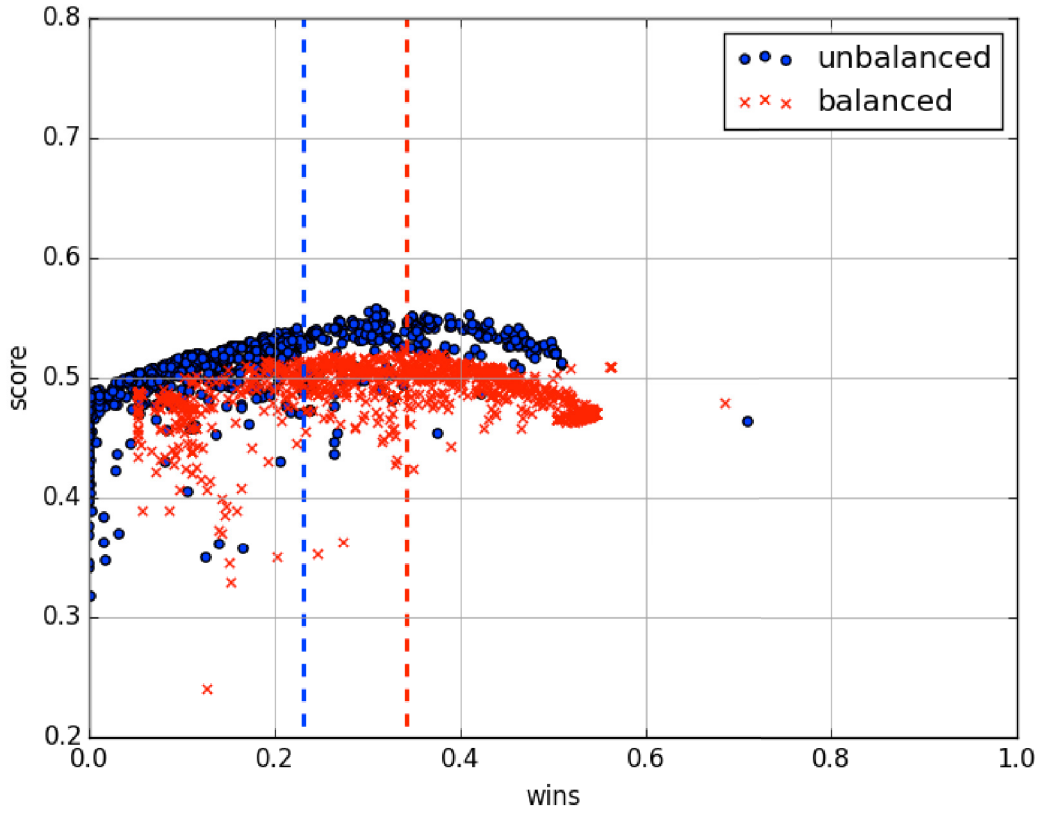
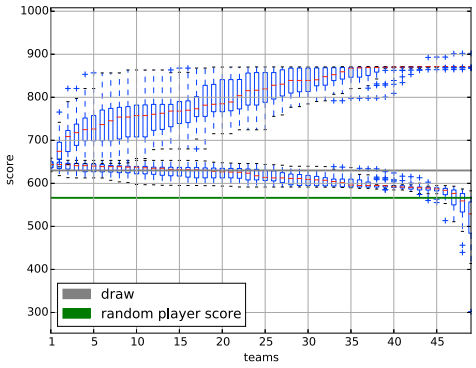
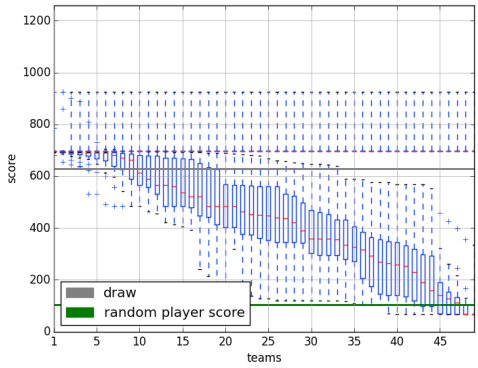


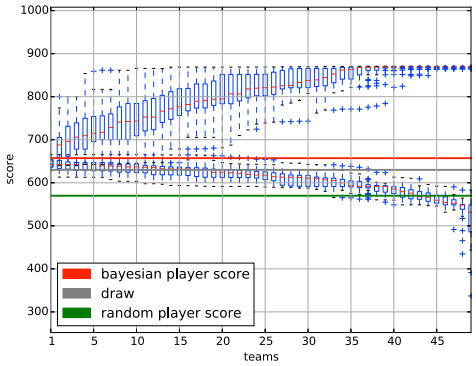
Figure 5.9: Win rates vs Score (chips gained) against the Bayesian opponent in 1260 hands across 25 runs, for a total of 1250 teams. The lines mark the means of the values.



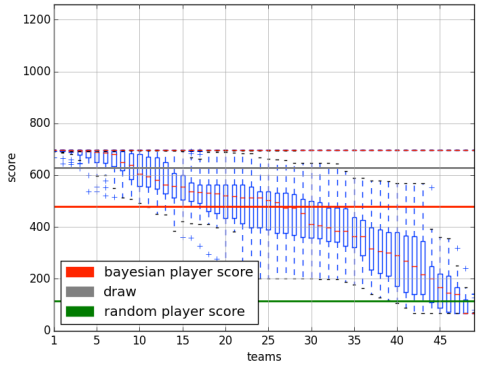
(a) Cumulative score, Bayesian opponent



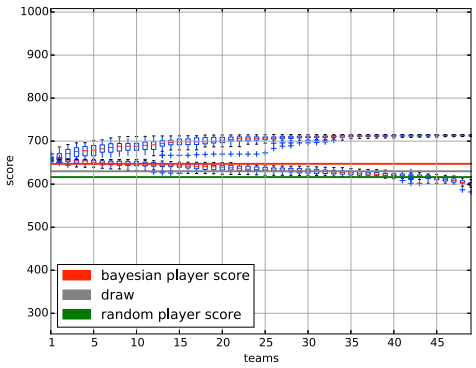
(b) Cumulative wins, Bayesian opponent



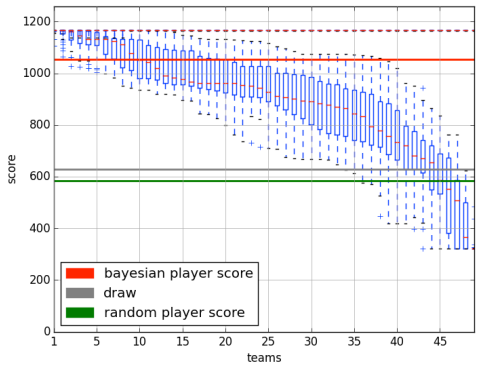
(c) Cumulative score, LA opponent



(d) Cumulative wins, LA opponent



(e) Cumulative score, TP opponent



(f) Cumulative wins, TP opponent

Figure 5.10: Comparison between cumulative score and cumulative wins, with 1260 balanced hands.

5.5.4 Most Used Inputs

Table 5.7 lists the most used inputs by the percentage of teams that had at least one program that used it. These results were obtained at the end of the training. The most used inputs were pot odds and effective potential, relevant to assess risks and hand value, followed by inputs dedicated to model the opponent. This agrees with the previous results on opponent modeling [5, 24, 2, 3], showing that it indeed is being important to the teams. The teams can use these inputs to predict the strength of the opponent's hands by observing its overall behavior.

A few of the inputs were nearly redundant, but they were available so the teams could 'choose' the ones that worked better. So it is interesting to see that more teams used 'effective potential' than 'hand strength', and that inputs that modeled the opponent's aggressiveness on short-term were more employed than the ones for long-term behavior. An exception is made for the input for the opponent's last action. It was the least used input, so it may be the case that such an immediate input did not give enough context to the teams.

By identifying a subset of inputs as the most relevant ones, it is possible to reduce the computational cost of supporting poker NPC characters. Overall, each team used 9.8 inputs, where around 4.7 inputs were used more than once inside the team. So it is possible to speculate that by using 5 inputs, probably the ones from the top 5, NPCs would be able to obtain good enough performances.

5.6 Summary

The main conclusions from this chapter are as follows.

- Teams trained and not-trained against the most complex opponent obtained an equivalent performance due to generalization of their strategies.

- Models trained with diversity maintenance techniques were able to improve the diversity of the agents. The successful models were trained with the methods: NCD and genotype, Hamming and genotype, novelty (with fitness), NCD, Hamming, and genotype. However, only two models translated an improved diversity into an improved performance, in comparison with a model with no

Table 5.7: Most used inputs per teams, by the percentage of teams that had at least one program that used it.

#	**Input**	**% Usage**
1	pot odds	82.7
2	effective potential	80.1
3	opp passive/aggressive	73.5
4	opp hand aggressiveness	72.0
5	opp short-term aggressiveness	71.9
6	hand strength	70.9
7	opp bluffing	69.9
8	round	68.8
9	self short-term aggressiveness	68.8
10	chips	67.4
11	opp tight/loose	65.0
12	betting position	64.5
13	opp aggressiveness	64.2
14	opp last action	63.8

diversity: NCD and genotype, and Hamming and genotype. Both models used fitness, genotypic diversity, and behavioral diversity as objectives. The main conclusions are that more diverse agents does not necessarily lead to stronger agents. However, genotypic and behavioral diversity maintenance along with fitness as objectives are essential components to evolve stronger agents.

- None of the models using novelty search was capable to significantly improve the agents' performances. Among the two models tested, novelty search alone and novelty search with fitness, the one with fitness obtained better results both in terms of diversity and of performance. Again, fitness is shown to be an important objective in order to deal with the complexity of the poker task.

- Teams that evolved with only novelty search bluffed significantly more than most of the other models in ten scenarios. This argues in favor that novelty search benefits the evolution of behaviors that may not be useful from start, but that can originate more complex and useful behaviors.

- The evolved teams produced diverse behavior on the axis of tight/loose and passive/aggressive strategies. They shift their behavior from the balanced to the unbalanced scenario, playing less hands and bluffing more in the hands that

they play, in order to avoid playing weak hands themselves while exploiting the opponent's weak hands.

- The best individual teams already won the hands that could be won, thus the main role of behavioral diversity was to improve the teams' ability to exploit and win more chips from each hand.

- The top 5 most used inputs by the teams were pot odds (to assess if the risk is worth it), effective potential (to asses the strength of one's own hand), and three inputs modeling the opponent.

Chapter 6

Conclusions and Future Work

This work reviewed and compared various methods of diversity maintenance for the task of Limit Heads-up Texas Hold'em Poker. Our goal was to analyze if behavioral diversity maintenance and novelty search are still beneficial under a deceptive task that posses a lot of ambiguity. Poker contains ambiguity due to imperfect information, stochasticity, and intransitivity. It is also deceptive, due to the complex strategies necessary to perform well in the game, such as bluffing. And finally, this task contains a behavior space that is extremely large, due to its many game states and decision points. As discussed on the previous works, up until now no research on diversity and novelty had studied a task that possessed these properties.

To analyze if diversity maintenance methods were indeed beneficial, two metrics were used. The first one, the diversity, was measured by the cumulative score of the teams. The second one, the performance, was measured by the best individual scores achieved by the best teams produced by the models. Nine models were compared with various combinations of objectives, such as fitness, genotypic diversity, behavioral diversity, and novelty search. The teams were evolved using the genetic programming framework Symbiotic Bid-Based (SBB). They were tested in ten scenarios, for combinations between hand balances (balanced and real-world) and opponents (rule-based classical opponents and an adaptive Bayesian opponent).

Six out of the eight diversity maintenance methods tested were able to produce agents that were significantly more diverse than in a model with no diversity maintenance. The only models that were not able to significantly improve diversity were the ones that used behavioral bid diversity and novelty search alone. Two models were capable to transform the diversity gain into a performance gain. These two models were the ones that combined fitness, genotypic diversity, and behavioral diversity. One used Hamming distance and the other used NCD as the behavioral distances. While not all models that improved diversity were able to improve performance, the

models that used both genotypic and behavioral diversity were able to achieve this improvement, which shows that they are important diversity maintenance methods for a deceptive and complex task such as poker.

Novelty search alone ended up not being enough to improve diversity and performance. It was one of the two that was not able to improve diversity in relation to a no diversity scenario. And regarding individual performance, six out of the eight other models outperformed it, while it is equivalent with the last two models. However, by coupling novelty search with the fitness function it was possible to produce diverse strategies and, even if not one of the best, by using fitness this model obtained a better performance than the one without fitness. These results suggest that even for a deceptive and ambiguous task such as poker, fitness is a necessary objective in order to tackle its complexity.

Additionally, it was speculated that novelty search would produce agents with a stronger tendency to bluff. The idea is that novelty search orients the evolution not by the immediate best solution (fitness), but by novel behaviors. So it incentives the development of strategies that may not be immediately beneficial, but that may lead to more complex strategies, such as bluffing. This was confirmed in our results, where teams that evolved with only novelty search bluffed significantly more than most of the other models.

Teams trained and not trained against a more complex and adaptive opponent (Bayesian opponent) were able to perform just as well against it, showing that the agents are able to generalize their strategies to deal with unseen opponents. The teams were also capable to adapt their strategies from balanced hands, where they trained, to unbalanced hands, the real-world task. They actually ended up performing even better in the real-world scenario than in the scenario they were trained.

The evolved agents were shown to be diverse in various axes, such as their strategies for betting, bluffing, and choosing which hands to play. Diversity was found to be useful mainly to increase the exploitation of chips per hand, since the best individual agents ended up already winning the maximum number of hands that they could possibly win.

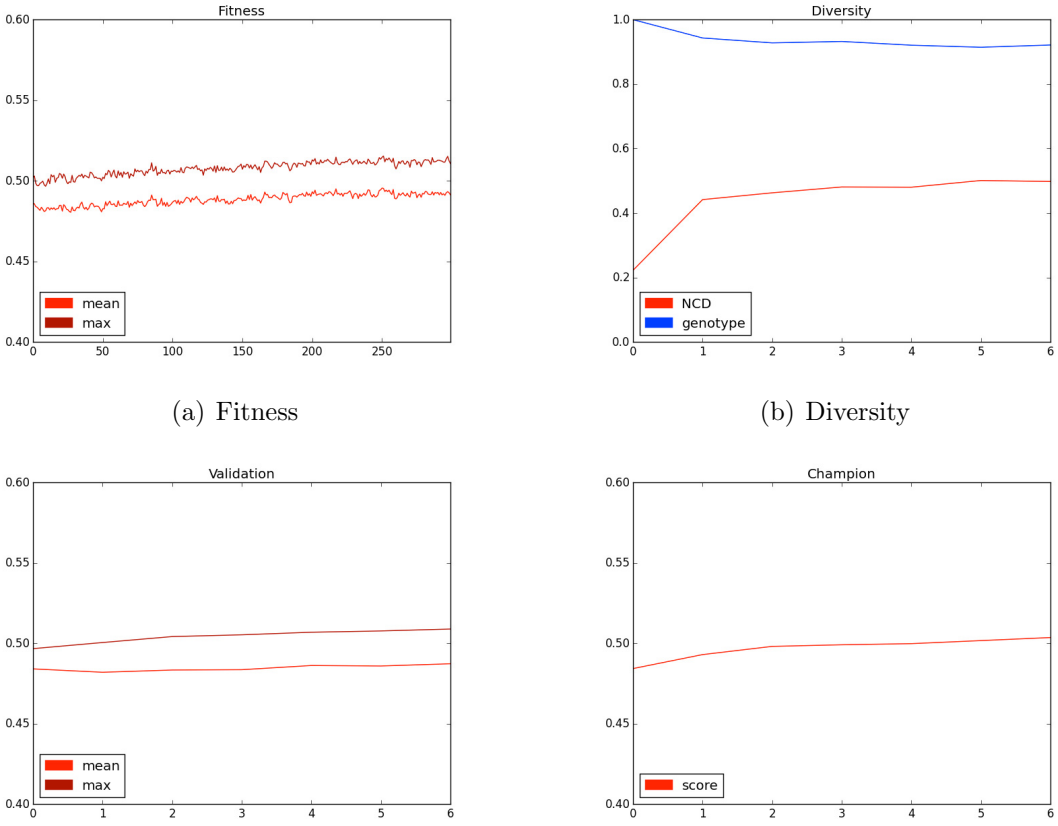
In terms of future work, the cumulative curves suggest that there is potential for deploying some subset of the Poker playing agents collectively, that may be able to

achieve an individual score as good as the cumulative score. Another path would be to further test diversity and novelty on an even more ambiguous and complex version of Poker, by adding more players and no limit on the bettings.

Appendix A

Training Performance

This appendix presents samples of the training for fitness, diversity, validation score, and champion score. Figure A.1 shows the charts for teams trained against the Bayesian opponent, while Figure A.2 shows the charts for the ones not trained against it. Both configurations included the four classical opponents and a HoF.



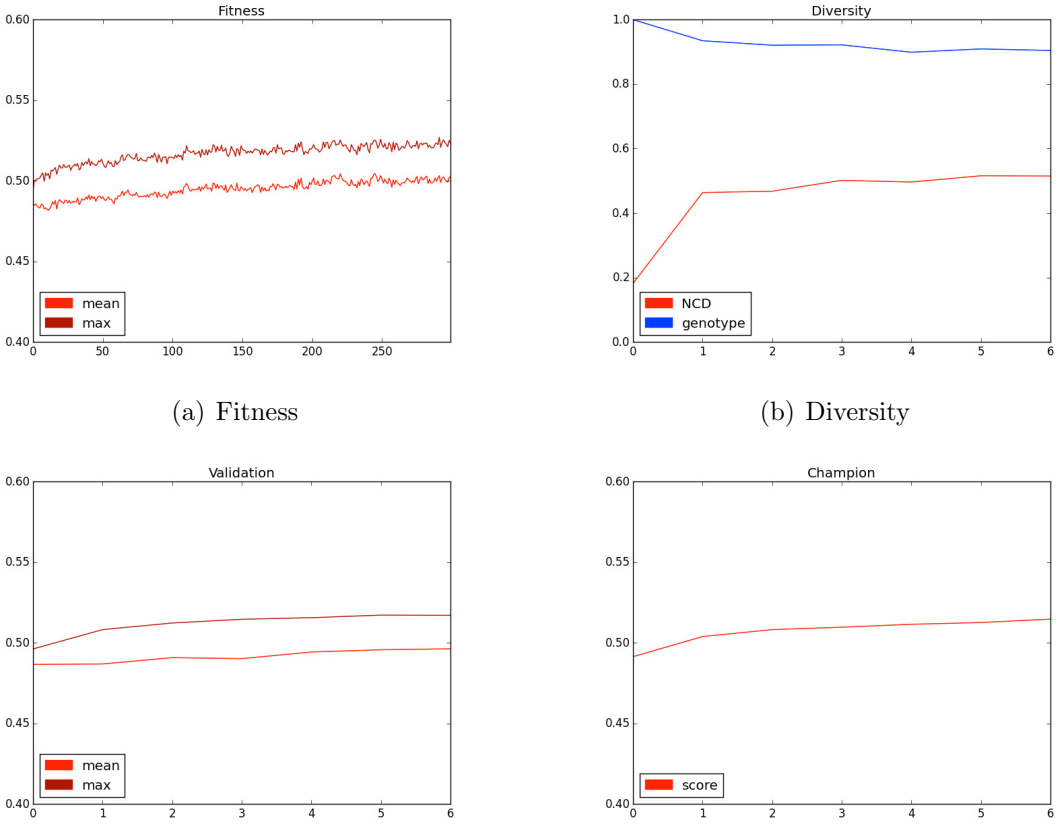
(a) Fitness

(b) Diversity

(c) Validation

(d) Champion

Figure A.1: Metrics for 300 generations across 25 runs with 6 validations in between for teams trained against the Bayesian opponent, with behavioral diversity (NCD).



(a) Fitness

(b) Diversity

(c) Validation

(d) Champion

Figure A.2: Metrics for 300 generations across 25 runs with 6 validations in between for teams not trained against the Bayesian opponent, with behavioral diversity (NCD).

Appendix B

Further Information on the Diversity Models

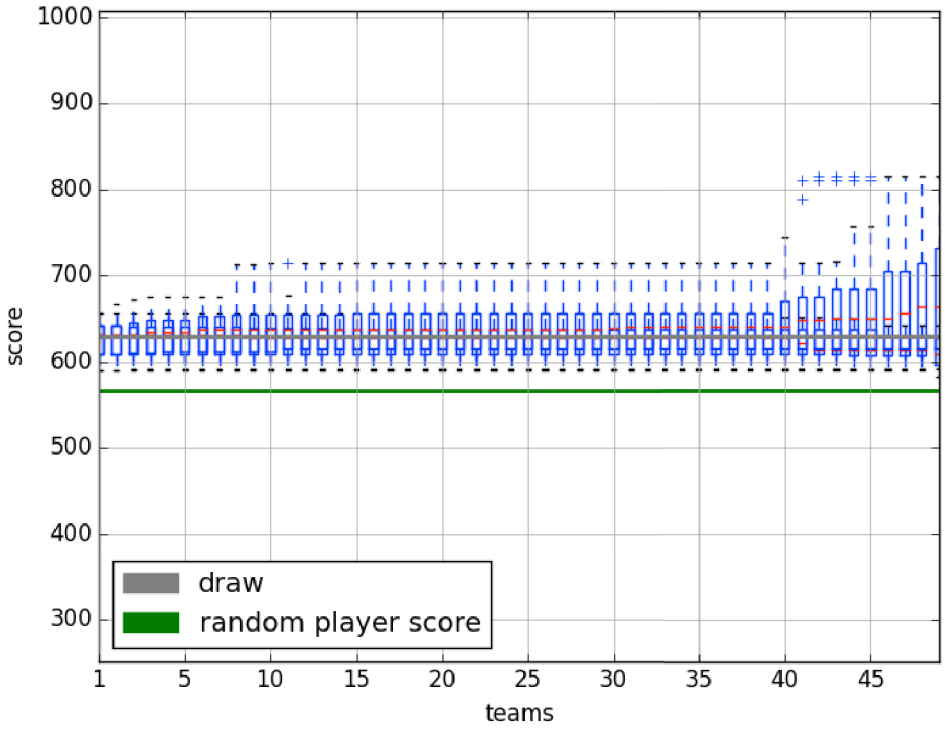
This appendix contains the data on the Diversity Models (Section 5.4) summarized by ranks (Table B.1 and Table B.2), and samples of the cumulative charts of all models against the Bayesian opponent for balanced and unbalanced hands. The models are as following: no diversity (Figure B.1); bid diversity (Figure B.2); fitness and novelty (NCD) as objectives (Figure B.3); novelty (NCD) as an objective (Figure B.4); fitness and behavioral diversity (NCD) as objectives (Figure B.5); fitness, behavioral diversity (NCD) and genotypic diversity as objectives (Figure B.6); fitness and behavioral diversity (Hamming) as objectives (Figure B.7); fitness, behavioral diversity (Hamming) and genotypic diversity as objectives (Figure B.8); and fitness and genotypic diversity as objectives (Figure B.9).

Table B.1: Comparison of the ranks for median normalized best individual scores for models with no diversity and with various types of diversity maintenance. Nine models are compared in ten scenarios (b and u marks the type of hand balance, and B, LA, LP, TA, and TP are types of opponents.

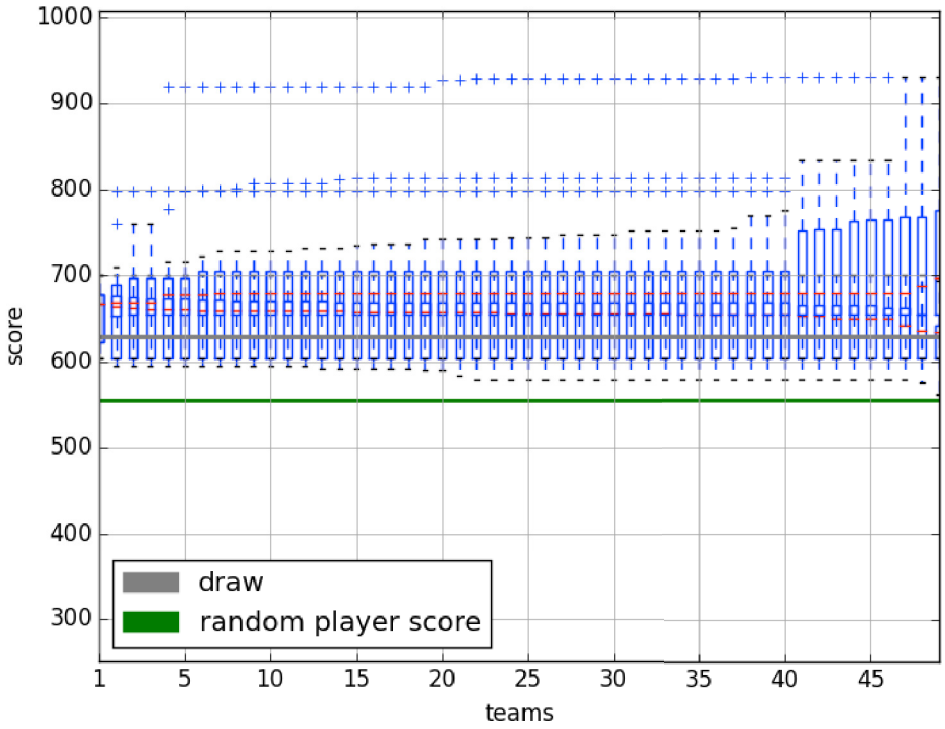
	no_div	bid	nov	only_nov	ncd	ncd_gen	ham	ham_gen	gen
bB	5	4	7	9	8	2	6	1	3
bLA	8	7	2	9	5	4	1	3	6
bLP	6	7	1	9	5	4	3	2	8
bTA	8	7	5	9	6	1	4	3	2
bTP	8	2.5	6	9	7	2.5	5	2.5	2.5
uB	6	5	7	9	8	3	4	2	1
uLA	8	5	3	9	7	4	6	2	1
uLP	4	2	6	9	7	5	1	3	8
uTA	8	6	7	9	5	2	4	2	2
uTP	8	2	6	9	7	4	5	1	3
rank	6.9	4.75	5	9	6.5	3.15	3.9	2.15	3.65

Table B.2: Comparison of the ranks for median normalized cumulative scores for models with no diversity and with various types of diversity maintenance. Nine models are compared in ten scenarios (b and u marks the type of hand balance, and B, LA, LP, TA, and TP are types of opponents.

	no_div	bid	nov	only_nov	ncd	ncd_gen	ham	ham_gen	gen
bB	9	8	5	6	7	1.5	4	3	1.5
bLA	9	8	6	7	5	1	3	2	4
bLP	9	8	4	7	6	1	2	5	3
bTA	9	8	1	3	2	4	5	6	7
bTP	9	8	1	5	2	3	4	6	7
uB	9	5	6	8	7	2.5	2.5	2.5	2.5
uLA	9	8	3	7	4	1	6	5	2
uLP	9	8	4	7	6	1	3	5	2
uTA	9	8	1	3	2	4	5	6	7
uTP	9	8	2	6	1	3	5	4	7
rank	9	7.7	3.3	5.9	4.2	2.2	3.95	4.45	4.3

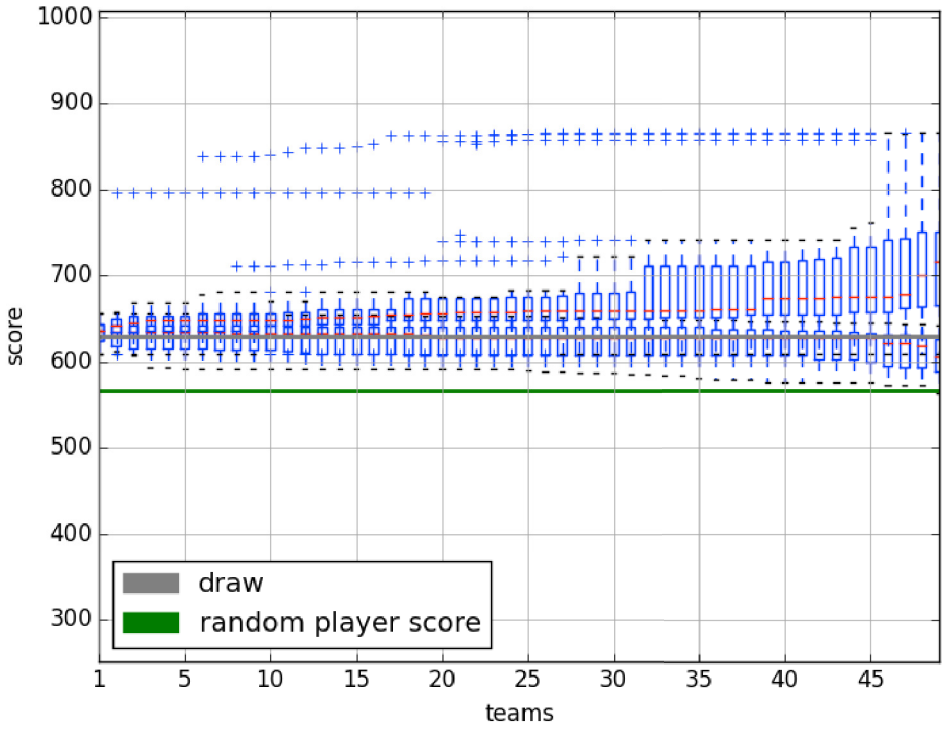


(a) Balanced hands

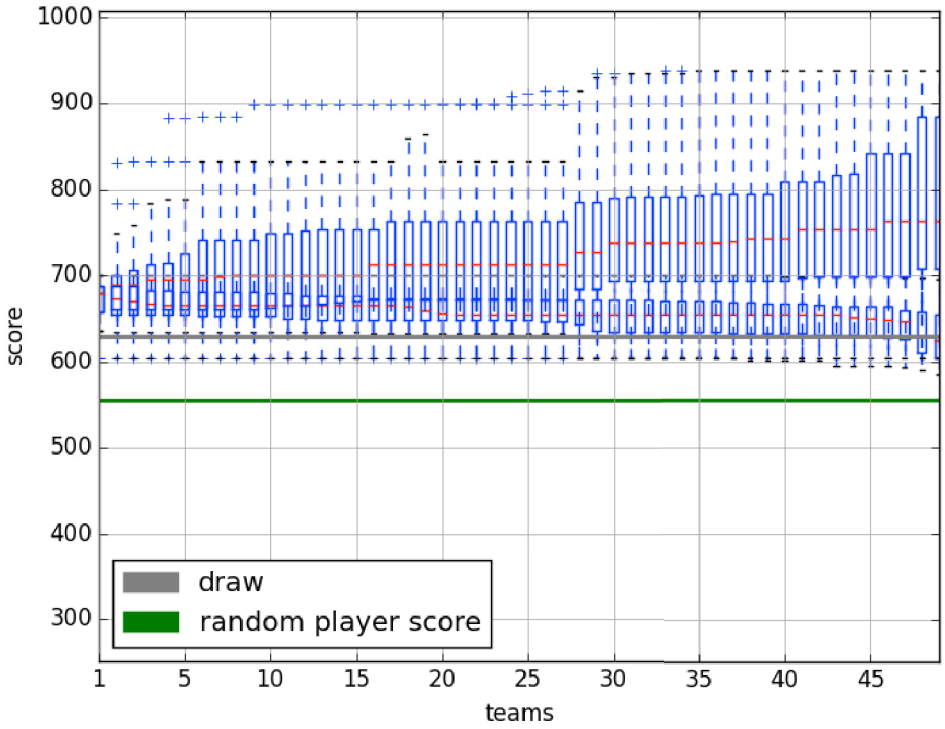


(b) Unbalanced hands

Figure B.1: Cumulative curves of SBB teams trained with no diversity, for 1260 hands against the Bayesian opponent.

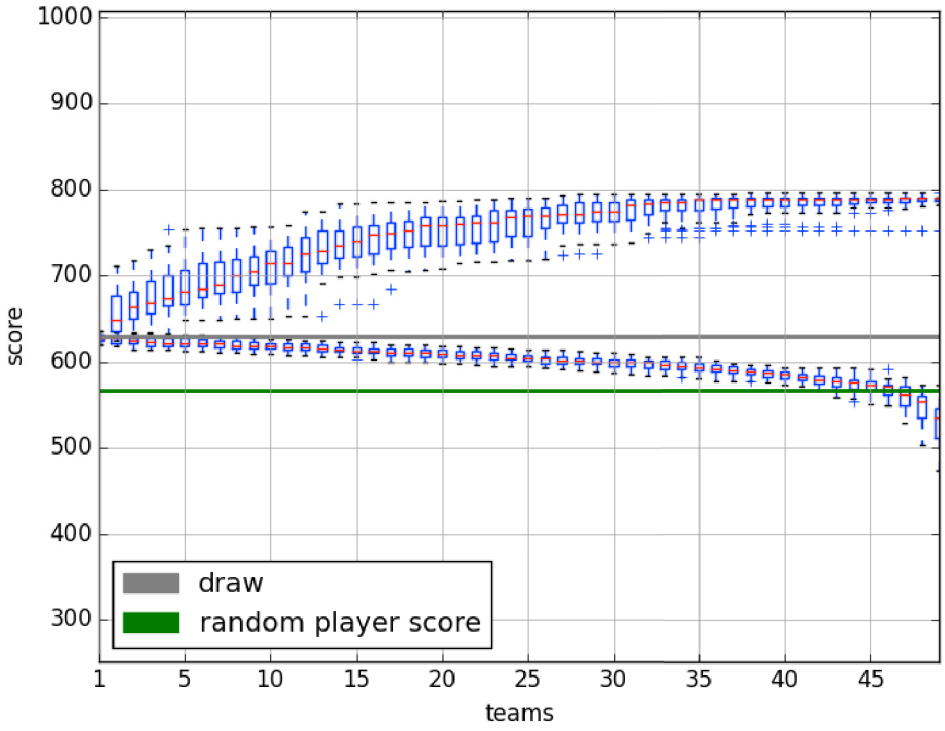


(a) Balanced hands

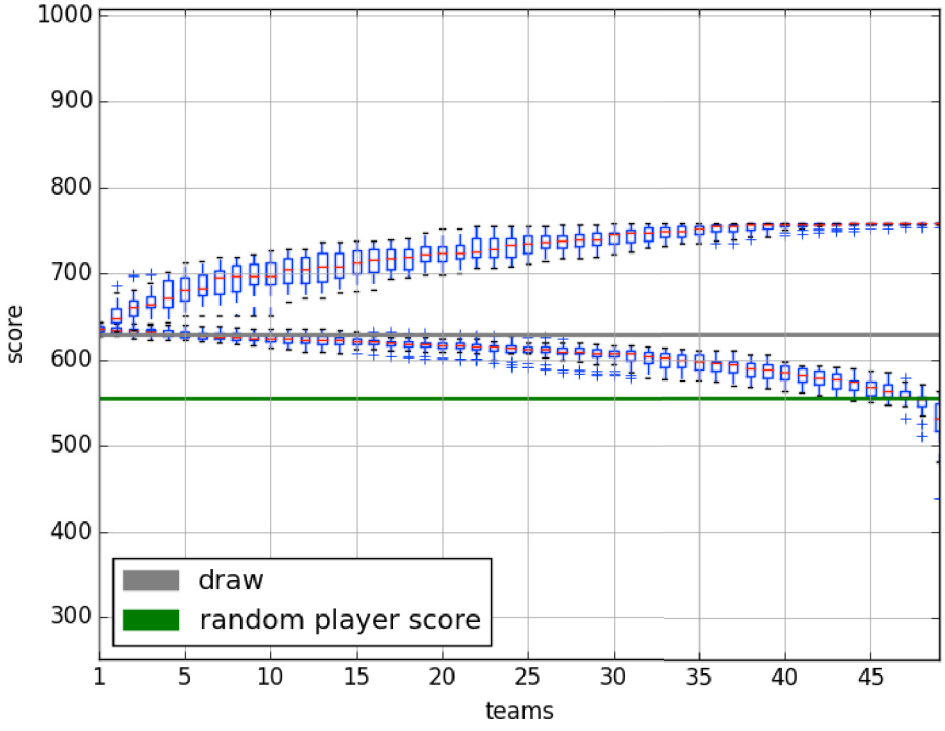


(b) Unbalanced hands

Figure B.2: Cumulative curves of SBB teams trained with bid diversity, for 1260 hands against the Bayesian opponent.

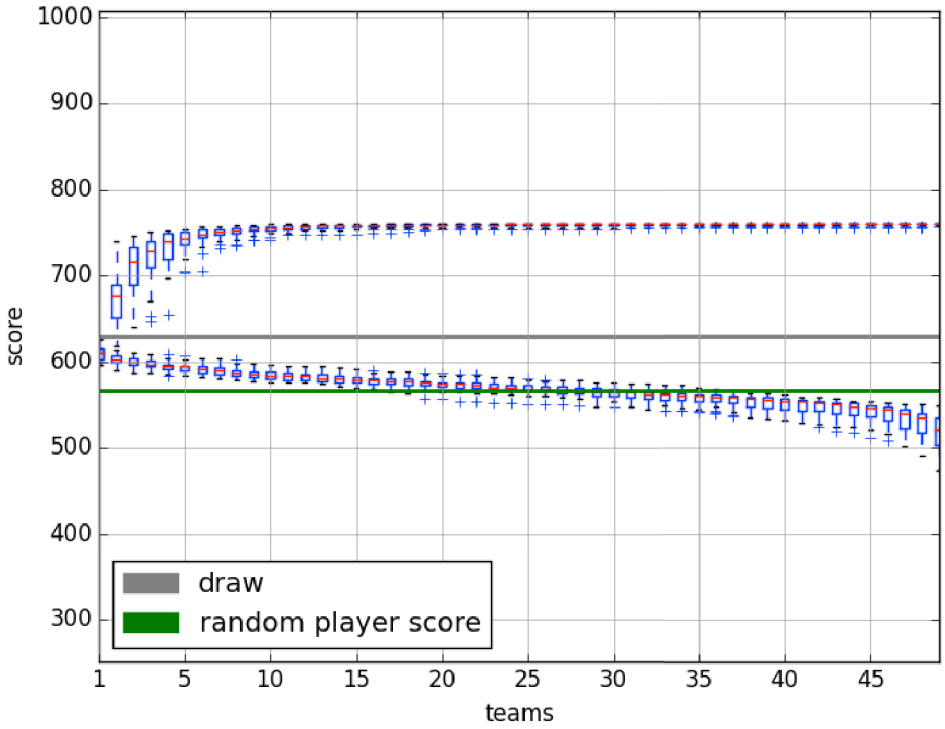


(a) Balanced hands

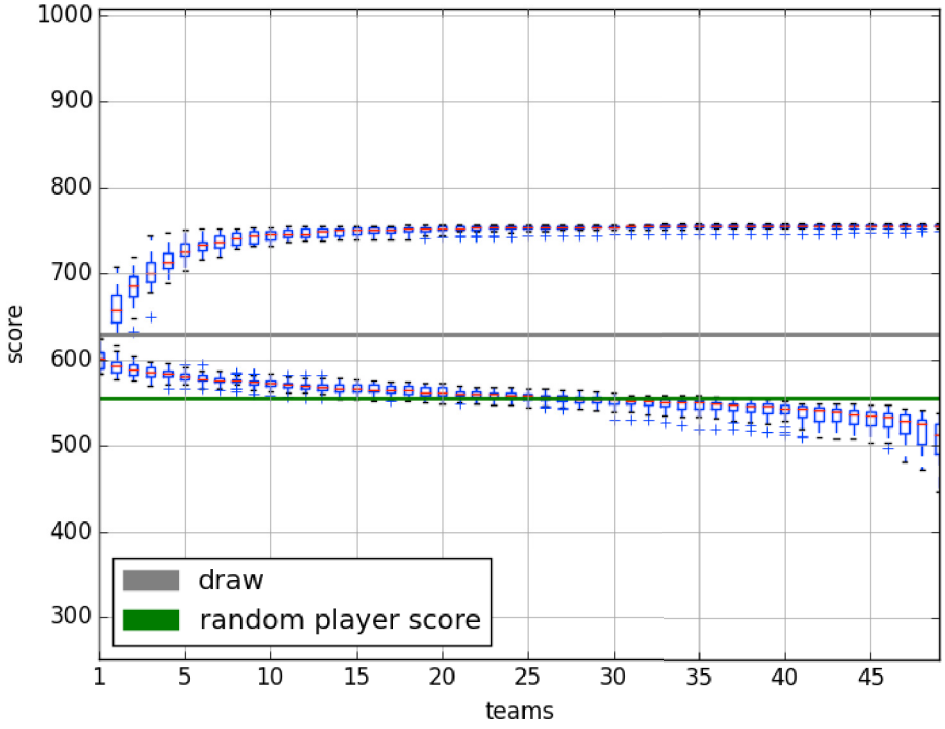


(b) Unbalanced hands

Figure B.3: Cumulative curves of SBB teams trained with fitness and novelty search (NCD), for 1260 hands against the Bayesian opponent.

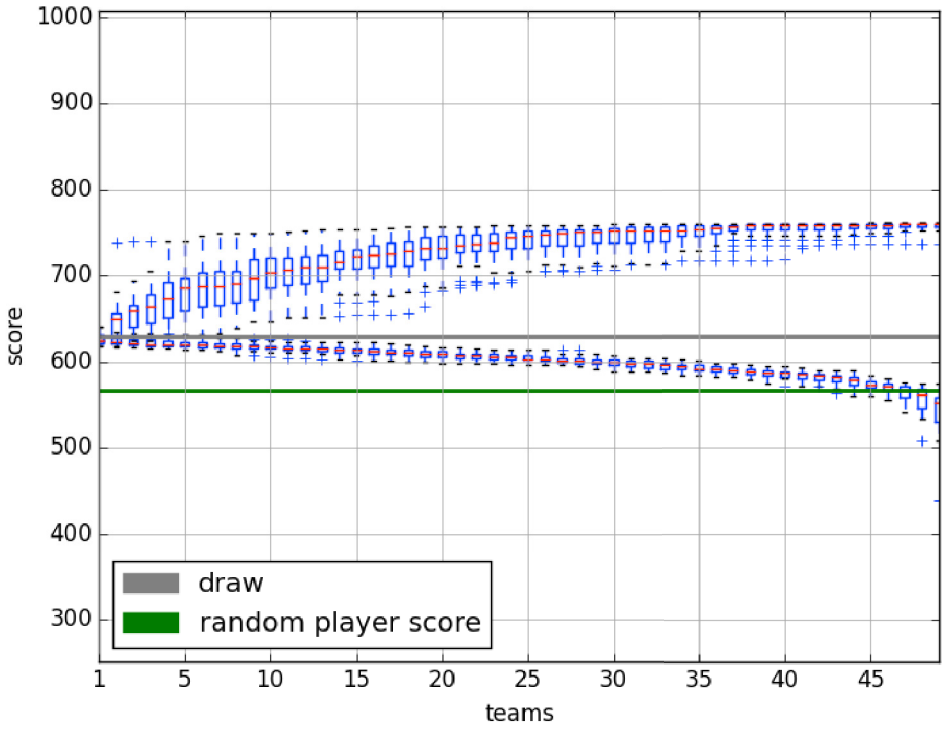


(a) Balanced hands

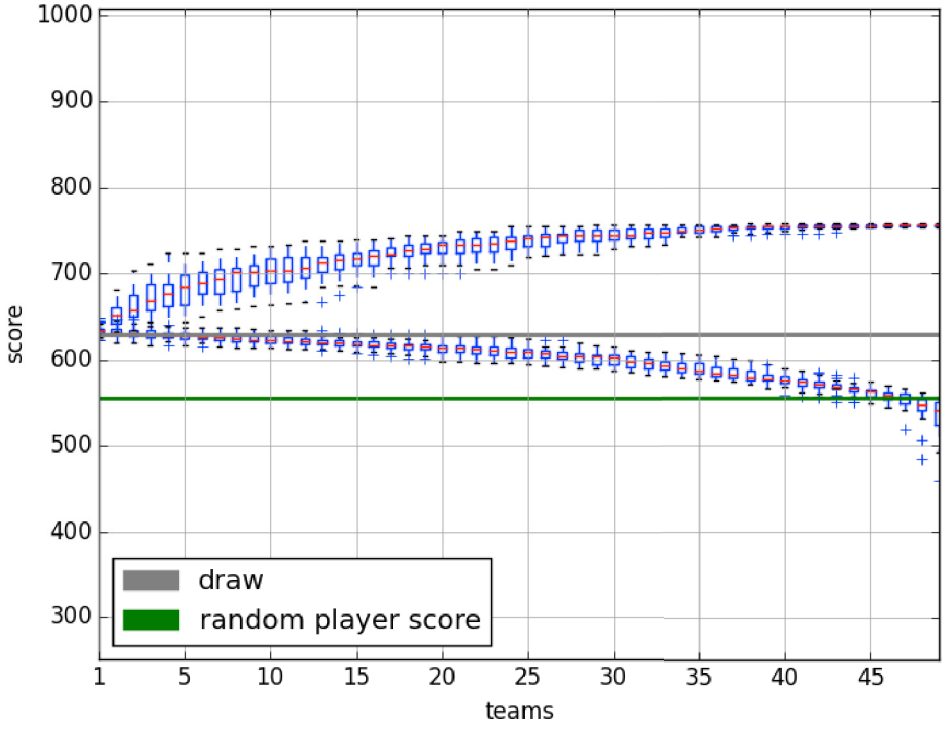


(b) Unbalanced hands

Figure B.4: Cumulative curves of SBB teams trained with only novelty search (NCD), for 1260 hands against the Bayesian opponent.

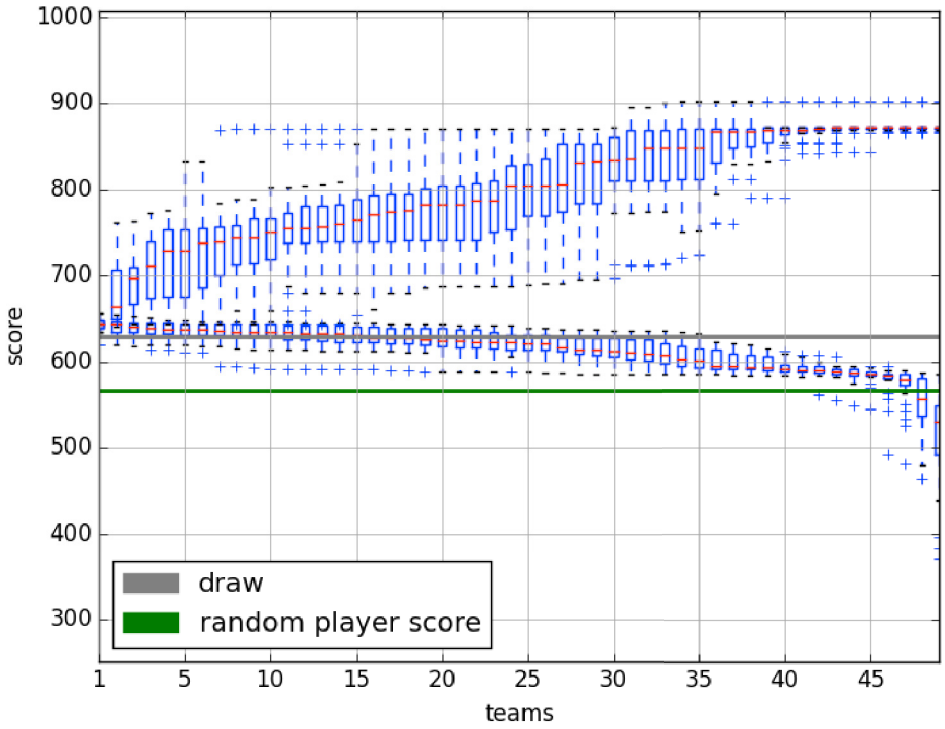


(a) Balanced hands

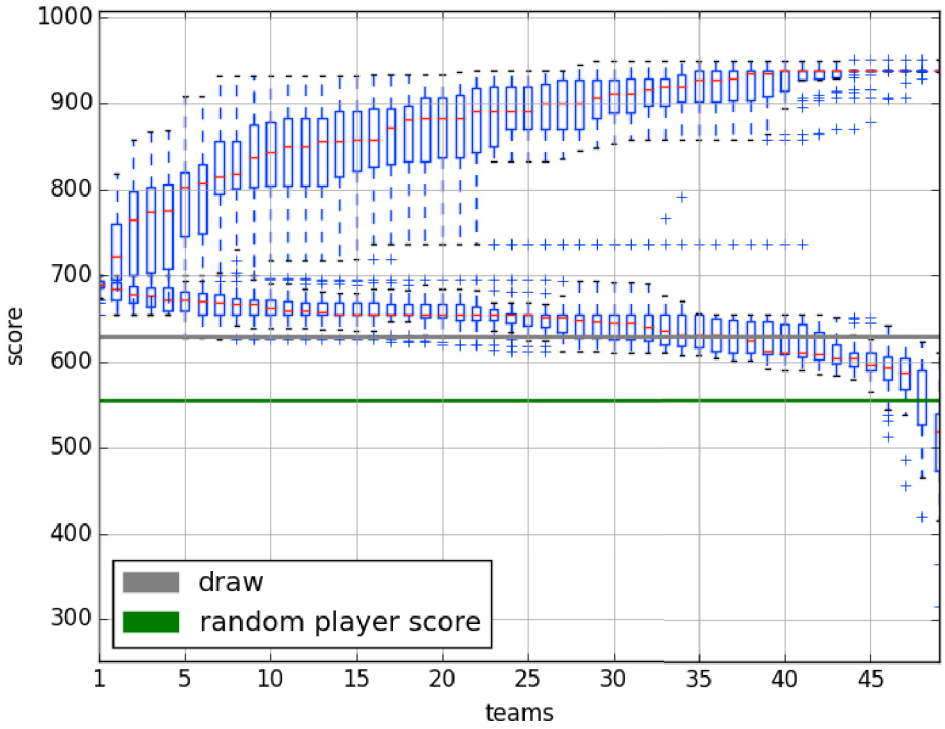


(b) Unbalanced hands

Figure B.5: Cumulative curves of SBB teams trained with fitness and behavioral diversity (NCD), for 1260 hands against the Bayesian opponent.

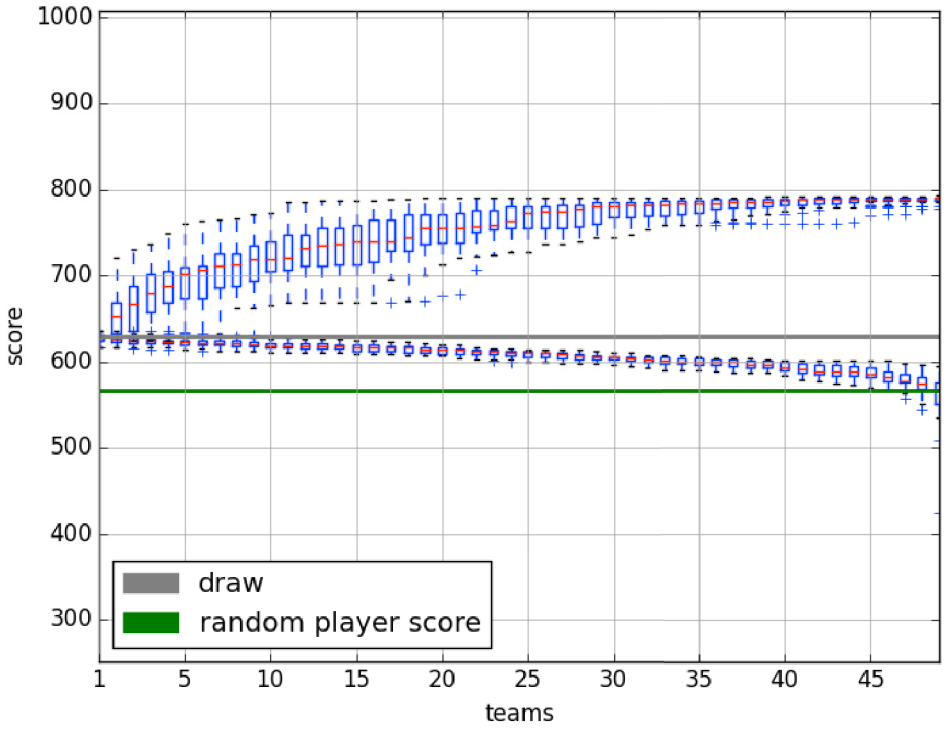


(a) Balanced hands

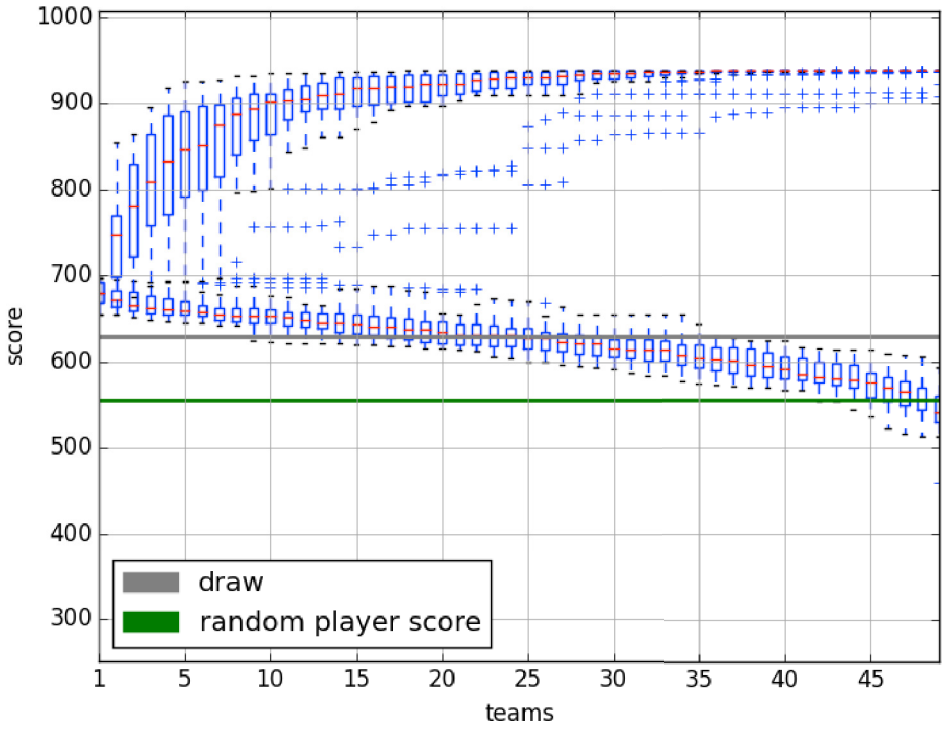


(b) Unbalanced hands

Figure B.6: Cumulative curves of SBB teams trained with fitness, behavioral diversity (NCD) and genotypic diversity, for 1260 hands against the Bayesian opponent.

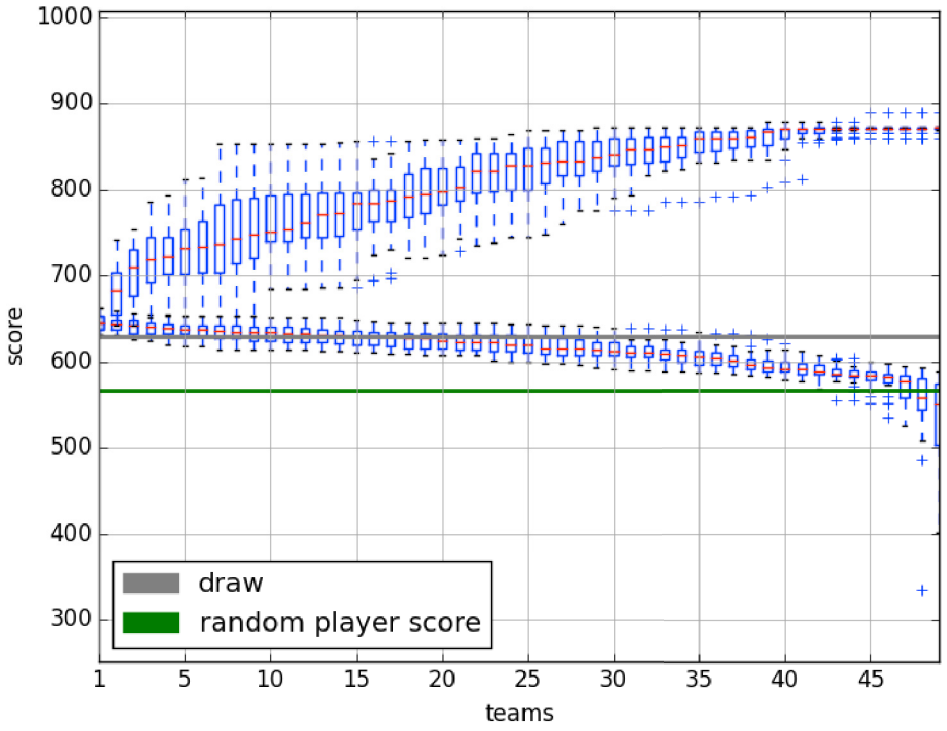


(a) Balanced hands

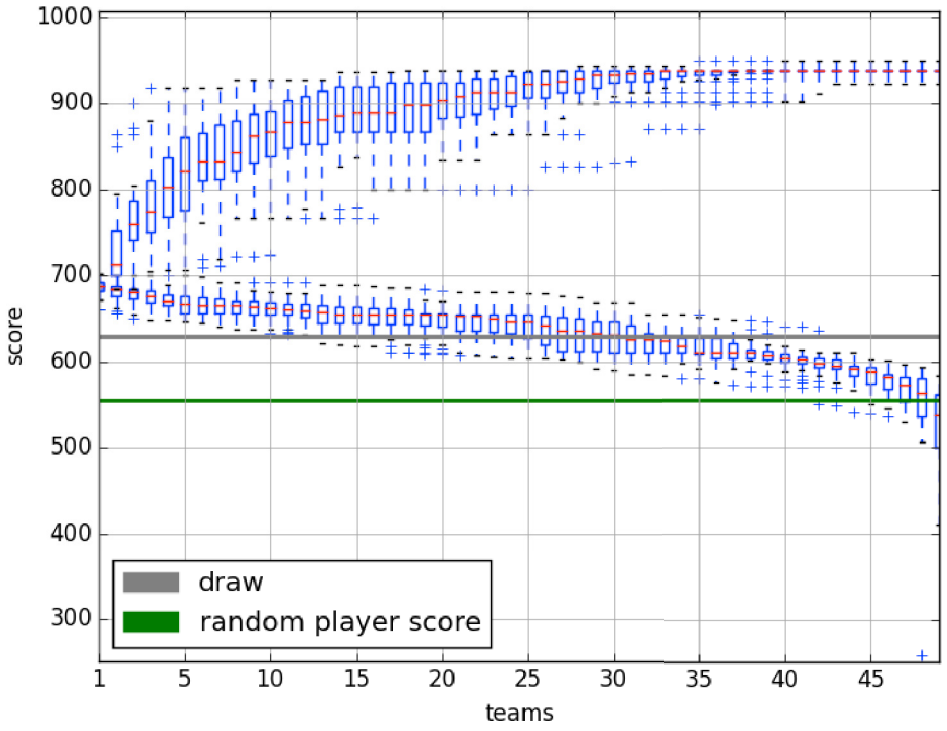


(b) Unbalanced hands

Figure B.7: Cumulative curves of SBB teams trained with fitness and behavioral diversity (Hamming), for 1260 hands against the Bayesian opponent.

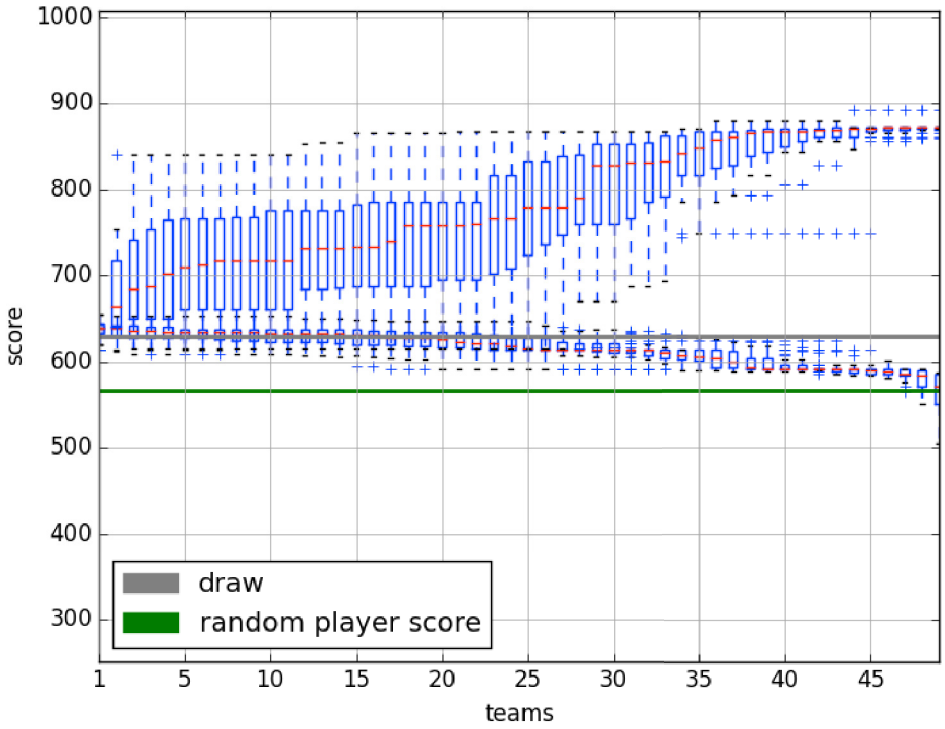


(a) Balanced hands

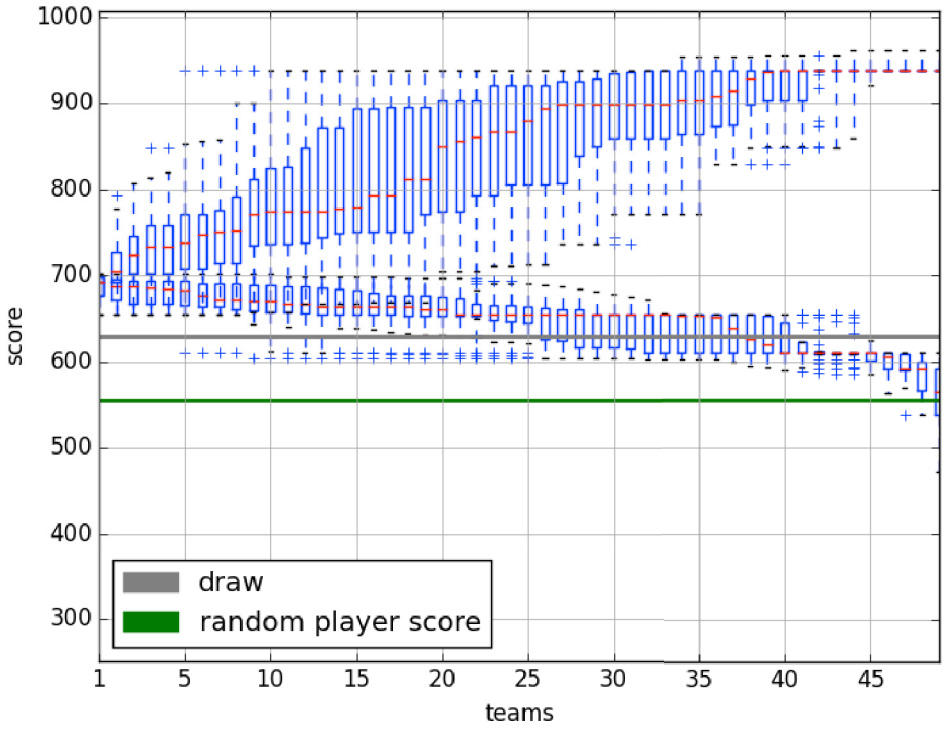


(b) Unbalanced hands

Figure B.8: Cumulative curves of SBB teams trained with fitness, behavioral diversity (Hamming) and genotypic diversity, for 1260 hands against the Bayesian opponent.



(a) Balanced hands



(b) Unbalanced hands

Figure B.9: Cumulative curves of SBB teams trained with fitness and genotypic diversity, for 1260 hands against the Bayesian opponent.

Appendix C

Behavior of Sampled Trained Teams

This appendix contains, as a matter of curiosity, details on sampled trained teams. Four teams were chosen according to their behavior and performance against the Bayesian opponent for unbalanced hands, among the teams that were evolved using fitness, genotypic diversity, and behavioral diversity (Hamming distance). The selected teams and an analyzes of their behaviors in shown below.

sbb_aggressive The best team for teams that played more than 80% of the hands.

- mean score: 0.541 (std: 0.44)
- aggressiveness: 0.65 (tight/loose: 0.81, passive/aggressive: 0.58)
- bluffing: 0.55

sbb_balanced The best team for teams that played between 40% and 60% of the hands.

- mean score: 0.556 (std: 0.33)
- aggressiveness: 0.47 (tight/loose: 0.6, passive/aggressive: 0.58)
- bluffing: 0.37

sbb_passive The best team for teams that played less than 20% of the hands.

- mean score: 0.523 (std: 0.2)
- aggressiveness: 0.15 (tight/loose: 0.19, passive/aggressive: 0.51)
- bluffing: 0.39

sbb_nonbluffer The best team for teams that bluffed less than 5% of the times. A specific team was chosen as a non-bluffer because all the best teams bluffed at least 30% of the time.

- mean score: 0.512 (std: 0.22)
- aggressiveness: 0.31 (tight/loose: 0.43, passive/aggressive: 0.56)
- bluffing: 0.04
- Observation: In the few times it bluffed, it preferred to raise instead of calling (75%).

Bibliography

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz. Handbook of evolutionary computation. *Release*, 97(1):B1, 1997.

[2] R. J. S. Baker and P. I. Cowling. Bayesian opponent modeling in a simple Poker environment. In *IEEE Symposium on Computational Intelligence and Games*, pages 125–131, 2007.

[3] R. J. S. Baker, P. I. Cowling, T. W. G. Randall, and P. Jiang. Can opponent models aid Poker player evolution? In *IEEE Symposium on Computational Intelligence and Games*, pages 23–30, 2008.

[4] L. Barone and L. While. Evolving adaptive play for simplified Poker. In *IEEE Congress on Evolutionary Computation*, pages 108–113, 1998.

[5] L. Barone and L. While. An adaptive learning model for simplified poker using evolutionary algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE, 1999.

[6] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, pages 661–668, 2003.

[7] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134:201–240, 2002.

[8] D. Billings, A. Davidson, T. Schauenberg, N. Burch, M. Bowling, R. Holte, J. Schaeffer, and D. Szafron. Game-tree search with adaptation in stochastic imperfect-information games. In *Computers and Games*, pages 21–34. Springer, 2004.

[9] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. Poker as a testbed for ai research. In *Advances in Artificial Intelligence*, pages 228–238. Springer, 1998.

[10] J. P. C. Bonson, S. Kelly, A. R. McIntyre, and M. I. Heywood. On synergies between diversity and task decomposition in constructing complex systems with GP. In *ACM GECCO*, 2016.

[11] M. Bowling, N. Burch, M. Johanson, and O. Tammelin. Heads-up limit holdem poker is solved. *Science*, 347(6218):145–149, 2015.

[12] M. Bowling, N. A. Risk, N. Bard, D. Billings, N. Burch, J. Davidson, J. Hawkin, R. Holte, M. Johanson, M. Kan, et al. A demonstration of the polaris poker system. In *Proceedings of The 8th International Conference on Autonomous Agents*

and Multiagent Systems-Volume 2, pages 1391–1392. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[13] G. Cuccu and F. Gomez. When novelty is not enough. In *European Conference on the Applications of Evolutionary Computation*, pages 234–243. Springer, 2011.

[14] A. Davidson. Opponent modeling in poker: Learning and acting in a hostile and uncertain environment. Master's thesis, University of Alberta, 2002.

[15] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International Conference on Parallel Problem Solving From Nature*, pages 849–858. Springer, 2000.

[16] S. Doncieux and J. Mouret. Behavioral diversity measures for evolutionary robotics. In *IEEE congress on evolutionary computation*, pages 1–8. IEEE, 2010.

[17] J. A. Doucette, P. Lichodzijewski, and M. I. Heywood. Hierarchical task decomposition through symbiosis in reinforcement learning. In *ACM Genetic and Evolutionary Computation Conference*, pages 97–104, 2012.

[18] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[19] F. J. Gomez. Sustaining diversity using behavioral information distance. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 113–120. ACM, 2009.

[20] N. Japkowicz and M. Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[21] S. Kelly and M. I. Heywood. Genotypic versus behavioural diversity for teams of programs under the 4-v-3 keepaway soccer task. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3110–3111, 2014.

[22] S. Kelly and M. I. Heywood. On diversity, teaming, and hierarchical policies: Observations from the Keepaway soccer task. In *European Conference on Genetic Programming*, volume 8599 of *LNCS*, pages 75–86, 2014.

[23] S. Kelly, P. Lichodzijewski, and M. I. Heywood. On run time libraries and hierarchical symbiosis. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[24] G. Kendall and M. Willdig. An investigation of an adaptive poker player. In *AI 2001: Advances in Artificial Intelligence*, pages 189–200. Springer, 2001.

[25] P. Krčah. Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty. In *Advances in robotics and virtual reality*, pages 167–186. Springer, 2012.

[26] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[27] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[28] J. Lehman and K. O. Stanley. Novelty search and the problem with objectives. In *Genetic Programming Theory and Practice IX*, pages 37–56. Springer, 2011.

[29] J. Lehman, K. O. Stanley, and R. Miikkulainen. Effective diversity maintenance in deceptive domains. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 215–222. ACM, 2013.

[30] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264, 2004.

[31] P. Lichodzijewski. *A symbiotic bid-based framework for problem decomposition using Genetic Programming*. PhD thesis, Dalhousie University, 2011.

[32] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with symbiotic bid-based genetic programming. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference*, pages 863–870, 2008.

[33] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under GP. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference*, pages 853–860, 2010.

[34] H. Moriguchi and S. Honiden. Sustaining behavioral diversity in neat. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 611–618. ACM, 2010.

[35] J. Mouret. Novelty-based multiobjectivization. In *New horizons in evolutionary robotics*, pages 139–154. Springer, 2011.

[36] J. Mouret and S. Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *2009 IEEE Congress on Evolutionary Computation*, pages 1161–1168. IEEE, 2009.

[37] J. Mouret and S. Doncieux. Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 627–634. ACM, 2009.

[38] J. B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation*, 20(1):91–133, 2012.

[39] G. Nicolai and R. J. Hilderman. No-limit texas hold'em poker agents created with evolutionary neural networks. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 125–131. Ieee, 2009.

[40] G. Nicolai and R. J. Hilderman. Countering evolutionary forgetting in no-limit Texas Hold'em Poker agents. In K. Madani et al., editor, *Computational Intelligence*, volume 399 of *SCI*, pages 31–48. Springer, 2012.

[41] J. Noble. Finding robust texas hold'em poker strategies using pareto coevolution and deterministic crowding. 2002.

[42] J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In *ACM GECCO*. Morgan Kauffman, 2001.

[43] M. Ponsen, K. Tuyls, M. Kaisers, and J. Ramon. An evolutionary game-theoretic analysis of poker strategies. *Entertainment Computing*, 1(1):39–45, 2009.

[44] S. Risi, C. E. Hughes, and K. O. Stanley. Evolving plastic neural networks with novelty search. *Adaptive Behavior*, 18(6):470–491, 2010.

[45] S. Risi, S. D. Vanderbleek, C. E. Hughes, and K. O. Stanley. How novelty search escapes the deceptive trap of learning to learn. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 153–160. ACM, 2009.

[46] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1998.

[47] J. Rubin and I. Watson. Computer poker: A review. *Artificial Intelligence*, 175(5):958–987, 2011.

[48] D. Sklansky. *The theory of Poker*. Two Plus Two Publishing LLC, 1999.

[49] R. Smith, S. Kelly, and M. I. Heywood. Discovering rubik's cube subgroups using coevolutionary GP - a 5 twist experiment. In *ACM GECCO*, 2016.

[50] C. Sun, Y. Liao, J. Lu, and F. Zheng. Genetic algorithm learning in game playing with multiple coaches. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 239–243. IEEE, 1994.

[51] T. Thompson, J. Levine, and R. Wotherspoon. Evolution of counter-strategies: Application of co-evolution to texas hold'em poker. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pages 16–22. IEEE, 2008.

[52] L. Trujillo, G. Olague, E. Lutton, and F. F. de Vega. Behavior-based speciation for evolutionary robotics. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 297–298. ACM, 2008.

[53] D. Whitley. Fundamental principles of deception. *Foundations of Genetic Algorithms 1991 (FOGA 1)*, 1:221, 2014.

[54] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2007.