JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN
FB15 Biologie und Informatik
Institut für Informatik

Proseminar "Komplexe Adaptive Systeme" WS2004/05

Advanced Evolutionary Design of Genreralized Recurrent Neural Networks

Evolutionärer Entwurf neuronaler Netze

Roozbeh Hassan Vand

Roozbeh@rz.uni-frankfurt.de

Inhalt

Einleitung
- Die Eigenschaften der evolutionären Algorithmen
- Die Topologie neuronaler Netze

Neuronale Netze
- Biologische Neuronen
- Künstliche Neuronen

Evolutionäre Algorithmen
- Evolutionäre Algorithmen als Such- und Optimierungsverfahren
- Vorgehensweise evolutionärer Algorithmen

Evolutionärer Ansatz zu Konstruktion neuronaler Netze
- Populationen und Individuen
- Evolutionäre (Genetische) Operatoren

Advanced Evolutionary Algorithmus (AEA)
-Verzweigung (Forking)
- Automatischer Austauchen der Wahrscheinlichkeitsverteilung der Mutation
- Lernen
- Verarbeitungsgeschwindigkeit der neuronalen Netze

Experimente
- Tomita Automaten und TXOR-Funktionen
- Endliche Automatenidentifikation
- Ameisen-Probleme

Ergebnisse
- Bessere Generalisierungsfähigkeit der AEA-Lösungen
- Höhere Verarbeitungsgeschwindigkeit des AEA

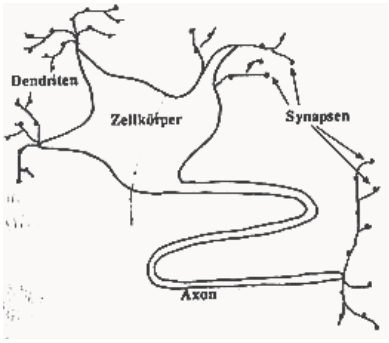
Literatur

Einleitung

Die Leistungsfähigkeit vom heutigen Rechner bezieht sich meist auf einem oder nur wenigen zentralen Prozessoren, die mit hoher Geschwindigkeit lange, sequentielle Programme abarbeiten.
Zur Programmierung solcher Rechner sind neuartige Algorithmen erforderlich, die eine Verteilung ihrer Rechenschritte auf eine Vielzahl von Prozessoren gestatten. Um die Integration solcher Algorithmen zu komplexen Softwaresystemen zu handhaben, müssen die Algorithmen fehlertolerant und lernfähig sein. Die Realisierung dieses Lösungsweges ist in biologischen Gehirnen mit zahlreichen Nerven-Zellen vorgezeichnet. Die oben genannten Algorithmen werden als „evolutionäre Algorithmen" bezeichnet, die sich an unterschiedlichen Vorbildern der natürlichen Evolution orientieren. Am häufigsten findet man den Ansatz evolutionärer Algorithmen für die Konstruktion der neuronalen Netze und Einstellung der Gewichte zu benutzen.
Es wurde ein neuer evolutionärer Algorithmus für neuronale Netze, die sich ständig verändern, entwickelt. Der Algorithmus hat viele fortgeschrittene Attribute, die optimale neuronale Netztopologie und Gewichte für das zu lösende Problem finden.
Bevor der allgemeine Ansatz des Entwurfs neuronaler Netze mit Anwendung evolutionärer Algorithmen und die in dem neuen evolutionären Algorithmus eingebauten Attributen beschrieben wird, werde ich zuerst neuronale Netze und evolutionäre Algorithmen kurz einführen. Zum Schluss werden die Ergebnisse dargelegt.

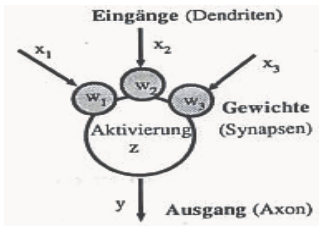
1.Neuronale Netze

Der wesentliche Kern der Informationsverarbeitung im biologischen Gehirn geschieht in der Gehirnrinde (Neokortex). Die elementaren Verarbeitungseinheiten in der Gehirnrinde sind die Neuronen. Sie sind Zellen, die untereinander auf elektrochemische Wege Signale austauschen und sich gegenseitig erregen können. Neuronen besitzen wie ein Prozess Eingabe und Ausgabe. Wenn die Summe der Eingabesignale als elektrisches Potential einen Schwellwert überschreitet, wird das Neuron aktiv und sendet ein Signal an andere benachbarten Neuronen. Die Gewichte(Synapsen) sind die Verbindungen im Netz, die die Eingabe bzw. den ankommenden Potentialwert abschwächen oder verstärken können, deshalb spielen sie im Lernverfahren eine wichtige Rolle.



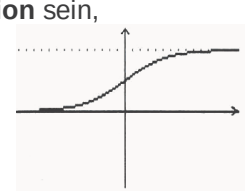
Ein künstliches, neuronales Netz besteht aus künstlichen Neuronen, die auf das mathematische Modell der natürlichen Neuronen basiert sind. Ein künstliches Neuron ist ein nicht lineares Element mit einige gewichtete Verbindungen, eine Ausgabeverbindung und eine Übertragungsfunktion f.
Wenn $y_i(u)$ der Ausgabewert des i-ten Neurons zum Zeitpunkt u, $v_{ij}(u)$ die Eingabewerte des Neurons und $w_{ij}(u)$ die Gewichtwerte entsprechen, wird der nächste Ausgabewert des Neurons so berechnet :

$$y_i(u+1) = f\left(\sum_{j=1}^{N_i} w_{ij} \cdot v_{ij}(u)\right),$$



,wobei N_i die Anzahl von Eingabeeinheiten bis zum i-ten Neuron entspricht. Die Verbindungen (Synapsen) eines Neurons nehmen Aktivierungen v_{ij} mit bestimmten stärken w_{ij} von anderen

Neuronen auf, summieren diese und lassen dann am Ausgang y_i des Neurons eine Aktivität entstehen, sofern die Summe vorher einen Schwellwert überschritten hat.
Der Schwellwert des i-ten Neurons wird durch das **Gewicht** w_{iNi} berechnet und der Eingabewert dieses Gewichts wird auf eins gesetzt oder wird mit eins initialisiert.
Die Übertragungsfunktion kann eine **Sigmoidfunktion** sein,



$$f(x) = 1 / (1 + e^{-x})$$

wobei f(x) die Stärke des auszusenden Signals ist.
Die neuronale Netze können in mehrere Schichten eingeteilt sein. Eine neuronale Netzschicht besteht aus Neuronen, die Informationen parallel oder gleichzeitig verarbeiten. Bei vorwärts gerichtete neuronale Netze werden die Signale (Informationen) von Eingabeschicht zur Ausgabeschicht verarbeitet, bei rekursive neuronale Netze aber in die beiden Richtungen. Neuronale Netze werden nach folgenden Kriterien unterteilt:

- Die Richtung der Signalpropagation
- Die Verbindung oder Relation zwischen Neuronen

2. Evolutionäre Algorithmen (EA)

EA orientieren sich an unterschiedlichen Vorbildern der **natürlichen Evolution (NE)**.
Bei der NE lassen sich die Evolution von lebenden und unbelebten Systemen unterscheiden. Für die EA ist das Vorbild der Evolution von lebenden Organismen. Unter NE wird der Prozess verstanden, welcher zu bestehenden Vervielfältigung der Organismenwelt –der Einzeller, Pflanzen, Tiere - geführt hat. Diese Vervielfältigung wird durch die Anpassung unterschiedlicher Arten(von lebenden systeme) an unterschiedliche Umweltbedingungen gewährleistet .EA verallgemeinern die grundlegenden evolutionstheoretischen Prinzipien der Vervielfachung, Veränderung und Auswahl. Sie bilden anwendbaren Such- und Optimierungsverfahren und werden in der simultanen Suche nach neuronalen Netztopologien und **Gewichte** eingesetzt. EA sind durch biologisch inspirierte Fachbegriffe gekennzeichnet. Die wichtigsten aus der Biologie übernommenen Begriffe werden im folgenden definiert:

Population (von Individuen): Menge von Strukturen (Lösungsalternativen)
Individuum: Struktur (enthält die Elemente einer Lösung)
Fitneß: Lösungsqualität hinsichtlich der relevanten Zielkriterien (z.B. Lernfunktion)
Eltern: Die zur Reproduktion ausgewählten Individuen
Nachkommen: Die aus den Eltern erzeugten Lösungen
Generation: Verfahrensiteration
Gen: Die kleinste Einheit der genetischen Information, das sich auf Chromosom befindet
Chromosom (Erbinformation) : besteht aus Genen und ist identisch mit Individuen, gelegentlich Kann ein Individuum sich aus mehreren Chromosomen zusammensetzen.
Genotyp: Erbinformation von Lebewesen(codierte Lösung) in den Chromosomen
Phänotyp: Erscheinungsbild von Lebewesen(decodiert Lösung) in den Chromosomen

Die evolutionäre (genetische) Operatoren, die man hier braucht ,werden im folgenden definiert:

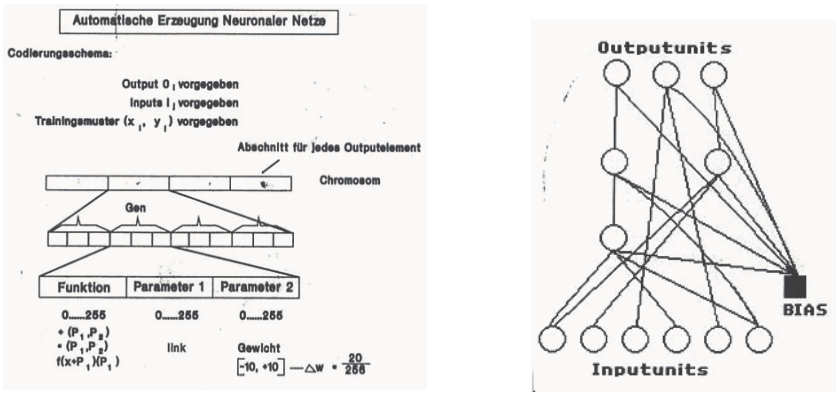
Selektion: Die Selektion ist im wesentlichen eine Verschiebung der Häufigkeit, mit der die einzelnen Individuen in der Population auftreten (bei EA hat die Aufgabe, die Suche eine bestimmte Richtung zu geben)
Mutation: Mutation verändert ein gegebenes Individuum
Cross-Over: Cross-Over überkreuzt zwei Chromosomen(Tupel),so dass sie an einer Stelle auseinandergeschnitten und überkreuzt zusammengesetzt werden

EA zeichnen sich durch eine Vielzahl an einstellbaren Parametern aus. Hierzu zählt die Wahl einer geeigneten Darstellung für das Problem, die richtige Populationsgröße, usw. Die Parameter erlauben eine hohe Anpassbarkeit des Algorithmus an das vorliegende Problem, so werden die Parameter, für die, die optimale Werte für ein vorgegebenes Problem gesucht werden, zu einem Tupel (Zahlentupel) zusammengesetzt, welches als Bauplan (Chromosom) eines Lebewesens präsentiert wird(z.B. das Tupel $t = (t_1,...,t_m)$). Im diesem Tupel ist die Stelle t_i als Gen und t selbst als Genotyp(Chromosom) bezeichnet. Die Bewertungsfunktion f(t) bewertet das Tupel t (f(t) bewertet die Qualität der Lösung und die Größe bzw. die Länge der Lösung “die Länge der mathematischen Formel“), um festzustellen, wie gut ein Chromosom das Optimum annähert.

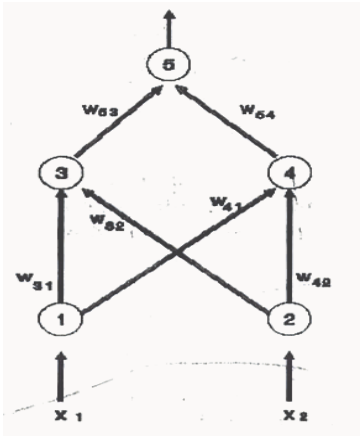
3. Evolutionärer Ansatz zu Konstruktion neuronaler Netze

Die Population (neuronale Netze, die durch EA in mathematische Formel codiert werden oder umgekehrt) besteht aus Individuen(auch ein neuronales Netz, das durch die Chromosomen, die als Zahlentupel dargestellt werden, präsentiert wird). Jedes **Individuum** ist eine Realisierung seines Chromosoms. In jeder Generation werden alle Individuen in der Population ausgewertet und die Erboperatoren(z.B. Mutation, Crossover) werden angewendet, um eine neue Population zusammenzustellen. Die Prozedur wird wiederholt bis eine Abbruchbedingung erreicht ist(bis die gewünschte Population "die optimale Lösung" erzeugt wird oder bis die Zeit abgelaufen ist).Der Suchraum ist Raum, wo das konkrete Problem gesucht wird und in Abhängigkeit von Anzahl der Parameter(Gene) kann n-Dimensional sein. Ein Chromosom ist ein Element, das einem Lösungsraum gehört, wobei in Lösungssubräume können neuronale Netze mit einer Lösung entdeckt werden. Die Bewertungsfunktion überprüft jedes Individuum in der Population und berechnet seine Leistung nach vorgegebenen Kriterien.
Lösungen werden in drei Gruppen unterteilt, nämlich:
1) Optimale Lösungen 2) Sub-Optimale Lösungen 3) Nicht-Optimale Lösungen
optimale Lösungen sind diejenigen, die völlig die vorgegebenen Kriterien(die gewünschte Ausgabe) erfüllen, sub-optimale Lösungen erfüllen die Kriterien nur Teilweise und nicht optimale Lösungen erfüllen die meisten Kriterien nicht.

Die folgende Abbildung zeigt der internen Codierung der mathematischen Formeln und bestes Ausgangsnetzwerk:



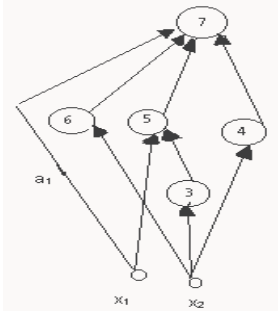
Beispiel 1: Ein 2-2-1 Netz wird (2 Input-Neuronen,2 Zwischenschicht-Neuronen,1 Output-Neuron) in eine mathematische Formel übersetzt.



$$y(x) = y(x_1,x_2) = f\,(w_{54}\,.f\,(w_{41}\,.x_1 + w_{42}\,.x_2) + w_{53}\,.f\,(w_{31}\,.x_1 + w_{32}\,.x_2))$$

Beispiel 2: Eine Komponente einer akzeptablen Formel wird in eine Netztopologie übersetzt.

$$A_i = f_7(a_1 .x_1 + a_6 .(f_4(a_2 .x_1)) + a_7 .(f_5(a_3 .x_1 + f_3(a_4 .x_2))) + a_8 .(f_6(a_5 . x_2)))$$



4.Advanced Evolutionary Algorithmus (AEA)

AEA ist eine Erweiterung des EA und unterscheidet sich von anderen evolutionären Algorithmen in den folgenden eingebauten Attributen:

- Verzweigung(Forking)
- Automatischer Austauchen der Wahrscheinlichkeitsverteilung der Mutation
- Lernen
- Automatische Feststellung/Ermittlung der optimalen Verarbeitungsgeschwindigkeit der neuronale Netze

Die Vorteile unser neuer evolutionärer Algorithmus (AEA) wurden auf drei Gruppen von Exprimenten getestet, nämlich:

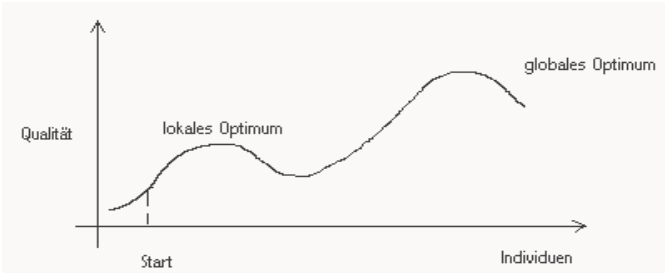
1) Identifikation modifizierter Tomita Automaten
2) Identifikation endlicher Automaten mit TXOR-Funktionen
3) Robotersteuerungsproblem

4.1. Verzweigung

Die Verzweigung ermöglicht eine Unterteilung des Suchraums in mehrere Subräume. Die Subräume werden dann von unabhängigen evolutionären Prozessen untersucht, um ein neuronales Netz mit passenden Topologie und Gewichte zu finden.
Jeder Lösungssubraum besteht aus neuronalen Netzen, die willkürliche Gewichtswerte und ein gleiche Anzahl von Neuronen haben. Die Lösungssubräume werden systematisch untersucht. Man beginnt mit dem Subraum, der die kleinsten neuronalen Netze besitzt bis zum Subraum mit den größten neuronalen Netzen.
Eine besondere evolutionäre Strategie bestimmt, wann ein neuer Lösungssubraum der alten ersetzt. Die Suche hört auf, sobald ein neuronales Netz in einer der Lösungssubräume gefunden wird, das

1) das Problem löst und
2) der kleinste Neuronenanzahl hat

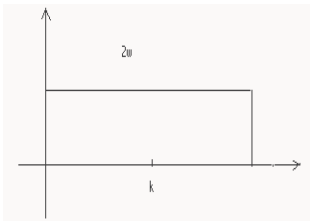
Ein evolutionärer Algorithmus mit Verzweigung kann zwei oder mehr evolutionäre Prozesse(**EP**) ausführen. Erstens wird ein grober EP über einen globalen Lösungsraum gestartet. Der globale Lösungsraum hat eine Population bestehend aus Individuen. Sobald der grobe EP einen lokalen Lösungsraum findet, der eine Optimale oder Sub-Optimale Lösung besitzt, wird dieser Lösungsraum aus dem globalen Lösungsraum entfernt. Gleichzeitig werden alle Individuen aus der Population, die zum lokalen Lösungsraum gehören, auf die lokale Population übertragen. Der grobe EP setzt sich mit dem kleineren globalen Lösungsraum fort. Ein neuer unabhängiger feiner EP wird auf den lokalen Lösungsraum angewendet. Dieser feine evolutionäre Prozess wird unabhängig ausgeführt bis 1) der Prozess eine Optimale Lösung im Lösungsraum findet oder 2) oder grobe EP ihn terminiert.



4.2. Das Anpassen der Parametern der Wahrscheinlichkeitsverteilungen (WV) der Mutation

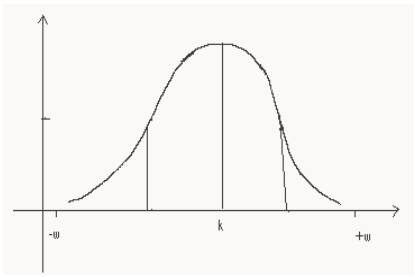
AEA hat eine eingebaute evolutionäre Strategie, um die Wahrscheinlichkeitsverteilungen **(WV**) der Mutation auszutauschen.
Verschiedene Studien zeigen, dass eine ausgewählte WV der Mutation die Konvergenz des evolutionäre Prozesses in Richtung Optimale Lösung erheblich beeinflussen kann.
Wir haben eine stetige (Kontinuierliche) Gleichverteilung und eine Normalverteilung(nach Gaußsche Formel) aus einem Intervall [-w+k , w+k] der Länge 2w und die Mittelwert **k** :

$$E(x,\kappa,\omega)=\begin{cases}\frac{1}{2\cdot\omega} & ; \quad -\omega+\kappa \le x < \omega+\kappa \\ 0 & ; \quad other\end{cases}$$



und

$$N(x,\kappa,\omega,\sigma)=\begin{cases}e^{-(x-\kappa)^2/(2\sigma^2)}\cdot\left[\int_{-\omega+\kappa}^{\omega+\kappa} e^{-(x-\kappa)^2/2\sigma^2}dx\right]^{-1} & ; \quad -\omega+\kappa \le x < \omega+\kappa \\ 0 & ; \quad other\end{cases}$$



4.3 Lernen und Evolution

Lernen kann die Evolution beschleunigen. Ein evolutionärer Algorithmus wird angewendet, um eine Lösung, die die optimale Lösung annähert, zu finden. Danach wird eine **Linear-basierte Lernmethode** eingesetzt, um die optimale Lösung zu finden. Die linear-basierte Lernalgorithmen können exakten oder genauen Gewichtswert finden. **AEA** hat eine eingebaute Lernprozedur “Real Time Recurrent Learning (RTRL) “.

4.4. Verarbeitungsgeschwindigkeit

Die Verarbeitungsgeschwindigkeit eines neuronalen Netzes bestimmt, wie oft ein neuronales Netz die gleiche Eingabe verarbeitet**,** bevor es mit der nächsten Eingabe fortsetzen kann. Ein einschichtiges neuronales Netz mit Verarbeitungsgeschwindigkeit von eins kann einige (sequentielle) Logikfunktionen nicht simulieren. Mehrsichtige neuronale Netze werden hier verwendet, um solche Probleme zu lösen.
AEA sucht nach einer Lösung innerhalb eines einschichtigen Netzes und die Verzweigung(Forking) wird angewendet, um die optimale Verarbeitungsgeschwindigkeit zu bestimmen.

5. Konvergenz der Attributen des AEA

Es wurde bewiesen, dass ein evolutionärer Algorithmus immer eine optimale Lösung in einer endlichen Anzahl von Schritten findet, wenn er die folgende vier Bedingungen erfüllt:

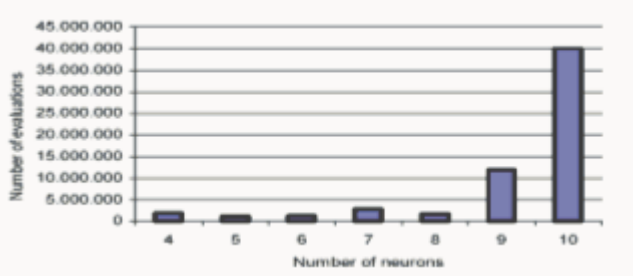
1) Jedes Individuum in der Population hat die Möglichkeit als Eltern einer neuen Population gewählt zu werden.
2) Es muß möglich sein, jede Lösung im Lösungsraum in jede andere Lösung im Lösungsraum durch ein endliche Anzahl von Mutationen zu verwandeln.
3) Die Wahrscheinlichkeit, dass ein Individuum aus der aktiven Population für die neue Population gewählt wird, muß höher als Null sein.

Das beste Individuum in der aktiven Population wird immer in der neuen Population aufgenommen.
Die 1. , 2. und 3. Bedingungen können durch eine modifizierte 2. Bedingung ersetzt werden:

2)* Der Mutationsoperator muss die Fähigkeit besitzen, jede Lösung im Lösungsraum in jede andere Lösung im Lösungsraum in genau einer Mutation zu verwandeln .Diese Bedingungen gelten für evolutionäre Algorithmen, die in endlichen Lösungsräume suchen und diskrete Zeitschritte haben.

Sobald es mindestens eine WV gibt, die die zweite modifizierte Bedingung erfüllt, ist die globale Konvergenz des AEA sicher.
Die neuronale Netze mit mehr Neuronen haben mehr freie Parametern als die Netze mit weniger Neuronen. Nicht- bestehenden Verbindungen werden wie Verbindungen mit einem Gewichtswert von 0 behandelt. Der Lösungsraum neuronaler Netze mit einer großen Anzahl von Neuronen ist größer als die Lösungsraum mit neuronaler Netze mit weniger Neurone, deswegen dauert die Evolution neuronaler Netze mit mehr Neuronen länger. AEA findet immer ein neuronales Netz mit der optimalen Neuronenanzahl oder mit einer Anzahl, die die optimale Anzahl durch die Vorwärtssuche nähert.
Wenn ein neuronales Netz mit einer **Sub-optimalen** Neuronenanzahl gefunden wird, sucht AEA rückwärts weiter. Er sucht nach einer Lösung mit der niedrigeren Neuronenanzahl, deshalb wird eine **optimale** Lösung gefunden. Dieses Phänomen wurde getestet und bewiesen.
Bild 1 verdeutlicht es durch ein Problem der Automatenidentifikation.

Bild 1:Vergleicht die Anzahl der Auswertungen, die man braucht, um optimale Gewichtswerte für neuronale Netze mit unterschiedlicher Neuronenanzahl, die Tomita-Automaten identifizieren, zu finden.



 # Auswertungen = #Generationen x (Große der Population).

AEA ist viel komplexer als andere evolutionäre Algorithmen, die gleichzeitig nach der optimalen Topologie und optimale Gewichtswerte neuronale Netze suchen. Trotzdem für einfache Probleme ist er im Durchschnitt fast **genauso schnell** wie einfache evolutionäre Algorithmen. Er hat auch die Fähigkeit viel komplexere Probleme zu lösen. AEA ist auch bei der Lösung **komplexe Probleme schneller, weil er Verzweigung und Lernen** verwendet, er tauscht der WV der Mutation aus und stellt automatisch die passende neuronale **Netzverarbeitungsgeschwindigkeit** fest.

6. Experimente

AEA wurde mit zwei Arten von Problemen getestet:

1)endliche Automatenidentifikation
2)Robotersteuerung
AEA wurde aus zwei Sichten getestet. **Erstens** wurden die statistisch ausgewerteten Ergebnisse mit den Ergebnissen anderen Algorithmen, die die gleiche Probleme lösen, verglichen. Alle statistischen Auswertungen sind auf 10 unabhängigen Ausführen des evolutionären Algorithmus basiert. Die Auswertungen haben zwei Messwerte: die durchschnittliche **neuronale Netzgröße** und die durchschnittliche **Konvergenzgeschwindigkeit.** Die von AEA gefundenen neuronalen Netze für ein gegebenes Problem hatten immer die gleiche Große, bei jeder evolutionäre Ausführung, da AEA immer

eine optimale Lösung finden konnte .**Zweitens** wurden die partielle Beschleunigungen der erweiterten Attributen des AEA gemessen. AEA hatte Verzweigung(Forking) in allen Versuchen angewendet.

Verzweigung (Forking) gibt AEA die Fähigkeit ,optimale neuronale Netzgrosse und optimale Verarbeitungsgeschwindigkeit zu bestimmen.

Neuronenanzahl wurde immer vor jeder evolutionären Suche mit 2 initialisiert oder auf 2 gesetzt. Die Anzahl der simultan durchsuchten Lösungssubräume wurde auf 10 gesetzt. Die Anzahl möglicher Verarbeitungsgeschwindigkeit wurde auf 1 und 2 begrenzt, um den Dauer der Evolution zu verkürzen. Die Große der Population in allen Versuchen war 100 Individuen. Alle neuronale Netze, die bei den Automatenidentifikationsprobleme benutzt wurden, hatten nur eine Eingabe(Eingabewerte 0 und 1) und eine Ausgabe. Die Ausgabewerte waren aus dem Reellenzahlintervall zwischen 0 und 1, die in diskreten Werten von 0 und 1 konvertiert wurden (die niedriger als 0.5 waren, wurden zum 0 diskretisiert und die übriggebliebene wurden zum 1 konvertiert).

6.1. endliche Automatenidentifikation

Die Versuche sind mit 4 Tomitaautomaten(4´,5,6,7´)und 4 zeitliche XOR-Funktionen (mit d=0,1,2,3)durchgeführt .Der gewünschte Wert zur Zeit u ist die XOR-Funktion der Eingaben zu Zeiten u-d und u-d-1. Es gibt insgesamt 7 Tomitaautomaten, die als Akzeptoren für Automatensprache angewendet werden. Fünf davon haben keine enge Verbindungen, deshalb werden einige interne Zustände nicht erreichbar aus einem willkürlichen internen Zustand. Die Automaten 4 und 7 wurden in 4´ und 7´ geändert, um Onlineidentifkation zu erlauben**. AEA wurde auch benutzt, um zeitliche XOR-Funktionen zu identifizieren**. Die Gewichtswerte der neuronalen Netze wurde aus dem Reellzahlintervall [-200,200] gewählt. Die Auswertungsfunktion basiert sich auf die Fehlerfunktion (Error- Funktion):

$$Err(t) = | o_d(t) - o(t) |$$

wobei $O_d(t)$ der gewünschte Ausgabewert und O(t) die diskretisiert neuronale Netzausgabe ist. Die Fehlern aus der Testfolge von Eingabe- und Ausgabewertpaare wurden addiert. **Das beste neuronale Netz in der Population hatte der niedrigste Gesamtfehler**. Die Testfolge hatte 125,000 Proben, erste 1000 Proben waren **in der Evolution verwendet** und die übriggebliebene 124,000 waren für das Testen. Eine entwickelte neuronale Netz wurde als eine Suboptimale Lösung betrachtet, wenn seine Gesamtfehler nach Testen der Testfolge null war. Eine zusätzliche Bedingung, damit ein neuronales Netz als optimal betrachtet werden kann, ist die Neuronenanzahl .Ein optimale neuronales Netz hatte die kleinstmögliche Neuronenanzahl.

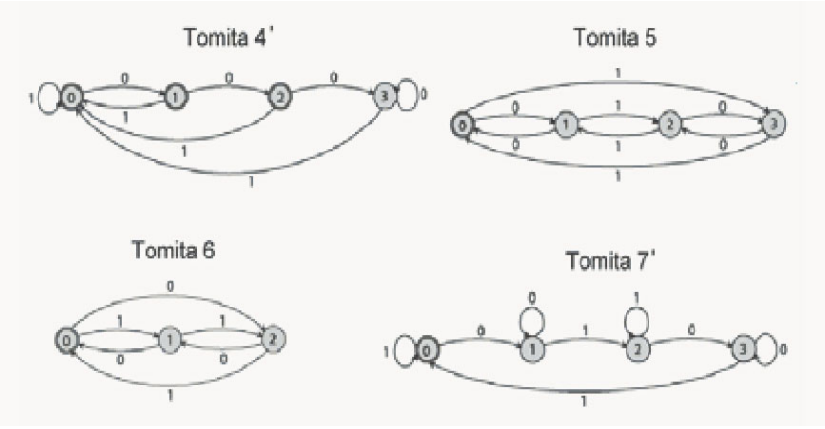


Bild 2.Die Zustandübergangs - Diagramme des Tomita Automaten. Die doppelt umkreisten Zustände haben das Ausgabezeichen 1 und die anderen Zustände haben das Ausgabezeichen 0. Der geänderte Tomita automata wird mit ' markiert.

Vier Folgen von Versuche wurden aufgeführt. Die ersten Folgen verglichen **die Größen der resultierenden neuronalen Netze**, die durch AEA, GNARL und GARNN basierend auf die Probleme von Tomita Automaten 4', 5, 6, 7' und logische Funktionen TXOR-0 - TXOR-3 erhalten wurden.
Bild 3 zeigt den Vergleich den verschiedenen neuronalen Netzgrößen. Die besten Ergebnisse wurden von AEA erreicht, d.h. AEA hatte die niedrigste Neuronenanzahl und kleinste neuronale Netzgröße.

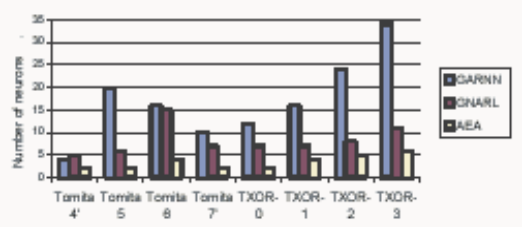


Bild 3: Vergleicht der neuronalen Netzgrößen. AEA zeigt eindeutig das beste Ergebnis.

Die nächsten Folgen verglichen **die Konvergenzgeschwindigkeit** von AEA und GNARL. GARNN wurde nicht in den Vergleich eingeschlossen, weil es eine linear-basierte Methode ist und viel schneller als evolutionäre Algorithmen ist.
Bild 4 vergleicht die AEA und GNARL miteinander. Die Konvergenzgeschwindigkeit wurde für die Tomita Automaten Identifikationsprobleme 4', 5, 6 und 7' gemessen.

Die Versuche in den letzten zwei Folgen wurden entworfen, um die Beschleunigung der Evolution wegen Verwendung des Lernens und wegen dem Austausch von WV der Mutation zu messen. Während Lernen eingeschaltet wurde, wurde der RTRL Lernalgorithmus benutzt, um 5% der besten aufführenden Individuen in der Population zu trainieren. Der durchschnittliche Fortschritt der Entwicklung des Lernens über zehn evolutionären Läufen, wurde mit dem durchschnittlichen Fortschritt der Evolution ohne das Lernen über zehn evolutionären Läufen verglichen. Der Einfluss des Lernens wurde zuerst für die Identifikationsprobleme von Tomita automata 4', 5, 6 und 7' gemessen. Die Ergebnisse werden in Bild 5 gegeben.
In allen Versuche außer für den Tomita Automat 4´ Identifikation, hat das Lernen der Evolution geholfen. Das Lernen beschleunigte die Evolution bis zu 30 % besonders für die schwierigeren Probleme wie Tomita Automat 6 Identifikation. Danach wurde der Einfluss des Lernens auf die Identifikationsprobleme zeitlicher XOR-Funktionen mit Verzögerungen von 0 bis 3 gemessen.

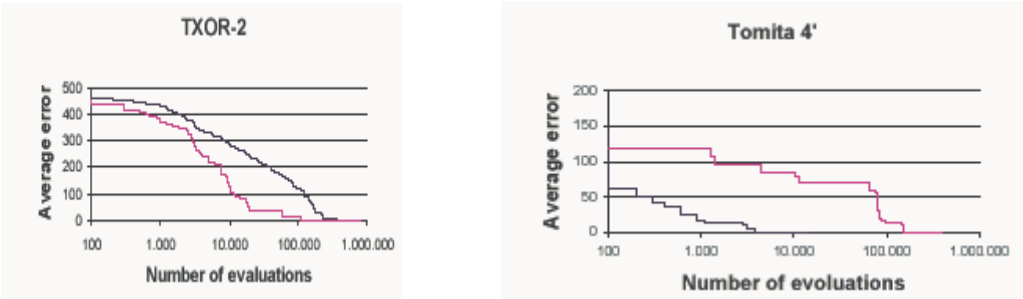


Bild 4 : Durchschnittlicher Beschleunigung der Evolution von optimalen neuronalen Netze, die TXOR-2 und Tomita 4´ Identifikationsprobleme lösen. Die punktierten Kurven zeigen durchschnittliche Evolution mit Lernen und die ununterbrochenen Kurven ohne Lernen.

Die Ergebnisse werden in Bild 4 gezeigt. Der letzte Reihe von Experimente schätzte die Leistung des AEA mit und ohne Austausch der WV. Einhundert evolutionäre Läufe wurden für jeden der Tomita automata 4', 5, 6 und 7' ausgeführt und für jede der logischen Funktionen aus TXOR-0 bis TXOR-3. Der erste Teil der Prüfung wurde entworfen, um zu sehen, wie AEA mit fester Wahrscheinlichkeitsverteilung auftritt.
Die Ergebnisse des letzten Reihe der Versuche zeigen, dass der Gebrauch der besten einzelnen WV für durchschnittlich leicht schnellere Evolutionen sorgt als die Reihe der Durchschnitt WV, jedoch ist für ein unbekanntes Problem der Austausch der WV der Mutation das Geeignetste keine einzelne WV die beste für alle Probleme ist.

6.2. ANT - Problem (Ameisen Problem)

Bei der Futtersuche markieren die Ameisen ihre Wege und orientieren sich mit größerer Wahrscheinlichkeit an solchen Wegen, auf denen sich mehr Duftstoff befindet. Dieses Verhalten wird nun zur Lösung von solchen Problemen imitiert, bei denen die Lösung als ein Weg in einem Graphen dargestellt werden kann. AEA wurde mit allen seinen erweiterten Attributen verwendet, um dieses Problem zu lösen.
Eine Ameise bewegt sich in einem diskretisierten **2D-Raum**, der ein Essenrest beinhaltet. Die Große der Ameise und die Große eines Essenpartikels bilden ein Feld. Jedes Feld im 2D-Raum bekommt den Wert

1,falls das Feld Essen hat und den Wert 0, falls das Feld kein Essen hat. Wenn die Ameise in einem Feld eintritt, frisst sie das Essen und setzt den Wert des Feldes auf 0. Der Ameise wird 200 Zeitschritte erlaubt, um der gesamte Essenrest von 89 Partikeln zu fressen. Die hat 4 Steuerungsaktionen:

1) bewegt sich
2) sich 90 grad links umdrehen
3) sich 90 grad rechts umdrehen
4)bleibt stehen.

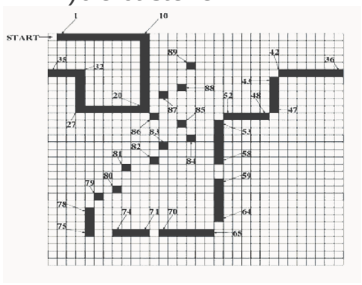


Bild 5: 2D-Raum mit einer Essensspur für das Ameisenproblem

Für jeden Aktion wird einen Strafpunkt gegeben mit nur eine Ausnahme: Wenn die Ameise sich in einem Feld mit Essen bewegt. Das neuronale Netz, das durch AEA bekommen wird, ist auf Bild 6 gezeigt. Es gibt 4 Steuerungsausgaben. Die Ausgabe mit dem höchsten Wert wird immer gewählt.

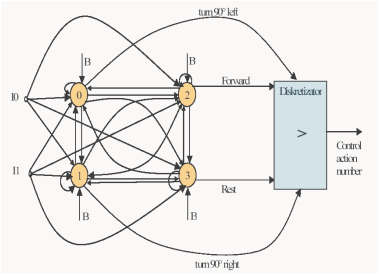


Bild 6: Das durch AEA gefundene neuronale Netz.

Die Gewichtswerte für die neuronale Netze wurden aus dem Reellzahlintervall [-500, 500] gewählt. AEA hat eine optimale neuronale Netzsteuerung entwickelt, mit durchschnittlichen 4 Neuronen aus **5,033,130** Auswertungen. Der durchschnittliche Verlauf von Zehn evolutionäre Ausführungen wird auf Bild 7 gezeigt.

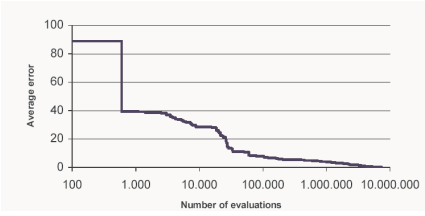


Bild 7: durchschnittlicher Verlauf von 10 evolutionäre Ausführungen (AEA).

Der schnellste aus den zehn Ausführungen dauerte nur **2,474,800** Auswertungen. Die Steuerung hat **4** Neuronen und **fraß** alle Essenpartikeln in **198** Zeitschritte. Andererseits die best GNARL neuronales Netzsteuerung hatte **12** Neuronen und brauchte **319** Zeitschritte, um alles zu fressen.[Angeline et al, 1994] . Deshalb konnte GNARL nur eine Teillösung finden.

6. Ergebnisse

AEA hat optimale Lösungen für alle zu lösender Probleme gefunden. Die durch AEA gefundenen neuronale Netze hatten höchstens halb so viele Neuronen als die vergleichbaren Lösungen. Die AEA –Lösungen haben deshalb bessere **Generalisierungsfähigkeiten**, weil sie die gleich schweren Probleme mit viel weniger Neuronen, die Informationen speichern können, lösen. Die Komplexität von AEA macht ihn fähig die schweren Probleme schneller zu lösen als die im vergleichstehenden evolutionären Algorithmen. Auf der anderen Seite AEA arbeitet nicht viel langsamer mit den einfachen Problemen.

AEA entwickelt rekursive neuronale Netze ohne Beschränkung im Bezug auf die Topologie. AEA ist gut geeignet, um durch die neuronale Netzlösungen verschiedener komplexe Probleme zu suchen.

Die einzige Beschränkung des AEA ist die evolutionär einschichtiges neuronalen Netzes .Aber diese Beschränkung betrifft nicht nur AEA, sondern fast alle andere evolutionäre Algorithmen, die neuronale

Netze entwickeln. Andererseits können einschichtige neuronale Netze die Schwierigsten Probleme lösen, solange sie mit den passenden Verarbeitungsgeschwindigkeiten laufen.
Bei der weiteren Entwicklung des AEA steht der folgende im Mittelpunkt:

Verbesserung der Methode, wie man die Wahrscheinlichkeitsverteilung der Mutation basierend auf vorhandene statistische Daten austauschen kann. Es wird dazu ein Modell eines allgemeinen mehrschichtigen neuronalen Netzes und eine Methode, um Neuronen optimal auf spezifische neuronale Netzschichten zu verteilen, entwickelt.

Literatur :

[Angeline et al, 1994] P. J. Angeline, G. M. Sauders, J. B. Pollack: *An evolutionary algorithm that constructs recurrent neural networks*. IEEE Trans. on Neural Networks, 5, (1):54-65, 1994.

[Bäck et al, 2000] T. Bäck, D. B. Fogel & Z. Mihalewicz: *Evolutionary Computation*. Institute of Physics Publishing, Bristol & Philadelphia, 2000.

[Corne et al, 1999] D. Corne, M. Dorigo and F. Glover: *New Ideas in Optimization*. McGraw-Hill, 1999.

[Dobnikar, 1995] A. Dobnikar: *Evolutionary design of application-specific neural networks: A genetic approach*. Neural Network World, 5(1):41-50, 1995.

[Gabrijel & Dobnikar, 2003] I. Gabrijel, A. Dobnikar: *On-line Identification and Reconstruction of Finite Automata with Generalized Recurrent Neural Networks*, Neural Networks, vol. 16, No. 1, 101-121, Jan., 2003.

[Haykin, 1998] S. Haykin: *Neural networks: A Comprehensive Foundation,* Prentice Hall, 1998.

[Pujol, 1999] J. C. F. Pujol: *Evolution of artificial neural networks using a two-dimensional representation*. Doktorat, School of Computer Science, University of Birmingham, 1999.

[Rudolph, 1997] G. Rudolph: *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovac, 1997.

[Tsutsui & Fujimoto, 1993] S. Tsutsui, Y. Fujimoto: *Forking genetic algorithm with blocking and shrinking modes*. Proceedings 5, International Conference on Genetic algorithms, 206-213, 1993.

[Yao, 1999]X. Yao: Evolving Neural Networks, Proceedings. of the IEEE, 1999.