Evolutionäre Algorithmen

Michael Herrmann mherrmann@gmx.org
Arno Mittelbach mail@arno-mittelbach.de

Abstract

Erste Versuche evolutionäre Algorithmen für die Programmierung von Agenten für Spiele mit unvollständigen Informationen einzusetzen zeigen, dass solche Agenten ihren statischen Alternativen weit überlegen sind. Auf den folgenden Seiten werden die Ergebnisse und die für die Implementierung eines Agenten wichtigen Details der Artikel (Kendall & Willdig, 2001), (Barone & While, 1997), (Barone & While, 1999), (Noble & Watson, 2001) und (Noble, 2002) zusammengefasst.

1. Einleitung

Die vorliegenden Artikel über evolutionäre Algorithmen in Pokerspielen lassen sich in drei größere Kategorien einteilen, an Hand derer wir auch diese Zusammenfassung strukturiert haben. Zunächst werden wir einen einfachen lernbasierten Algorithmus (Kendall & Willdig, 2001) beschreiben, um daran anschließend auf evolutionäre Strategien einzugehen (Barone & While, 1997), (Barone & While, 1999). Im Anschluss daran beschäftigen wir uns mit Pareto-Koevolutionären Algorithmen, behandelt in (Noble & Watson, 2001) und (Noble, 2002), in denen versucht wird einige Probleme mit evolutionären Ansätzen zu lösen. Abschließend werden wir in einem Fazit die für die Implementierung wichtigen Details und Hinweise noch einmal kurz und bündig zusammenzufassen.

2. Lernbasierter Ansatz

Der einfachste Ansatz um einen sich entwickelnden Pokerspieler zu generieren, ist durch einen lernbasierten Algorithmus. Ein solcher Algorithms wird in (Kendall & Willdig, 2001) beschrieben. Es wird gezeigt, dass ein solcher Algorithmus einem statisch programmierten Spieler überlegen ist.

Als Pokervariante wird eine Form des *Draw Poker*[1] mit zwei Wettrunden verwendet.

2.1 Statisch programmierte Spieler

Um die vier klassischen Pokerspieler zu simulieren wird jedem Spieler statisch vorgegeben was er bei welcher Handstärke machen soll (Fold, Call oder Raise und ggf. um wieviel erhöht werden soll).

Als Beispiel sei hier der Unterschied zwischen einem Loose Aggressive Spieler und einem Tight Passive Spieler angegeben. Der Loose Aggressive Spieler steigt aus dem Spiel aus, wenn er nichts besseres als ein Achter-Pärchen hat. Der Tight Passive Spieler steigt hingegen bei allem bis einschließlich einem Ass-Pärchen aus.

1. Siehe http://de.wikipedia.org/wiki/Draw_Poker

Weiterhin unterscheiden sich die statischen Spieler im Setzverhalten. Der Loose Aggressive Spieler geht im Spiel mit (Call) bei allem zwischen einem Neuner- und einem Königs-Pärchen. Der Tight Passive Spieler geht erst ab zwei Pärchen und einer Dreier-Karte bis einer Straße mit Ass als Kicker mit.

2.2 Sich entwickelnder Spieler

Der sich entwickelnde Spieler speichert für jede mögliche Hand einen Lernwert. Dieser wird mit 10 initialisiert, und kann keine Werte größer als 10 und kleiner als 0 annehmen. Der Lernwert wird so interpretiert, dass eine hohe Zahl für eine gute Hand und eine kleine Zahl für eine schlechte Hand steht.

Hat der Spieler nun eine bestimmte Hand, dann holt er sich den dazugehörigen Lernwert und generiert nach einem Algorithmus[2] seine Spielentscheidung. Nach Ende des Spiels wird überprüft ob der Spieler mit der Hand erfolgreich war oder nicht. War er erfolgreich, so wird der Lernwert zur Hand um 0,1 erhöht. Im Gegenzug wird er bei Misserfolg um 0,1 verringert. Dadurch werden gute Hände über die Zeit einen hohen Lernwert behalten wobei schlechte Hände einen niedrigen Lernwert bekommen.

In den Algorithmus, der die Spielentscheidung generiert, gehen verschiedene Größen ein. Diese sind z.B. der Lernwert der entsprechenden Hand, die momentane Potgröße und die Anzahl der Spieler, die im Spiel verblieben sind.

2.3 Experimentergebnis

Da der sich entwickelnde Spieler mit einem Lernwert von 10 für jede Hand startet, wird er am Anfang bei jeder Hand spielen und erhöhen. Er benötigt also eine gewisse Anlaufzeit, um zu lernen, welche Hände schlecht und welche gut sind. Deswegen verliert er am Anfang sehr viel Geld.

Für das Ergebnis ist es egal, ob man den Spieler gegen Loose Aggressive oder Tight Aggressive Spieler spielen lässt. Der grobe Ablauf bleibt der gleiche. Am Anfang verliert der Spieler, aber nach einer gewissen Anzahl an gespielten Händen gewinnt er sein zuvor verlorenes Kapital zurück. Nach diesem Zeitpunkt gewinnt der Spieler stetig mehr Geld. Das lässt sich dadurch erklären, dass die Lernwerte sich inzwischen gut entwickelt haben und der Spieler nur noch bei Spielen mitspielt bei denen er realistische Gewinnchancen hat. Somit ist klar, dass der Spieler die statischen Spieler schlagen kann.

Für genauere Informationen, wie Spieltabellen und Testergebnisse, sei der Leser auf den Artikel (Kendall & Willdig, 2001) verwiesen.

3. Evolutionärer Ansatz

Ein weiterer Ansatz einen sich entwickelnden Pokerspieler zu schreiben, ist die Verwendung von evolutionären Algorithmen. In diesem Abschnitt wird auf die Artikel (Barone & While, 1997) und (Barone & While, 1999) eingegangen.

2. Der genaue Algorithmus ist in (Kendall & Willdig, 2001) angegeben

3.1 Einleitung

Ein evolutionärer Algorithmus ist ein Optimierungsverfahren, das als Vorbild die biologische Evolution hat. Dabei durchlaufen die Individuen einer bestehende Population immer wieder folgende Aktionen, um effektivere Lösungsansätze zu finden:

Selektion Durch die Selektion wird die Richtung vorgegeben in der sich das Erbgut (die Repräsentation des Lösungsansatzes) weiterentwickelt.

Rekombination Die Rekombination oder das *Crossover* von Erbgut liegt hinsichtlich ihres Beitrags zur Zielfindung zwischen Mutation und Selektion. Um gute Ansätze noch zu verbessern wird beispielsweise das Erbgut zweier guter Individuen kombiniert, in der Hoffnung daraus eine noch bessere Lösung zu erhalten.

Mutation Die Aufgabe der Mutation ist es neue Varianten und Alternativen zu erzeugen, um dadurch lokale Maxima zu überwinden.

In den beiden behandelten Artikeln wird eine $(1 + 1)$ Evolutionsstrategie angewandt. Das bedeutet, dass zur Erzeugung von einem Kind ein Elternelement benutzt wird, also asexuelle Evolution stattfindet. Bei den Pareto-Koevolutionsansätzen (Abschnitt 4) hingegen kommt eine $(2 + 2)$ Evolutionsstrategie zum Einsatz. Hier werden also aus zwei Eltern zwei Nachkommen generiert.

Für eine weitere Beschreibung von evolutionären Algorithmen verweisen wir auf (Wikipedia, 2008a).

3.2 (Barone & While, 1997)

Barone und While unterteilen ihren Pokerspieler (Pokeragenten) in mehrere Komponenten. Jede dieser Komponenten kümmert sich um eine bestimmte Aufgabe. Der vorgeschlagene Agent benutzt die Komponenten Handstärke, Position und Risikomanagement. Diese werden durch den *Resolver* befragt, wenn eine Spielentscheidung ansteht. Die Komponenten unterrichten den *Resolver* über ihre Spieleinschätzung, also mit welcher Wahrscheinlichkeit sie mitgehen, erhöhen oder aus dem Spiel rausgehen wollen. Der Resolver gewichtet diese einzelnen Meinungen und trifft eine Entscheidung. Wie er diese Komponentenmeinungen am besten gewichtet, wird ebenso, wie die Einschätzungen der einzelnen Komponenten, bei einem Evolutionsschritt angepasst.

3.2.1 Population der Pokerspieler

Die Anzahl an Pokeragenten variiert abhängig vom Experiment. Jeder Pokeragent hat eigene Komponenten für Handstärke, Position und Risikomanagement sowie einen *Resolver*. Jede Simulation geht über 500 Generationen und jede Generation spielt 100 Hände. Bei dem sich entwickelnden Spieler findet nach jeder Generation eine Mutation statt.

3.2.2 Experimentergebnis

Bei den Experimenten zeigt sich, dass die sich entwickelnden Pokerspieler zu Beginn den statischen unterlegen sind. Das kommt daher, weil die sich entwickelnden Pokerspieler am

Anfang durch Zufallswerte zusammengesetzt sind. Nach einer gewissen Zeit werden sie aber besser und schlagen die statischen Spieler. Dadurch ist es ihnen möglich ihr zuvor verlorenes Spielgeld wieder zurück zu gewinnen und am Experimentende sogar einen Überschuss erspielt zu haben.

Ein sich entwickelnder Pokerspieler der gegen *loose aggressive* Spieler spielt entwickelt ein risikofreudigeres Spiel als ein Pokerspieler der gegen *tight aggressive* Spieler spielt.

Beide sich entwickelnden Pokerspieler gehen bei einem *Straight Flush* aus dem Spiel. Da dieses Blatt so selten ist, konnten sie nicht lernen, dass hier ein sehr gutes Blatt vorliegt.

Ein überraschendes Ergebnis tritt auf, wenn ein sich entwickelnder Spieler gegen *loose aggressive* Spieler spielt, wird er mit hoher Wahrscheinlichkeit aus dem Spiel herausgehen, wenn er in später Position sitzt.

Für die vollständigen Ergebnisse verweisen wir auf (Barone & While, 1997).

3.3 (Barone & While, 1999)

Im Gegensatz zu (Barone & While, 1999) wird hier die Population nicht über Pokeragenten erstellt, sondern durch Kandidaten. Jeder Kandidat besteht aus drei Funktionen. Eine für die Spielaktion Mitgehen, eine für aus dem Spiel gehen und eine für Erhöhen. Diese Funktionen erhalten als Argument einen zu der Gewinnwahrscheinlichkeit des aktuellen Spielstandes korrelierenden Wert. Zudem besteht jede dieser Funktionen aus weiteren Konstanten, die die Form der Funktion definieren.

3.3.1 Evolutionäre Struktur

Jeder sich entwickelnde Pokerspieler besteht aus einem Hypercube, der in zwei Dimensionen unterteilt wird. Eine Dimension steht für das Riskomanagement und eine andere für die Position. Die Positionskomponente ist noch einmal in drei Divisionen unterteilt (eine für frühe, mittlere und späte Position). Die Risikomanagementkomponente ist in vier Divisionen[3] unterteilt. Dies führt zu zwölf Hypercubeelementen. Jedes Hypercubeelement besteht aus N Kandidaten und jeder dieser Kandidaten aus sieben reellen Zahlen. Diese Zahlen stehen für die nötigen Konstanten, die die Form der oben genannten Funktionen definieren.

An jedem Punkt, an dem der Pokerspieler eine Spielentscheidung treffen muss, wählt er zuerst das entsprechende Element aus dem Hypercube durch Herausfinden der aktuellen Position und Anzahl der Wetten bei denen er mitgehen müsste. Danach nimmt er einen Kandidaten und wertet damit die oben genannten Funktionen aus, um den nächsten Spielzug zu berechnen.

Die N Kandidaten eines Hypercubeelementes werden der Reihe nach befragt.

Evolution findet dann statt, wenn alle Kandidaten oft genug befragt wurden. Dabei werden die besten $N/2$ Elemente der Population behalten und generieren $N/2$ Nachfolger, die der Population dann hinzugefügt werden. Nachfolger werden durch Mutation der sieben Zahlen eines Kandidaten generiert.

3. eine für keine Wette, eine für eine Wette, eine für zwei Wetten und eine für drei oder mehr Wetten, die mitzugehen sind

3.3.2 Experimentergebnis

Bei allen Experimenten werden wieder die üblichen vier Standardpokerspieler verwendet. Für die Experimente werden zwei Tische definiert. Am ersten Tisch (genannt *Loose Table*) sitzen neun *loose aggressive* Spieler und ein sich entwickelnder Spieler. Am zweiten Tisch (genannt *Tight Table*) sitzen neun *tight passive* Spieler und ein sich entwickelnder Spieler.

Die sich zu entwickelnten Pokerspieler setzen sich nach einer anfänglichen Verlustphase gegen die statischen Spieler durch.

Im nächsten Experiment werden trainierte Pokerspieler untersucht. Ein Spieler der am *Loose Table* gespielt hat (Spieler A^{loose}) zeigt wesentliche Unterschiede zu einem Spieler der am *Tight Table* (Spieler A^{tight}) gespielt hat. A^{tight} hat beispielsweise gelernt, dass wenn ein Gegenspieler erhöht, dass dies ein Anzeichen dafür ist, dass er vermutlich eine starke Hand hat und er selbst besser aus dem Spiel gehen sollte. A^{loose} hingegen hat gelernt, dass er bei Erhöhungen der Gegenspieler ruhig mitgehen kann, wenn er selbst eine starke Hand hat, da die Gegenspieler oft schwache Hände spielen.

In einem weiteren Experiment wird untersucht, ob die Pokerspieler A^{tight} und A^{loose}, an einem für ihren antrainierten Spielstiel besseren Tisch auch besser abschneiden. Diese naheliegende Vermutung wird durch das Experiment bestätigt.

Für die genauen Ergebnisse und Tabellen verweisen wir den Leser auf (Barone & While, 1999).

4. Pareto-Koevolution

Ein großes Problem von Spielstrategien, die von evolutionären Algorithmen dadurch gewonnen werden, dass die Individuen der Population ausschließlich gegen sich selbst antreten ist, dass diese häufig zu sehr auf einen kleinen Ausschnitt des Lösungsraumes spezialisiert sind. Eine wünschenswerte Strategie hingegen, sollte allgemein genug sein, um gegen viele verschiedene Spieler erfolgreich abzuschneiden.

Dieses Problem lässt sich unter anderem dadurch erklären, dass die Überlegenheitsrelation (Strategie A gewinnt gegen Strategie B) nicht transitiv ist. Dies lässt sich an einem kleinen Beispiel auf Grundlage des Spiels Stein-Schere-Papier leicht erklären: Nehmen wir an, Spieler A spielt immer Papier, Spieler B immer Stein und Spieler C immer Schere. Es gilt nun, dass Spieler A immer gegen Spieler B gewinnt, der wiederum immer gegen Spieler C gewinnt. Jedoch wird Spieler A immer gegen Spieler C verlieren, obwohl er den gegen C überlegenen Spieler B immer schlägt.

Die Existenz einer solchen, nicht transitiven Überlegenheitsrelation kann bedeuten, dass eine Suche zwar ständig neue Strategien findet, die besser als die vorherigen sind, jedoch keine Strategie findet die gegen viele verschiedenen Strategien im Allgemeinen gewinnt.

Diesem Problem soll mit Hilfe der *Pareto-Koevolution* entgegengewirkt werden.

4.1 Begriffserklärung

Bevor wir uns mit *Pareto-Koevolution* in evolutionären Algorithmen beschäftigen, wollen wir uns kurz den Begriffen *Pareto* und *Koevolution* widmen.

4.1.1 Koevolution

Koevolution, auch Coevolution, bezeichnet im Rahmen der biologischen Evolutionstheorie einen evolutionären Prozess der wechselseitigen Anpassung zweier stark interagierender Arten aufeinander, der sich über sehr lange Zeiträume in der Stammesgeschichte beider Arten erstreckt (Wikipedia, 2008c).

Übertragen auf evolutionäre Algorithmen in Spielen würde dies bedeuten, dass sich verschiedene Strategien gegenseitig beeinflussen und sich wechselseitig anpassen. Das ist ein Unterschied zu rein evolutionären Ansätzen, bei denen sich Lösungsansätze nur ihrer Umgebung anpassen.

4.1.2 Pareto-Optimum

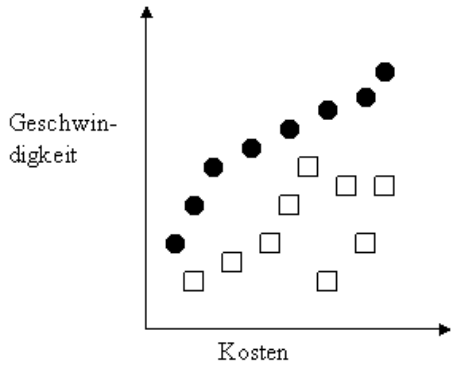

Figure 1: Lösungen des hypothetischen Herrstellungsproblems eines Autos. Die schwarzen Kreise beschreiben die Pareto-Menge.

Das Pareto-Optimum ist ein Begriff aus der VWL und beschreibt all die Lösungen eines Problems mit mehreren Freiheitsgraden, in denen keine relevante Größe (oder Dimension) weiter verbessert werden kann, ohne dass sich eine andere verschlechtert. Als Beispiel soll uns hier die Herstellung eines Autos dienen, wobei die für uns relevanten Dimensionen Kosten und Höchstgeschwindigkeit sind. Ein mögliches Pareto-Optimum könnte zum Beispiel bei 15.000 EUR und 160 km/h liegen. Dies würde bedeuten, dass man für 15.000 EUR kein Auto herstellen kann, dass schneller als 160 km/h fährt und dass man für ein Auto, dass 160 km/h fahren soll mindestens 15.000 EUR zahlen muss. Ein anderes Optimum kann zum Beispiel bei 20.000 EUR und 200 km/h liegen. All jene Lösungen (siehe Figure 1), dieser optimalen Kompromisse bezeichnet man auch als *Pareto-Menge* (im Englischen Pareto-optimal set oder Pareto-front). In den meisten Fällen wird diese Menge mehr als ein Element enthalten, so dass sie beispielsweise als Auswahlkriterium für evolutionäre Algorithmen benutzt werden könnte.

Ein weiterer wichtiger Begriff in diesem Zusammenhang ist der, der *Pareto-Dominanz* (oder der *Pareto-Superiorität*). Man sagt eine Lösung Pareto-dominiert eine andere Lösung, falls sie in allen relevanten Größen (Dimensionen) mindestens genauso gut und in mindestens

einer relevanten Größe besser ist als die andere Lösung. So wird beispielsweise dass Herstellungsverfahren, bei dem 160 km/h schnelle Autos für 20.000 EUR hergestellt werden von dem Design Pareto-dominiert, dass es erlaubt 160 km/h schnelle Autos für 15.000 EUR herzustellen.

4.2 Optimieren mit Hilfe des Pareto-Optimums

Im Gegensatz zu gewöhnlichen Herangehensweisen an Optimierungsprobleme können Pareto-Optimale Lösungen, nach dem Pareto Ansatz nicht mit Hilfe einer Kostenfunktion bewertet werden, so dass zwei Pareto-optimale Lösungen zunächst einmal nebeneinander existieren ohne dass eine Aussage getroffen werden kann, welche der beiden besser ist. Nehmen wir zum Beispiel an, dass 10 Äpfel auf 2 Personen aufgeteilt werden sollen, so sind die Lösungen (10,0) und (5,5) zwei mögliche Verteilungen. Beide Lösungen sind Pareto-optimal, da keine Dimension des Problems weiter verbessert werden kann (eine Person erhält weitere Äpfel), ohne dass eine andere Dimension verschlechtert wird (eine Person bekommt Äpfel abgenommen). Jedoch ist die zweite Lösung "gerechter" als die erste (Wikipedia, 2008d).

Der Pareto-Ansatz in Optimierungsproblemen ist daher nicht die Bestimmung der "besten" Lösung anhand einer Kostenfunktion, sondern die Bestimmung der Pareto-Menge, damit ein menschlicher Entscheidungsträger, oder eine andere Instanz aus dieser Menge geeignete auswählen kann.

4.3 Pareto-Koevolution in evolutionären Algorithmen

Wie bereits in der Einleitung erwähnt, kann die nicht transitive Überlegenheitsrelation von Spielen dazu führen, dass evolutionäre Algorithmen sehr spezielle, statt allgemein gute Strategien entwickeln. Die Idee ist nun, verschiedene Spieler als unterschiedliche Dimensionen (Individuen der Population) eines mehrdimensionalen Optimierungsproblems anzusehen. Darauf sollen nun Prinzipien, wie die Pareto-Dominanz angewandt werden, um allgemeingültige und robuste Strategien zu entwickeln. Ein Individuum ist unter diesem Ansatz umso besser, je mehr Spieler "es schlägt" (hierfür wird im folgenden noch versucht eine Definition zu finden).

Man beachte den sich von "normalen" evolutionären Algorithmen unterscheidenden Ansatz, die ein Individuum umso besser bewerten, je höher der Wert einer Fitness-Funktion (zum Beispiel die Anzahl der Chips beim Pokern) ist.

4.4 Pareto-Koevolution in Texas Hold'em

Im folgenden werden die Ergebnisse des Artikels (Noble & Watson, 2001), in dem an Hand einer einfachen Implementierung untersucht werden soll, ob Pareto-Koevolution evolutionäre Algorithmen verbessern kann, zusammengefasst. Im Anschluss daran werden wir den Folgeartikel (Noble, 2002) diskutieren, bei dem einige Ansätze untersucht werden, um Algorithmen mit Pareto-Koevolution weiter zu verbessern. Im Gegensatz zu allen bisher besprochenen Artikeln (über evolutionäre Algorithmen), wird in beiden Artikeln Texas Hold'em Poker nicht vereinfacht, in einer Limit 2$/4$ Version gespielt.

4.4.1 (Noble & Watson, 2001)

Noble und Watson wählen in ihrer Untersuchung eine einfache Datenstruktur, um verschiedene Pokerstrategien abzubilden. Sie betrachten lediglich die Handstärke sowie die aktuelle Wetthöhe, um sich für Fold, Call oder Raise zu entscheiden. Somit lassen sie viele Feinheiten des Pokerspiels, wie Position, ob die beiden Pre-Flop-Karten connected oder suited sind etc. außer Acht. Dieses zusätzliche Wissen ist aber für die Fragestellung des Artikels auch nicht weiter von Bedeutung, da es den Autoren um die Bewertung der Pareto-Selektion und nicht um die Erstellung eines guten Pokerspielers geht.

Die Strategien werden mit 2 Wahrscheinlichkeiten und 24 Integern (6 für jede Wettrunde) kodiert. Die beiden Wahrscheinlichkeitswerte geben an, mit welcher Wahrscheinlichkeit der Spieler blufft bzw. mit welcher Wahrscheinlichkeit er sich für ein Check-Raise anstatt eines normalen Raise entscheidet (falls er in die Situation kommt). Bei den Integern geben zwei die Mindeststärke einer Hand an, bei der ein Spieler im Spiel bleibt. Zwei weitere beschreiben die Karten, die ein Spieler als eine starke Hand ansehen würde. Die letzten beiden Integer beschreiben den Einsatz, den der Spieler zu dieser Zeit gerne setzen würde und den Wert den er bereit ist maximal mitzugehen. Zusätzlich wurden vier Binärwerte eingeführt, die das Spielverhalten der Spieler in jeder Wettrunde modifizieren können. Ein Bit gibt zum Beispiel an, ob ein Spieler, hält er starke Karten auf der Hand, seine eigentliche Strategie für diese Runde nicht beachtet und stattdessen den maximal möglichen Betrag setzt.

Für eine ausführliche Beschreibung der verwendeten Zahlen verweisen wir den Leser auf (Noble & Watson, 2001).

Ein einfacher Pareto-Koevolutions-Algorithmus Als Algorithmus wurde ein einfacher genetischer Algorithmus gewählt der Reproduktion mit Hilfe von multi-point Crossover und Mutation simuliert. Die Pareto-Dominanz wurde für die Selektion herangezogen.

Es wurde mit einer Population von 100 zufälligen Pokerstrategien angefangen, von denen je 10 zufällig ausgewählt wurden, um 50 Hände Poker zu spielen. 200 solcher Spiele wurden gespielt (so dass jede Strategie im Schnitt 1000 Hände spielen konnte), bevor sich die nächste Generation entwickelt. Die Ergebnisse der 10.000 Hände einer Generation werden für die Selektion in einer Matrix akkumuliert. Mit Hilfe dieser Matrix kann nun die Pareto-Menge, durch paarweise Vergleiche (welcher Spieler hat wie viele Chips von wem gewonnen) entwickelt werden, die dem Algorithmus als Selektionsgrundlage dient.

Ergebnisse im Vergleich mit regulären, koevolutionären, genetischen Algorithmen Als Vergleichsalgorithmus[4] wurde ein ähnlich gestalteter genetischer Algorithmus herangezogen, der für die Selektion jedoch nicht die Pareto-Menge, sondern die verbleibenden Chips der Spieler nutzt. Um die beiden Ansätze miteinander vergleichen zu können, wurden zwei Referenztische mit je 5 handgeschriebenenen Strategien[5] gebildet. Jedes Individuum aus den Endpopulationen (Pareto und regulär) musste nun für ein Spiel, mit 1000 Händen, an beiden Referenztischen antreten.

4. Die Individuen des Vergleichsalgorithmus durchlaufen die gleiche Trainingsphase, wie der Pareto-Algorithmus.
5. Die Referenzspieler wurden mit Hilfe der gleichen Datenstrukturen kodiert.

Das Ergebnis legt nahe, dass Strategien, die mit Hilfe von Pareto-Koevolution entwickelt werden den Strategien überlegen sind, die durch reguläre, genetische Algorithmen erzeugt werden. So schnitten die Pareto Spieler gegen beide Referenzgruppen im Schnitt deutlich besser ab, als die Vergleichsstrategien.

In der Untersuchung aufgetretene Probleme Neben der relativ eindeutigen Aussage, dass Pareto-Strategien im Schnitt besser abschneiden, als ihre rein genetischen Verwandten, wurden auch einige Probleme offensichtlich.

Zum einen zeigt sich, dass die entwickelten Strategien nicht als besonders stark angesehen werden können. Sowohl die Individuen der Pareto-Population, wie auch die Individuen der Vergleichspopulation schafften es nicht an einem der Referenztische Gewinn zu erspielen[6]. Zum anderen scheinen die Pareto-Strategien nicht an ihrem Kollektiv-Wissen festhalten zu können. Dies zeigt sich durch eine sehr kurze Verweildauer von Strategien in der Pareto-Menge (im Schnitt zwei Generationen) und keinem stetigen Wachstum der Spielstärke.

Für eine ausführliche Diskussion der entwickelsten Strategien und der aufgetretenen Probleme sei auf den Artikel (Noble & Watson, 2001) verwiesen.

4.4.2 (Noble, 2002)

In seinem Folgeartikel (Noble, 2002) versucht Jason Nobel die in (Noble & Watson, 2001) aufgetretenen Probleme (siehe 4.4.1) anzugehen, um robuste Texas Hold'em Poker Strategien zu entwickeln. Hierbei wurden die wesentlichen Strategien beibehalten, jedoch kommt eine weitaus feinere Darstellung der Pokerstrategien und eine Technik zur Erhaltung der Vielfalt (*Deterministic Crowding*) zum Einsatz.

Als Darstellungsform der Strategien wurde diesmal auf neuronale Netze[7] zurückgegriffen. Eine Strategie besteht hierbei aus zwei Netzen: Eines für das Pre-Flop-Spiel, bestehend aus 69 Eingängen und 5 versteckten Neuronen und eines für das Spiel auf dem Flop, Turn und River, mit 109 Eingängen und ebenfalls 5 versteckten Neuronen. Beide Netze verfügen über drei Ausgangsneuronen (für Fold, Call und Raise), die über die Strategie des Spielers entscheiden. Um die Komplexität jedoch etwas zu reduzieren wurden die Netze nicht vollständig verbunden[8]. Zu jeder Zeit existieren jeweils nur 50 Verbindungen. Für Informationen zu Mutations- und Rekombinationsstrategien sei der Leser auf (Noble, 2002) verwiesen.

Die Selektion wurde ähnlich, wie in (Noble & Watson, 2001) implementiert (siehe 4.4.1), jedoch wurde die Populationsgröße auf 20 Individuen beschränkt. Da Poker einen nicht zu vernachlässigenden stochastischen Anteil besitzt, wurde eine 100$ Fehlergrenze eingeführt. Das bedeutet, dass Spieler A Spieler B dann dominiert, falls er nicht mehr als 100$ schlechter als Spieler B gegen alle anderen Spieler spielt und besser als 100$ gegen mindestens einen Spieler als Spieler B.

6. Jedoch war der gemittelte Verlust der Pareto-Population deutlich geringer.
7. Für weitere Informationen siehe (Wikipedia, 2008b)
8. Währen sie vollständig verbunden, müssten für das Pre-Flop-Netzwerk 567 Gewichte und für das Post-Flop-Netz 887 Gewichte gepflegt werden.

Deterministic Crowding ist ein einfacher Mechanismus, der versucht die Vielfalt in den Lösungen zu erhalten. Hierbei werden zwei Eltern zufällig ausgewählt, um zwei Nachkommen, mit Hilfe von Mutation und Crossoverfunktionen zu erzeugen. In der folgenden Epoche treten die Nachkommen jeweils gegen den Elter an, dem sie am meisten ähneln. Um den Elter zu ersetzen, müssen die Nachkommen ihren zugewiesenen Elter Pareto-dominieren. Somit sind zu jeder Zeit, mit Ausnahme der ersten Epoche, 10 Individuen in der Pareto-Menge und 10 Nachkommen kämpfen darum, ihren zugewiesenen Elter zu ersetzen.

Als Ergebnis lässt sich festhalten, dass Pareto-Koevolution den einfachen genetischen Algorithmen deutlich überlegen ist. Es wurden verschiedene Experimente mit unterschiedlichen Ausgangslagen[9] durchgeführt. Als Vergleichsgruppe wurde wieder ein ähnlicher einfacher Koevolutionsalgorithmus ohne Pareto-Selektion gewählt. Die Individuen der Endpopulationen traten nun wieder gegen die Referenzspieler aus (Noble & Watson, 2001) (siehe 4.4.1)[10] an. In allen Experimenten waren die Pareto-Spielern ihren einfachen evolutionären Kollegen überlegen. Auch die Einführung von *Deterministic Crowding* (siehe 4.4.2) scheint sich rentiert zu haben. Die Pareto-Population akkumuliert stetig ihr Wissen über neue Generationen hinweg und die durschnittliche Aufenthaltszeit eines Individuums in der Pareto-Menge wird mit zunehmender Generationenzahl immer länger, was sich mit den immer stärker spielenden Individuen gut erklären lässt.

Probleme Auch wenn die Ergebnisse zunächst sehr vielversprechend klingen, so gibt es dennoch einige Beeinträchtigungen. So wurden zwar im Grunde die Ziele erreicht, jedoch erspielen nur die Spieler aus Experiment 3 und 4 (die Startpopulation fing nicht mit zufälligen Spielern sondern mit handkodierten neuronalen Netzen an) am Referenztisch einen Gewinn. Andererseits scheinen die stärksten Individuen aus Experiment 4 sehr gute und robuste Strategien entwickelt zu haben[11].

Für eine ausführlichere Diskussion der Experimente und der aufgetretenen Probleme sei auf den Artikel (Noble, 2002) hingewiesen.

5. Fazit

Als Ergebnis dieser Seminararbeit lässt sich zusammenfassen, dass alle drei vorgestellten Ansätze rein statischen Spielern nach einer Trainingsphase überlegen sind. Dabei zeigt sich, dass evolutionäre Ansätze rein Lernbasierten überlegen sind, jedoch die Pareto-Koevolution der vielversprechenste Ansatz für die Implementierung eines Pokeragenten ist. Besonders der Artikel (Noble, 2002) liefert wichtige Erkenntnisse über eine mögliche Implementierung.

Trotz aller guten Ansätze liegen keinerlei Erkenntnisse über evolutionäre Algorithmen für Texas Hold'em No Limit (sowie Turnierspiele) vor. Die Ansätze können daher nicht direkt auf das Praktikum übertragen werden. Wir erwarten jedoch, dass die oben genannten

9. Experiment 1: Untrainierte Spieler spielen gegeneinander.
   Experiment 2: Es sitzen handgeschriebene Referenzspieler mit am Tisch.
   Experiment 3: Die neuronalen Netze wurden "sinnvoll" vorbelegt.
   Experiment 4: Kombination von 2 und 3.
10. Die Spieler traten nur an dem stärkeren der beiden Referenztische an.
11. Der Autor des Artikels muss zugeben gegen den besten Spieler aus Experiment 4 im Schnitt zu verlieren.

Artikel auch für das Praktikum und die Implementierung eines Pokeragenten, für Texas Hold'em No Limit, eine gute Ausgangsbasis bilden.

References

Barone, L., & While, L. (1997). Evolving computer opponents to play a game of simplified poker. In Royle, G., & Backman, R. B. (Eds.), *8th University of Western Australia Computer Science Research Conf.*

Barone, L., & While, L. (1999). An adaptive learning model for simplified poker using evolutionary algorithms. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., & Zalzala, A. (Eds.), *Proceedings of the Congress on Evolutionary Computation*, Vol. 1, pp. 153–160 Mayflower Hotel, Washington D.C., USA. IEEE Press.

Kendall, G., & Willdig, M. (2001). An investigation of an adaptive poker player. In *Australian Joint Conference on Artificial Intelligence*, pp. 189–200.

Noble, J. (2002). Finding robust texas hold'em poker strategies using pareto coevolution and deterministic crowding..

Noble, J., & Watson, R. A. (2001). Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., & Burke, E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 493–500 San Francisco, California, USA. Morgan Kaufmann.

Wikipedia (2008a). Evolutionärer algorithmus — wikipedia, die freie enzyklopädie.. [Online; Stand 27. März 2008].

Wikipedia (2008b). Künstliches neuronales netz — wikipedia, die freie enzyklopädie.. [Online; Stand 26. März 2008].

Wikipedia (2008c). Koevolution — wikipedia, die freie enzyklopädie.. [Online; Stand 25. März 2008].

Wikipedia (2008d). Pareto-optimum — wikipedia, die freie enzyklopädie.. [Online; Stand 25. März 2008].