Theoretical Computer Science 785 (2019) 1–16

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Evolving novelty strategies for the Iterated Prisoner's Dilemma in deceptive tournaments [☆]

C.R. Noordman [*], G.A.W. Vreeswijk

Utrecht University, Buys Ballotgebouw, Princetonplein 5, 3584 CC Utrecht, Netherlands

ARTICLE INFO

Article history:
Received 1 January 2018
Received in revised form 18 May 2018
Accepted 22 October 2018
Available online 1 November 2018
Communicated by W. Banzhaf

Keywords:
Evolutionary games
Iterated prisoner's dilemma
Novelty search
Zero-determinant strategies

ABSTRACT

This paper proposes that the concept of deception brought forward by novelty search research can be applied to the Iterated Prisoner's Dilemma problem, and in doing so simultaneously fights the claim that Zero-determinant strategies can outperform *any* evolutionary opponent. Zero-determinant strategies are a special class of strategies where its moves are probabilistically conditioned on the previous outcome through careful mathematics. When compared with behaviors that merely attempt to obtain the highest score possible through objective search, more complex and above all unique behaviors generated from novelty search allows us to transcend the deception problem that come with certain configurations of an Iterated Prisoner's Dilemma tournament.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

Recent research has brought exciting new developments in the field of evolutionary game theory [1]. Zero-determinant strategies [2] are a new class of strategies for an old problem which excels at outperforming slowly evolving strategies such as those created by evolutionary algorithms (by slowly trailing the fitness gradient), and novelty search [3] is a new method of an evolutionary algorithm that ignores the concept of fitness and searches merely for unique behaviors.

One of the most heavily analyzed games in game theory, the Prisoner's Dilemma [4], shows how two players are unable to cooperate even when those players are considered fully rational. The extended iterated Prisoner's Dilemma is where players repeatedly play the same game, but here with the opportunity to punish the other when they are betrayed. In a single game, defecting is the dominant strategy, and mutual defection is the only Nash equilibrium in the game. When repeated, strategies become more fluid, and a number of strategies have posed as a significant opponent over the many years of analyzing the Iterated Prisoner's Dilemma (IPD). The goal of an IPD strategy is to accumulate the most rewards after n iterations.

The Prisoner's Dilemma, framed in the 1950s, is presented as a game where two persons are arrested and imprisoned. They are in solitary confinement and are not allowed to communicate with each other. The prosecutors lack sufficient evidence to convict the two players, but they are both offered a bargain. They are given the opportunity to either defect by testifying that the other committed the crime, or implicitly cooperate with the other by remaining silent. The payoff matrix that goes with this story is displayed in Fig. 1.

[☆] This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

[*] Corresponding author.
E-mail addresses: cr_noordman@protonmail.com (C.R. Noordman), gv@uu.nl (G.A.W. Vreeswijk).

https://doi.org/10.1016/j.tcs.2018.10.026
0304-3975/© 2018 Elsevier B.V. All rights reserved.

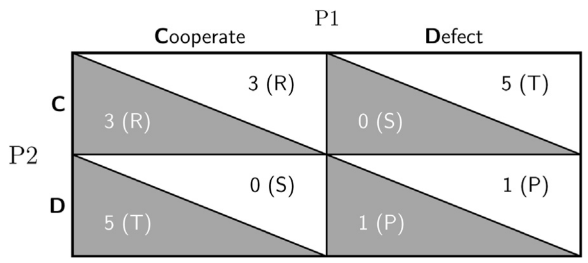

Fig. 1. The payoff matrix for the Prisoner's Dilemma. If players P1 and P2 cooperate, they each earn a reward R. When one defects, the defector receives a defector reward T (temptation), and the betrayed receives S (sucker). (T, R, P, S) were first defined by [4]. When both defect, both are rewarded with P (punishment). Any payoff matrix is a prisoners dilemma when it satisfies $T > R > P > S$, which guarantees that the only Nash equilibrium is mutual betrayal, and $2R > T + S$, which makes mutual cooperation the best cumulative outcome. The above matrix shows the most commonly used values for (T, R, P, S).

Strategies that are developed for the IPD are varied, and almost always follow static rules. Always Defect (ALL-D) is simply set up to always choose the defect option, no matter past actions. Its counterpart, Always Cooperate (ALL-C), is to always cooperate irrelevant of past actions. These are simple strategies that can barely be called a strategy, but are important as they are the outer edges within the set of strategies applicable to the IPD. Renowned strategies also within this set include Tit-for-Tat [4] (TFT, repeat opponent's last choice), Pavlov [5] (repeat last choice if it was T or R) and the recently formulated Zero-Determinant strategy (conditional probabilities based on the last round of play). Most competitive strategies are variants of these strategies. An IPD tournament is a setting where some set of strategies must play a set number of games against each other strategy. The winner of a tournament is the strategy which obtains the highest cumulative score over all other strategies.

In this paper, we present a novel neuroevolution-based strategy that does not necessarily perform better than any other valid strategy for the IPD, but will likely have good performance when up against specialized strategies. Most strategies have a weak point, and there always exists a strategy that directly focuses on that weak point. Normal evolutionary strategies have the weakness that they always slowly try to work their way up towards a higher payoff, and there exist strategies that punish exactly such behavior. This evolutionary strategy, we call NOVELTY, maximizes exploration and is not expected to have a weak point. This does not mean that it will win all possible tournaments, but it does have a profound advantage over other strategies in any tournament.

The rest of this paper is structured as follows. First, the background of the IPD is examined in section 2. Following that a system is devised which can model any valid strategy that can be used for in a tournament, in section 3. This is important for comparing our artificial neural network, NOVELTY, with a large variety of strategies. The strategy that explicitly punishes the general evolutionary strategy is described in subsection 3.2. The concept of NOVELTY, named simply after its core technique, is detailed in sections 4 and 4.1. The experimental design and its results of NOVELTY's hypothetical performance is demonstrated in sections 5 and 6. Finally, we discuss our findings and its implications in section 7. There are two appendices. Appendix A show more concrete examples of the model presented in section 3. Appendix B provides the software and the parameters used during experimentation with NOVELTY.

2. Background

The search for successful strategies for the IPD goes back quite a few decades. Some are a clever set of conditional instructions, others employ mathematics, and others even attempt to game the system surrounding an IPD tournament. We discuss some history of the strategies developed for these IPD tournaments.

Static strategies are strategies which have their rules defined a priori. This implies that for every static strategy there exists some set of possible optimal counter strategies. A counter strategy can be devised by considering the rules of which some static strategy relies, and then subsequently exploiting those rules to maximize payoff. The very idea of counter strategies has led to numerous tournaments revolving around the aforementioned Prisoner's Dilemma. In 1980, people sent in their strategy proposals for the first Iterated Prisoner's Dilemma tournament [6]. The winner was TFT, due to it having a number of characteristics that are found to be advantageous during conflict games. It starts peaceful (starts with cooperating), can be provoked (defecting leads to it defecting), but can also forgive (cooperating leads to it cooperating once again) and its methods are clear. Numerous tournaments followed, even until this day, and many strategies are designed to win by exploiting other known strategies, or colluding with other strategies.

For example, OmegaTitForTat [7] is a strategy which plays like TFT but can recover from mutual defection deadlock situations. A good example of such a situation is pitting TFT against DTFT (like TFT, but starts with a defection). Both will immediately go into a defection deadlock, even though they could conceptually cooperate as well as a TFT mirror match-up. Another interesting concept [7] designed was roughly the following: consider two strategies, Lord and Peon. Lord is essentially TFT, while Peon plays a repeating list of choices which strongly resembles ALL-C. Lord can detect whether it is playing versus Peon because it knows this list of choices, and once detected, exploits Peon. The idea here is to submit a single Lord strategy to a tournament, and many Peons, so the Lord can exploit a large share of the strategies. To do this, the

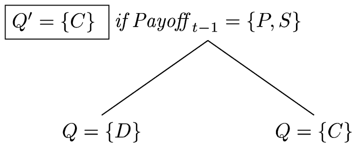

Fig. 2. Model for the TFT strategy. It starts with cooperating, so Q initializes to $\{C\}$ by copying Q'. When Q is empty, it requests a new Q from the decision tree E. When the payoff from the previous game resulted in either a P or S, then the next choice will be to defect. Q is set to the sequence $\{D\}$, which is its decision in the current game it's playing. The alternative would have resulted in Q being set to the sequence $\{C\}$.

authors created multiple email accounts and faked a large number of varying personas in order to get their Peon strategies submitted.

Presumably, the motivation for any strategy that wishes to outperform all others is to find the quickest way to develop a counter strategy of any arbitrary strategy that is valid within the IPD. These need not necessarily be rational strategies. The thought originated from two ideas: the idea that the very best strategy (call it BEST) in any game would be one that immediately constructs the counter strategy to any opposing strategy. The other idea is that many, if not all, non-cooperative games are of a deceptive nature. This is because actions need not necessarily be immediately rational, but be part of a rational plan down the line.

A combination of these two ideas would lead to a strategy that uses the deceptive nature of the game to its advantage, and conceptually comes very close to the abstract strategy BEST that can instantly develop a counter strategy. It is, in a way, the inverse of the Adaptive strategy. The Adaptive strategy first analyses the opponent, then objectively picks choices which have given the best average score re-calculated after every iteration.

As an example, imagine some IPD tournament that plays n games using normal scores as described in Fig. 1. Adaptive is up against a rather strange TFT strategy (sTFT), where it will always cooperate until game 20, and then turns into normal TFT. If, during the analyzing stage, Adaptive cooperates six times, then defects five times, the average score from cooperating will be 3 (R), and the average score from defecting will be 5 (T). When sTFT starts following TFT rules from game 20 onwards, the average score from defecting will slowly move towards 1 (P). It will be until game 33 until the average scores for playing either cooperate or defect are equal and Adaptive might start reacting to the opponent. The concept of probing, where some random intervention allows any strategy to make a random choice instead of its normal choice, would only prolong the adaptation. Specifically, Adaptive would play C instead of D due to random intervention in our example, receiving the S reward, because sTFT still plays D. This decreases the average score for C. Then, the next iteration Adaptive goes back to playing D (no intervention), while sTFT plays C as per TFT rules. Adaptive receives T, and sTFT receives S. This increases the average score for D. Therefore, it takes an even longer time for the average score for D to reach the average score for C. Probing only hurts Adaptive in this case, and many similar cases.

3. Set of all valid strategies

NOVELTY is motivated by the desire to come close to the abstract strategy BEST. In order to properly examine how NOVELTY would perform against tournaments with arbitrarily large number of strategies competing, we need to find a way to populate the set of all valid strategies, T, with something other than the abstract concept of a valid strategy.

3.1. Generating strategies

We devise an explicit system that can properly model all variations of what a strategy can be when its behavior does not change during a tournament. Current implementations of syntactically defining a strategy are insufficient, as highly specialized strategies are abstracted or merely defined in words. The motivation was to define an implementation where programmatically we could create any valid static strategies of near-infinite complexity. This could subsequently be used to generate a statistical sampling of the set of all valid strategies T, for use during the experimentation of our NOVELTY concept. Inspired by ideas from genetic programming, what follows is a very brief explanation of the system, with more details to be found in Appendix A.

In our system, any static strategy is expressed as (Q, E). The strategy outputs a decision from the decision set $\{C, D, R\}$, where R represents a random choice, by extracting the first element from the sequence of decisions Q. If Q is empty, it instead evaluates decision tree E first, then tries to extract from Q again. E is expected to populate Q, it consists of a hierarchical set of nodes, where at the leafs it assigns a sequence to Q. Fig. 2 is an example of the TFT strategy.

Using this system, we can randomly generate any valid strategies for the IPD. Visualizing the set of all valid strategies T is very useful in understanding what NOVELTY tries to overcome. Because these generated strategies are completely defined by chance, a large number are likely to be rather underperforming in a tournament. However, some of these strategies need not become very complex to be of good performance. We discuss an important subset of all valid strategies.

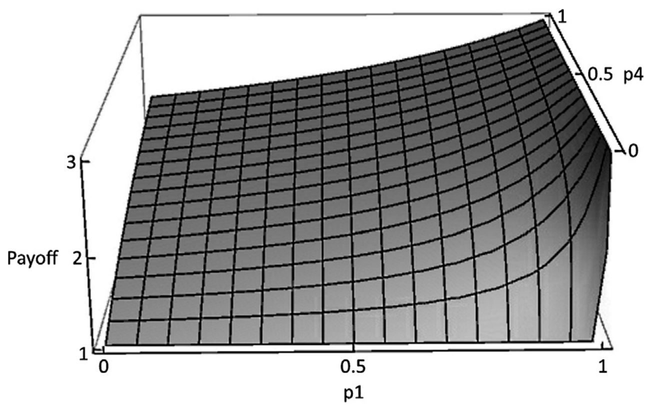

Fig. 3. Plot for all possible payoffs in the standard IPD as shown in Fig. 1 for player Y given the values for elements p_1 and p_4 for some $\bar{p}$. It shows that player X can set all scores $P \leq S_Y \leq R$, and it is up to the strategy of player Y to expand this to $S \leq S_Y \leq T$. It also shows that there is no need for player X to explicitly react to the strategy of player Y, because only Y controls scores outside the $[P, R]$ range.

3.2. Zero-determinant strategies

In [2], Press and Dyson have devised a strategy they call Zero-Determinant (ZD) strategies, which unilaterally sets the score of their opponent in an effort to stay on top. It is important to understand the workings of this strategy, as it claims to be able to defeat any evolutionary opponent (such as NOVELTY) because it can outmanoeuvre its slowly adapting opponent.

Mathematically, ZD strategies are a vector of probabilities (p, q) conditioned on the four outcomes of the previous move. [2] show that some fixed strategy vector $\bar{p}$ for player X (or ($\bar{q}$ for player Y), which is based on the previous move, gives the linear equation between the scores for players X and Y: $\alpha S_X + \beta S_Y + \gamma = 0$. The elements in a strategy vector $p = (p_1, p_2, p_3, p_4)$ define the probabilities for when to cooperate when the previous outcome was R, T, S and P respectively. For example, if $p_1 = 0.75$ and the previous outcome ended in R for the player, there is a 75% chance this player would cooperate. The vector $p = (1, 0, 1, 0)$ equals the TFT strategy. This linear relation is constructed from the determinant of the dot product between the Markov matrix v and any vector f: $v \cdot f \equiv D(p, q, f)$. They show that using substitution of the probability values for $\bar{p}$, Y's score eventually can be expressed as

$$S_Y = \frac{(1-p_1)P + p_4 R}{(1-p_1) + p_4} \tag{1}$$

where p_1 and p_4 are elements in the vector $\bar{p}$. In other words, player X can unilaterally set player Y's score, as p_1 and p_4 are under control of player X. Solutions for $\bar{p}$ are when $p_1 \leq 1$ and $p_4 \geq 0$. Fig. 3 shows the possible payoffs for all valid solutions for $\bar{p}$.

The idea behind zero-determinant strategies lies in the extortionate share. Two new variables are introduced, χ and ϕ, where

$$0 < \phi \leq \frac{(P-S)}{(P-S) + \chi(T-P)} \tag{2}$$

This range is necessary for the variable to make sense in some underlying equations omitted in this explanation. By balancing these two variables, we can create new valid solutions for $\bar{p}$ that allow the player to get payoffs higher than R. X and Y's best respective scores become:

$$S_X = \frac{2 + 13\chi}{2 + 3\chi}, S_Y = \frac{12 + 3\chi}{2 + 3\chi} \tag{3}$$

Fig. 5 shows the scores for both players given a certain χ value. In ZD strategies, the extortion factor χ defines the strategy $\bar{p}$, not the strategy of the opponent. For player Y, the optimal counter strategy is to have an extortion value exactly equal to that of player X. In this case, the player strategies cancel each other out, and effectively sets them back to fair strategies where $\chi = 1$. [2] argue that these single-memory ZD strategies always dominate payoff-maximizing evolutionary strategies.

[8] has defined that any strategy for game theory succeeds in a population only if it can compete versus itself. That is, a strategy only succeeds if it can also fare well against strategies that are very much like themselves. These are called evolutionary stable strategies. [9] show that extortionist strategies are evolutionary unstable, because they make themselves extinct whenever they play against themselves (or opponents very similar to themselves). An example of an extortionist strategy is shown in Fig. 4. We read from Fig. 5 that with ZD strategy X and non-ZD strategy Y, $S_X > S_Y$ for all $\chi > 1$. For all $\chi > 1$, $p_4 = 0$ to keep $\bar{p}$ valid, meaning the strategy will never cooperate when the last round was P. Therefore, when ZD strategy X plays against ZD strategy X', mean payoff becomes P. In other words, $S_X = S_{X'} = P$. However, Fig. 5 also

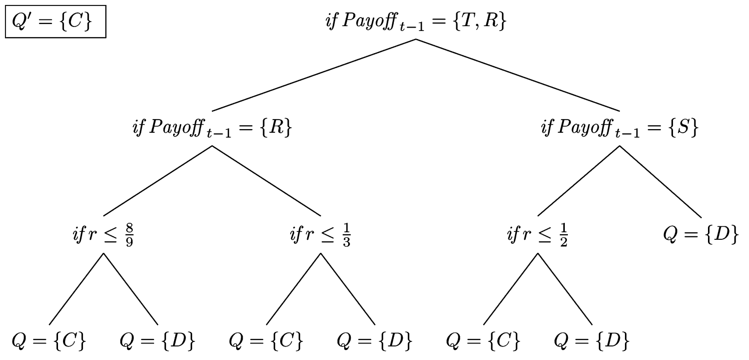

Fig. 4. Model for the successful strategy Extort-2, named this way because its extortion rate χ is set to 2 and is thus an extortionate ZD strategy. In essence, ZD strategies are simply a set of conditional probabilities based on the last round of play. Because this strategy uses probability, we use r as a shorthand for $z \sim U(0, 1)$. The extortion component is best visible in the left branches of the decision tree, where a cooperative outcome R has a 1/9 chance to result in a defection (this is extortion). The lack of generosity in this strategy is visible in the right branch of the decision tree, where mutual defection will always lead to more mutual defection.

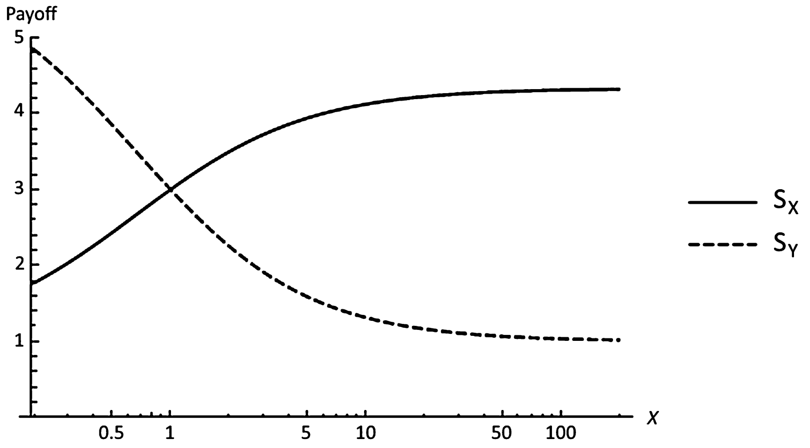

Fig. 5. Graph showing the payoffs for S_X and S_Y given the extortion policy for player X. $\chi < 1$ are generous ZD strategies, $\chi = 1$ are non-ZD strategies, and $\chi > 1$ are extortionate ZD strategies. The limit for this extortion policy lies at $S_X = 13/3$, but this is obviously infeasible, as $S_Y = 1$, and few strategies allow for such extortion to continue. Perhaps strategies with $\chi < 1$ would allow it.

shows that for all $\chi > 1$, $S_Y > P$, excluding $\lim_{\chi\to\infty}$. It is the generous ZD strategies, where $\chi < 1$, that are evolutionary stable instead. These strategies, when confronted with each other, will have a mean payoff of R. Therefore, extortionate ZD strategies will eventually go extinct as they cannot face each other, and the idea that for a single game of the Prisoner's Dilemma, defecting is the optimal strategy while the IPD the cooperating subset of strategies are the stable strategies, lives on.

4. Deception

We can now map arguably the most important subset of all valid strategies in the IPD, the set of all memory-one strategies, in Fig. 6. It shows the relations between cooperating strategies, defecting strategies and the previously explained ZD strategies.

A problem is deceptive when lower-order evolutionary building blocks that are combined into larger building blocks do not lead to a global optimum. A more theoretical formulation of this concept is given by [10], where given a fitness function on some optimization problem, when all of the lowest-order hyperplanes associated with a solution can be correctly determined, then the problem is not deceptive. If any of these lowest-order hyperplanes are incorrect, the intersection of all hyperplanes will end up somewhere vastly different than with a non-deceptive problem. Thus, a problem is deceptive when at least one part of the problem is deceptive.

Intuitively it is quite easy to show examples of deceptive problems. An example brought forward by [11] is the Chinese finger trap toy, a simple puzzle that traps the fingers in both ends of a small cylinder. The objective here, is to free your fingers. The first impulse one would have would be to pull apart your hands, but this only tightens the trap's grip on your fingers. Instead, the solution is to push your fingers towards each other, which is exactly the opposite of what one would expect to do when presented with such a problem. In more abstract terms, when one presents this problem to an evolutionary algorithm with the objective is to maximize the distance between fingers, any input corresponding to "pull

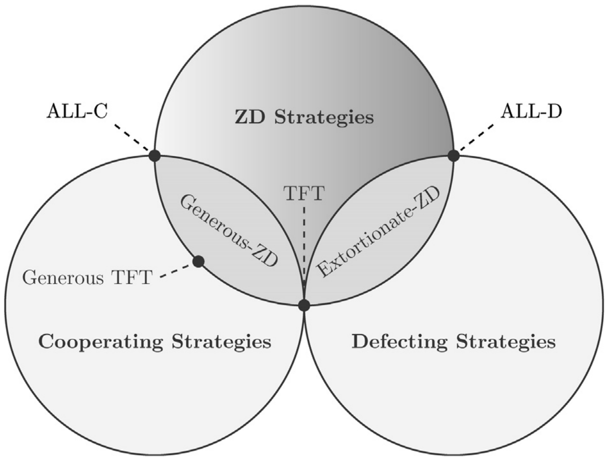

Fig. 6. A Venn diagram of the three sets of single-memory strategies. Single memory strategies are strategies that only consider the most recently played game. ALL-C is a cooperating strategy that is of maximum generosity, and so sits furthest away from defecting strategies. ALL-D is the exact opposite, being maximally extortionate. The shading shows the intuitionally effective display of variations in ZD strategies with respect to its χ, with low χ on the left and high χ on the right. TFT, technically a ZD strategy, sits in the middle of all three, being equally cooperative as defecting. Its extortion rate is at exactly 1. The subsets Generous-ZD and Extortionate-ZD are where extortion policies are $\chi < 1$ and $\chi > 1$ respectively.

Table 1
The conditional probabilities listed for six ZD strategies. The last three we consider deceptive, the first three are not deceptive. $P(C|Y)$ shows the probability for playing cooperate on the next round given that the previous round's outcome was of Y (for example, the probability that EXTORT-2 cooperates after being betrayed equals 1/2).

	ALL-D	ALL-C	TFT	GTFT	ZDGTFT-2	EXTORT-2
$P(C\|CC)$	0	1	1	1	1	8/9
$P(C\|CD)$	0	1	0	1/3	1/8	1/2
$P(C\|DC)$	0	1	1	1	1	1/3
$P(C\|DD)$	0	1	0	1/3	1/4	0

both hands" when maximized will lead to high fitness values (that is, distance is at its highest when both hands are pulled with the greatest force). It is easy to see that such behavior will lead to a local optimum, but it would be very difficult for the algorithm to evolve from this behavior to the behavior that is exactly opposite of these supposedly successful behaviors.

To relate this to Prisoner's Dilemma strategies, non-deceptive strategies belong to the set of either cooperating strategies or defecting strategies. The ultimate intention of any such strategy would be to either cooperate as often as possible, or defect as often as possible. Essentially, if the last game round was of mutual cooperation, a defecting strategy will attempt to betray such cooperation, and a cooperating strategy will attempt to continue cooperating. By contrast, all strategies in the set of memory-one ZD strategies, exempting ALL-D, TFT and ALL-C, are deceptive. Notice that these are exactly the three strategies that lie in the intersections of the circles of the shaded Venn diagram in Fig. 6. This is not coincidental. These are the only strategies that are part of the set of ZD strategies for which all their probabilities for cooperating are either set to 0 or 1. Examine Table 1, which shows the conditional probabilities for a select number of elements in the set of ZD strategies. The deceptive strategies GTFT, ZDGTFT-2 and EXTORT-2 each have at least one conditional probability set between the range of 0 and 1. These three strategies have performed exceedingly well in the latest IPD tournament [12], coming high in either score or wins (but not both). We will empirically show in later sections that it is indeed these strategies that are difficult to solve for a normal evolutionary player, and that NOVELTY is capable of dealing with these deceptive strategies.

4.1. Novelty search

NOVELTY makes use of a new and, in many applications, yet unproven method named novelty search [3]. Current research using novelty search is popular and far reaching, from evolving artificial Poker players [13] to space-exploring robots that evolve their locomotion strategy at varying gravity levels [14]. The main motivation for this research was the argument that our application of fitness in evolutionary methods, where the goal is always to obtain a higher fitness, determines the direction of evolution. Evolutionary algorithms run the risk of becoming stuck in a local optimum, and there is a lot of research dedicated to solving that issue. For example, simulated annealing sometimes performs harmful changes to the evolutionary path in order to break free from potential local optima whirlpools [15]. Instead, novelty search ignores the entirety of the objective function, which is rarely a perfectly fitting objective function with respect to the problem domain, and searches explicitly for behaviors that are unique.

To apply novelty search for evolutionary algorithms, generally the NeuroEvolution of Augmenting Topologies (NEAT) method is used, developed by [16]. NEAT begins evolution with a population of artificial neural networks, initialized to

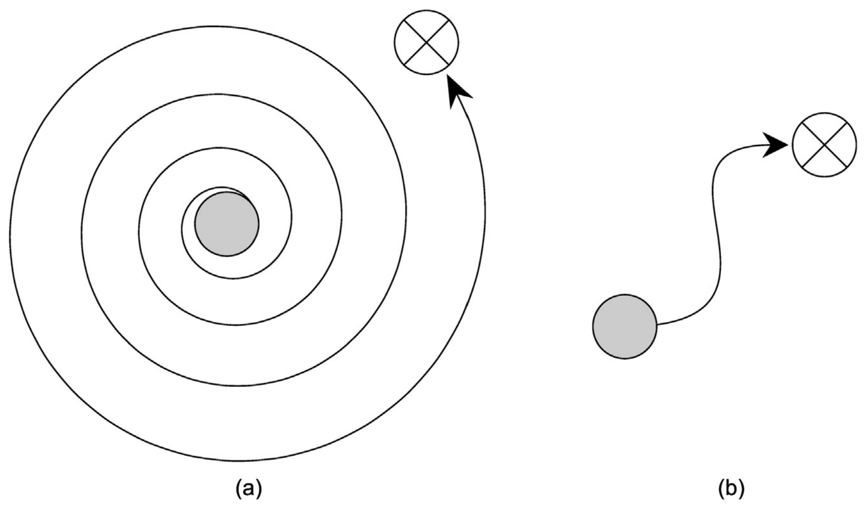

Fig. 7. Conceptual representation of the evolutionary path of any evolutionary algorithm using an objective function (a) and novelty search (b) to find the solution (marked by a crossed-out circle). The further away from the start, the higher the complexity of the solution. The spiral-themed path in (b) represents how the algorithm does a thorough search through the search space, starting at low complexity, and not affected by any goal other than the search for novelty. When a problem is deceptive, the novelty search is undisturbed by deceptions, whereas in (a) the path might not be as optimal as presented, or ever even reach the solution.

only the bare essentials. A neural network in its most basic form is a set of input nodes and output nodes, where each input node is connected to each output node. Whenever an individual from the population, a phenotype,[1] is evaluated, each input node is given some value based on the problem domain. Upon activation, each node, starting from the input nodes, transforms its input into another value and communicates that value with every node it is connected to. After activation, the values in each output node are used to specify the behavior of the phenotype. These neural networks grow in complexity by adding hidden nodes and allowing for speciation within genomes. Hidden nodes lie between input and output nodes, and only exist to further refine the possible outputs. This demonstrates the ability of NEAT to encounter simple behaviors before more complex ones. Speciation is grouping specific structures in neural networks together to form a species, protecting their innovation from the global population and allowing for such new species to have a chance against older, more sophisticated species. This allows for the diversity in genomes to exist that a technique such as novelty search needs to flourish.

Novelty search replaces the objective function. Instead, it looks for genomes that are more different than others, while making sure that the low complexity search space is examined first before moving on to more complex, yet novel, behaviors. Novelty is evaluated by using a domain-dependant metric. For example, in our simple Chinese finger trap example, the novelty metric would look for behaviors where the fingers end up at different distances from each other. In contrast, an objective function would simply reward high distances, keeping the search only to behaviors that purely obtain high distances at all times. It would likely only solve the trap once a behavior is sufficiently (perhaps exceedingly) complex enough that it contains a step in its behavior where distances between fingers are very short. It can be intuitively guessed which method would find behaviors that solve the trap faster. Fig. 7 shows a conceptual comparison between the two search methods.

The novelty metric computes the sparseness of a behavior within the search space. It does this by making use of a novelty archive, where old behaviors are kept and used to calculate the behavioral distance of a new individual. Generally, this measure of sparseness is calculated through a k-nearest neighbors algorithm. The sparseness p at point x is given by

$$p(x) = \frac{1}{k}\sum_{i=0}^{k} dist(x, \mu_i) \tag{4}$$

where μ_i is the ith nearest neighbor of x with respect to the novelty metric $dist$. The greater this Euclidean distance, the greater the novelty. A threshold is maintained that can rise and fall depending on the rate of approval of new sufficiently novel genomes. This archive gives a useful sample in showing where the search has been and where it is going. We can make use of a goal function to determine when to stop the search, but we must make sure not to use any such objectivity during the search.

We have so far discussed the intricacies of novelty search and its potential towards having greater success in solving deceptive problems, but an important issue remains: it does perform worse when confronted with non-deceptive yet rela-

[1] A genome is converted to a phenotype in order to distinguish it of its immutability in its structure.

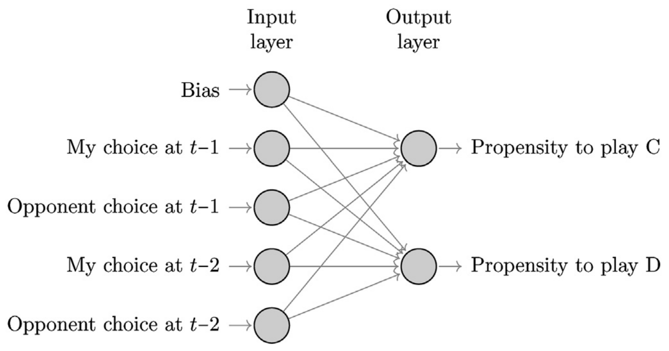

Fig. 8. An NOVELTY phenotype in its most basic state. This phenotype can look up to two iterations back in the past (memory-2). Hidden nodes will be added evolutionarily as generations transpire. After initialization, input and output nodes do not change.

tively complex problems. This makes intuitive sense, as shown in Fig. 7, the length of the path in (a) is decidedly shorter than that for (b), where the distance from start to goal is supposed to illustrate complexity. This impediment could possibly be overcome by introducing a two-step search, where it starts a search using objective measures, and when fitness is no longer increasing sufficiently each generation (indicating deception), reset and switch to novelty search.

Another concern that we have yet to properly discuss is the novelty metric used. Imagine if we had an evolutionary algorithm which uses a badly designed objective function, and subsequently proceeds to function unacceptably. Indeed, there exists a large volume of literature expanding on noisy and uncertain problems, for when problems have a set of objective results and clear decisions on which behavior is superior becomes much more difficult [17]. While the novelty metric is expected to properly identify key differences between two behaviors, there is no such things as having a multitude of performance criterion. There is only the uniqueness of a behavior that counts.

In [10], Whitley makes the claim that only challenging problems are deceptive. While this work is no good place to discuss the claim to the fullest extent, there is good reason to believe that within the confines of the Prisoner's Dilemma only the deceptive strategies are challenging for an evolutionary algorithm to overpower.

5. Experimental design

The software for which the empirical data has been generated is a modified version of [18], which can be found in Appendix B. The main objective for the experiment was to find whether there was a significant change in phenotype evaluation when using novelty search rather than objective functions. An experiment consisted of a large number of parameters, but most are set to default states. The fitness of a NOVELTY phenotype is evaluated by playing n games against each strategy in the strategy pool and calculating its score. When a strategy employs some form of probability, then n games are played an additional 99 times and the average score is calculated instead, to make sure high scores are not attributed to sheer luck. The total score, the cumulative of scores a player has obtained during each game against each opponent in a tournament, is considered a phenotype's fitness. An objective-function based search therefore searches for high scores.

There are numerous methods for a network to come to a decision given a problem. For example, a recent work writes about Markov Brains [19], where each node does not only have varying connections driven by evolution, but also their very function may change, a remarkable innovation. Our network of a phenotype, shown also in Fig. 8, has a variable number of inputs nodes, and exactly two output nodes. Inputs nodes must be at least 3, and must be an odd amount. The first input node is the bias node. Every subsequent pair of input nodes represents the outcomes of past iterations, where the first pair is the previous iteration its set of choices, and the ith pair the $t-i$th iteration its set of choices. The two output nodes signal its propensity to playing C or D respectively. All values are normalized to $[0, 1)$.

The reasoning for two output nodes instead of one, is to account for potential uncertainty within the network. To explain the difference precisely, imagine two countries with compulsory voting. Both countries employ a two-party system, and have exactly the same populations and both parties have the same ideologies. The only difference between the two countries is that in country A voters have a choice between party "Cooperate" and "Defect", and in country B voters have a choice between party "Cooperate", "Defect" or to vote blank. For the algorithm, country A represents a single output node where values greater or equal to 0.5 would result in a Cooperate action, and country B represents the double output version, where the node with the higher value results in its respective action. In country A the election results are 55% for "Cooperate" and 45% for "Defect" (in the algorithms terms, this would result in a single output value of 0.55). Country B's election results are instead 35% for "Cooperate", 40% for "Defect" and 25% for "indecisive" (two output values of 0.35 and 0.40), it shows that "Defect" has more decisive voters, and should be given the preference. The uncertain voters have been filtered out in country B.

Novelty search is implemented using a behavior archive and a novelty metric that is based on the choices made during a fitness evaluation. This sequence of choices a phenotype has made during each of its n games is called a phenotype's behavior characterization. A behavior archive, also known as a novelty archive, is a collection of behaviors that are sufficiently different from others during the same time frame. The behavior archive starts empty, and is filled with any behavior until its size equals k, which is the minimum amount in order to calculate a behavior's distance to the k nearest behaviors. When the archive reaches size k, it obtains the distance threshold p_{min} by calculating the average distance these first k behaviors are from each other. Each new behavior is compared to the k nearest behaviors, and is either accepted into the archive when the distance is greater than p_{min}, or refused otherwise. Each time a behavior is accepted, p_{min} is increased by 5 percent point. When refused, p_{min} is decreased by 0.2 percent point.

This same archive is used to show the empirical data found. Because it contains the behaviors of only novel phenotypes, it is also used for objective-based searches, because the visualizing the archive clearly demonstrates the search for both methods. During objective searches, the archive functions well as a filter for phenotypes that are too much alike in behavior, which would merely bloat the data with duplicate behaviors.

6. Results

Each of the following experimental results have a number of important parameters adjusted. The full details of all parameters for each experiment are relegated to Appendix B. This data is extracted from the novelty archive. During objective searches, the novelty archive was only used to show where the search was headed in the search space, but completely ignored for actual searching purposes. It was slightly modified to allow for better visualization of the search trend.

The goal of these experiments was to find whether novelty search performs better in finding behaviors which consistently have top scores in an Iterated Prisoner's Dilemma tournament versus exclusively "challenging" opponents. Challenging opponents here imply extortionate ZD strategies.

The following graphs in this section are visualizations of the novelty archive during its search. The following explanation is valid for all graphs in this section. The X axis is the index number of each element in the novelty archive, sorted by seniority. Early behaviors found in the first few generations are on the left side of the graph, and this should be interpreted as a temporal measure. The left-Y axis is the score obtained by a behavior, the right-Y axis is the number of wins obtained by a behavior. One strategy wins over the other when that strategy has accumulated a greater score than the other. Each small vertical bar on the graph displays an archived behavior's score (left Y axis). Scoring is the accumulation of rewards for each round played. This allows for nice visualization of the search, because a tournament of any size has a minimum obtainable score and a maximum obtainable score, and all scores in between show the possible scores obtainable by a behavior. These minima and maxima are dictated by the opponent pool of any such tournament. The average-wins curve shows the average number of wins each behavior up to some index i have obtained. Wins are not equal to a behavior's ranking. The average-ranking shows the average ranking each behavior has obtained up to some index i. The Y-axis for this curve is not displayed, as it is always a static value from 0 to 1, where 1 means having the highest score in a tournament, and 0 means having the lowest score in a tournament. Ranking r is calculated by

$$r = \frac{(n - p_i)}{(n - 1)} \tag{5}$$

where n is the number of players in a tournament, and p the position of player i after a tournament has ended. For example, third place is converted to a ranking of $0.\bar{3}$ in a tournament with 4 players. When the average-ranking curve climbs, the behaviors in the archive increase in quality.

In Fig. 9 we find a comparison between objective search and novelty search when the tournament pool is composed of only a large number of arbitrary non-extortionate strategies. It neatly displays the main difference between these two search methods. Objective search searches for the high score, and searches the narrow search space around such high-scoring behaviors. The big cluster of behaviors just under 30000 is where the objective search will eternally remain. This particular tournament only contains non-extortionate, non-deceptive strategies. Both the average-wins and average-ranking curves are near identical, because objectively maximizing either will lead to a tournament-winning behavior.

Novelty search very clearly demonstrates its exploratory power. It has considered most, if not all, behaviors that lead to all possible scores, including the ones objective search has spent the same time rediscovering over and over. However, because the problem is not deceptive, this more thorough search through the search space is a wasted effort. The tournament does not contain a strategy that is strong against evolutionary opponents.

We empirically display the recipe for a deceptive strategy. EXTORT-2 is a strategy that outperforms objective-searching evolutionary opponents, as high scores are obtained by cooperating with EXTORT-2. During tournaments, this would be an error, as going for high scores against EXTORT-2 means being extorted by EXTORT-2. Allowing EXTORT-2 to obtain an inevitably higher score increases its chances of winning the tournament, and thus decreasing the phenotype's chances. In Fig. 10, the number of behaviors that have won against EXTORT-2 is much greater in novelty search than in objective search. Additionally, the actual scores obtained in novelty search are also greater, because novelty search is taking more time in investigating that search space area. The average-ranking curve is climbing as it moves into greater complexity space, suggesting complexity is necessary to defeat EXTORT-2. The optimal behaviors found by objective search are discarded immediately, as their fitness are too low to be included in the next generation. They are found in the archive especially because the result of their behavior is drastically different from the rest. None of the high scores that are clustered during objective search win against EXTORT-2.

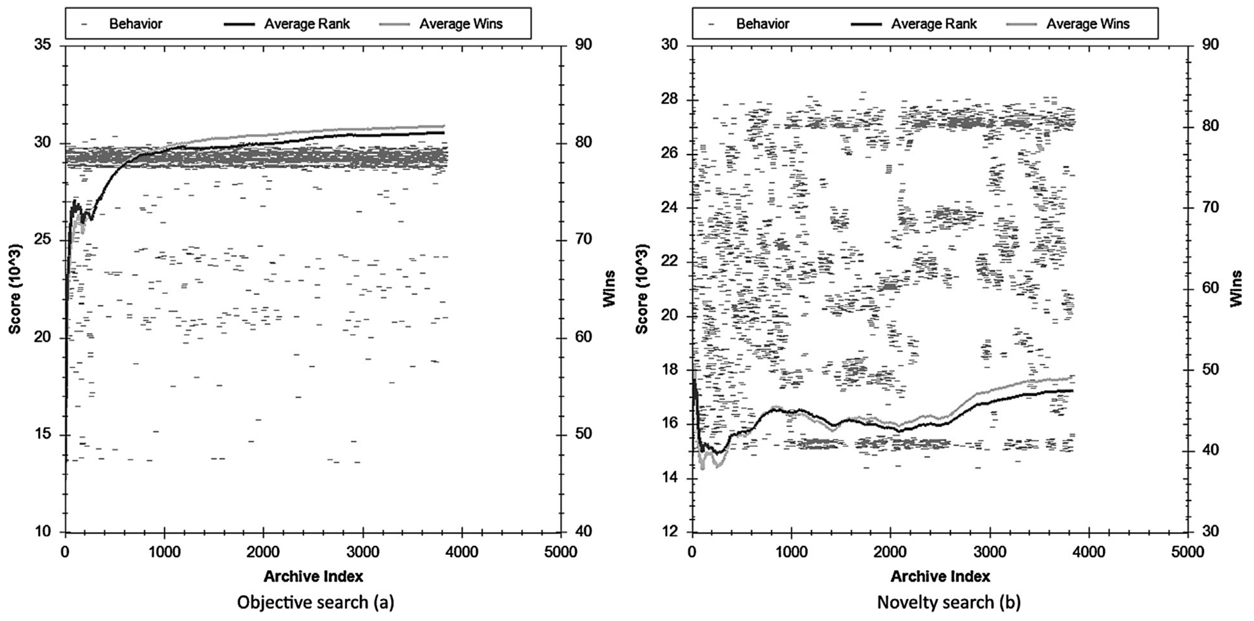

Fig. 9. IPD 100-round tournament with 100 randomly generated non-extortionate strategies, using the model from section 3. Both systems are using a population of 150 genomes with memory-5 and $K = 3$.

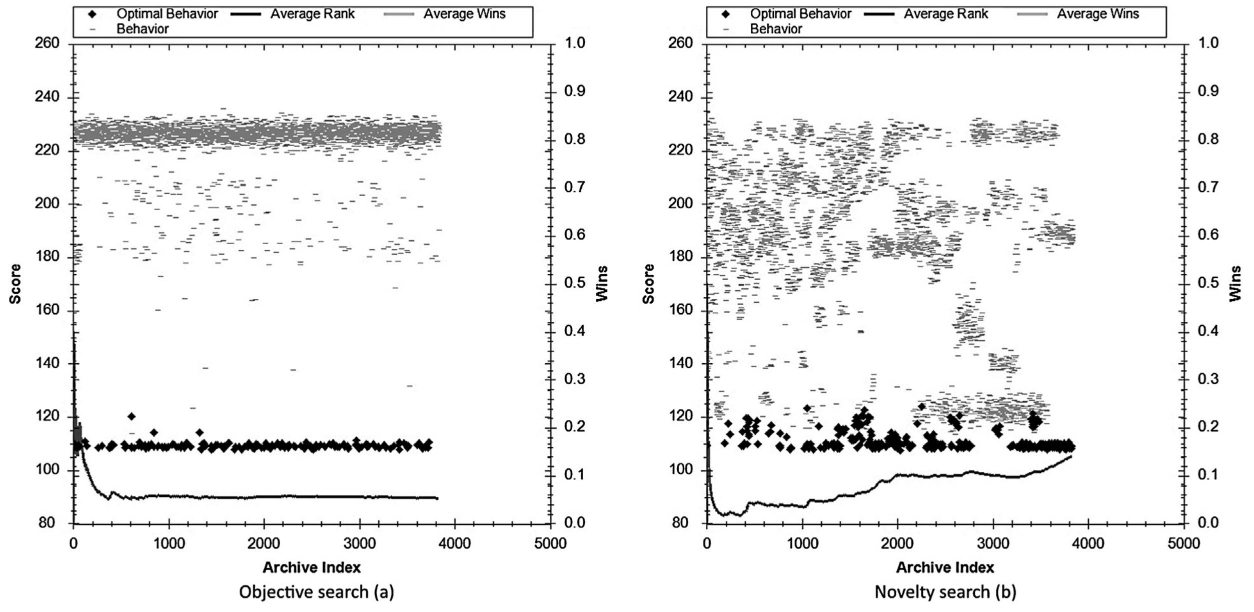

Fig. 10. IPD 100-round tournament with only EXTORT-2. Both systems are using a population of 150 genomes with memory-5 and $K = 4$. Some behavior bar symbols are replaced with diamond symbols to highlight behaviors which have obtained a higher score than EXTORT-2.

The concept of deception in strategies can be expanded to tournaments by including a number of deceptive strategies to a tournament. We use the ZD strategies found in Table 1 and graph the results in Fig. 11. We see similar behavior as in Fig. 10, where objective search is only interested in obtaining top scores, and novelty search explores all obtainable scores. The deception is shown in by comparing the average-ranking curves for both graphs. Objective search exclusively finds rather high scores, but its ranking remains forever what is the equivalent of 3rd place. Novelty search has managed second place by exploring all possibilities, and therefore breaks free from the deceptive trap. It is not optimal, it does not win the tournament, but it does perform better than the normal evolutionary phenotype.

Another object of deception is best explained by observing Tables 2 and 3. Each row shows the score the row strategy has obtained versus the column strategy. The score column is the cumulative score of a row, and the wins column displays the number of times the row strategy has won against a column strategy. A strategy obtains a 0.5 win when it ties with a

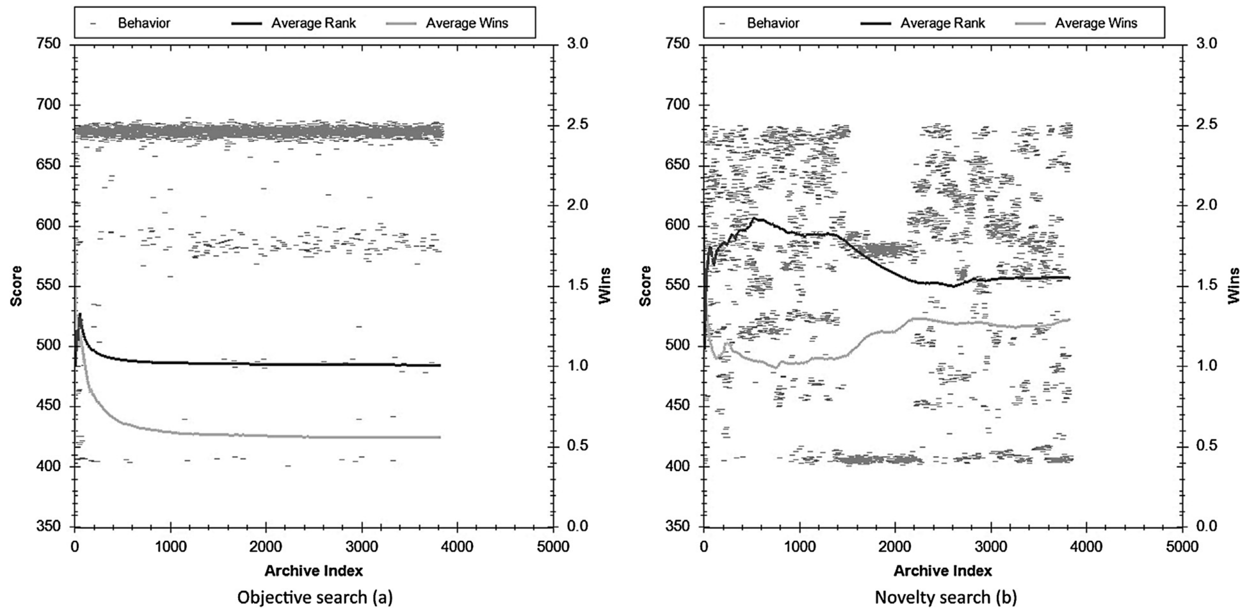

Fig. 11. IPD 100-round tournament with GTFT, ZDGTFT-2 and EXTORT-2. Both systems are using a population of 150 genomes with memory-5 and $K = 4$.

Table 2
The best phenotype's performance found during the objective search. GTFT wins this tournament, EXTORT-2 has outperformed all other opponents. Our phenotype is third place.

	Phenotype	GTFT	ZDGTFT-2	EXTORT-2	Score	Wins
Phenotype		150	300	213	663	0.5
GTFT	400		245	101	**746**	2
ZDGTFT-2	300	150		181	631	0.5
EXTORT-2	358	106	256		720	**3**

Table 3
The best phenotype's performance found during the novelty search. ZDGTFT-2 wins this tournament, our phenotype has the most wins. Our phenotype is second place.

	Phenotype	GTFT	ZDGTFT-2	EXTORT-2	Score	Wins
Phenotype		102	288	106	496	**2.5**
GTFT	102		245	101	448	1.5
ZDGTFT-2	253	150		181	**584**	0
EXTORT-2	101	106	256		463	2

strategy. Top scores and top wins are emphasized in bold. These tables show the relative strength of NOVELTY over classical evolutionary strategies.

Comparing the two tables, average scores for the objective phenotype and novelty phenotypes are respectively 690 and 498. The novelty phenotype is much lower, as it disallows the other strategies from obtaining great amounts of payoff from playing against it. It is much less cooperative, which is useful when playing against extortionate strategies. The novelty phenotype does cooperate, with ZDGTFT-2, but is generally less inclined. This behavior has allowed it to obtain second place as opposed to third place. Furthermore, notice that EXTORT-2 has won against all opponents in Table 2, yet did not win the tournament. Thus, winning a tournament is not as easy as simply defeating all opponents. It is also critical to consider by how much one is winning (or losing).

We have shown how NOVELTY has a better chance in withstanding ZD strategies than the normal evolutionary strategy. Deception is suggested to exist in an IPD tournament environment, and this deception is only present together with ZD strategies. While NOVELTY is not yet finding any behaviors that explicitly wins any configuration of an IPD tournament, the deceptive obstacles that ZD strategies have brought to IPD tournaments have been significantly transcended.

7. Discussion

Novelty search is principally inspired by evolution in nature. Nature's evolution is virtually unguided, yet has produced solutions to unimaginably difficult problems. Such solutions are never quite precise, they are very often estimations, never perfect. An artificial neural network is neither as precise. Depending on the problem, phenotypes can be structurally very different, but exhibit similar behavior. If the solutions these networks produce needed to be very precise, it often meant a solution either could not be found, or the objective function was overfitted so excessively that it could not produce any other solution than a precise one.

Slime molds are a great example in nature of novelty search in action. These giant multinucleated cells can extend more than a square decimeter. In an experiment, these molds are put on a plate that is connected by a bridge over to another plate loaded with their food. However, this bridge is covered in a bitter but harmless substance. After exhausting all other options, the slime mold extended a single thin tentacle across the bitterness to explore what was on the other side, and subsequentially finds the food [20]. In other words, after having exhausted all other options, novel behavior emerged which has led to the organism to overcome the deceptive bridge. If this problem was attempted by an artificial evolutionary algorithm, abstractly, the designers would have had three options. An intuitive objective function (positive fitness for finding food, negative fitness for finding bitter substances), an overfitting objective function (positive fitness for finding food, and for exploring landscapes with bitter substances), or the novelty search function. The first function might possibly never find food. The second function would find the food, but if the experiment was adjusted to also add food nearby without any deceptive bitter bridges, it would prefer the less optimal one. The third function would function similarly to how the slime mold functioned.

Nature's most remarkable creations are often very much general-purpose. Slime molds have shown to solve a wide array of problems, such as building networks, avoid traps, solving mazes and anticipating periodic events. Their adaptability is an impressive and valuable trait. The "No Free Lunch" (NFL) theorem states that if an algorithm performs well on a certain set of problems, then that same algorithm will have paid in performance when used for a different set of problems. More specifically, [21] explains that if an algorithm performs better than a random search on a set of problems, random search must perform better than that algorithm on another set of problems. This is an important specification, as novelty search is a generalist that essentially employs random search in a structured way. Novelty is explicitly a new behavior generated through randomization, rather than simply a random behavior.

Reduce this abstract set of problems to a set of tournaments $\kappa \subset K$, we have found novelty search to be a quick estimator for a solution of any κ, whether it contains deceptive strategies or not. One could design an algorithm that specifically overfits for ZD strategies, but as the NFL dictates, it will pay in performance against a different class of tournaments. Particular reports of strong results such an overfitted strategy may produce in a particular tournament κ are of limited utility.

Given these implications, there exist a large number of multi-agent learning algorithms that are not properly compared. [22] makes the important case that experiments are often designed to advocate for their newly-designed algorithm rather than surveying the landscape with their own algorithm as one extra addition. That these experiments are also often conducted until different environment, making comparisons even more difficult. Here, too, these comparisons are not made empirically. This is due to an important observation made by [22, p. 21]: "Algorithm performance depended substantially on which opponent was played.". This observation is perfectly in line with the NFL theorem. A strong performance versus one algorithm means a weak performance versus another. These same algorithms are also, in theory, sensitive to being deceived. For example, no-regret learning is an algorithm, where regret is considered the difference between playing the best pure static strategy τ_s, and the payoff it received when playing some sequence of actions. Obtaining no regret is when this difference is zero, meaning every time the algorithm picks an action different to τ_s static action, it leads to a higher payoff.

However, such an algorithm is deceived by extortionate strategies. There is reason to believe that no-regret learning will always become an ALL-D type strategy, as any time the algorithm is extorted it will shift its weights down to more defections until its only defecting. It will never realize that balancing out the extortions with your own extortions will increase your overall payoff without necessarily increasing the extortioner's payoff relative to your own. This fact is elaborated properly when contrasting it against a TFT strategy. Assume for illustrative purposes that we already know the best pure static strategy is to cooperate. When playing n repeated games, it will find that defecting at the nth game is beneficial to its regret. We can continue this logic until the regret algorithm will defect continually against TFT strategies. Fictitious play is another algorithm that perhaps stands a better chance against at least extortionate strategies, as it assumes its opponent is playing an unknown and potentially mixed stationary strategy [23, pp. 76–90]. More generally however, further research into such algorithms would be welcome, as only a comparison between evolutionary players has been explored thus far.

When we consider that novelty search in essence is a very structured random walk, such that it is quite balanced with respect to the NFL theorem (that is, the algorithm does not pay much performance when up against any alternative set of problems), there might remain methods to upgrade the algorithm without upsetting this NFL-balance. Novelty search can be extended by enforcing minimal criteria, which prunes the space of viable behaviors, increasing efficiency [11]. In evolution, this minimum criterion is the ability to survive long enough to reproduce. While there are countless ways to organisms to survive and reproduce, the outcome is the same. Often, and especially during the experiments performed in this paper, our algorithm has found many behaviors which are structurally different yet express the same functionality. The minimum criteria in natural evolution is survival and reproduction, but in our algorithm it can correspond to anything. When the minimum criteria are not met then the behavior is discarded, otherwise the algorithm proceeds as normal.

In our IPD problem, an interesting minimum criteria design could be to generate the minimum criteria after performing some tests on a tournament prior to actually initiating the algorithm. We generate the minimum criteria by playing a number of very simple non-deterministic strategies, probably ALL-D and ALL-C. When these test strategies have not won the tournament after n trials, we can safely assume that these simple test strategies will never win. We could assume any strategy which perfectly mimics any test strategy to be under the minimum criteria and thus be pruned out of the search. We can easily find out whether a behavior equals a test strategy, such as ALL-D, by inspecting the response of the behavior using all possible input permutations and comparing it with the test strategy. We want to use non-deterministic strategies as they minimize the chance of randomness plays a role in the creation of minimal criteria. This concept would prune a large number of possibly complex structures which express very simple actual behavior.

Another upgrade to novelty search might be to initially start with an objective search-based algorithm prior to a novelty search-based algorithm. Objective search often is quicker at finding solutions to less challenging problems, but these are still problems that need solving. We can stop the search either when the goal solution has been found, or the top fitness found during a generation has not changed in the last n generations. When the latter is the case, we can assume there exists some deception and the novelty search algorithm can be started.

Novelty search has spawned a class of research which has been termed quality diversity search. The conclusions brought forward by [24] shows how the degree to which finding novelty tends to also lead to higher fitness should significantly influence what exact method is to be used. Finding the optimal quality diversity algorithm for this particular problem, outperforming deceptive strategies in an IPD tournament, may be akin to looking for a needle in a haystack. Good understanding of the strengths and weaknesses of these recent and future algorithms are critical to developing improvements to NOVELTY. However, it goes without saying that there is still much to be discovered on the application of novelty search and its offspring with regards to the IPD.

In summary, novelty search has allowed us to find strategies for IPD tournaments where the objective is not as simple as attempting to obtain the highest score versus other strategies. The novelty search allows us to find complex behaviors that can break through this deception and find strategies that consistently perform better in any tournament configuration.

7.1. Conclusion

This paper has made use of the novelty search algorithm to produce better performing strategies to play an Iterated Prisoner's Dilemma tournament. Winning a tournament tends to be a highly deceptive endeavor due to the interactions among the strategies. We have presented what constitutes a deceptive tournament, and have shown novelty search to more consistently find a greater variety of behaviors that break through the deceptive barriers than objective search would. Finally, we have discussed how the "No Free Lunch" theorem is respected by using novelty search, and have explored some ideas on how to improve the used methods. The conclusion is that finding evolutionary behaviors in Iterated Prisoner's Dilemma tournaments is difficult when the opponents are of the extortionate kind, and that novelty search offers an approach for evolutionary algorithms to combat these recently developed strategies.

Appendix A. Strategy models

The models in this appendix are examples of our system that can randomly generate valid strategies for the IPD, such that it could be considered a statistical sampling of the set of valid strategies for the IPD (Figs. A.1–A.5). What follows is a complete exposition of the system.

Any static strategy is expressed as (Q, E). The strategy outputs a decision from the decision set $\{C, D, R\}$, where R represents a random choice, by extracting the first element from the sequence of decisions Q. If Q is empty, it instead evaluates decision tree E first, then tries to extract from Q again. E is expected to populate Q, it consists of a hierarchical set of nodes, where at the leafs it assigns a sequence to Q.

The decision tree E contains various types of nodes. The root and internal nodes are conditional nodes, and the leafs either assign a sequence to Q or jump to another conditional node. The payoff-conditional node is an evaluation of payoffs,

$$\langle \mathrm{L}: \mathit{if}^{\alpha,\beta} \mathit{Payoff}_{t-k} = \chi \rangle \tag{A.1}$$

where $k > 0$, and $\chi \in \mathcal{P}(T, R, P, S)$. L represents a textual label. At the start of every run through decision tree E, two variables $A = B = 0$ are initialized. When a conditional node evaluates, α increases A and we jump down to the left node as the node evaluates to true, or β increases B and we jump down to the right node as the node evaluates to false. α, β and L are optional, $\alpha = \beta = 0$ when undefined. Note that $\{T, R, P, S\} \Rightarrow \top$ and $\emptyset \Rightarrow \bot$. The first iteration of an IPD is when $t = 0$. The value-conditional node is an evaluation of values,

$$\langle \mathrm{L}: \mathit{if}^{\alpha,\beta} u \le v \rangle \tag{A.2}$$

where $u, v \in \mathbb{Z}$, but u and/or v may also be functions that output a value that is a member of $\mathbb{Z}$. When a payoff-conditional fails to evaluate, which is when $t - k < 0$, then Q becomes the result of a predefined function Q' and the evaluation of E immediately ends. Computer scientists may imagine E to be inside a try-block, and Q' to be the contents of a catch-block. Q and Q' are a sequence of choices of the form $Q = [i \le j]\{q_1\}\{q_2\}$, or without a conditional, $Q = \{q_1\}$. In general, q_n is shorthand for $\{x_0^{y_0}, x_1^{y_1}, \cdots, x_n^{y_m}\}$, where $x \in \{C, D, R\}$ and $y \in \mathbb{N}_0$. When the evaluation of $[i \le j]$ is true, $Q = q_1$, otherwise $Q = q_2$. y_i denotes the number of times x_i is repeated, so for example, $\{C^3, D\} = \{C, C, C, D\}$. To avoid empty sequences, any Q that happens to be empty will default to $\{R\}$, such as in the case of $\{C^0\}$ for example.

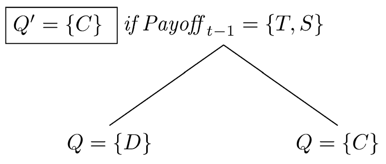

Fig. A.1. Model for the Pavlov strategy, or win-stay lose-switch strategy. Pavlov has no randomization component, so ξ is 0, it starts with cooperating, so Q initializes to $\{C\}$. When Q is empty, it requests a new Q from the tree. Pavlov considers both T and R a win, both of which result from a defect and cooperate choice respectively. If it encounters a T, it will "stay" by defecting once more. If it encounters a S, which means it has cooperated last round, it will "switch" to defecting. Q', α and β are omitted because they serve no purpose in this model.

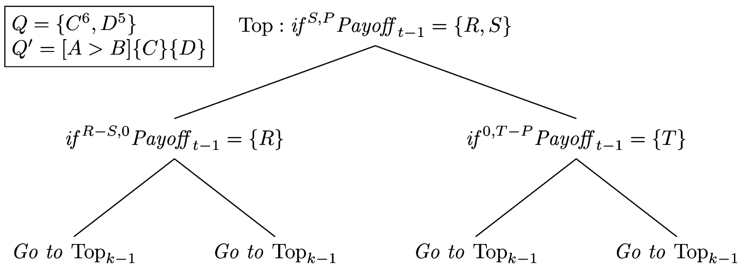
$Q = \{C^6, D^5\}$
$Q' = [A > B]\{C\}\{D\}$
$\text{Top} : if^{S,P}Payoff_{t-1} = \{R, S\}$
$if^{R-S,0}Payoff_{t-1} = \{R\}$
$if^{0,T-P}Payoff_{t-1} = \{T\}$
$Go\ to\ \text{Top}_{k-1}$
$Go\ to\ \text{Top}_{k-1}$
$Go\ to\ \text{Top}_{k-1}$
$Go\ to\ \text{Top}_{k-1}$

Fig. A.2. Model for the Adaptive strategy. Adaptive generally starts with six cooperates followed by five defections, then takes choices which have given the best average score, recalculates after every move. T, R, P, S values for α and β refer to the actual payoff values for the PD.

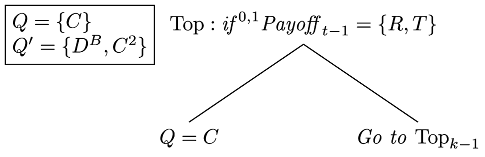
$Q = \{C\}$
$Q' = \{D^B, C^2\}$
$\text{Top} : if^{0,1}Payoff_{t-1} = \{R, T\}$
$Q = C$
$Go\ to\ \text{Top}_{k-1}$

Fig. A.3. Model for the Gradual strategy. Gradual cooperates until the opponent defects, in such case defects the total number of times the opponent has defected during the game. Followed up by two cooperations.

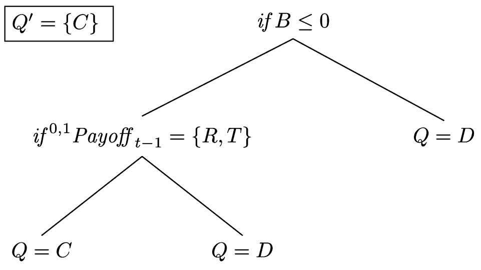

Fig. A.4. Model for the Grudger strategy. Cooperate until the opponent defects. Then always defect unforgivingly.

At the leafs of the decision tree E, we find result nodes or jump nodes. Result nodes can assign a sequence to Q,

$$\langle Q = [i \le j]\{q_1\}\{q_2\}\rangle \tag{A.3}$$

or may jump to a conditional node for re-evaluation, using the labels L as references. Naturally, when a conditional node has no label, it cannot be jumped to. A jump node is of the form

$$\langle Go\ to\ \text{L}_{k-1}\rangle \tag{A.4}$$

$k - 1$ signifies that any conditional nodes that use the k variable have those values reduced by one every time this jump node is visited. Value- and payoff-conditional nodes are always the internal or root nodes of the tree, and result or jump nodes are always at the leafs of the tree. The tree terminates when Q is assigned a new sequence. A valid model is where Q must initialize with at least one element, and the tree contains no duplicate conditional nodes.[2]

[2] Two conditional nodes are the same when they both have the same defined L label, or their k and χ variables are equal.

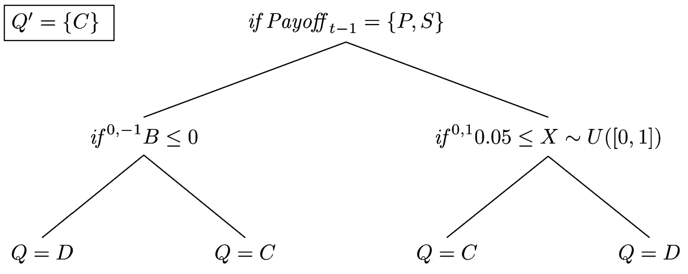
$Q' = \{C\}$
$if\ Payoff_{t-1} = \{P, S\}$
$if^{0,-1} B \leq 0$
$if^{0,1} 0.05 \leq X \sim U([0,1])$
$Q = D$
$Q = C$
$Q = C$
$Q = D$

Fig. A.5. Model for the Remorseful Prober strategy. Repeat opponent's last choice (i.e. Tit For Tat), but sometimes probe by defecting in lieu of co-operating. If the opponent defects in response to probing, show remorse by co-operating once. We use the B counter to keep track of whether we probed or not, but we could also look further back in the history of the outcomes to see whether we probed.

Appendix B. Experiment

B.1. Software

Author-modified source code forked from [18] can be found here: https://github.com/XGDragon/INTEL-sharpneat/releases/tag/v2.3.1-intel

B.2. Experimental parameters

The following tables show the parameters used for all experiments found in section 6. If a parameter varied between experiments, it will be marked *var*.

B.2.1. Evolution specific

Activation scheme	Acyclic
Population	150
Number of species	10
Elitism proportion	20%
Selection proportion	20%
Probability asexual offspring	50%
Probability crossover	50%
Probability interspecies mating	1%

B.2.2. Genome specific

Connection weight range	5
Probability connection weight mutation	98.8%
Probability neuron add mutation	0.1%
Probability neuron add connection	1%
Probability neuron remove connection	0.1%

B.2.3. Domain specific

Number of games	100
Strategy pool	*var*
Random robust check[a]	100
Random player seed	9865
Memory ($inputnodes \times 2 - bias$)	5
K	*var*
Evaluation mode	*var*
Evaluation limit	100000

[a] The amount of runs done versus a non-deterministic strategy. The resulting payoffs from each rerun is averaged and returned.

References

[1] J.W. Weibull, Evolutionary Game Theory, MIT Press, 1997.
[2] W.H. Press, F.J. Dyson, Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent, Proc. Natl. Acad. Sci. USA 109 (26) (2012) 10409–10413.
[3] J. Lehman, K.O. Stanley, Abandoning objectives: evolution through the search for novelty alone, Evol. Comput. 19 (2) (2011) 189–223.
[4] R. Axelrod, W.D. Hamilton, The evolution of cooperation, Science 211 (4489) (1981) 1390–1396.
[5] M. Nowak, K. Sigmund, A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game, Nature 364 (6432) (1993) 56.
[6] R. Axelrod, D. Dion, The further evolution of cooperation, Science 242 (4884) (1988) 1385–1390.
[7] W. Slany, W. Kienreich, On some winning strategies for the Iterated Prisoner's Dilemma, or, Mr. Nice Guy and the Cosa Nostra, in: The Iterated Prisoners' Dilemma 20, 2007, p. 171.
[8] J.M. Smith, Evolution and the theory of games, in: Did Darwin Get It Right?, Springer, 1988, pp. 202–215.
[9] C. Adami, A. Hintze, Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything, Nat. Commun., arXiv:1208.2666.
[10] L.D. Whitley, Fundamental principles of deception in genetic search, in: Foundations of Genetic Algorithms, vol. 1, Elsevier, 1991, pp. 221–241.
[11] J. Lehman, K.O. Stanley, Revising the evolutionary computation abstraction: minimal criteria novelty search, in: 12th Annual Conference on Genetic and Evolutionary Computation, 2010, pp. 103–110.
[12] A.J. Stewart, J.B. Plotkin, Extortion and cooperation in the Prisoner's Dilemma, Proc. Natl. Acad. Sci. USA 109 (26) (2012) 10134–10135.
[13] J.P. Bonson, A.R. McIntyre, M.I. Heywood, On novelty driven evolution in poker, in: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on, IEEE, 2016, pp. 1–8.
[14] G. Methenitis, D. Hennes, D. Izzo, A. Visser, Novelty search for soft robotic space exploration, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, 2015, pp. 193–200.
[15] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[16] R. Miikkulainen, K.O. Stanley, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2002) 99–127.
[17] E.J. Hughes, Evolutionary multi-objective ranking with uncertainty and noise, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2001, pp. 329–343.
[18] C. Green, SharpNEAT – evolution of neural networks [online], http://sharpneat.sourceforge.net/. (Accessed 11 May 2017).
[19] A. Hintze, J.A. Edlund, R.S. Olson, D.B. Knoester, J. Schossau, L. Albantakis, A. Tehrani-Saleh, P. Kvam, L. Sheneman, H. Goldsby, et al., Markov brains: a technical introduction, arXiv preprint, arXiv:1709.05601.
[20] D. Vogel, A. Dussutour, Direct transfer of learned behaviour via cell fusion in non-neural organisms, Proc. R. Soc. Lond., B Biol. Sci. 283 (1845) (2016).
[21] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[22] E. Zawadzki, A. Lipson, K. Leyton-Brown, Empirically evaluating multiagent learning algorithms, CoRR, arXiv:1401.8074.
[23] H.P. Young, Strategic Learning and Its Limits, OUP, Oxford, 2004.
[24] J.K. Pugh, L.B. Soros, P.A. Szerlip, K.O. Stanley, Confronting the challenge of quality diversity, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, 2015, pp. 967–974.