Advanced strategy representations for the iterated prisoner's dilemma

Rafael Slodzinski, Rüdiger Ehlers, Danny Seis, Thomas Bartz-Beielstein, Karlheinz Schmitt, Jörn Mehnen

Abstract— **Although many studies have been conducted on evolving strategies for the iterated prisoners dilemma (IPD) using evolutionary algorithms, most of them use rather simple representations like look-up tables. Knowledge about the evolution of strategies with complex representations remains limited. Complex representations may lead to new strategies and may be closer to the human decision making process ("thinking"'). Our main question is: Do complex represantations have substantial advantages over simpler ones in context of the IPD?**

In this paper we introduce two novel advanced representations: an exhaustive approach based on linear genetic programming and an evolution strategy based approach which reflects a psychological model. We show that both approaches lead to the generation of usable tournament strategies. We also use a cooevolutionary environment to study the evolution of cooperation, leading to diverse results. Additionally, the application of decision tree techniques to generate a strategy training set from real world tournaments is discussed. Strategies evolved with such training sets may perform well in subsequent real world tournaments, as other participants tend to imitate successful strategies of previuos tournaments.

I. Introduction

In the past, many researchers studied using evolutionary algorithms for the automatic generation of strategies for the iterated prisoner's dilemma (IPD)[1]. In contrast to the traditional way of developing a strategy by hand, only the representation and the evolutionary operators are to be defined and the strategy itself is the result of evolution. This is a very tempting approach, since tradeoffs in design of strategies are made automatically.

A couple of representations have been studied so far. For example, Axelrod [2] examined lookup-table based representations, Fogel [3] used finite automata and a couple of authors [4] suggested neural networks as a well suited representation. Of course, this list is far away from being complete since the amount of senseful representations can be considered infinite. However, most of these represetations follow the rule to be as simple as possible to avoid problems that arise with increasing complexity. Furthermore, simple representations limit the set of strategies than can be evolved. For example, the strategy GRADUAL[5] cannot be represented by a lookup-table.

Previous work on this topic differs in the goal of the evolution, defined by the fitness function. In the context of the IPD there are two completely different ways of defining the fitness function:

- The fitness of a strategy depends on the average payoff it gains when playing against the other members of the population of the evolutionary algorithm.
- The fitness of a strategy depends on the average payoff it gains when playing against a fixed set of strategies typically found in tournaments. This set is called a training strategy set.

The first way is also called coevolutionary, since members of the population adapt their behavior to the behavior of the other strategies in the population. It is favoured when evolution of cooperation is studied, but often the generated strategies do not behave well in other environments.

The second way is inspired by the idea that by using a training set of strategies which resembles most commonly found reaction scemes in practical application of the IPD, the evolved strategies play well in these practical applications and in IPD tournaments as well. In the past, these tournaments [1] were organized to bring together the strategies of state-of-the-art research and to provide an overview over those reaction schemes that can be expected when IPD players behave in a well-planned way.

A. The goal and structure of the paper

The central question we want to answer with this paper is whether the coevolutionary approach leads to the generation of strategies that have the ability to perform well in an IPD tournament and therefore compare it to the evolution against a fixed set of strategies. We discuss details of both approaches (section II.A) and for the later approach, we show a novel approach on how to get a representative set of strategies based on immitation of strategies that played in preceding tournaments.

Furthermore we use two different reprensentations, one complex and exhaustive approach based on linear genetic programming and one far simpler approach based on psychological models (section II.B). This enables us to examine both evolutionary environments with two representations, leading to a total of four combinations and a greater scope of the result.

To get a good comparison, it is neccessary to optimize the parameters of each representation approach. For this reason, we not only discuss the design decisions but also use a regression tree based approach for half-automatic optimization (section II.C).

Afterwards, we discuss how the interpretation of the results of evolution can be done. We introduce a set of test strategies, which is used to play against the evolved strategies and rate the evolved strategeies (section II.D). Then we shortly present a method to test if in case of the complex representation the generated strategies use the enlarged possibilities that are introduced by the complexity. This method is based

on the same tools we use to immitate strategies in chapter II.A and allows us to see if there is any difference in the usage of this enlarged possibilities beween the coevolution and the evolution against a fixed set of strategies (section II.E)

The results are written down in section III, which is divided by the destinction between the two evolutionary environments (coevolution and evolution against a fixed set of strategies) and the two representations. Afterwards, we analyse the results (section IV) and discuss outstanding differences between the two evolutionary environments and finally come to a conclusion (section V).

II. Beschreibung der AnsĎtze

A. Die verglichenen ReprĎsentationen

1) ES-Ansatz: Die Psychologie ist eine Wissenschaft, die das menschliche Erleben und Verhalten zu beschreiben, zu erklären und vorherzusagen versucht. Ihr vorrangiges Ziel ist es also, GesetzmäSSigkeiten hinsichtlich der Faktoren, die das Erleben und Verhalten beeinflussen und den Folgen dieses Verhaltens aufzudecken. In der Literatur finden sich zahlreiche Theorien und Modelle zur Wahrnehmung, dem menschlichen Denken und Motivation. Diese Funktionsbereiche lassen sich ferner aus verschiedenen Perspektiven betrachten (z.b. sozial-, allgemeinpsychologisch oder differentiell). Die in den nächsten Abschnitten vorgestellte Herangehensweise ist durch einige dieser Theorien inspiriert, wobei insbesondere Überlegungen der "'Motivationspsychologie"' und Konzepte der differentiellen Psychologie als Ausgangspunkte dienten [6] [7] [8].

Das angestrebte Ziel, eine günstige (erfolgreiche) Strategie für das IPD zu finden wird durch Nachahmen des menschlichen Verhaltens zu erreichen versucht. Einerseits soll der Ansatz der Tatsache Rechnung tragen, dass eine Person auf verschiedenartige Situationen flexibel reagiert, sie nicht selten sogar zu ihrem Nutzen zu beeinflussen versucht. Andererseits soll er interindividuelle Unterschiede in der Persönlichkeit berücksichtigen. So werden voraussichtlich auf dieselbe Situation zwei Personen unterschiedlich reagieren bzw. in ihr agieren. Um die erwähnten Modelle auf das IPD zu übertragen, soll eine Strategie in der Lage sein zwischen freundlichen und unfreundlichen Kontrahenten zu unterscheiden, zusätzlich aber ihre Schwächen erkennen können und nach Möglichkeit sie zu eigenem Vorteil ausnutzen. Darüber hinaus sollen sich Strategien in ihren Grundeigenschaften unterschieden können, insbesondere in Bezug auf ihre Aggressivität, Nachgiebigkeit oder Erinnerungsvermögen.

Das beobachtbare Verhalten einer Strategie wird also von zwei Komponenten bestimmt. Erstens wird es von den individuellen Eigenschaften einer Strategie wie dem Verhalten im ersten Zug, einigen Schwellwerten für Kooperation bzw. Defektion oder GedächtnisgröSSe bestimmt, zweitens von einer situationsabhängigen Komponente. Diese Komponente äuSSert sich in Form einer abschnittsweise definierten Aktivierungsfunktion $M(x)$ (siehe Fig.1, konstruierter Verlauf). Sie liefert zu jeder wahrgenommenen Situation, gemessen an der durchschnittlichen Punkten aus den letzten Zügen und auf das geschlossene Intervall $(0, 1)$ transformiert (x-Achse), den zugehörigen Aktivierungsgrad der Strategie (y-Achse). Dabei gilt die Grundregel: Je höher der Aktivierungsgrad, desto gröSSer die Tendenz zur Kooperation. Jedoch erst das Zusammenspiel beider Komponenten bestimmt das endgültige Verhalten einer gegebenen Strategie.Für die Suche nach günstigen Parametern wird auf eine Evolutionsstrategie zurückgegriffen. Die Strategien selbst sind während des Evolutionspozesses durch den folgenden Genotyp repräsentiert:

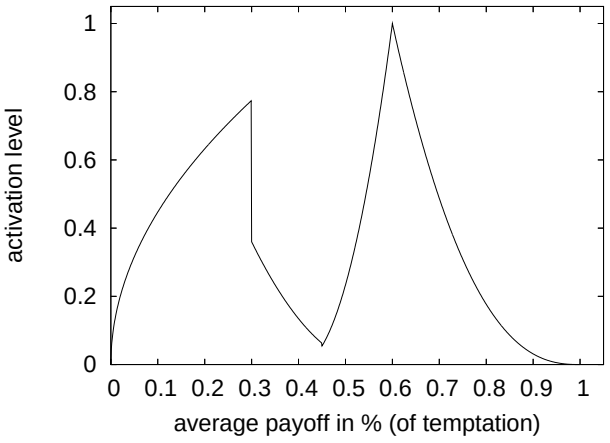

Fig. 1. Aktivierungsfunktion $M(x)$ als Bestandteil der situationsabhängige Komponente einer Strategie

- *GRUNDLEGENDE EIGENSCHAFTEN :*
- *firstMove* das Verhalten in der ersten Runde
- *memSize* Anzahl der maximal erinnerter Züge
- *quitThreshold* Schwellwert für ständige Defektion
- *threshold* Schwellwert für Kooperation
- *AKTIVIERUNGSFUNKTION :*
- *mainPeak*Gipfelpunkt der Freundlichkeit
- exp_{left} , exp_{right} transition's smoothness
- *auxiliaryPeak* Zwischengipfel der Freundlichkeit
- $auxiliaryExp_{left}, auxiliaryExp_{right}$ transition's smoothness

Die zweite Komponente, die bereits thematisierte Funktion $M(x)$, wird im Genotyp durch die letzten sechs Gene repräsentiert. Die Peaks können als Kontrollpunkte einer abschnittsweise definierten Aktivierungsfunktion aufgefasst werden, aus denen der endgültige Verlauf der Funktion noch zu generieren ist. An dieser Stelle sei angemerkt, dass Funktion in der Anzahl ihrer (Kontrollpunkte) Peaks variabel ist. Jedoch ist die Anzahl für ein bestimmtes Individuum während Evolution und folglich für eine Strategie fest. Sie enthält aber mindestens einen Peak, den *MainPeak* und die zugehörigen Parameter exp_{left} *und* exp_{right}. Für die Dauer des Optimierungsprozesses gilt, dass der *MainPeak* und sofern vorhanden die *auxiliaryPeak* lediglich den Abszissenwert des jeweiligen Kontrollpunktes speichern. Die Ordinatenwerte werden implizit auf 1 für den *MainPeak* bzw. *threshold*+0.1 für die *auxiliaryPeaks* gesetzt. Bei der

Konstruktion der endgültigen Strategie aus dem Genotyp wird die Aktivierungsfunktion $M(x)$ um zusätzliche Kontrollpunkte erweitert. Explizit werden die Punkte $P_0 = (0,0) und P_{last} = (1,0)$ eingefügt. Sofern mindestens zwei Peaks im Genotyp vorhanden sind, wird zwischen diese ebenfalls ein neuer Kontrollpunkt eingefügt. Bezeichnen also X_1 und X_2 zwei benachbarte Punkte im Genotyp (Peaks), so wird zwischen diesen ein neuer Punkt X_{bottom} an der Position $(X_2 - X_1)/2$ mit dem Funktionswert $M(X_{bottom}) = threshold/2$ eingefügt. Die Parameter exp_{left} , exp_{right} bzw. $auxiliaryExp_{left}$, $auxiliaryExp_{left}$ sind für den Verlauf der Funktion zwischen zwei Kontrollpunkten verantwortlich. Parameter mit dem Index $_{left}$ beziehen sich hierbei auf den Verlauf links eines Peaks. Analog gilt die Aussage für den Index $_{rigtht}$. Da $M(x)$ abschnittsweise definiert ist , müssen vor jeder Berechnung die korrekten Grenzpunkte $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ lokalisiert werden, so dass $x_1 < x < x_2$ gilt. Für diese gegebenen Punkte P_1, P_2 und ein beliebiges Argument $payoff$ mit $x_1 < payoff < x_2$ errechnet sich $M(payoff)$ nach folgender Formel :

$$M(x) = ((payoff - x_1) * (y_2 - y_1)/(x_2 - x_1) + y_1)^{e}xponent$$

Zu Beginn eines Spiels bestimmt *firstMove* ,ob die Strategie das Spiel mit einer Kooperation bzw. Defektion eröffnen soll. In der folgenden zweiten Runde wird unabhängig von der erhaltenen Auszahlung das Gegenteil der ersten Runde gespielt. Hat also die Strategie im ersten Zug kooperiert, so wird sie nun defektieren und umgekehrt. Weiterhin wird ab der zweiten Runde und bis zum Spielende fortlaufend das Gedächtnis der Strategie mit den erhaltenen Auszahlungen gefüllt. Wird hierbei die Speicherkapazität ,die durch *Memsize* begrenzt ist, überschritten, wird die älteste Auszahlung durch die zuletzt Erhaltene ersetzt.Anschliessend berechnet die Strategie ihre Punkteausbeute aus den erinnerten Runden. Liegt nun die Punktausbeute unterhalb von *quitThreshold*, so wird die Strategie für den Rest des Spiels ständig defektieren (AllD spielen). Überschreitet die Ausbeute jedoch diesen Schwellwert, so kann die wahrgenommene Situation von der Strategie bewertet werden.Hierzu wird die aktuelle Punktausbeute $payoff$ auf das geschlossene Intervall $(0, 1)$ abgebildet, dies entspricht der Division von Ausbeute durch Temptation (höchste Auszahlung). AnschlieSSend wird der erhaltene Wert in die Aktivierungsfunktion $M(x)$ eingesetzt. Im letzten Schritt wird der Funktionswert von $M(payoff)$ mit *threshold* verglichen. Ist der Aktivierungsgrad $M(payoff)$ höher als *threshold*, wird die Strategie kooperieren, im anderen Fall defektiert sie.

2) LGP-Ansatz: In this approach, the technique of linear genetic programming [9] is being used. A strategy consists of a list of commands which are executed by a simulated 3-register-machine[1] before each move the strategy makes. The list of available commands is given below. We have chosen to use stronly typed genetic programming with three different register types: "double", "integer" and "boolean". There are 4 internal registers of each of these types and an additional set of two boolean input register and one boolean output register.

Before the calculation of the next move, the last move of the opponent is stored into the first input register. If there is no last move because the strategy is about to make its first move, the second input register is filled with a boolean "true" value instead. Afterwards, the register machine gets started. When the genetic program has finished, output register contains the move the strategy will make. Between the rounds of a game the contents of the internal registers are maintained, they are only erased when the game has ended.

The usage of linear genetic programming instead of the more common, tree-like representaion has the advantage of simplifying describing loops and the transfer of internal states into the next round of the game at the same time. Furthermore, in practice common strategies usually only have a very small memory (for example, the strategy SPITE only stores if there has been a defection in the current game), which is reflected by the LGP very well. The available commands are:

- Add, Subtract, Multiply, Divide, Or, And, Not
- LoadRandom, LoadConstant
- SetIfZero, SetIfNotZero, SetIfGreaterZero, SetIfLessThanZero (Sets a boolean register if the contents of the second listed register fulfills a given criterium)
- Copy (Also copies values between different register types)
- Conditional Skip, Conditional Jump Back (Here the information whether to jump or not is taken from the specified boolean register)

All commands operate on all register types which are senseful for the specific command, leading to a total of 34 command/parameter combinations. Due to the fact that a 3-register machine is simulated, commands of all groups except the last one not only have one or two input registers, but also have an output register specified. To constrain the computation time, the maximum allowed repetitions of the "conditional jump back"-command is restricted to 100 times (per move). After that, this command is ignored. The maximum number of commands per strategy is 48.

Note that if there had been no restriction on the number of registers and commands per strategy, this approach would have been turing-complete[2] and would be able to represent all possible strategie for the IPD. However, due to limited computation time, these restrictions are neccessary. Anyway this can be called an exhaustive approach.

It can easily be seen that this approach is even more complex than the evolution strategy based one. Thus, evolution is even more computationally expensive and a higher computation time can be expected. To reduce this effect a little bit, self-adaptation has been added using an adaptation scheme similar to those commonly found in evolution strategies. Each individual has its own probability vector containing the probabilities of insertion, mutation and deletion of a command. These probabilities are multiplied with exponential

[1]TODO: Nachgucken, wie das Teil auf Englisch wirklich heiSSt

[2]Wie schreibt man das?

normally distributed random numbers before each generation. If a vector of these mutation probabilities is better than others, the chances that a good individual is created are higher and therefore the chances of survival are better as well. This way better mutation probability vectors are favoured. To fulfill all requirements for a working self adaptation scheme, the insertion/deletion probablities are constrained to a minimum of $0.1/length(Individual)$ and a maximum of 0.15 and the mutation probability is constrained to a minimum of $0.5/length(Individual)$ and a maximum of 0.1. This prevents probabilities getting too low (which sets the LGP stuck in a local minimum) and getting too high as well (this would turn out to a more or less random search).

B. Die evolutionĎren Umgebungen

1) Coevolution: In this contex, coevolution means, that the fitness of the strategies is detemined by a round robin tounament between all strategies of the population. This differs from most other EAs, because the fitness of a strategy now depends on the other strategies in the population. So even if the genotype of a strategy remains the same from one generation to another, the fitness may change, because other strategies change.

More precisly described we do the following to calculate the fitness:

1) Play a round robin IPD tournament with all strategies in the population.
2) To even out the influence of randomized strategies, step 1. is repeatet several times.
3) Calculate the fitness for all strategies as average payoff over all played rounds (accumulated points divided by the total number of played rounds).

The number of tournaments and the number of rounds per tournament is a parameter of the EA, see table ...

This approach resembles the natural evolution more closely than EAs where the calculation of fitness for a certain strategy is independent from the other strategies in the population, because in the natural evolution indivdiduals are not independant: the interact and influence each other. This opens the oppurtunity to examine the evolution of cooperation. Axelrod was successful in evolving cooperative population for simple strategy representations (Reference ..). Here we study the behaviour of the population with more complex representations.

2) Anpassende Umgebung: One of the most important questions in context of the evolution against a fixed set of strategies is how to get a good set of training strategies. These training strategies are best if they resemble the behaviour of most other commonly found strategies and are therefore representaive for the strategies to be expected as opponents. The aim of using this representative set is to have a strategy generated by evolution against this set which is capable of playing well against most other opponent strategies found in tournaments. If possible, the generated strategy ought to rip off the opponent, otherwise cooperation should be established.

TABLE I

IMMITATION TRAINING DATA TABLE FOR THE STRATEGY PERNASTY(WHICH IS THE STRATEGY TO IMMITATE), CONTAING A GAME AGAINST TFT. TO PROVIDE A BETTER OVERLOOK, ONLY THE LAST THREE MOVES INSTEAD OF THE LAST TEN MOVES ARE SHOWN.

Own last 3 moves (PerNasty)			**Enemy's last 3 moves (TFT)**			**Own move**
-3	**-2**	**-1**	**-3**	**-2**	**-1**	
N	N	N	N	N	N	D
N	N	D	N	N	C	D
N	D	D	N	C	D	C
D	D	C	C	D	D	D
D	C	D	D	D	C	D
...	...	...	...	...	...	...

In his publication about the evolutionally generation of lookup-table based strategies [2], Axelrod used the strategies he recieved for the tournament he organised. He made the observation that the overall performance of a strategy in the tournament can be calculated using its performance against a set of eight of the tournament strategies and infered from this fact that these eight strategies are suitable for being a training set of strategies as well.

Due to the fact that we have access to the strategies that have been recieved for the tournament organized by the project group 474 at the University of Dortmund, we use all of these strategies for adaptive evolution. However, most of these strategies were created by amateurs in scope of the IPD, therefore they are not sufficient for being the complete training set of strategies.

A good way to enlarge this set is to examine tournaments that took place before. Strategies that performed well and strategies similar to those are likely to be submitted to the next tournaments as well. However, the authors of these strategies normally do not publish details about the behavoiur of their strategies. One way to solve this problem is to apply machine learning techniques to the tournament results. This requires the results to contain a list of all moves the strategies made in all games. The goal of this approach is to deduce rules which are sufficient to create a strategy that immitates the behavoiur of a participent of the tournament. In most cases, this will not work exactly, but this no requirement anyway, since similar behaviour is well enough to serve as a part of the trainings set of strategies.

One machine learning technique that fits to this task is the deduction of decision trees. Each node represents one of the last 10 own or the opponent's moves. To get to know which move a strategy representated by a decision tree will make, the decision tree is traversed from the root to a leaf, always taking the branch which corresponds to the move that is represented by the node. The leafes and the branches of a node are either labeled with "C" (for cooperation) or "D" (for defection). For the branches, there is an additional labeling "N" which means that the branch will be taken in the case that the game has just begun and there there is no move

corresponding to the node. After the traversal, the strategy plays the move described by the leaf.

Before the generation of a decision tree can start, immitation training data tables are created. They contain lines holding information about the last 10 own moves, the last 10 enemy's moves and how the strategy to immitate reacted in that situation. Table I shows an example of such a immitation training data tables. Note that for simplicity only one game is shown, whereas in practice, lines from several games occur. After the table has been created, a C4.5-algorithm [10] is used to generate a decision tree out of it. For measuring the classification error, 5% of the lines of the table are removed and used as testing set.

Although many classical strategies can be represented by decision trees (e.g. TFT, ALLC, Spite), this does not apply to all of them (e.g. Random, Joss, GRADUAL). If immititation is tried in these cases, the $C4.5$-algorithm generates an decision tree which does not immitate the strategy in a complete way. To get to know if a decision tree generated out the tournament protocol of a participating strategy renders it correctly, taking a look at the classification error is a simple but effective way. It is determined by iterating through all the lines in the testing set. For each line, the last 10 own moves and enemy's moved are used as input for the decision tree and the outcome is compare to the move the strategy to immitate made in that situation (the last column of the table).

For determing the maximum classification error which can be allowed in order to make sure that only decision trees that immitate their corresponding strategies well enough will be part of the set of trainings strategies, we determine the classification error that occur when we try to immitate strategies that we know to be unimmitatable. The lowest classification error is the threshold value we use.

Diese Vorgehensweise sorgt natürlich nicht dafür, dass ganz genau die Strategien, welche sich durch Entscheidungsbäume darstellen lassen, auch tatsächlich in die Referenzgegnermenge übernommen werden. Diese Exaktheit ist aber auch gar nicht notwendig, da ja nicht genau die Strategien des Turniers repräsentiert werden müssen, sondern für die Benutzung als Referenzgegner auch sich einfach nur ähnlich verhaltende Strategien ausreichen.

Für diese Versuche wurden die Turnierprotokolle der Turniere 1 und 4 der CIG2005 verwendet. Dabei traten insgesamt 176 verschiedene Strategien an. Zusätzlich zu den auf diese Weise erhaltenen Strategien bestand die Referenzstrategiemenge aus weiteren 18 klassischen in der Literatur vorkommenden Strategien, wie z.B. GRADUAL, TfTT und PerNasty sowie den 36 der 56 bereits erwähnten beim Turnier der Projektgruppe 474 der Universität Dortmund eingeschickten Strategien.

C. Auswahl der Testmenge

Zur Bewertung des Ergebnisses der Evolution ist es sinnvoll, die erzeugten Strategien gegen eine weitere Menge von Gegnern (der sogenannten Testgegnermenge) spielen zu lassen, um festzustellen, wie gut sie sich gegen diese schlagen. Dadurch kann festgestellt werden, ob das Ergebnis der Evolution auch genügend generalisiert und wie gut es insgesamt ist. Eine solche Testgegnermenge sollte natürlich, wie die Trainingsgegnermenge, repräsentativ sein, mit ihr aber nicht übereinstimmen, da ansonsten bei der anpassenden Evolution nicht festgestellt werden kann, ob Überspezialisierung vorliegt oder nicht.

Als Testmenge werden die klassischen Strategien ALLC, ALLC, TFT, Pavlov, TfTT, STFT und GRADUAL sowie die 20 bei der Gegnertrainingsmenge nicht verwendeten Strategien aus dem Turnier die Projektgruppe 474 benutzt.

D. Wahl und Optimierung der Parameter

Bei der anpassenden Evolution und der Koevolution gibt es eine Reihe von Parametern, welche vor der Durchführung der Experimente mit Werten belegt werden müssen. Zur besseren Vergleichbarkeit der Ansaetze wurden diese moeglichst mit gleichen Werten belegt, an einigen Stellen sind aber Abweichungen sinnvoll. Beispielsweise wird die Evolution linearer Programme ueblicherweise nicht mit elitistischer Selektion (d.h. es werden fuer die naechste Generation immer die besten Individuen der vergangenen Evolution gewaehlt) betrieben, waehrend dies bei Evolutionsstrategien immer der Fall ist. Für viele andere Parameter ist es jedoch sinnvoll, diese automatisch zu optimieren. In Tabelle II ist für die jeweiligen Ansätze dargestellt, welche Parameter vorgegeben und welche automatisch optimiert wurden.

Für die automatische Optimierung der Parameter wird für jeden Parameter $2-3$ mögliche Werte festgelegt und mit einer kleineren Trainingsstrategiemenge für einige Kombinationen der Werte verschiedener Parameter die Evolution durchgeführt. AnschlieSSend wird ein sogenannter Regressionsbaum erstellt, welcher die Güte der Lösungen anhand der eingestellten Parameter vorhersagt. Diese Vorhersage ist natürlich sehr ungenau, es lässt sich jedoch ein Trend ablesen, welcher aussagt, bei welchen Parametereinstellungen die besten Ergebnisse zu erwarten sind. Das gesamte Verfahren wird in [11] beschrieben.

E. Interpretation of the evolved strategies

To find advantages and disadvanteges of our more complex representations we need to look at the strategies evolved by the correspondig EAs. The relationship between genotypes and phenotypes (the behaviour) is not direct, so usually it isn't possible to tell what a strategy does just by taking a quick look at the genotype. Interpreting the evolved strategies is a difficult problem, this is the price of complexity.

Interpretation is done in two ways:

1) Manual interpretation: When using LGPs interpretation is much like understanding uncommented assembler code. This is also called reverse-engeenering and usually it is a very time consuming process, but it is possible. Removing redundant commands (optimizing the code) helps, but it is still manual work. Roughly the same is true for the ES represantation, although it is a bit less complex, as there are lesser internal states of the strategy to care for. In both cases we also take a look on how the strategy plays against classic strategies like TFT.

TABLE II

OVERVIEW OF ALL PARAMETERS INVOLVED IN EVOLUTION - IN THE FIRST TWO ROWS, ALL PARAMETERS OF EVOLUTION ARE LISTED. FIXED PARAMETERS ARE FILLED WITH VALUES WE CONSIDERED SENSEFUL, ALL OTHER PARAMETERS ARE OPTIMIZED USING REGRESSION TREES. WHENEVER PRACTICAL REASONS LED TO A CHANGE OF A PARAMETER VALUES FOR COEVOLUTION, THE NAME OF THE PARAMETER IS LISTED IN THE THIRD ROW.

	Linear genetic programming	**Evolution strategy**
fixed parameters	mutation rate (self-adaptation) maximum program length (48 commands) available commands	maximum number of peaks
regression tree optimized parameters	population size (3000) children per parent (3) randomization of selection (low)	population size children per parent
different values in coevolution	population size randomization of selection (none)	population size

2) Interpretation with decisions trees: In fact, the decision trees introduced earlier, are a simple represantation. By applying the C4.5 algorithm to a move data set, gathered by playing an extensive tournament, we can "convert" a strategy with a complex represantation to a strategy with a simple representation: a decision tree. This decision trees are usually very easy to interpret. As long as they are small, this way is much faster than interpretation of the original genotype. Of course, simplicity comes with a price too: reduced expressiveness. A big class of strategies can't be represented with decision trees (for example all strategies using randomization).

But this limitation is a helpful tool to analyze our approach: if a strategy can be succesfully classified (with no classification errors) our complex representation is superfluous[3], a simpler representation would have sufficed. In the other case, if C4.5 fails to generate an error-free decision tree, we most-likely found a strategy that uses the superior expressiveness of our representations, which is clearly a benefit.

So we try using this way of interpretation, but fall back to manual interpretation if it doesn't work. Because of the time needed we can only interpret a limited number of strategies.

III. RESULTATE

A. Vorbereitung zur Evolution – Parameteroptimierung

Zusammentragung der Fakten:

Beim LGP ist der wichtigste Parameter die Anzahl der Kinder. Es gab da 3 Möglichkeiten: 1,3 oder 5 Kinder pro Elter. Im Baum steht ganz oben, dass die Anzahl der Kinder nicht 1 sein sollte - Eigentlich logisch. Danach kam sofort die Randomisierung der Selektion. Diese sollte kleiner als 0.35 sein. Zur Verfügung standen 0, 0.3 und 0.7. Eigentlich auch logisch, wenn man die Ähnlichkeit der Individuen in der Fitness berücksichtigt und wieviel Randomisierung so ein Wert von 0.3 oder 0.7 in die Sache bringt. Danach kam die Frage, ob einige Runden koevolutiv berechnet werden sollten. Diese Frage war noch relativ wichtig, Unterschied in der Fitness von 0.14 oder 0.28. Koevolution sollte nicht stattfinden. Letzten Endes die Frage, wie groSS die Population sein sollte. Dabei kan <= 5500 heraus. Es gab Werte von 1000 und 10000. Der Unterschied lag dabei zwischen 2.94 und 3.08.

[3]More precisely it may have influenced the search process in a positive way, but this is rather unlikely.

B. Vorbereitung zur Evolution – Referenzstrategiegewinnung

Der erste Schritt ist hier die Bestimmung des maximal zulässigen Klassifikationsfehlers. Zusätzlich zu dem vorhandenen Satz an klassischen Strategien wurde aufgrund ihrer Einfachheit noch eine weitere Strategie aus dem PG474-Turnier berücksichtigt, welche eindeutig nicht durch Entscheidungsbäume darstellbar ist. Diese Strategie namens Gralla50 spielt abwechselnd immer 50 Züge Kooperation und 50 Züge Defektion.

Ein diese Strategie immitierender Entscheidungsbaum könnte also eine Gestalt haben, in welcher die immitierende Strategie ihren letzten Zug immer wiederholt. Dies führt nur am Übergang zu Klassifikationsfehlern, so dass bereits dieser triviale Entscheidungsbaum nur eine Fehlerquote von 2% hat.

Interessanterweise ist dies auch die nicht durch Entscheidungsbäume vollständig darstellbare Strategie, welche beim Versuch, diese die eben diese darzustellen, die niedrigste Fehlerquote von nur $1,3732\%$ erreicht. Dies wird also die Grenze darstellen.

Von den 176 in den Turnieren 1 und/oder 4 der CIG 2005 teilgenommen habenden Strategien konnte dabei in insgesamt 144 Fällen aus dem Turnierprotokoll ein Entscheidungsbaum generiert werden, dessen Klassifikationsfehler unter der eben festgelegten Grenze lag.

C. Results of the evolution

For each of the four EAs, we took 20 strategies from different runs and generations to examine them further. To cover a broad range we didn't took only the fittest strategies, but also some from inbetween.

To measure the quality of this strategies, we played tournaments against the test enemy set. This set consists of 27

strategies, so our selected strategies may get rangs ranging from 1 to 28

Afterwards we interpetet the strategies i.e. we studied their behaviour,

1) Anpassende Evolution: (Rüdiger: hier fehlt noch das Abschneidne der strategien im Testturnier!)

Zusammentragung der Fakten: Beim LGP wurden 10 beste Individuen eines Lauf sowie 10 aus der Mitte eines Laufes genommen. Dabei kamen die folgenden Strategien heraus:

- 0+5x Spite pur. Plätze in Testmenge und Trainingsmenge: 14-16, 15-21
- 1+0x 2 Spite-Strategien abwechselnd, es wird also für gerade und ungerade Züge unabhängig gemerkt, ob schon einmal defektiert wurde. Plätze in Testmenge und Trainingsmenge: 1 und 10
- 0+1x Spite mit Vergebeung gegnerischer Defektion in der ersten Runde, danach Vergebung mit 50
- 1+0x Spite mit 7/8 Wahrscheinlichkeit für Vergebung. Plätze in Testmenge und Trainingsmenge: 19,6.
- 0+1x Tit-for-Tat, welchselt jedoch in ALLD nach zweimaliger Defektion des Gegners hintereinander. Plätze 1,13.
- 4+2x Spite mit 1/2 Wahrscheinlichkeit für Vergebung. Plätze in Testmenge und Trainingsmenge: 9-16,11-13.
- 4+0x Spite, jedoch mit Vergebung im 2. Zug (d.h. im ersten Zug darf der Gegner ruhig defektieren). Plätze 15, 6-9
- 0+1x Spite, jedoch mit einem "'Freischuss"', d.h. einmal darf die Gegnerstrategie defektieren.

Übrigens wurden die Integer-Register nur für diese 1/8-Wahrscheinlichkeit und für die Detektion des ersten Zuges (durch Überschreiben des Wertes während des Programmdurchlaufes während der Wert beim ersten Lauf ja noch mit 0 initialisiert ist) wirklich benutzt, die Double-Register werden gar nicht benutzt. In beiden Fällen kommen die entsprechenden Befehle natürlich in den Introns vor.

Ach ja, in der Trainingsmenge beste Strategie ist GreenShark, fügt man diese aber zur Testmenge hinzu und lässt dort ein Turnier spielen, erreicht sie Platz 20, beides mit einigem Abstand zum Vorder- und Hintermann. Das wäre also schon einmal "'das Hauptverantwortliche"' für das Abschneiden.

2) Co-Evolution: Zunächst betrachten wir die Evolution der Kooperation: Hierzu wurde die durchschnittliche Fitness gegen die Generation einiger repräsentativer Läufe abgetragen (Fig. 2).

Wie man sieht entwickelt sich beim ES-Ansatz recht schnell eine stabile Kooperation, d.h. eine durchschnittliche Auszahlung von 3 (entspricht gegenseitiger Kooperation). Beim LGP Ansatz kann man dies leider nicht beobachten. Die durchschnittliche Auszahlung bleibt in der Nähe von 1 (d.h. gegenseitiger Defektion. Nach Axelrod ist TFT erst vorteilhaft, wenn eine gewisse kritische Grenze von kooperativen Individuen überschritten ist; daher wurde der Population ein Anteil von 5% TFT hinzugefügt. Wie man sieht ist dies ausreichend, um Kooperation entstehen zu lassen![4]

Im Hintergrund dieser Ergebnisse betrachten wir nun das Abschneiden der evolvierten Strategien in einer Testumgebung: Hier zeigt sich klar die Überlegenheit der Strategien, die aus einer kooperierenden Population entnommen wurden (also aus einem Lauf des ES-CE oder LGP-CE mit 5% TFT). Die aus der LGP-CE mit 5% TFT entnommenen Strategien konnten, obwohl sie nicht auf bereits bekannte Strategien hin optimiert wurden durchaus passable Plätze in den Testturnieren erzielen: Das beste getestete Individuum erreichte Platz 7. Viele Strategien lagen im Mittelfeld (Platz 14-17). Dagegen landeten die Strategien, die aus der unkooperativen Population entnommen wurden meist nur auf dem letzten oder vorletzten Platz.

Strategien aus dem ES-Ansatz belegten meist die unteren Plätze (Platz 22-27), eine ereichte allerdings auch den ersten Platz (diese wird im nächstem Abschnitt detaillert dargestellt).

3) Interpretation of the evolved strategies: Die Benutzung des C4.5 Algorithmus zur KLassifikation der evolvierten Strategien mit Entscheidungsbäumen war nur beim LGP-CE erfolgreich. Hier konnten alle ausgewählten Strategien fehlerfrei klassifiziert werden. Die Interpretation der Entscheidungsbäume ergab folgendes: Alle Strategien, die aus unkooperativen Populationen entnommen wurden defektierten immer, also entsprechen ALLC. Bei den kooperativen Strategien dagegen fand sich einmal TFT(diese Strategie erreichte den besten Platz im Testturnier), einmal ALLC(immer kooperieren) und 8mal Spite, eine klassische Strategie, die kooperiert bis der Gegner das erste mal defektiert und dann nur noch defektiert.

Die teilweise dutzende Befehle langen Strategien aus dem LGP-AE konnten nicht mit Entscheidungsbäumen klassifiziert werden und mussten daher manuell interpretiert werden: (Rüdiger: bitte ergänzen!) (hier fehlt noch der ES Ansatz)

IV. Analyse

A. Analyse Strategiegewinning

Wie zu sehen ist, konnten über 80% der Strategien ausreichend genau immitiert werden. Der Erfolg des Verfahrens kommt nicht vollkommen unerwartet. Beispielsweise hat Axelrod(*Referenz einfügen*) in seinen Analysen von einer guten Strategie gefordert, dass diese nicht nachtragend sein darf. Bei einer nicht nachtragenden Strategie sind vor einiger Zeit gemachte Züge für die jeweils aktuelle Entscheidungssituation nicht mehr von Belang. Bei vielen solchen Strategien ist diese "'Vergebungszeit"' kleiner als 10 Runden, so dass diese, wenn sie nicht randomisiert sind, durch einen Entscheidungsbaum, dessen Knoten jeweils einen der letzten 10 Züge des Gegners oder immitieren Strategie repräsentieren, dargestellt werden können.

[4]Genauer gesagt: In etwa der Hälfte aller Läufe des LGP-CE mit 5% TFT entsteht gegenseitige Kooperation innerhalb der ersten 100 Generationen.

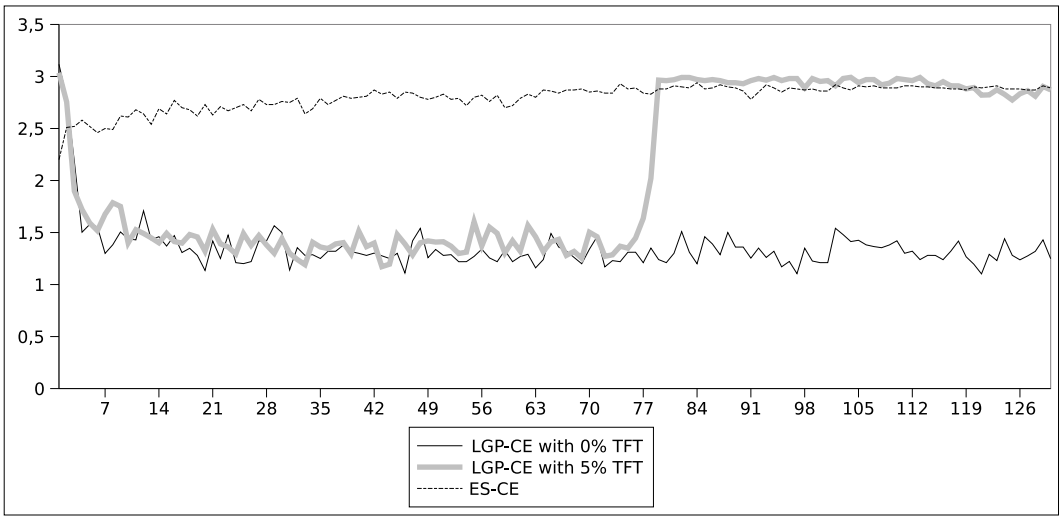

Fig. 2. Coevolutionary approach: the evolution of cooperation

Bei diesem Ergebnis ist es aber wichtig, zu beachten, dass solche immitierende Entscheidungsbäume den "'Kern"' einer Strategie eventuell nicht wiedergeben. Bei einer Strategie, welche jeweils abwechselnd 100 Züge defektiert und kooperiert (also gleichsam "'Gralla100"'), könnte einfach ein Entscheidungsbaum erzeugt werden, dessen Klassifikationsfehler geringer als der maximal zulässige ist. Diese durch den Entscheidungsbaum immitierte Strategie würde mit Kooperation beginnen und dann immer seinen letzten Zug wiederholen. Die Eigenart, nach jeweils 100 Zügen zwischen Kooperation und Defektion zu wechseln, würde jedoch nicht erfasst werden, obwohl diese essenziell für die Strategie ist. Im Allgemeinen ist der Klassifikationsfehler also als Kriterium für eine gute Immitation nicht ausreichend (es sei denn, er ist 0% und das Turnierprotokoll enthält Spiele gegen wirklich alle zu erwartenden Strategien). Für die Benutzung als Referenzgegner erscheint es jedoch sinnvoll, diese dennoch zu verwenden, da hier in jedem Fall eine bessere Anpassung an die konkreten Gegnerstrategien, welche in einem nächsten Turnier wieder auftreten können, zu erwarten ist.

B. Ergebnisse Anpassende Evolution

Zusammentragung der Fakten: Beim LGP sind verschiedenartige Resultate zu sehen. Zwar kommen eine Strategie aus dem Mittelfeld und eine der Spitze wirklich auf Platz 1 in der Testmenge, diese sind aber in der Trainingsmenge nicht so gut.

Was zu sehen ist, ist, dass durchaus einige Strategien herausgekommen sind, welche z.B. mit Lookup-Tables nicht zu beschreiben waren - AuSSerdem welche, an die niemand gedacht hätte und die somit in weniger komplizierten EA-Ansätzen möglicherweise schon durch den Aufbau ausgeschlossen worden wären (Frage an Rafael: Bei den Franzosen auch?). Beispielsweise wurde die Möglichkeit zur Randomisierung genutzt.

Zu sehen ist auSSerdem die Spite-Lastigkeit der Ergebnisse. Dies könnte auf die Test- und Trainignsmenge zurückzuführen sein. Sie ist möglicherweise zufälligerweise einfach blöd aufgesplittet. Andererseite könnte bei den Gegnern Spite-Artiges wirklich die beste Wahl sein, da die, die nicht freundlich sind, meist auch versuchen, den gegner rigeros auszunehmen. Zu guter Letzt könnte es auch sein, dass die Essenz der Gegnerstrategien durch die Immitation wirklich nicht übernommen worden ist.

Ideen zur Analyse / Conclusion:

- theoretische Überlegungen: Axelrod sagt, dass einfache Strategien am besten sind, aber er untersucht garkeine komplizierten strategien. dadurch, dass komplexe strategien von vorherein vom suchprozess ausgeschlossen sind, weil sie nicht darstellbar sind, kann man auch keine aussagen darüber machen. Man blendet eine ganze klasse von strategien aus, theoretisch betrachtet können die meisten einfachen repr. nicht mal reguläre sprachen erkennen (bsp lookup table ist untermenge der reg sprachen), die lgps dagegen können alle berechnenbaren sprachen erkennen (bzw könnten bei unendl. speicher), also eine wesentlich grössere klasse. bisher wurde also nur ein sandkorn am strand betrachtet. (hm, diese überlegung würde vllt auch in die einleitung passen: als motivation, aber auch in die conclusion: clearly further examination is needed because: ..) (danny)

C. Ergebnisse Co-Evolution

Passend zu: [12]. *Diesen Text schreibt wahrscheinlich Danny*

V. Conclusion

Dies ist ein weiterer Testtext. *Wird wahrscheinlich in einer gemeinsamen Redaktionssitzung geschrieben.*

Acknowledgment

The authors would like to thank Christian Lasarczyk for providing tips about genetic programming.

REFERENCES

[1] R. Axelrod, *The evolution of cooperation*. London: Penguin Books, 1990.

[2] R. M. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma." in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Morgan Kaufmann, 1987, ch. 3, pp. 32–41.

[3] D. B. Fogel, "Evolving Behaviors in the Iterated Prisoner's Dilemma," *Evolutionary Computation*, vol. 1, pp. 77–99, 1993.

[4] S. Y. Chong and X. Yao, "Behavoiral Diversity, Choices and Noise in the Iterated Prisoner's Dilemma," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 540–551, 2005.

[5] B. Beaufils, J.-P. Delahaye, and P. Mathieu, "Our meeting with gradual, a good strategy for the iterated prisoner's dilemma," in *Proceedings of the Fifth International Workshop on the Synthesisand Simulation of Living Systems*, C.-G. Langton and K. Shimohara, Eds. Cambridge, MA, USA: The MIT Press/Bradford Books, 1996, pp. 202–209.

[6] H. Heckhausen, *Motivation und Handeln*. Springer-Verlag, 1989, vol. 2. Auflage.

[7] P. G. Zimbardo, *Psychology and life*. Scott, Foresman (Glenview, Ill.), 1988, vol. 12th Edition.

[8] J. Asendorpf, *Psychologie der Persönlichkeit*. Springer-Verlag, 1996.

[9] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, Jan. 1998.

[10] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

[11] T. Bartz-Beielstein and S. Markon, "Tuning search algorithms for real-world applications: A regression tree based approach," in *Proc. 2004 Congress on Evolutionary Computation (CEC'04)*, G. W. Greenwood, Ed. Piscataway NJ, 2004: IEEE Press, 2004, pp. 1111–1118.

[12] P. Darwen and X. Yao, "On evolving robust strategies for iterated prisoner's dilemma," in *Progress in Evolutionary Computation*, ser. Lecture Notes in Artificial Intelligence, vol. 956, 1996, pp. 276–292.