Untersuchungen von evolutionären Algorithmen zum Training neuronaler Netze in der Sprachverarbeitung

Diplomarbeit

von

Markus Wolkenhauer

Institut für Angewandte Physik

Johann Wolfgang Goethe-Universität Frankfurt am Main

Oktober 1997

Inhaltsverzeichnis

1 Einleitung

Von großem wissenschaftlichem und praktischem Interesse ist die Untersuchung dynamischer Prozesse und Zeitreihenanalysen, wie beispielsweise in der Börsenkursprognose und der Sprachverarbeitung. Dabei erweisen sich rekurrente, neuronale Netze (RNN) als eine geeignete Methode. In der Regel werden die Gewichte eines neuronalen Netzes durch gradientenbasierte Verfahren anhand von Trainingsdaten berechnet. Diese haben i.a. Schwierigkeiten zu einer optimalen Lösung zu gelangen, wenn die Fehlerfunktion multimodal ist, da sie dann oft in lokale Extrema konvergieren. Zudem hat sich gezeigt, daß die zum Training von RNN eingesetzten Gradientenverfahren Abhängigkeiten über zeitlich beliebige Dauer nicht lernen können [B+94]. Es wurden verschiedene Möglichkeiten zur Lösung oder Umgehung dieses Problems [Wü94, G+96] vorgeschlagen. Diese haben jedoch den Nachteil des erheblich vergrößerten Speicherbedarfs und einer drastisch erhöhten Anzahl der Verbindungsgewichte.

In dieser Arbeit werden evolutionäre Algorithmen (EA) als alternativer Ansatz zur Bestimmung der Verbindungsgewichte von RNN betrachtet. Die hier untersuchten Verfahren sind evolutionäre Strategien (ES) und genetische Algorithmen (GA). Dabei handelt es sich um gerichtete stochastische Suchverfahren, die auf Modellvorstellungen über die der natürlichen Evolution zugrundeliegenden Prozesse basieren. Prinzipiell gelingt es mit derartigen Optimierungsverfahren lokale Extrema der Fehlerfunktion zu überwinden. Zudem wird nicht die Differenzierbarkeit der Fehlerfunktion gefordert. Nachteilig ist jedoch der im Vergleich zu Gradientenverfahren drastisch erhöhte Rechenaufwand und die i.a. verringerte Konvergenzgeschwindigkeit.

Nach einer kurzen Darstellung rekurrenter Netze und des zugehörigen gradientenbasierten Lernverfahrens werden die Grundlagen der evolutionären Algorithmen vorgestellt. Anschließend wird die prinzipielle Leistungsfähigkeit unteschiedlicher evoluti-

onärer Algorithmen zum Training der Gewichte eines mehrschichtigen Perzeptrons an einem Prädiktionsproblem untersucht und miteinander verglichen. Im weiteren wird anhand der Klassifikation von Zufallssignalen mit RNN die Leistungsfähigkeit von Gradientenverfahren und ES hinsichtlich des Erlernens von langreichweitigen zeitlichen Abhängigkeiten untersucht. Abschließend wird anhand eines, aufgrund des hohen numerischen Aufwands der ES, sehr eingeschränkten Vokabulars der Einsatz der evolutionsbasierten Optimierungsverfahren zum Training von RNN für die sprecherunabhängige Einzelworterkennung diskutiert. Die evolutionären Algorithmen werden in dieser Arbeit auch im Hinblick darauf untersucht, ob sie die Beschränkungen der Gradientenverfahren hinsichtlich der Lernbarkeit von Abhängigkeiten über zeitlich beliebige Dauer überwinden können.

2 Grundlagen neuronaler Netze mit unterschiedlicher Topologie

Neuronale Netze bestehen aus formalen Neuronen, welche durch gewichtsbehaftete Verbindungen miteinander verknüpft sind. Ein formales Neuron selbst bildet dabei eine gewichtete Summe, die sogenannte Aktivierung, über alle zu ihm hinlaufenden Eingangsdaten. Der Aktivität genannte Ausgabewert des Neurons wird berechnet, indem eine nichtlineare Transferfunktion auf die Aktivierung angewandt wird. Die am häufigst verwendeten Transferfunktionen sind Sigmoidfunktionen, wie beispielsweise der Tangens-Hyperbolicus oder die Fermi-Funktion.

Durch die unterschiedliche Anordnung der Neuronen in einem Netz entstehen verschiedene Netzwerk-Topologien. Im einfachsten Fall des Perzeptrons besteht das Netz nur aus einer Eingabeschicht und einer Ausgabeschicht. Die Eingabeschicht wird nur aus formalen Gründen aus Neuronen bestehend bezeichnet, da diese in keiner Weise an der Informationsverarbeitung direkt beteiligt sind.

Damit neuronale Netze die Fähigkeit zur Lösung von nichtlinear separierbaren Aufgabenstellungen erlangen, muß eine weitere Schicht von Neuronen zwischen Eingangsschicht und Ausgangsschicht eingefügt werden. Diese wird als verborgene Schicht bezeichnet und besitzt eine nichtlineare Transferfunktion. Findet die Informationsweiterleitung innerhalb des Netzes unidirektional von der Eingangsschicht über die verborgene Schicht zu der Ausgangsschicht statt, so bezeichnet man diese Struktur als mehrschichtiges Perzeptron (MLP). Die Schwellwerte der informationsverarbeitenden Neuronen werden dadurch gebildet, indem ein Neuron dem Netz zusätzlich hinzugefügt wird, welches als konstante Ausgabe den Wert 1 hat. Diese Aktivität wird in alle informationsverarbeitenden Neuronen über gewichtete Verbindungen eingespeist. Damit geht die Aktivität des Schwellwerts in Form eines Gewichtes ein.

Wird die geschichtete Struktur des Netzwerkes aufgegeben und jedes Neuron mit jedem anderen und sich selbst verknüpft, so erhält man ein vollvernetztes, rekurrentes neuronales Netz (RNN). Ein solches Netz mit 5 Neuronen und einem Schwellwertneuron ist in Bild 1 dargestellt.

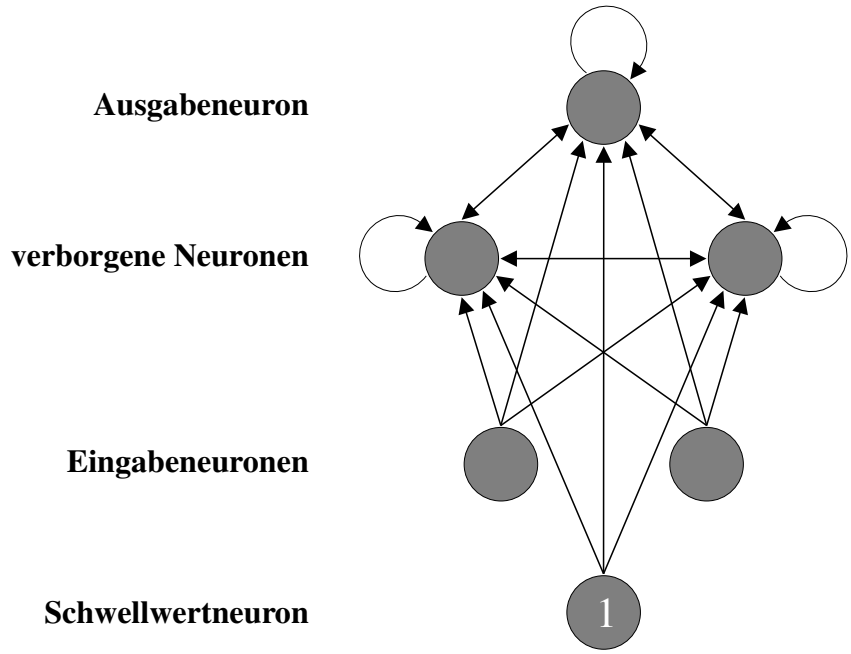

Bild 1: Struktur eines RNN mit 5 Neuronen und einem Schwellwertneuron

Die Informationsweiterleitung innerhalb des Netzes ist nicht mehr unidirektional. Vielmehr ist eine Unterscheidung zwischen vorwärts- und rückwärtsgerichteten Verbindungen nicht mehr möglich. Damit ein RNN physikalisch realisierbar wird, müssen alle Verbindungen eine Zeitverzögerung, von mindestens einem Zeittakt erhalten. Da alle Neuronen im Netz gleichberechtigt sind, muß durch die Auswahl von Teilmengen der Neuronen eine Eingabe-, Verarbeitungs- und Ausgabestruktur erzeugt werden. Eine Teilmenge E mit N_e Neuronen wird als Eingabeneuronen ausgewählt, deren Aktivitäten gleich der angelegten Eingabewerte sind. Eine andere Teilmenge A mit N_a Neuronen repräsentiert die Ausgabeneuronen, deren Aktivitäten als Ausgabe gewertet werden. Ein RNN wird somit durch eine $N * (N - N_e)$ Gewichtsmatrix beschrieben.

Innerhalb des RNN kommt es zu Signalrückkopplungen, so daß ein zu einem beliebigen Zeitpunkt anliegendes Eingangsmuster die zukünftigen Aktivitäten aller informationsverarbeitenden Neuronen im Netz für alle weiteren Zeiten beeinflußen kann. Dies bedeutet, daß ein RNN prinzipiell ein unendlich langes Gedächtnis hat, da eine Abhängigkeit der aktuellen Neuronenaktivierungen von allen früheren Aktivierungen existiert. RNN mit nichtlinearen Transferfunktionen stellen nichtlineare, dynamische Systeme dar [Pin89, Pea89]. Die über die Zeit beobachteten Aktivitäten der Ausgangsneuronen können daher eine komplexe zeitliche Dynamik bis hin zu chaotischem Verhalten aufweisen.

Das grundlegende Ziel des Trainings eines neuronalen Netzes ist die Minimierung eines beliebigen, differnzierbaren Fehlermaßes über die gesamte Menge der Trainingsmuster, wobei zumeist der quadratische Fehler verwendet wird.

2.1 Der Trainingsalgorithmus BPTT

Der gradientenbasierte Lernalgorithmus, der zu den Vergleichsmessungen benutzt wird, ist eine Modifikation des Error-Backpropagation-Algorithmus für MLP, welchen man als Error-Backpropagation-Through-Time (BPTT) [Wer90] bezeichnet. Gemein ist diesen Verfahren, daß der Gesamtfehler durch ein Gradientenabstiegsverfahren minimiert wird. Dabei existieren verschiedene BPTT-Varianten: Backpropagation-Through-Time, Truncated-Backpropagation-Through-Time und Epochwise-Backpropagation-Through-Time [Wer90]. Der Gradient ΔW ist definiert als die partielle Ableitung des Fehlers nach den Netzgewichten $\underline{\underline{W}}$ gemäß

$$\Delta W_{ij} = \frac{\partial F^{Total}(ta, te)}{\partial W_{ij}}. \tag{1}$$

Der Zeitparameter kann eliminiert werden, indem das rückgekoppelte Netz als MLP betrachtet wird [RM86], dessen Anzahl der Schichten mit der Zeit t zunimmt. Beginnend mit einer Schicht zur Zeit $t = 0$, fügt man mit jedem weiteren Zeittakt

eine Schicht mit identischer Gewichtsmatrix hinzu. Somit wird der Zeitparameter t zu einem Parameter der Schichtenanzahl eines in der Zeit entfalteten MLP. Auf diese Weise läßt sich eine zeitliche Entkopplung beim Backpropagation-Algorithmus erreichen. Bild 2 zeigt für den Zeittakt $t = 3$ ein einfaches rückgekoppeltes Netz (a) und ein dazu äquivalentes MLP (b).

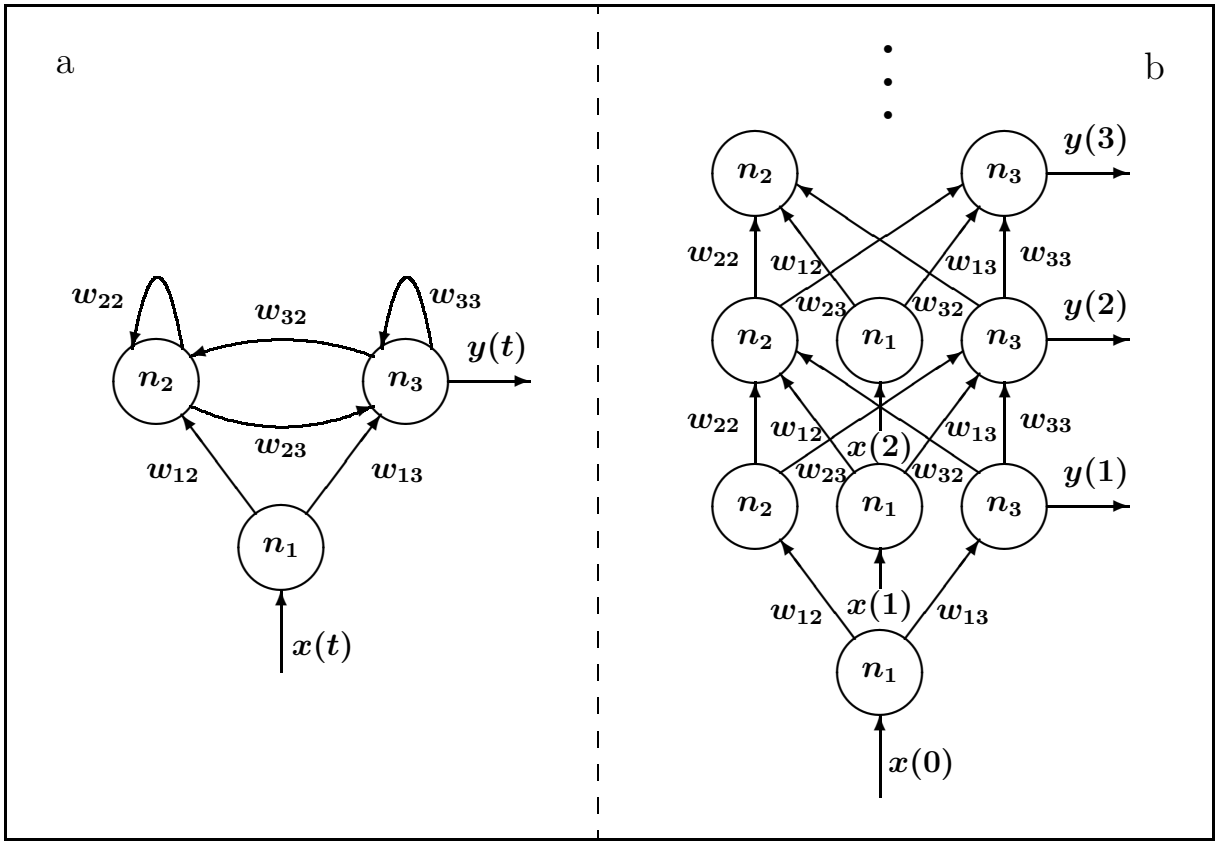

Bild 2: Äquivalente Darstellung eines RNN (a) durch ein MLP (b)

Für die weitere Beschreibung des Trainingsalgorithmus werden folgende Notationen und Definitionen eingeführt:

- t : diskreter Zeitpunkt
- N : Gesamtanzahl der Neuronen im Netz
- N_e : Anzahl der Eingabeneuronen

- N_a : Anzahl der Ausgabeneuronen
- M : N - N_e
- E : Indexmenge der Eingabeneuronen
- A : Indexmenge der Ausgabeneuronen
- U : Indexmenge der verborgenen Neuronen
- $T(t)$: Indexmenge der Ausgabeneuronen für die zur Zeit t ein Ausgabewert bekannt ist
- $\underline{\underline{W}}$: (N x M)-Matrix der Verbindungsgewichte w_{ij}
- $\underline{h}\,(t)$: Aktivierungen zur Zeit t
- $\underline{X}\,(t)$: Aktivitäten aller Neuronen zur Zeit t
- $f(x)$: Transferfunktion
- $f'(x)$: Ableitung der Transferfunktion
- $\underline{e}\,(t)$: Eingabewerte für die Menge der Eingabeneuronen zur Zeit t
- $\underline{a}\,(t)$: Sollausgabewerte für die Menge der Ausgabeneuronen zur Zeit t
- $\underline{\epsilon}\,(t)$: Fehlerwerte der Menge der Ausgabeneuronen zur Zeit t

Damit ergibt sich die neue Netzaktivität $\underline{X}\,(t+1)$ nach Setzen der Eingabewerte $\underline{e}\,(t)$ aus der Netzaktivität $\underline{X}\,(t)$ durch

$$h_k(t+1) = \sum_{l \in E \cup A \cup U} X_l(t) W_{lk} \text{ für } k \in A \cup U \tag{2}$$

$$X_k(t+1) = f_k(h_k(t+1)) \text{ für } k \in A \cup U. \tag{3}$$

Entsprechend ist die Ausgabe des Netzes zum Zeitpunkt $t+1$ gegeben durch die Aktivitäten der Ausgabeneuronen $X_k(t+1)$, wobei $k \in A$. Für alle Ausgabeneuronen

$T(t) \subset A$, für die ein Sollausgabewert $a(t)$ zur Zeit t bekannt ist, errechnet sich damit der Netzfehler als

$$\epsilon_k(t) = \begin{cases} a_k(t) - X_k(t) & : \quad k \in T(t) \\ 0 & : \quad \text{sonst} \end{cases} . \tag{4}$$

Mit dieser Definition des Fehlers ist es nicht notwendig, daß zu jedem Zeitpunkt für alle Ausgabeneuronen ein Sollausgabewert bekannt ist. Der Gesamtfehler $F(t)$ des Netzes zum Zeitpunkt t ergibt sich zu

$$F(t) = \frac{1}{2} \sum_{k \in A} \left[\epsilon_k(t)\right]^2, \tag{5}$$

bzw. über eine Zeitperiode mit Anfangszeitpunkt t_a und Endzeitpunkt t_e als

$$F^{Total}(t_a, t_e) = \sum_{t=t_a+1}^{t_e} F(t). \tag{6}$$

Damit ist die rekursive Berechnung des Gradienten nach jedem Zeittakt von $\tau = t$ bis $\tau = 1$ gegeben durch

$$\delta_k(\tau) = \begin{cases} f'\left[h_k(\tau)\right] \epsilon_k(\tau) & : \quad \tau = t \\ f'\left[h_k(\tau)\right] \left[\epsilon_{\mathbf{k}}(\tau) + \sum\limits_{\mathrm{l}} \mathrm{W_{kl}} \delta_{\mathrm{l}}(\tau + 1)\right] & : \quad 0 < \tau < t, \end{cases} \tag{7}$$

wobei die Gewichtsänderung gemäß

$$\Delta W_{ij} = -\eta \sum_{\tau=1}^{t} \delta_j(\tau) X_i(\tau - 1) \tag{8}$$

erfolgen kann. Der Faktor η bestimmt die Lernschrittweite und liegt im Intervall $0 < \eta < 1$. Da das dem RNN äquivalente MLP mit jedem Zeittakt um eine weitere Schicht wächst, steigt der Verarbeitungsaufwand beträchtlich an. Bild 3 zeigt graphisch den Verlauf der Fehlerrückführung beim BPTT. Zur Kennzeichnung der verschiedenen Fehler wurden die Indizes s und z benutzt. Der Index s kennzeichnet den aus späteren Netzaktivitäten hervorgerufen Fehler der zurückpropagiert wird, analog dem Fehler der verborgenen Schichten in einem MLP. Index z bezeichnet den Fehler, der zum aktuellen Zeitpunkt durch die Netzaktivitäten und der Sollausgabe $\underline{a}\,(t)$ gebildet wird.

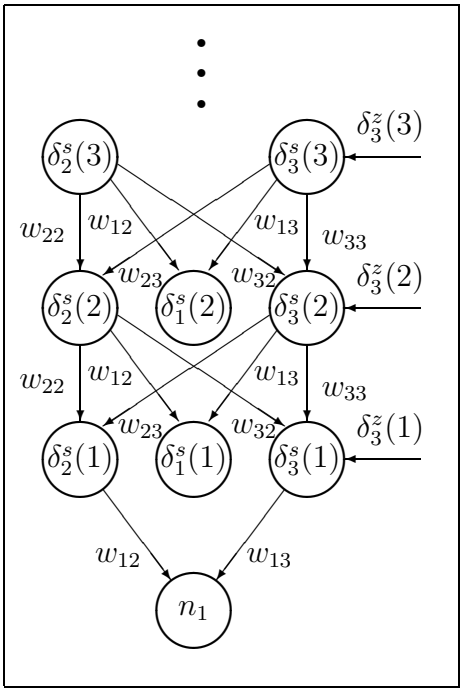

Bild 3: Fehlerrückführung beim BPTT

Ist die Zeitspanne, in denen der Fehler zurückgeführt werden muß, sehr lang, so ist der Aufwand sehr groß, da für jeden Zeitpunkt die Netzaktivierung zwischengespeichert und nach jedem Zeittakt die Gradientenberechnung immer wieder bis zum Zeitpunkt Null rückgerechnet werden muß.

Um diesen Rechenaufwand zu reduzieren, kann der Algorithmus zu einem segmentweisen Algorithmus verändert werden. Dies setzt voraus, daß eine Segmentierung der Lernmenge existiert oder die Lernmenge insgesamt nicht zu viele Trainingsmuster enthält. Bei dieser Form des BPTT erfolgt die Gradientenberechnung erst am Ende eines Segments, wodurch der Rechenaufwand erheblich sinkt. Für diese Art der Berechnung muß die Netzaktivierung eines Segments gespeichert werden. Zusätzlich müssen die anfallenden Netzfehler der Ausgabeneuronen zwischengespeichert werden, um sie bei der Gradientenberechnung am Segmentende einbeziehen zu können. Zu diesem erhöhten Speicheraufwand gegenüber dem Grundalgorithmus kann eine unter Umständen größere Anzahl der notwendigen Trainingsiterationen gegenüber

dem zuvor beschriebenen BPTT hinzukommen, da pro Segmentdurchlauf nur je eine Gewichtsänderung erfolgt. Dadurch kann der erzielte Aufwandsvorteil wieder zunichte gemacht werden. Die modifizierte Gleichung mit den zwischengespeicherten Fehlern ergibt sich zu

$$\delta_k(\tau) = \begin{cases} f'\left[h_k(\tau)\right] \epsilon_k(\tau) & : \quad \tau = t \\ f'\left[h_k(\tau)\right] \left[\epsilon_{\mathbf{k}}(\tau) + \sum\limits_{\mathrm{l}} \mathrm{W_{kl}} \delta_{\mathrm{l}}(\tau+1)\right] & : \quad t - R < \tau < t. \end{cases} \tag{9}$$

Die hier verwendete Formel entspricht einer MLP-Backpropagation-Formel, bei der die Sollausgabewerte nicht nur für Neuronen der Ausgabeschicht, sondern auch für Neuronen der versteckten Schichten bekannt sind. Man spricht in diesem Fall vom Einfügen des Fehlers zur Zeit t. Dieser Algorithmus wurde im weiteren verwendet. Der Fehler wurde, wenn nicht anders beschrieben, immer nur am Ende eines Segments berechnet und anschließend zurückpropagiert.

2.2 Eigenschaften von Gradientenverfahren

Allgemein ist die Leistungsfähigkeit von Gradientenverfahren stark von der Topologie der Funktionslandschaft, auf der sie angewendet werden abhängig. Dies bedeutet bei einer großen Anzahl von Maxima und Minima auf dieser Landschaft, daß die Wahrscheinlichkeit dafür, daß dieser Typus von Verfahren in ein lokales Optimum hineinläuft und nicht mehr aus diesem hinausfindet, ebenfalls groß ist.

Bei rückgekoppelten neuronalen Netzen, die Probleme mit langen Zeitabhängigkeiten, im Sinne von langreichweitigen Abhängigkeiten in den Eingangsmustern lösen sollen, besteht noch eine weitere Schwierigkeit. Dieser Netzwerkstyp kann auf Grund der Rückkopplung ein sehr komplexes, nichtlineares Verhalten in der Zeit aufweisen, da ein zum Zeitpunkt $t = 0$ dem Netz präsentiertes Eingabemuster alle folgenden Ausgaben des Netzes beeinflußt. Betrachtet man zum Beispiel den Fall, daß die Klassifikation einer Mustersequenz der Länge T nur von den ersten L Merkmalsvektoren, mit $L < T$, abhängt und das Klassifikationsergebnis nur am Ende einer

Sequenz bekannt ist, so stellen alle Merkmalsvektoren $t_L < t \leq T$ Rauschen dar. Um das Problem lösen zu können, muß das Netz lernen, die Informationen, welche zu den Zeitpunkten $0 \leq t_L$ anliegen, robust über $(T - L)$-Zeittakte zu speichern. Die Ausgabe am Ende der Sequenz ist abhängig von allen vorangegangen Netzaktivitäten zu den Zeitpunkten $t < T$ und damit auch von den Störungen durch die Merkmalsvektoren die zu den Zeitpunkten $t_L < t \leq T$ anliegen, welche Rauschen darstellen.

Da das Netz ein nichtlineares Verhalten hat, können schon kleine Störungen dazu führen, daß die Netzaktivitäten sich völlig anders in der Zeit entwickeln als im ungestörten Fall. Damit wird aber auch die Berechnung des Gradienten gestört, da dieser von der Ausgabe des Netzes zur Zeit T abhängt und nur zu diesem Zeitpunkt der Netzfehler berechnet werden kann. Diese Störungen werden mit wachsenden Sequenzlängen, d.h. mit steigender Anzahl der Zeittakte, in der das Netz die Information speichern muß, immer stärker die Gradientenberechnung beeinflußen. Damit kann ein Lernen der robusten Informationsspeicherung für diese Art des Trainingsalgorithmus, ohne kompensierende Maßnahmen, nicht mehr möglich sein.

Verbunden damit ist das Problem des gegen Null strebenden Gradienten bei langen Zeitabhängigkeiten. In [B+94] wird eine analytische Beschreibung dieses Sachverhaltes dargestellt. Die Analyse geht von dem zuvor beschriebenen Sachverhalt bezüglich der Eingabe- und Sollausgabemuster aus. Folgende grundlegenden Anforderungen werden an ein parametrisches, dynamische System gestellt, welches lernen soll relevante Zustandsinformationen zu speichern:

1. Das System soll in der Lage sein, Informationen über eine beliebige Zeitdauer zu speichern.

2. Das System soll robust gegenüber Störungen sein (beispielsweise Variationen in den Eingabemustern, die Rauschen darstellen oder irrelevant für die Berechnung der richtigen Ausgabe sind).

3. Die Systemparameter müssen in begrenzter Zeit trainierbar sein.

Das Ergebnis dieser Analyse ist, daß ein solches System entweder stabil und unempfindlich gegenüber Störungen oder gut trainierbar durch ein Gradientenverfahren ist, jedoch nicht beides gleichzeitig. Es zeigt sich, daß bei dem Versuch die ersten beiden Bedingungen zu erfüllen, die Ableitung des Systemzustandes $\underline{X}(t)$ zum Zeitpunkt $t = \tau$ nach dem Systemzustand zum Zeitpunkt $t = 0$ exponentiell mit wachsendem τ verschwindet. Das bedeutet, daß der Zustand des Systems zum Zeitpunkt t im abnehmenden Maße von den vorherigen Zuständen bestimmt ist. Aus diesem Grund sind diese Algorithmen nicht in der Lage Zeitabhängigkeiten von langer Zeitdauer im Gradienten zu repräsentieren.

In der Regel werden alle hier angesprochenen Probleme beim Training eines RNN in kombinierter Form auftreten.

3 Evolutionäre Algorithmen

3.1 Historische Bemerkungen

Im Jahre 1859 veröffentlichte Charles Darwin (1809 - 1882) sein aufsehenerregendes Werk *On the Origin of Species by Means of Natural Selection*, in dem er die Entwicklung der Arten auf einen Vorgang der Anpassung an die jeweilige ökologische Nische durch die Auslese der am besten angepaßten Individuen einer Population zurückführte. Bis dato war man im allgemeinen von einer Konstanz der Arten ausgegangen, welche seit Anbeginn der Zeiten, d.h. seit der Urschöpfung existierten. Charles Darwin war nicht der Erste, der den Gedanken der Konstanz der Arten zu widerlegen suchte, sondern er nahm die Gedanken anderer auf, zitierte und erweiterte diese in seinem Werk. So erkannte bereits Jean Baptiste de Lamarck, daß sich Arten derart entwickeln, daß sie sich optimal an ihren Lebensraum anpassen. Er konnte jedoch nicht schlüssig erklären wie diese Anpassung erfolgt und ging letztlich von bestimmten Bedürfnissen der einzelnen Arten aus. So sprach er den Arten einen Drang zur Vervollkommnung zu, welcher die Triebfeder der Anpassung sei.

Wesentliche Teile der Evolutionstheorie wurden bereits 1855 in einem Artikel mit dem Titel *Über das Gesetz, welches die Einführung von neuen Arten reguliert* von Alfred Russell Wallace (1825 - 1913) formuliert. Darwin war sich der Tatsache bewußt, daß er den Evolutionsgedanken nicht erfunden hat und nennt in der deutschen Übersetzung seines Werkes [Dar56] über zwei Dutzend Autoren, die Teile dieses Gedankenwerkes schon vor ihm formulierten. Die Begriffe der *Evolution* für den Prozeß und *survival of the fittest* für das Selektionsprinzip wurden von einem Interpreten der Werke Darwins, dem Philosophen Spencer (1820 - 1905) eingeführt. Charles Darwin entwickelte den Evolutionsgedanken aus den drei folgenden Beobachtungen und Grundannahmen:

1. In der Natur wird ein Überschuß von Lebewesen in Form von Nachkommen produziert und trotzdem bleibt die Populationsgröße beschränkt. Allein durch die Tatsache, daß die Nahrungsmittel meist nur arithmetisch zunehmen, die Populationen jedoch geometrisch wüchsen, wenn das Wachstum ungehemmt wäre, entsteht ein Selektionsdruck, um die für die ökologische Nische angepaßte Populationsgröße zu erreichen.

2. Die Individuen einer Art sind nicht identisch, sondern unterscheiden sich. Eine Art besitzt eine mehr oder weniger starke Variationsbreite ihrer Erbanlagen.

3. Haben sich Varianten von Erbanlagen im Kampf um das Überleben bewährt, finden sich diese bevorzugt in den Folgegenerationen wieder.

Aus diesen drei Grundeinsichten leitete Darwin folgendes ab: Aufgrund des Selektionsdruckes werden die Individuen mit den relativ zu den anderen besseren Erbanlagen selektiert, so daß die im Regelfall nur geringen Abweichungen des Erbgutes in der nächsten Generationen häufiger auftreten und damit langfristig eine schrittweise bessere Adaption an die herrschenden Umweltbedingungen erfolgt.

Auf diesem Konzept basiert auch das heutige Verständnis der Evolution, jedoch ist es noch wesentlich erweitert worden. So um die Auffassung der Evolution als rückgekoppelter Prozeß [FM93] , da die Arten wiederum auf ihre ökologische Nische einwirken und sie verändern, zum Beispiel durch die Entnahme von Nahrung aus ihrer Umgebung. Durch die Verringerung im Nahrungsangebot steigert sich der Selektionsdruck.

Die Erforschung der Informationsweitergabe an Nachkommen auf Zellebene durch die Meiose und Zellkernverschmelzung und speziell die Entdeckung der Codierung und Speicherung der Erbinformationen in der DNS führten zu einem umfassenderen Verständnis. Das erweiterte Wissen um die Vererbung auf zellularer Ebene und der ihr zugrundeliegenden Mechanismen, wie die Speicherung bzw. die Codierung der

Erbinformation, sowie der Informationsweitergabe und des Informationsaustauschs und der damit verbundenen Veränderung des Erbmaterials mittels Rekombination und Mutation, ermöglichte in Verbindung mit der Entwicklung von leistungsfähigen Computern die technische Anwendung dieser Optimierungsverfahren. Die darauf basierenden Arten der Optimierungsverfahren stellen idealisierte Modelle der Evolution dar und unterscheiden sich unter anderem in ihrem Grad der Idealisierung.

Die bei den Optimierungsverfahren verwendeten Begriffe sind der Biologie entlehnt, haben aber teilweise abweichende Bedeutungen. Ein Individuum ist in der Biologie ein einzelnes Mitglied einer Gesellschaft. Gekennzeichnet ist das Individuum dabei durch seine phänotypische Ausprägung. In den EA wird der Begriff synonym sowohl für den Phänotypus, als auch für die „Erbinformation“, d.h. die Vektoren der problembeschreibenden Parameter, benutzt. Insbesondere können diese Vektoren sowohl die eigentlichen Objektvariablen, die die problembeschreibenden Parameter repräsentieren, als auch zusätzlich die einem Individuum zugeordneten Parameter zur Steuerung der Optimierung enthalten.

Der Begriff der Population bezeichnet die Gesamtheit aller Individuen in einem Gebiet die sich miteinander fortpflanzen können und wird auch bei den Optimierungsverfahren in dieser Weise benutzt.

Die Algorithmen unterscheiden sich in der Repräsentation der problembeschreibenden Parameter. So existieren Algorithmen, die eine Codierung dieser Variablen benutzen, wie dies in der Natur der Fall ist. Wird eine solche Repräsentation benutzt, dann werden Vektoren dieser Art auch als Chromosomen bezeichnet. In der Natur wird die „Erbinformation“ in Sequenzen von vier verschiedenen Basen in der DNS abgelegt, während in der technischen Anwendung der Optimierungsverfahren binäre Codierungen bevorzugt Verwendung finden.

Die Nachkommenerzeugung geschieht in erster Linie durch Verdoppelung der „Erbinformation“ der Elternindividuen und Mutation. Sind mehrere Eltern beteiligt,

so findet vor der Mutation eine Rekombination statt. Wie in der Natur bewirken Mutation und Rekombination eine Veränderung der in der Population enthaltenen Individuen im Laufe der Zeit.

Mutation bedeutet eine zufällige Veränderung der „Erbinformation“. In der Natur ist dies durch Strahlung, bestimmte chemische Verbindungen und durch die Einwirkung von Wärme, die zu Chromosomenbrüchen führen kann, gegeben. Dabei sind verschiedene Formen der Mutation zu beobachten. Zum einen können einzelne Basen aus der DNS-Sequenz durch eine andere ausgetauscht werden, wobei dies als Punktmutation bezeichnet wird. Zum anderen können ganze Gruppen von Basen herausgelöst und anschließend in der Abfolge vertauscht wieder in die DNS eingebaut werden. Diesen Prozeß bezeichnet man als Inversion. Weiterhin können in eine bestehende Basensequenz neue Basen eingebaut oder bestehende herausgelöst werden, was als Insertion bzw. Deletion bezeichnet wird. In ihrer Wirkung auf den Phänotyp unterscheiden sich die einzelnen Formen der Mutation erheblich. Während Punktmutationen bei einer Basensequenz, die zum Beispiel ein Protein codiert, die Funktionsfähigkeit weniger oft gravierend beeinflußen, ist dies bei Insertion und Deletion und insbesondere der Inversion eher die Regel. Alle genannten Formen der Mutation finden in den Optimierungsverfahren Anwendung. Insertion und Deletion werden allerdings seltener genutzt, da meist mit konstanten Vektorlängen gearbeitet wird.

Als zweite Möglichkeit der Informationsweitergabe und der Veränderung bzw. der Kreation von neuen Individuen dient die Rekombination. Dabei tauschen mehrere Elternindividuen zufällig ausgewählte Teilstücke ihrer „Erbinformation“ aus, die zu einem neuen Individuum zusammengesetzt werden. Durch die Verwendung der Rekombination kann Information innerhalb einer Population verteilt und ausgetauscht werden. Die Rekombination findet in der Natur innerhalb der Meiose, dem Vorgang der Erzeugung der haploiden Geschlechtszellen, statt.

3.2 Evolutionäre Strategien

Die ES wurden in den 60er und 70er Jahren von Ingo Rechenberg [Rec73] entwickelt und später hauptsächlich im deutschsprachigen Raum weiterentwickelt. Der Pseudocode der ES lautet wie folgt:

Wahl einer geeigneten reellwertigen Darstellung des zu optimierenden Problemes
Initialisierung der Anfangspopulation
do

- Bewertung des Grads der Optimierung der Individuen mittels einer Qualitätsfunktion
- Erzeugung der Nachkommenschaft mittels Mutation und je nach Verfahren auch mit Rekombination
- Bewertung des Grads der Optimierung der Nachkommen mittels einer Qualitätsfunktion
- Bildung einer neuen Generation durch Auswahl von Individuen aus den Scharen von Eltern und Nachkommen durch ein Selektionsschema

while
Abbruchkriterien nicht erfüllt

Zuerst wird eine Anfangspopulation initialisiert, wobei im Gegensatz zu GA als Repräsentation eine reellwertige Darstellung einer Optimierungsaufgabe gewählt wird. Dies bedeutet, daß die relevanten Informationen eines Individuums in Vektoren $\underline{x}_i$ von reellen Zahlen codiert werden, weshalb dieser Ansatz als phänotypisch orientiert bezeichnet wird. Die Erbinformationen werden dadurch auf die qualitativen Merkmalsausprägungen, dies sind die Parameterwerte, reduziert. Häufig ist jedoch

zusätzlich noch eine phänotypische Interpretation bzw. Umrechnung der Vektoren erforderlich, um den Grad der Optimierung eines Individuums bewerten zu können. Beispielsweise sind bei einem Roboterarm, der den kürzesten Weg zwischen zwei Punkten zurücklegen soll, die Winkelstellungen in der Zeit als reellwertiger Vektor codiert. Die physikalische Realisierung dieser Information ist jedoch die Trajektorie, die der Arm zurücklegt. Diese muß aus dem Winkelvektor errechnet werden.

Aus der Grundpopulation werden einzelne Individuen mit gleichverteilter Wahrscheinlichkeit ausgewählt. Diese dienen dann als Eltern von Nachkommen, die durch Mutation und je nach Variante der ES auch durch Rekombination erzeugt werden. Eine neue Generation wird durch Verwendung eines Ersetzungsschema mittels der Bewertung der Individuen durch eine Qualitätsfunktion $Q(\underline{x})$ gebildet, die die Güte der Optimierung beschreibt. Dieser Ablauf wird beginnend mit der Auswahl von Individuen solange iterativ wiederholt, bis ein Abbruchkriterium erreicht worden ist. Die Qualitätsfunktion sollte sich idealerweise bei kleinen Änderungen auch nur geringfügig ändern, d.h. streng kausal verhalten. Das Abbruchkriterium besteht im Regelfall entweder aus dem Erreichen einer maximal zulässigen Anzahl von Generationen oder dem Unter- bzw. Überschreiten einer Schranke der Qualitätsfunktion.

3.2.1 Mutation

Der Mutation kommt bei den ES eine entscheidende Rolle als treibende Kraft bei der Optimierung zu. Das Hauptprinzip der ES ist die Nachkommenerzeugung durch die Verdoppelung von zufällig ausgewählten Individuen. Hierbei wird der Objektvariablenvektor eines Elter dupliziert und anschließend mutiert. Die Mutation erfolgt dadurch, daß auf die durch reellwertige Zahlen dargestellten Objektvariablen eine Zufallszahl, die aus einer mittelwertfreien Normalverteilung stammt, addiert wird. Dies bedeutet, daß auf den Ausgangsvektor des Elter ein Zufallsvektor addiert wird.

Dabei gilt

$$\underline{x}_{neu} = \underline{x}_{alt} + \underline{n}(0, \sigma) \tag{10}$$

bzw. für die einzelnen Komponenten

$$x_{j,neu} = x_{j,alt} + n, \tag{11}$$

wobei der Summand n durch einen zweistufigen Zufallsprozeß bestimmt wird. Im ersten Schritt wird eine Zufallszahl z aus einer Normalverteilung bestimmt. Nur falls diese kleiner als der Mutationsgrenzwert α_m ist, findet im zweiten Schritt die Mutation statt. Der zweite Schritt besteht darin, eine Zufallszahl n aus einer mittelwertfreie Gaußverteilung mit Varianz σ_m zu bestimmen. Aus der Wiederholung des Zufallsprozeß für jede Vektorkomponente j ergibt sich der beschriebene Zufallsvektor $\underline{n}(0, \sigma_m)$, dessen Komponenten unabhängig voneinander sind. Die Mutation ist somit von zwei Parametern, den Strategieparameter α_m und σ_m, abhängig. Der Mutationsgrenzwert α_m regelt, ob eine Vektorkomponente mutiert wird oder nicht. In der ursprünglichen Formulierung der ES war der Parameter α_m nicht vorhanden. σ_m ist die Varianz der Gaußverteilung, aus der die zu addierende Zufallszahl ermittelt wird und bestimmt damit die Schrittweite der Mutation.

Damit bilden die Nachkommen eines Elter eine Hyperkugel um das Elterindividuum. Aufgrund der Charakteristik der Gaußverteilung nimmt die Dichte der Nachkommen um das Elter mit größer werdendem radialem Abstand ab, wobei durch die Mutation keine Richtung im Suchraum bevorzugt wird.

Beide Strategieparameter sind an das Problem adaptiert zu wählen. Sowohl eine zu kleine, als auch eine zu große Wahl der Strategieparameter führt zu extrem langen Konvergenzzeiten oder zu einem vorzeitigen Ende der Optimierung in einem lokalen Extremum. So entartet der Algorithmus bei einer zu großen Wahl der Strategieparameter zu einem total stochastischen Suchverfahren und profitiert nicht mehr von der impliziten, gerichteten Suche. Dieser Sachverhalt wird von Rechenberg [Rec92]

durch den Begriff des Evolutionsfensters beschrieben. Eine Variation der Strategieparameter verändert die Konvergenzgeschwindigkeit des Algorithmus. Innerhalb eines Sub-Intervalls des Intervalls aus denen die Strategieparameter bestimmt werden, steigt die Konvergenzgeschwindigkeit an, um dann wieder abzufallen. Damit ist ein Optimum der Konvergenzgeschwindigkeit gegeben. Bild 4 zeigt schematisch die Entwicklung der Konvergenzgeschwindigkeit in Abhängigkeit von einem Strategieparameter.

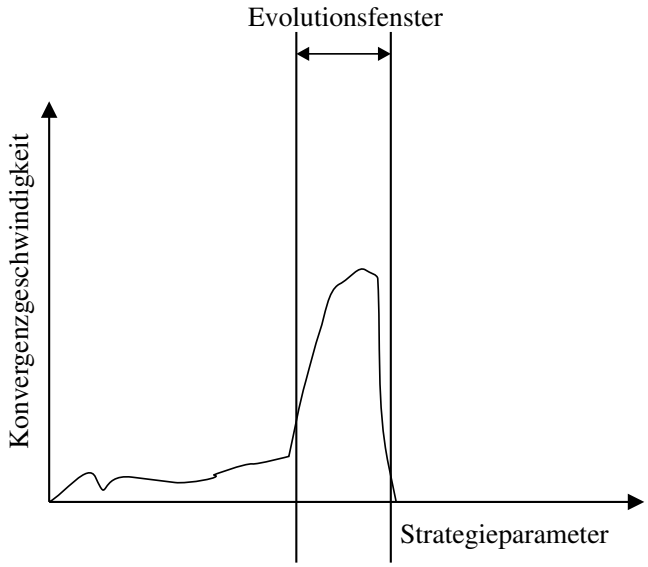

Bild 4: Evolutionsfenster

Es existieren verschiedene Möglichkeiten der Adaption dieser Parameter. Einerseits ist es möglich, diese Parameter global für die gesamte Population und alle Generationen – d.h. für alle Individuen und alle Zeiten – konstant zu wählen. Andererseits kann man den Mutationsgrenzwert α_m wie auch die Schrittweite σ_m für alle Individuen zu Beginn gleich wählen, wobei diese Werte im Laufe der Optimierung jedoch verändert werden. Die verschiedenen vorgeschlagenen Verfahren zur Parameterveränderung sind alle stark von Heuristiken abhängig [Bä93]. Eine häufig verwandte Methode nutzt als Kriterium die Geschwindigkeit der Konvergenz. Falls diese sehr gering ist und die Güte der Optimierung nur in geringem Maße variiert, werden Mutationsgrenzwert und Schrittweite vergrößert, da anzunehmen ist, daß man sich in einem

lokalen Optimum oder auf einem flach verlaufenden Stück der Qualitätsfunktion befindet. Solange die Qualitätsfunktion starken Schwankungen unterliegt werden die Parameter verkleinert, da anzunehmen ist, daß der Suchalgorithmus entweder nur rein stochastisch den Raum absucht oder ein Optimum gefunden hat, jedoch auf Grund der zu großen Parameter wieder aus dem Optimum herausgetrieben wird.

Die Strategieparameter werden daher zu bestimmten, vom Adaptionsverfahren abhängigen Zeitpunkten anhand von

$$\alpha_{m,neu} = \begin{cases} \alpha_{m,alt}\xi^{sp}_{\alpha_m} & : \text{ wenn } \underline{Q}(\underline{x}) \text{ alternierend und } \xi^{sp}_{\alpha_m} < 1 \\ \alpha_{m,alt}\xi^{st}_{\alpha_m} & : \text{ wenn } \underline{Q}(\underline{x}) \text{ stagnierend und } \xi^{st}_{\alpha_m} > 1 \\ \alpha_{m,alt} & : \text{ sonst} \end{cases}, \tag{12}$$

und

$$\sigma_{m,neu} = \begin{cases} \sigma_{m,alt}\xi^{sp}_{\sigma_m} & : \text{ wenn } \underline{Q}(\underline{x}) \text{ alternierend und } \xi^{sp}_{\sigma_m} < 1 \\ \sigma_{m,alt}\xi^{st}_{\sigma_m} & : \text{ wenn } \underline{Q}(\underline{x}) \text{ stagniernd und } \xi^{st}_{\sigma_m} > 1 \\ \sigma_{m,alt} & : \text{ sonst} \end{cases} \tag{13}$$

berechnet. Dabei sind allerdings zusätzlich die Parameter $\xi^{sp}_{\alpha_m}, \xi^{st}_{\alpha_m}, \xi^{sp}_{\sigma_m}$ und $\xi^{st}_{\sigma_m}$ empirisch zu bestimmen.

Als dritte Möglichkeit können die Parameter selbst als zu optimierende Variablen aufgefaßt werden. Jedem einzelnen Individuum i wird dazu ein Parameterpaar $\alpha_{m,i}$ und $\sigma_{m,i}$ zugeordnet, welche die Mutation dieses Individuums steuern. Sowohl die Strategieparameter als auch die Objektvariablen werden dann dem selben Optimierungsverfahren unterworfen. Die Optimierungsparameter werden in jeder Generation durch

$$\alpha^{neu}_{m,i} = \alpha^{alt}_{m,i} + n_i, \tag{14}$$

$$\sigma^{neu}_{m,i} = \sigma^{alt}_{m,i} + n_i, \tag{15}$$

mit n_i wie bei den Gleichungen (10) und (11) beschrieben, neu berechnet.

Nicht selten werden Kombinationen der beiden letzten beschriebenen Verfahren realisiert. Die Leistungsfähigkeit von ES hängt in einem sehr hohen Grad von der Entwicklung einer problemadaptierten Mutationsregel ab, da diese die Konvergenzgeschwindigkeit bestimmt und bei einer fehlerhaften Anpassung die Konvergenz verhindern kann.

3.2.2 Rekombination

Bei einigen Varianten der ES erfolgt die Bildung der Nachkommen nicht ausschließlich durch Mutation. Vielmehr erfolgt zusätzlich eine als Cross-over bezeichnete Rekombination mehrerer Elternindividuen, bevor die entstehenden Nachkommen mutiert werden. Dabei werden mehrere – im Regelfall zwei – Eltern zufällig ausgewählt und die Objektvariablenvektoren an beliebig vielen zufällig bestimmten – jedoch für alle Eltervektoren gleichen – Orten auseinandergeschnitten. Sodann werden die entstehenden Bruchstücke ausgetauscht. Die Rekombination ist dabei von zwei Parametern, den Strategieparametern α_r und σ_r abhängig. Der Rekombinationsgrenzwert α_r und die Varianz σ_r regelt, ob ein Individuum mittels Rekombination erzeugt wird. Ähnlich wie bei der Mutation wird eine Zufallszahl aus einer mittelwertfreien Gaußverteilung mit Varianz σ_r bestimmt und im Falle, daß diese kleiner als der Rekombinationsgrenzwert α_r ist eine Rekombination durchgeführt.

Findet eine Rekombination statt, so werden durch weitere Zufallsprozesse die Partnerindividuen und die Position des Crossover im Vektor der Objektvariablen bestimmt. Durch Wahl der Zufallsverteilung können dabei bestimmte Individuen oder Positionen in den Objektvariablen bei der Rekombination bevorzugt werden. Bild 5 zeigt die Rekombination von drei Elternvektoren und zwei Schnittpunkten, bei Bildung dreier Nachkommen.

Es existieren unterschiedliche Formen der Rekombination. Zum einen sind sowohl die Anzahl der beteiligten Eltern, als auch die Anzahl der Schnittpunkte zu be-

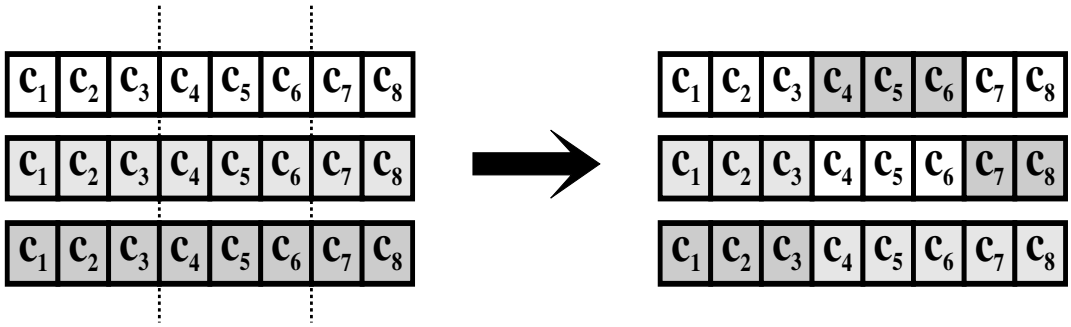

Bild 5: Rekombination dreier Eltervektoren mit zwei Schnittpunkten

stimmende Variablen. So unterscheidet man nach der Anzahl der Schnittpunkte zwischen einpunktiger, zweipunktiger oder im Fall, daß an allen möglichen Schnittstellen eines Objektvariablenvektors geschnitten wird, uniformer Rekombination. Andererseits sind auch verschiedene Arten der Kombination der Bruchstücke zu einem neuen Individuum möglich. So ist neben dem einfachen Fall des Austauschs von Bruchstücken die Rekombination durch eine Mittelwertbildung der einzelnen Komponenten der beteiligten Bruchstücke realisierbar.

Unabhängig von der algorithmischen Gestaltung ermöglicht die Rekombination den Informationsaustausch zwischen den Individuen.

3.2.3 Varianten der ES

Aus Kombinationen der unterschiedlichen Arten der Nachkommenerzeugung, im besonderen von verschiedenen Selektions- bzw. Ersetzungsschemata, der Wahl der Größe und Anzahl der Populationen sowie der Art des Informationsaustauschs ergeben sich Varianten der ES mit unterschiedlichem Verhalten und Aufwand der Berechnung bei der Optimierung. Um die Varianten bezeichnen zu können, führte Rechenberg [Rec92] nachfolgende Notation ein, welche die verschiedenen Varianten in groben Zügen charakterisiert.

- () Das runde Klammersymbol bezeichnet eine Population.
- + Das Pluszeichen gibt an, daß eine elitäre Auswahl aus der Schar der Elternindividuen und der Nachkommenindividuen erfolgt.
- , Das Komma gibt an, daß die Elternindividuen durch die Nachkommenindividuen ersetzt werden.
- # Dieses Zeichen dient als Platzhalter für das Plus und das Komma bei der Beschreibung.
- / Der Schrägstrich bedeutet entweder die Verwendung von Rekombination bei der Erzeugung von Nachkommen innerhalb einer Population, oder bei mehreren Populationen den Austausch ganzer Individuen zwischen den Populationen.
- μ Gibt die Anzahl der Elternindividuen an.
- λ Gibt die Anzahl der Nachkommenindividuen an.
- ρ Im Fall der Verwendung der Rekombinationen zur Nachkommenerzeugung gibt dieser Parameter die Anzahl der jeweils beteiligten Elternindividuen an.
- [] Das eckige Klammersymbol bezeichnet eine Schar von Populationen.
- θ Gibt die Anzahl der Populationen an.
- ξ Gibt die Anzahl der Generationen der isolierten Entwicklung an.

Die einfachste Variante ist eine $(\mu + \lambda)$-Strategie, bei der aus einer einzigen Population von μ Elternindividuen λ Nachkommenindividuen erzeugt werden, und die Selektion derart erfolgt, daß nur die besten Individuen beider Individuenscharen in die nächste Generation übernommen werden. Ein Problem bei der Anwendung dieser Variante besteht häufig in der Konvergenz in ein lokales Extremum. Dieses Verhalten

liegt darin begründet, daß ein Individuum, welches hinsichtlich der Qualitätsfunktion wesentlich besser ist als alle anderen Individuen, über mehrere Generationen „überlebt“. Unter der Voraussetzung einer kausalen Qualitätsfunktion hat das Individuum viele Nachkommen in den Folgegenerationen, so daß die Population in ein lokales Optimum gezwungen wird. Ein solches Verhalten kann bei einem (μ, λ)-Algorithmus nicht auftreten, da die gesamte Elternpopulation durch die μ-besten Nachkommen ersetzt wird, und somit kein Individuum länger als eine Generation „lebt“.

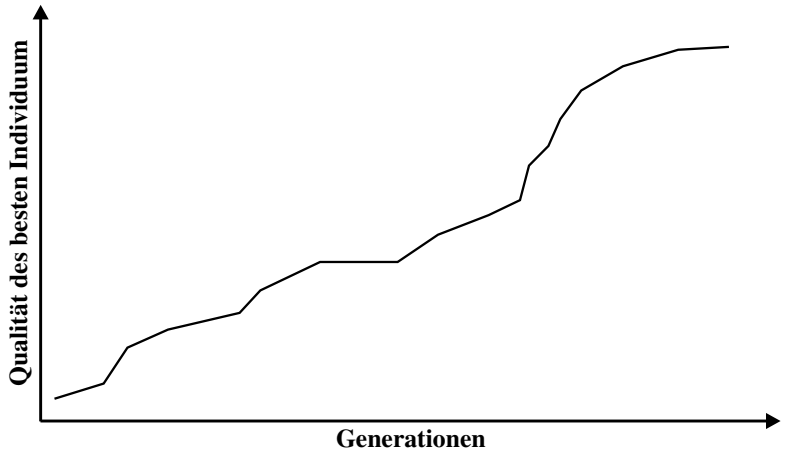

Bild 6: Zeitliches Verhalten der Qualitätsfunktion bei einer $(\mu + \lambda)$-Strategie

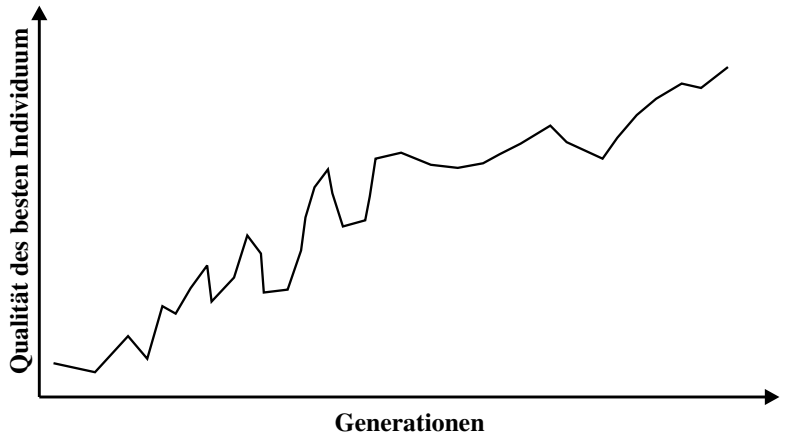

Bild 7: Zeitliches Verhalten der Qualitätsfunktion bei einer (μ, λ)-Strategie

Während die Qualitätsfunktion im Fall der $(\mu+\lambda)$-Variante monoton verläuft, kann sie bei der (μ,λ)-Strategie auch im Vergleich zu vorigen Werten kleinere Werte annehmne. Dies ist in den Bildern 3.2.3 und 3.2.3 schematisch dargestellt.

Durch die Einführung der Rekombination entsteht eine $(\mu/\rho\#\lambda)$-Strategie, bei der aus μ Eltern immer ρ Eltern ausgewählt werden, um einen der λ Nachkommen zu zeugen. Sodann wird nach dem durch # bestimmten Ersetzungsschema die nächste Generation gebildet. Der Selektionsdruck wird durch das Verhältnis von μ/λ bestimmt: Je kleiner das Verhältnis ist, desto stärker ist die Auslese. Bei großen Werten von μ/λ findet dagegen so gut wie keine Auslese statt.

Die in der Natur beobachtete temporäre Isolation von verschiedenen Populationen gleicher Art wird bei den ES in einer $[\theta,\xi,(\mu/\rho\#\lambda)]$-Strategie genutzt. Eine Anzahl θ einzelner Populationen wird darin getrennt voneinander über eine vorgegebene Anzahl von ξ Generationen entwickelt. Danach tauschen die Populationen Individuen aus indem entweder einzelne Individuen zwischen den Populationen getauscht werden, oder im Extremfall eine große gemeinsame Population gebildet, Nachkommen erzeugt und die Population wieder getrennt wird. Dadurch lassen sich analog zur Natur geschützte Subpopulationen simulieren, wie zum Beispiel Gruppen von Jungtieren, welche sich erst nach einer Entwicklungszeit dem uneingeschränkten Wettbewerb stellen müssen.

3.3 Genetische Algorithmen

Ein anderes Optimierungsverfahren, das auf der Grundlage der Evolution basiert, sind die GA. Diese wurden von John Holland in den 60er und 70er Jahren entwickelt [Hol75]. Der Pseudocode eines GA lautet wie folgt:

Wahl einer geeigneten Codierung des Problems
Initialisierung der Anfangspopulation
do

Bewertung aller Individuen einer Generation und Berechnung ihrer Fitness

Auswahl von Paaren/Gruppen von Individuen und Erzeugung von Nachkommen durch Rekombination nach Heiratsschema

Mutation der Nachkommen

Bildung einer neuen Generation durch Ersetzung von Individuen der aktuellen Generation durch die Nachkommen gemäß Ersetzungsschema

while
Abbruchkriterien nicht erfüllt

Bei den GA wird im Gegensatz zu den ES zuerst eine geeignete Form der Codierung zur Darstellung der problembeschreibenden Parameter gesucht. Dies erfolgt analog zur Natur, in der die verschiedenen Genome durch Basenpaarsequenzen codiert sind, weshalb dieser Ansatz auch als genotypisch orientiert bezeichnet wird. Im Unterschied zur Codierung mit vier Basenpaaren in der Natur, bevorzugt man aus Gründen der effizienteren numerischen Behandlung bei GA binäre Codierungen.

Nach der Wahl der Codierung findet eine Initialisierung der Anfangspopulation statt. Die Individuen der aktuellen Generation werden mittels einer Bewertungsfunktion, die die Güte des Individuums darstellt, bewertet. In einem zweiten Schritt wird aus diesem Wert die individuelle Fitness berechnet. Durch ein zufallsgesteuertes Auswahlverfahren, in das die Fitness der Individuen eingeht, werden Chromosomenvektoren zur Nachkommenerzeugung ausgewählt. Diese ausgewählten Chromo-

somenvektoren bilden durch Rekombination neue Nachkommen, wobei die Auswahl und die Rekombination durch ein Heiratsschema gesteuert wird. Anschließend wird aus den Nachkommen und den Elternchromosomen durch das Ersetzungsschema die neue Generation gebildet und der Vorgang, beginnend mit der Bewertung der Individuen der aktuellen Generation, bis zum Erreichen eines Abbruchkriteriums wiederholt durchlaufen.

3.3.1 Codierung

GA bieten durch die Codierung eine Möglichkeit auf diskrete Optimierungsprobleme angewendet zu werden. Unabhängig davon ist die Frage der Codierung bei GA von zentraler Bedeutung, da eine geeignete Wahl ein Problem unter Umständen gut lösbar machen kann. Jedoch entstehen durch die Codierung auch Probleme.

So existiert bei der Codierung von reellen Zahlen die Problematik, daß bei einer regulären binären Codierung in Mantisse und Exponent eine bitweise und damit kleine Veränderung im Chromosom starke Veränderungen im Phänotyp hervorrufen kann. Dies resultiert bereits bei Betrachtung von Exponent oder Mantisse aus der Tatsache, daß sich im Bitmuster sehr ähnliche Zahlen in der phänotypischen Ausprägung sehr unterscheiden können. Andererseits kann bei Betrachtung der gesamten binären Repräsentation eine Änderung an unterschiedlichen Stellen im binären Muster unterschiedlich starke Veränderungen in der Zahl zur Folge haben, da ein Eingriff in dem Teil der binären Sequenz, der die Mantisse codiert, geringere Auswirkungen zeigt, als eine Änderung im Bereich des Exponenten.

Zur Umgehung des erstgenannten Problems bietet sich die Verwendung eines Gray-Codes an, der den Vorteil hat, daß bei Änderung eines Bits des Bitmusters die resultierende Zahl in der Nachbarschaft der Ursprungszahl liegt. Das zweite Problem kann durch Verwendung von an die Darstellung in Mantisse und Exponent adaptierten Rekombinations- und Mutationsoperatoren gelöst werden.

Ein Vorteil der binären Codierung ist ihre in einem herkömmlichen Computer einfache Speicherung. Zusätzlich existieren in der Regel effiziente Methoden zur Ausführung von bitweisen Operationen, allerdings kann der Vorteil durch die Rückrechnung der in codierter Form vorliegenden Information bei der Gütebewertung teilweise oder auch völlig wieder kompensiert werden.

3.3.2 Heiratsschema

Anders als bei den ES, bei denen die Wahrscheinlichkeit eines Individuums zur Vermehrung gleichverteilt ist, ist die Wahrscheinlichkeit der Auswahl der Elternchromosomen bei den GA von der individuellen Fitness abhängig. Die individuelle Fitness wird aus dem Optimalitätsgrad des Individuums auf Basis der Bewertungsfunktion bestimmt.

Die Form der Auswahl wird durch das Heiratsschema beschrieben, wobei sehr viele spezielle Arten dieses Schemas genutzt werden. Eine einfache Form der Auswahl ist die Erzeugung einer Rangordnung auf Basis der Gütefunktion und der Zuordnung eines mit besserem Rang größer werdenden Subintervalls aus einem Intervall positiver reeller Zahlen. Je besser ein Individuum bewertet wird, um so größer ist das ihm zugeordnete Intervall. Dann wird eine Zufallszahl generiert, deren Wert innerhalb des Intervalls liegt, durch die das Individuum bestimmt wird, in dessen vorher zugeordneten Subintervall diese Zahl liegt.

Die Zuordnung des Ranges zu den disjunkten Subintervallen erfolgt durch die Fitnessfunktion. Mit Variation der Größe der Subintervalle kann die Auswahl der Individuen verändert werden. Diese Auswahl kann entweder konstant für die Dauer der Optimierung sein, oder die Intervallgröße kann direkt proportional dem Optimierungsgrad sein oder an andere Funktionsverläufe angepaßt werden, wobei eine gebräuchliche Funktion die Exponentialfunktion ist. Zielsetzung dabei ist, daß Individuen mit einer hohen Bewertung in den Folgegenerationen mehr Nachkommen

haben als solche mit nur durchschnittlicher Bewertung. Die Wahl des Heiratsschema hat Einfluß auf das Konvergenzverhalten des Optimierungsprozesses. Speziell in einem frühen Stadium der Optimierung führt eine zu starke Bevorzugung von Individuen mit einer guten Bewertung zu einer zu diesem Zeitpunkt unerwünschten Konzentration der Population auf ein Gebiet des Suchraums und damit zu einer vorzeitigen Konvergenz in ein lokales Optimum. Im Gegensatz dazu kann im Fall eines zu geringen Selektionsdrucks eine sehr geringe Konvergenzgeschwindigkeit oder das rein stochastische Absuchen des Suchraums die Folge sein.

Der Hauptmechanismus der Nachkommenserzeugung bei GA ist die Rekombination, wie sie bei ES beschrieben wurde. Die anschließende Mutation der Nachkommen besteht bei binären Chromosomenvektoren im allgemeinen in der zufallsgesteuerten Inversion von einzelnen Komponenten der Vektoren. Die Wahrscheinlichkeit einer Inversion ist dabei in der Regel sehr gering und führt, insbesondere bei sehr langen Chromosomenvektoren, nur ein Rauschen in den Suchprozeß ein. Beide Mechanismen, Rekombination und Mutation, werden oft in spezieller Weise an das bestehende Problem bzw. die Codierung angepaßt. Dies ist zum Beispiel bei der Standard-Codierung für reelle Zahlen der Fall, wenn die Größe der Veränderung von der Position innerhalb des Chromosoms abhängig ist.

3.3.3 Ersetzungsschema

Die Bildung der Folgegeneration wird durch das Ersetzungsschema, das auch Einfluß auf die Auswahl der Individuen zur Nachkommenerzeugung haben kann, geregelt. Die wichtigsten Varianten sind folgende :

1. **Komplette Ersetzung:**
   Ersetze die aktuelle Generation vollständig durch die Nachkommen.

2. **Elitismus:**
   Übernehme die n besten Chromosomenvektoren der aktuellen Generation in unveränderter Form in die nächste Generation.

3. **Schwacher Elitismus:**
   Übernehme die n besten Chromosomenvektoren der aktuellen Generation in mutierter Form in die nächste Generation.

4. **Letzte-Elimination:**
   Lösche die n schlechtesten der aktuellen Generation und bilde aus den verbleibenden Individuen n neue Individuen mittels Heiratsschema. Diese bilden zusammen mit den alten Individuen die neue Generation.

5. **Zufalls-Elimination:**
   Lösche n zufällig ausgewählte Individuen der aktuellen Generation und bilde aus den verbleibenden Individuen n neue Individuen mittels Heiratsschema.

6. **Zufalls-Elimination mit Elitismus:**
   Ersetze n zufällig ausgewählte Individuen der aktuellen Generation durch n Nachkommen, wobei die besten Individuen der aktuellen Generation stets übernommen werden.

7. **Tournament:**
   Mit gleicher Wahrscheinlichkeit werden eine Anzahl q von Individuen mit $1 < q$ aus der Gesamtpopulation ausgewählt und deren Güte bestimmt. Das beste Individuum wird in die nächste Generation übernommen und der Vorgang solange wiederholt, bis eine neue Generation generiert wurde.

Die verschiedenen Ersetzungsschemata haben unterschiedliche Auswirkungen auf das Konvergenzverhalten und sind wesentlich für die Optimierung. Bei der kompletten Ersetzung der aktuellen Generation durch die folgende kann die Bewertung,

sowohl des jeweils besten Individuums als auch die durchschnittliche der Population in der Folgegeneration schlechter werden, weshalb die Bewertungsfunktion in der Zeit nicht monoton steigend und daher die Konvergenzgeschwindigkeit geringer sein kann. Dem steht der Vorteil gegenüber, daß eine Dominanz von einigen guten Individuen zu Beginn der Optimierung durchbrochen wird, die unter Umständen zu einer vorzeitigen Konvergenz in einem lokalen Optimum führen könnte.

Um die Monotonie der Bewertungsfunktion zumindest für die besten Individuen zu erreichen, wird als Ersetzungsschema der Elitismus eingeführt. Die n besten Individuen werden in unveränderter Form in die nächste Generation übernommen. In der Regel werden allerdings nur sehr wenige Individuen übernommen. Um eine bei diesem Verfahren mögliche Dominanz eines „Superindividuums“ in der Vermehrung zu verhindern, kann der Elitismus derart abgeschwächt werden, daß die n besten Individuen nur in mutierter Form in die nächste Generation übernommen werden.

Die Variante bei der die schlechtesten n gelöscht werden hat die selben Vor- und Nachteile wie der Elitismus, konzentriert jedoch in der Regel, abhängig von der Mutationsrate, die Individuen schnell auf einen Teil des Suchraums. Das Eliminieren der schlechtesten Individuen ist zusätzlich oft von Nachteil, da diese auch in Teilen Informationen enthalten können, die zum Lösen eines Problems wichtig sind, die aber nur in Kombination mit anderen Teilen zu einer guten Bewertung führen. Diese Erkenntnis führte zur Entwicklung der beiden letzten Ersetzungsschemata. Zur Kombination der verschiedenen (Sub-)Populationen miteinander existieren auch bei GA im Prinzip dieselben Verfahren wie bei ES, um auch hier Effekte natürlicher Evolution nutzbringend implementieren zu können.

3.3.4 Schemata-Theorem

Im Rahmen der Konvergenzanalyse von GA wurde der Begriff des Schema [Hol75] eingeführt. Ein Schema ist ein Chromosom, das die Information in binär codierter

Form repräsentiert, jedoch an einer oder mehreren Stellen eine Variable enthält. Als Symbol wird das Zeichen # eingeführt und entspricht entweder einer „1“ oder „0“, so daß demnach ein Schema der Länge l ein Element aus $\{0, 1, \#\}^l$ ist. Der Gesamtraum M^l aller möglichen Chromosomen enthält 3^l Unterräume aller möglichen Schemata. Als Instanz eines Schema $\underline{h} \in \{0, 1, \#\}^l$ wird ein Chromosom $\underline{c} =< c_1, c_2, \ldots, c_l > \in M^l$ bezeichnet, wenn alle Komponenten des Instanzen-Chromosoms die ungleich #, gleich denen des Schema-Chromosoms sind. Jedes Element aus dem M^l stellt eine Instanz von 2^l Schemata dar. Folglich repräsentiert eine Population mit p Individuen zwischen 2^l und $p * 2^l$ Instanzen. Es existieren 2^l Instanzen wenn alle Chromosomen gleich sind und $p * 2^l$ Instanzen falls alle Chromosomen unterschiedlich sind. Damit wird pro Generation eine große Zahl von Unterräumen gleichzeitig abgesucht und bewertet [Gol89]. Dieser Prozeß wird als implizite Parallelität der GA bezeichnet.

Die Weitervererbung eines Schema unterliegt den restriktiven Kräften der genetischen Operatoren, der Selektion, Rekombination und Mutation. Durch die Anwendung dieser genetischen Operatoren wird ein Schema mit einer gewissen Wahrscheinlichkeit zerstört. Wenn ein Schema eine besondere Teillösung eines Problems enthält, so ist es unerwünscht wenn dieses Schema zerstört wird. Das Schema-Theorem [Hol75] gibt eine Antwort auf die Frage, welche „Überlebenschance“ ein Schema hat und lautet wie folgt:

$$m(h, t+1) \geq m(h, t) \frac{f(h)}{< f >} (1 - p_r \frac{d(h)}{l-1} - p_m o(h)). \tag{16}$$

Es sind $m(l, t)$ die Instanzen eines Schema h innerhalb einer Population von Chromosomenvektoren, $f(h)$ die durchschnittliche Fitness aller Instanzen von h und $< f >$ die durchschnittliche Fitness der gesamten Population zum Zeitpunkt t. Desweiteren sind p_r und p_m die jeweils auf ein Chromosom bezogene Rekombinations- bzw. Mutationswahrscheinlichkeit. Die Ordnung $o(h)$ eines Schema h ist die Anzahl der Positionen, die nicht das Variablensymbol # enthalten. Der Durchmesser oder die definierende Länge $d(h)$ eines Schema h ist der Abstand $k - i$ zwischen der ersten c_i und der letzten c_k von der Variable # verschiedenen Positionen in h.

Das Schema-Theorem läßt sich wie folgt erklären. Sind die fixen Positionen eines Schema für die überdurchschnittliche Fitness eines Chromosoms verantwortlich, so wird dieses Schema bei einem Heiratsschema, welches die Individuen proportional zu ihrer Fitness bestimmt, entsprechend häufiger ausgewählt. Damit hängt die Häufigkeit des Auftretens davon ab, um wieviel die Fitness der Instanzen des Schema die durchschnittliche Fitness übertrifft. Jedoch können die Schemata durch Rekombination und Mutation zerstört werden. Dabei spielt die Ordnung und der Durchmesser des Schema eine Rolle, denn je länger ein Schema ist, desto größer ist die Wahrscheinlichkeit, daß es durch Rekombination „zerbrochen“, oder von Mutation betroffen ist.

Das Schema-Theorem macht eine Aussage darüber, welche Eigenschaften eine effektive Codierung haben muß. Es besagt, daß überdurchschnittliche Schemata von kurzer definierender Länge und niedriger Ordnung in den Folgegenerationen häufiger ausgewählt werden als solche, die diese Eigenschaften nicht besitzen. Selbst wenn bei der Rekombination und Mutation Information zerstört wird, behält der GA dieses Verhalten bei. Daraus läßt sich ableiten, daß eine Codierung inhaltlich zusammengehöriger Information nicht getrennt auf dem Chromosom liegen sollten und die Codierung möglichst kurz sein sollte.

Aus dem Schema-Theorem wird die Building-Block-Hypothese abgeleitet, welche besagt, daß die GA aus Schemata niedriger Ordnung, kleinem Durchmesser und überdurchschnittlicher Fitness, den sogenannten Building-Blocks, die Lösung sukzessive aufbauen. Diese Hypothese erklärt jedoch nicht das Lösungsverhalten von GA bei Problemen, die nur durch Kombination von Schemata mit geringer Fitness gelöst werden können. Daher läßt sie sich nicht verallgemeinern und ist nur bedingt als Erklärungsmodell tauglich. Jedoch kann man daraus ableiten, daß ein GA Schwierigkeiten bei der Lösung von Problemen haben wird, bei der zum einen die Rekombination von kompakten Schemata mit sehr geringer Fitness zu Chromosomen mit hoher Fitness führt und zum anderen, die Rekombination von Schemata

mit hoher Fitness zu solchen mit geringer Fitness führt.

3.4 Vergleich zwischen ES und GA

Beiden Verfahren ist zunächst einmal gemein, daß sie zu Beginn eines Optimierungsprozesses mit einer breiten Verteilung im Suchraum starten und sich zum Abschluß der Optimierung auf einen Ausschnitt des Lösungsraums konzentrieren. Es existieren für beide Arten von EA allerdings mittlerweile eine nahezu unüberschaubare Anzahl von Varianten, die auch die Grenze zwischen beiden Algorithmen fließend werden lassen. So werden beispielsweise bei den GA reelle anstatt binärer Vektoren bei bestimmten Problemstellungen verwandt.

Auch eine allgemeingültige Antwort auf die Frage welcher der Algorithmen in einem speziellen Fall besser ist, kann nicht gegeben werden. Jeder der beiden Algorithmen funktioniert je nach Aufgabenstellung [HB92] unterschiedlich gut.

Allgemein läßt sich nur sagen, daß beide Algorithmen eine Lösung nur schwer finden, wenn die Landschaft der Bewertungs-, bzw. Qualitätsfunktion in der Nachbarschaft eines sehr ausgeprägten Optimums sehr flach ist, d.h. daß die Fehlerfunktion sich kaum oder überhaupt nicht ändert. Dort hat der Algorithmus keine Anhaltspunkte, in welche Richtung er die Population verschieben soll, wobei jedoch auch die meisten klassischen Verfahren hierbei kein Lösung finden. Jedoch haben die evolutionären Algorithmen, aufgrund der stochastischen Komponente, die Möglichkeit das Optimum zufällig zu finden.

Neben der Wahl des geeigneten Verfahren ist die Wahl der einzelnen Strategieparameter der Optimierung stark problemabhängig und es existieren keine generell gültigen Vorschriften wie diese einzustellen sind, um eine gute Konvergenz zu gewährleisten. Für beide Verfahren existieren eine Reihe von Heuristiken bezüglich

dieser Problematik. So ist ein häufig gewähltes Verfahren die Einbeziehung der Steuerparameter in den Optimierungsprozeß. Jedoch beeinflußt auch die Anforderungen die an die Genauigkeit einer Lösung gestellt werden die Parameterwahl. So steigt in der Regel neben der Genauigkeit auch die Rechenzeit beträchtlich. Insgesamt ist es jedoch die Feinabstimmung des, Algorithmus die einen beträchtlichen Aufwand mit sich bringt.

Zum aktuellen Stand des theoretischen Verständnisse sei folgendes zitiert ([B+97], Seite 11):

> *In contrast, the theoretical foundations are to some extent still weak. To say it more pithy: We know that they work, but we don't know why. As a consequence, inexperienced user fall into the same traps repeatedly, since there are only few rules of thumb for design and parametrization of evolutionary algorithms.*

4 Simulationsexperimente mit Zufallssignalen und einer deterministischen Funktion

4.1 Lernen am Beispiel der logistischen Abbildung

Um die prinzipielle Leistungsfähigkeit sowohl der GA, als auch der ES zu testen, wurden neuronale Netze anhand verschiedener Testbeispiele mittels dieser Algorithmen trainiert. Zur Untersuchung des Verhaltens der Algorithmen ist zunächst eine Prädiktionsaufgabe ohne explizite Zeitabhängigkeit gewählt worden.

Die Aufgabe des neuronalen Netzes im Testbeispiel der logistischen Abbildung ist die Approximation der quadratischen Abbildung

$$x_{t+1} = r x_t (1 - x_t). \tag{17}$$

Die logistische Abbildung beschreibt rekursiv das Wachstum von Populationen bei vollständigem Generationenwechsel, wobei der Faktor r die Vermehrungsrate darstellt und x die auf den Wert eins normierte Population ist. Für bestimmte Werte von r weist die logistische Abbildung ein chaotisches Verhalten auf [KS86]. Die Test- und Trainingsmenge enthielt jeweils $P = 1000$ Muster. Erzeugt wurden die Mengen durch 2000 Iterationen von Gleichung (17), wobei der Parameter r zu 3.9 und als Initialwert x_0 ein Wert von 0.9 festgesetzt wurde. Mit dem so gewählten Wert für r weist die Funktion kein chaotisches, sondern ein quasiperiodisches Verhalten auf. Das Bild 8 zeigt einen Ausschnitt dieser Abbildung.

Als neuronales Netz wurde ein mehrschichtiges Perzeptron mit einer verborgenen Schicht bestehend aus $N = 10$ Neuronen, sowie einem Eingabe- und einem Ausgabeneuron eingesetzt. Als Aktivierungsfunktion der verborgenen und der Ausgabeneuronen fand die Fermi-Funktion Verwendung. Zum Training des Netzes wurde das Muster x_t am Eingang und die Sollausgabe x_{t+1} am Ausgabeneuron präsentiert.

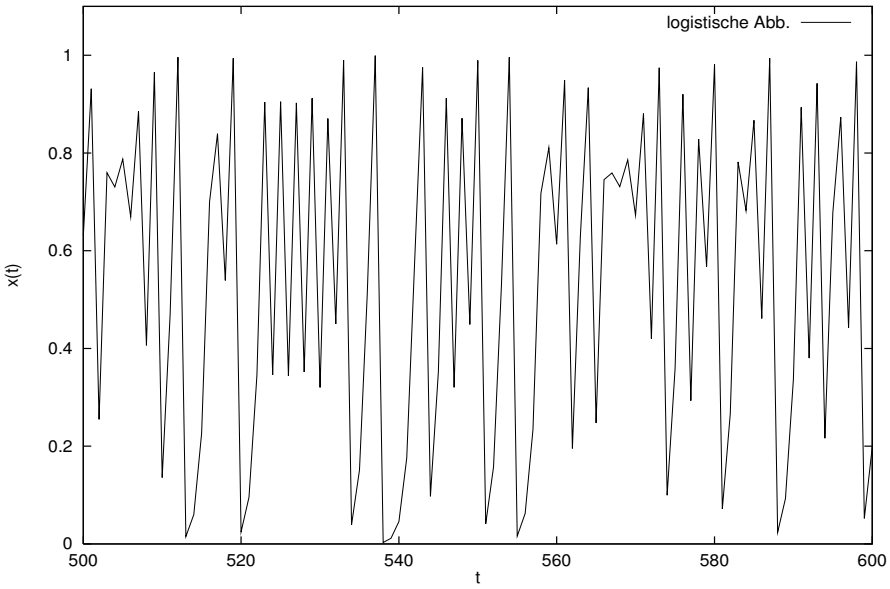

Bild 8: Logistische Abbildung

Der mittlere quadratische Fehler pro Muster 18 stellt die Qualitäts- bzw. Bewertungsfunktion dar.

$$MSE = \frac{1}{P}\sum_{t=1}^{P}(x_t - x_t^{\text{soll}})^2 \tag{18}$$

4.1.1 Logistische Abbildung mit GA

Bei den GA findet durch die Codierung der Gewichtsvektoren eine Quantisierung des Suchraumes statt. Dadurch wird der Suchraum bei diesen Algorithmen absichtlich verkleinert. Das Intervall $[-g, g]$ wurde mit nach außen abnehmender Genauigkeit quantisiert. Dies bewirkt eine Annäherung der Häufigkeitsverteilung an eine Gaußglocke, da diese Verteilung oft bei trainierten Netzen zu beobachten ist. In Vortests wurde als Lösungsraum das Intervall $[-5.0, 5.0]$ bestimmt.

Als Codierungsverfahren fand ein Codebuch der quantisierten Gewichte Verwendung, dessen Indizes durch einen Gray-Code nachbarschaftserhaltend angeordnet

wurden. Diese Art der Codierung vermeidet, daß Ortsabhängigkeiten innerhalb der Chromosomen bei der Mutation oder der Rekombination berücksichtigt werden müssen. Die Gewichte der Ausgabeschicht und der verborgenen Schicht wurden als zwei getrennte, voneinander unabhängige Chromosomen betrachtet und verarbeitet. Als Ersetzungsschema wurde einmal *Zufalls-Elimination* und zum anderen *Letzte-Elimination* mit $n = 10$ gewählt. Zusätzlich fand ein Test der beiden Varianten mit und ohne Elitismus statt.

Vortests zeigten, daß eine uniforme Rekombination nicht oder nur sehr langsam zu einer Konvergenz führt. Daher wurde die Untersuchung auf die Rekombination an einer Stelle des Chromosomenstranges beschränkt. Der Mutationsoperator wurde durch eine punktuelle Bitinversion pro Chromosom realisiert, mit einer Mutationswahrscheinlichkeit von einer Inversion auf 10^3 Bits.

Die Population bestand bei allen Tests aus 50 Individuen, da mit kleineren Populationsgrößen in Vortests keine Konvergenz zu erreichen war. Die Fitness eines Individuums wurde proportional zur Bewertung gewählt und nach Erreichen von 1000 Generationen wurde das Training abgebrochen. Das Kriterium für die Lösung des Problems bestand darin, daß der MSE kleiner 0.1 auf der Testmenge war.

Tabelle 1: MSE der logistischen Abbildung, GA mit Elitismus

Ersetzungsschema	Quant.-Stufen	Generationen	MSE
Letzte-Elimination	2^8	1000	0.14
Zufalls-Elimination	2^8	1000	0.17
Letzte-Elimination	2^{12}	780	0.065
Zufalls-Elimination	2^{12}	840	0.071
Letzte-Elimination	2^{16}	810	0.043
Zufalls-Elimination	2^{16}	920	0.057

Tabelle 2: MSE der logistischen Abbildung, GA ohne Elitismus

Ersetzungsschema	Quant.-Stufen	Generationen	MSE
Letzte-Elimination	2^8	1000	0.16
Zufalls-Elimination	2^8	1000	0.15
Letzte-Elimination	2^{12}	800	0.085
Zufalls-Elimination	2^{12}	820	0.092
Letzte-Elimination	2^{16}	850	0.080
Zufalls-Elimination	2^{16}	950	0.079

In den Tabellen 1 und 2 sind die Ergebnisse der einzelnen Tests, MSE pro Muster und Anzahl der benötigten Generationen, geordnet nach Ersetzungsschema und Anzahl der Quantisierungsstufen dargestellt. Tabelle 1 enthält die Ergebnisse für die GA mit Elitismus und Tabelle 2 diese für die GA ohne Elitismus. Wie in den Tabellen 1 und 2 zu sehen, gelingt es bei einer geringen Anzahl von Quantisierungsstufen nicht, eine Lösung zu finden. In den Bildern 9 und 10 sind sowohl die Ist-, als auch die Soll-Ausgaben dargestellt, indem die Ausgabewerte $x(t+1)$ und die Eingabewerte $x(t)$ gegeneinander aufgetragen sind. Bild 9 zeigt dies für das beste Netz mit 2^{16} Quantisierungsstufen und Bild 10 für 2^{12} Stufen.

Bei zunehmender Anzahl der Quantisierungstufen findet der Algorithmus eine Lösung, jedoch nimmt neben der Approximationsgenauigkeit, zugleich die Anzahl der benötigten Generationen zu. Bei Verwendung zu weniger Quantisierungsstufen kann das Lösungsverhalten qualitativ wie folgt beschrieben werden. Nach einer raschen Abnahme des quadratischen Fehlers bis auf einen bestimmten Fehlerwert war die weitere Verbesserung nur noch marginal. Dabei beschrieb der Fehler der veränderbaren Individuen eine Punktwolke um den besten Fehlerwert, mit einer starken Ausprägung zu schlechteren Fehlerwerten hin. Im Gegensatz dazu wurde der Fehler des jeweils besten Individuums im Experiment mit einer höheren Anzahl

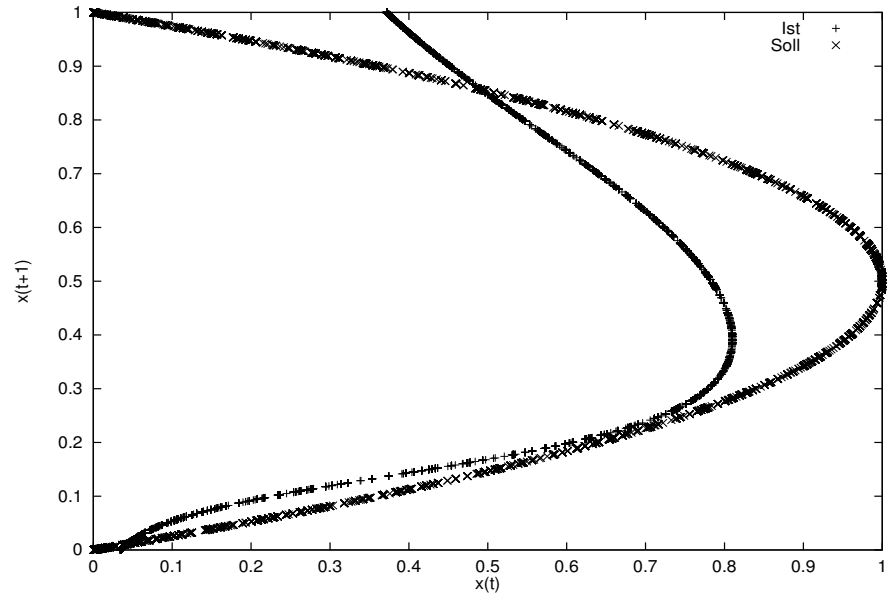

Bild 9: Logistische Abbildung, 2^{16} Quant.-Stufen, $MSE = 0.045$

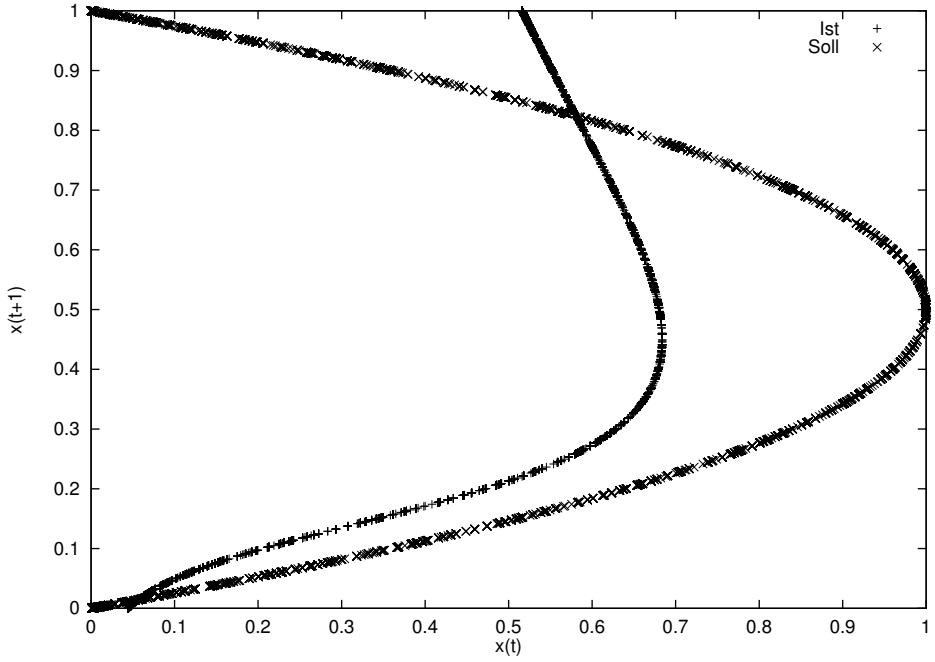

Bild 10: Logistische Abbildung, 2^{12} Quant.-Stufen,$MSE = 0.065$

von Quantisierungsstufen nahezu kontinuierlich kleiner, wobei dieser Effekt zunahm, je höher die Stufenanzahl war. Dieses Verhalten des GA bei einer zu geringen Anzahl der Quantisierungsstufen resultiert aus der Tatsache, daß der Algorithmus nicht mehr Informationen über eine Entwicklungsrichtung im Suchraum gewinnen kann, da die Änderungen der Bewertungsfunktion zu groß waren. Außerdem gleichen sich die Individuen einer Population im Laufe der Optimierung immer mehr an, d.h. sie werden sich immer ähnlicher. Die Bevorzugung von Individuen mit besserer Bewertung bewirkt auf diese Weise, daß die anfängliche Verteilung im Suchraum zunehmend schmaler wird und die Suche leichter in einem lokalen Minimum enden kann. Verstärkung erfährt dieser Effekt zusätzlich durch die geringe Anzahl der Quantisierungsstufen, weil damit die Anzahl der möglichen Individuen, die durch Anwendung der genetischen Operatoren aus einem Individuum entwickelbar sind, eingeschränkt ist. Auch die Verwendung einer Variante des GA mit höherer Mutationsrate führte nicht zu einer Lösung. Mit wachsender Anzahl der Quantisierungsstufen ist der Algorithmus in der Lage, auch notwendige kleine Schritte auszuführen, die sukzessive zu einer Lösung führen.

Für die Lösbarkeit und Genauigkeit der Approximation zeigten die unterschiedlichen Varianten des Algorithmus bei gleicher Anzahl der Quantisierungsstufen kaum einen Unterschied. Allerdings differierten Konvergenzgeschwindigkeit und Konvergenzverhalten. Die größere Anzahl der Quantisierungsstufen führte einerseits zu einer genaueren Approximation der logistischen Abbildung mit kleinerem Fehler, andererseits war die Konvergenzgeschwindigkeit durch die geringere Schrittweite der genetischen Operatoren geringer. Wie aus den Tabellen 1 und 2 ersichtlich ist, zeigen die Varianten des GA, die die Vermehrung der Individuen mit einer höheren Bewertung bevorzugen, d.h. jene mit Ersetzungsschema *Letzte-Elimination* und Elitismus, eine in der Regel höhere Konvergenzgeschwindigkeit.

4.1.2 Logistische Abbildung mit ES

Eine ES-Variante mit der das neuronale Netzwerk trainiert wurde, war eine (25/2,25)-Strategie mit und ohne Elitismus. Die Mutationrate war global für alle Individuen gleich durch den Mutationsgrenzwert $\alpha_m = 1.25$ und die Varianz $\sigma_m = 0.5$ festgelegt. Die Häufigkeit für eine Rekombination wurde durch den Rekombinationsgrenzwert $\alpha_r = 0.5$ und die Varianz von $\sigma_r = 1.0$ bestimmt. Neben diesen Varianten wurde ein Algorithmus getestet, der eine Mischform zwischen ES und GA darstellt. Dabei wurde ein Ersetzungsschema mit Fitnessfunktion anstatt der Bewertungsfunktion der ES benutzt. Das Ersetzungsschema selbst war *Letzte-Elimination* mit $n = 10$.

Auf diese Art sollte getestet werden, ob eine Bevorzugung der besten Individuen auch bei diesem Algorithmus die Konvergenz beschleunigt. Wie bei GA wurde nach Erreichen der maximalen Anzahl von 1000 Generationen das Training abgebrochen. Das Problem galt als erfolgreich gelöst, wenn der $MSE \leq 0.1$ auf der Testmenge war.

Tabelle 3: MSE der logistischen Abbildung bei Gewichtstraining mit ES unter Verwendung verschiedener Ersetzungsschemata

Ersetzungsschema	Elitismus	Generationen	MSE
Standard	Ja	770	0.04
Standard	Nein	940	0.045
Letzte-Elimination	Ja	600	0.03
Letzte-Elimination	Nein	700	0.047

In Tabelle 3 ist der MSE pro Muster, geordnet nach dem verwendeten Ersetzungsschema, Elitismus und der Generationenanzahl, dargestellt. Wie der Tabelle zu entnehmen ist, konvergieren solche Strategien schneller, bei denen die Individuen mit

besserer Bewertung bevorzugt werden. Insbesondere die Variante *Letzte-Elimination* mit Elitismus zeigt bei vergleichbarem Fehler eine deutlich höhere Konvergenzgeschwindigkeit als alle anderen Varianten.

In einem separaten Test wurden Strategien zur Optimierung der Mutationsparameter und Einflüsse der Populationsgröße untersucht, wobei die beste ES-Variante aus dem vorangegangenen Test verwandt wurde. Im Falle der größeren Population mit 100 Individuen war die Anzahl der zu löschenden Elemente $n = 35$. Einmal wurden global für alle Individuen in Vorversuchen optimierten Parameter der Mutation während des Trainings schrittweise auf ihren halben Ausgangswert am Ende des Trainingszyklus verringert. Diese Vorgehensweise wurde gewählt, um zu prüfen, ob der Algorithmus durch Reduktion der Mutation und damit der Schrittweite, leichter ein Optimum erreicht. Vergleichen läßt sich diese Vorgehensweise mit der Verringerung der Lernschrittweite η bei Backpropagation-Algorithmen. Zum anderen wurden die Parameter selbst als zu optimierende Variablen aufgefaßt und jedem Individuum ein eigenes Parameterpaar zugeordnet.

Tabelle 4: MSE der logistischen Abbildung bei Optimierung der Mutationsparameter

Mutations-Optimierung	Individuen	Generationen	MSE
Global	25	600	0.03
Global	100	550	0.016
Global mit Reduktion	25	480	0.014
Global mit Reduktion	100	570	0.01
Individuell	25	1000	0.14
Individuell	100	720	0.005

Tabelle 4 zeigt den Fehler pro Muster und die benötigte Generationenzahl geordnet nach der verwendeten Mutationsart und der Anzahl der Individuen. Die Ergebnis-

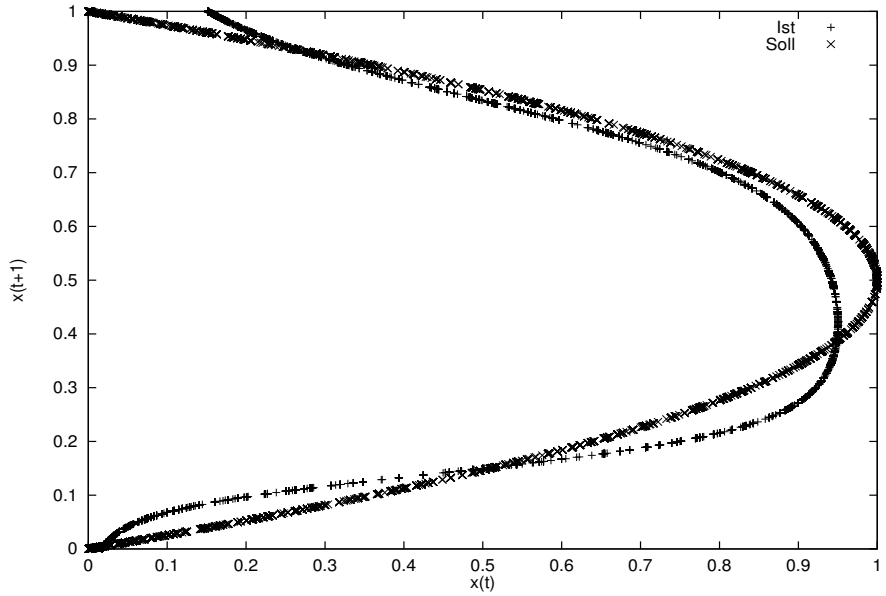

Bild 11: Logistische Abbildung, $MSE = 0.005$

se zeigen zum einen, daß der Aufwand, gemessen durch die Anzahl der getesteten Individuen als Produkt aus Populationsgröße und Zahl der durchlaufenen Generationen, mit der Populationsgröße stark ansteigt. Andererseits steigt die Genauigkeit der Approximation der Abbildung, wie in Bild 11 zu sehen ist, so daß die Wahl der Populationsgröße maßgeblich von der gewünschten Qualität der Lösung abhängt.

4.1.3 Bewertung und Vergleich

Insgesamt zeigt sich, daß ES eine höhere Konvergenzgeschwindigkeit aufweisen als GA. Zum einen profitieren sie von der direkten Verwendung der reellen Zahlen, auf die die genetischen Operatoren angewendet werden. Zum anderen bietet offensichtlich die Verwendung der Mutation als treibender Optimierungsoperator den Vorteil, den Lösungsraum effizienter absuchen zu können. Die Populationsgröße hat zunächst die Eigenschaft, die Konvergenzgeschwindigkeit zu reduzieren. Allerdings ist bei größeren Populationen die Parameterwahl dann weniger empfindlich. Dies erleichtert die Bestimmung derselben, speziell wenn man die Optimierung der Pa-

rameter automatisiert in den eigentlichen Optimierungsprozeß integriert.

Desweiteren hat die Populationsgröße Einfluß auf die Genauigkeit der Lösung, da Varianten mit geringeren Populationsgrößen vorzeitig in ein lokales Optimum konvergieren können. Dies kann speziell bei Problemstellungen die eine exakte Problemlösung verlangen – beispielsweise bei solchen, deren Fehlerlandschaft in der Umgebung eines Optimum sehr steil ist – bedeuten, daß eine Lösung nur dann gefunden wird, wenn die Populationsgröße einen bestimmten Wert überschreitet. Neben der geringeren Populationsgröße und der für gute Konvergenz benötigten höheren Generationenanzahl sind die GA auch bezüglich der reinen Rechengeschwindigkeit langsamer, da trotz der Mutation auf binärer Ebene schon allein die Decodierung zusätzliche Rechenzeit benötigt. In den weiteren Tests wurden daher nur ES betrachtet.

Aus den vorgestellten Ergebnissen sollte nicht der falsche Eindruck entstehen, daß eine Lösung mit einer relativ geringen Populationsgröße (≤ 100), also einem niedrigen Rechen- und Zeitaufwand zu finden ist. Bei kleinen Populationen ist die Parameteroptimierung sehr aufwendig, da diese exakt eingestellt werden müssen, um einerseits eine vorzeitige Konvergenz und andererseits eine ungerichtete, zufällige Suche zu verhindern. Die Verteilung der Population wird im ersten Fall zu schmal, d.h. die Population kontrahiert und konzentriert dadurch die Suche in einem lokalen Optimum. Im zweiten Fall werden lediglich zufällige Ergebnisse erzielt und es erfolgt keine zielgerichtete Absuche optimumverdächtiger Gebiete des Suchraumes. Daher sind die Ergebnisse kleinerer Netze wesentlich von einer guten Initialisierung, d.h. von guten Startpunkten und einer guten Verteilung der Population im Suchraum abhängig.

4.2 Latching-Problem

Ein überschaubares Problem zur Untersuchung der Frage, ob ein Algorithmus in der Lage ist lange Zeitabhängigkeiten zu erkennen und zu trainieren bzw. Informationen über lange Zeitdauern zu speichern,ist das Latching-Problem [B$^+$94]. Das verwendete neuronale Netz, in Bild 12 dargestellt, besteht aus einem einzigen Ausgabeneuron $y(t)$ mit einer rekurrenten Verbindung und drei Eingabeneuronen $e_1(t), e_2(t)$ und $e_r(t)$. Es ist ein Klassifikationsproblem mit zwei Klassen, das in der modifizierten Variante von [G$^+$96] wie folgt definiert ist.

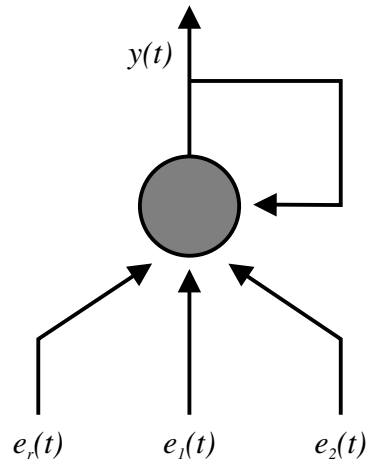

Bild 12: Netz des Latching-Problems

Beide Eingabeneuronen $e_1(t)$ und $e_2(t)$ sind Null für alle Zeiten $1 < t \leq T$. Zum Zeitpunkt $t = 1$ sind für Muster der Klasse 1 $e_1(t) = 1$, $e_2(t) = 0$ und für Muster der Klasse 2 $e_1(t) = 0$ und $e_2(t) = 1$. Das Eingabeneuron $e_r(t)$ injiziert ab dem Zeitpunkt $t > t_L$ ein gleichverteiltes Rauschen $U(-b, b)$ aus dem Intervall $[-b, b]$ mit $b = 0.155$:

$$e_r(t) = \begin{cases} 0 & : \quad t \leq t_L \\ U(-b,b) & : \quad t_L < t \leq T \end{cases} . \tag{19}$$

Insbesondere haben zur Zeit $t = \{2, 3\}$ alle Eingabeneuronen die Aktivität Null. Die Transferfunktion des Ausgabeneurons ist der Tangens-Hyperbolicus und das Gewicht $w_r = 1.25$ der rekurrenten Verbindung ist so eingestellt, daß das Netz zwei

stabile Fixpunkte um ± 0.710 und einen instabilen Fixpunkt bei Null hat. Dieses Gewicht ist konstant während des gesamten Trainings, ebenso das Gewicht von dem Eingabeneuron $e_r(t)$ das konstant $w_n = 1.0$ ist. Das heißt, daß nur die Gewichte $w_{1,2}$ von den beiden Eingabeneuronen $e_1(t)$ und $e_2(t)$ trainierbar sind, wobei die Initialwerte der Gewichte aus der selben Verteilung $U(-b, b)$ wie das Rauschen stammen. Alle Muster haben die gleiche Länge T und das Rauschen beginnt zum Zeitpunkt $t_L = 3$. Der Fehler für die anliegenden Muster wird nur am Schluß einer Mustersequenz, d.h. zum Zeitpunkt $t = T$ berechnet. Als Sollausgabe für die Klasse 1 wird $a_{soll,1}(T) = 0.8$ und für die Klasse 2 $a_{soll,2}(T) = -0.8$ gewählt. Ein Muster gilt als richtig klassifiziert, wenn die quadratische Abweichung $F(T) \leq 0.2$ ist.

Befindet sich das Netz in einem der Fixpunkte und wird im weiteren lediglich Rauschen aus dem angegebenen Intervall angelegt, so verbleibt es im angenommenen Fixpunkt. Der Trainingsalgorithmus muß daher die Gewichte $w_{1,2}$ derart verändern, daß das Netz bei Anlegen eines Musters zum Zeitpunkt $t = 1$, in einen der entsprechenden Musterklasse zugeordneten Fixpunkt getrieben wird, den es dann während der gesamten Mustersequenz nicht mehr verläßt. Dies ist mit Sicherheit dann der Fall, wenn die Gewichte $w_1 \geq 0.6$ und $w_2 \leq -0.6$ sind. Hat der Algorithmus eine derartige Lösung gefunden, so kann das Netz Muster mit beliebiger Sequenzlänge T korrekt klassifizieren, da die Einstellungen des Netzes und der Muster so gewählt sind, daß das Netz nicht mehr aus einem eingenommenen, stabilen Zustand herauskommt. Also fällt die Fehlerebene in der direkten Umgebung des Optimums steil ab und das Optimum ist eine ausgedehnte, ebene Hyperfläche. Damit ist dieses Problem prinzipiell sehr leicht zu lösen, speziell für einen Optimierungsalgorithmus der stochastisch basiert ist.

Für die vergleichenden Messungen wurden jeweils eine Trainings- und eine Testmenge bestehend aus 30 Mustern pro Klasse generiert, wobei die Mustersequenzlänge T von 20–200 in Zweierschritten zunehmend vergrößert wurde. Eine Messung wurde dann als erfolgreich gewertet, wenn spätestens nach Erreichen einer festgelegten

Anzahl von Trainingsschritten die Erkennungsrate auf der Testmenge 100 Prozent war.

Das Netzwerk wurde zum einen mittels BPTT trainiert, mit einem Wert für η von 0.1 und einer maximalen Anzahl von 200 Epochen, wobei im Regelfall die Konvergenz innerhalb einiger Dutzend Epochen erfolgte. Jeder Meßpunkt mit unterschiedlichem T wurde zur Vermeidung von Abhängigkeiten von der Anfangsinitialisierung, 100 mal gemessen. Zum anderen wurde das Netzwerk durch eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ trainiert. Der Mutationgrenzwert war $\alpha_m = 1.5$ und die Varianz $\sigma_m = 0.05$. Der Rekombinationgrenzwert war $\alpha_r = 0.5$ und die Varianz $\sigma_r = 1.0$. Das beste Netz wurde unverändert in die nächste Generation übernommen. Speziell die Mutationsparameter wurden klein gewählt, um zu verhindern, daß die Lösung durch einen rein stochastischen Suchvorgang gefunden wird. Aufgrund der Parameterwahl wurde die maximale Anzahl der Generationen auf 300 begrenzt und aus Gründen des höheren Rechenaufwands wurde für jeden Meßpunkt nur 10 mal statt wie beim BPTT 100 mal gemessen.

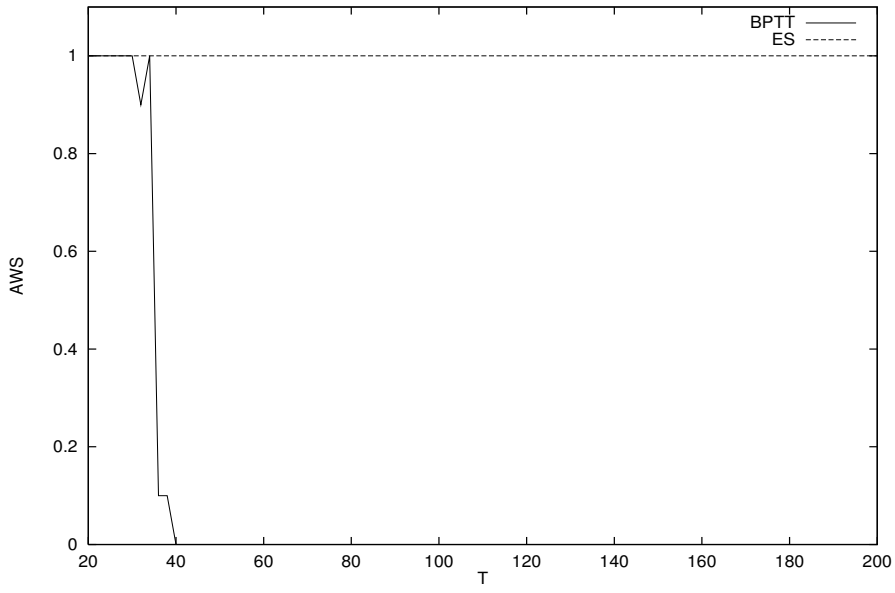

Bild 13: AWS in Abhängigkeit von der Sequenzlänge T

Das Konvergenzverhalten ist bei beiden Verfahren sehr ähnlich. Nach geringer sukzessiver Abnahme des Fehlers, wird innerhalb weniger Trainingsschritte der Fehler sehr klein und die Erkennungsrate hoch. In Bild 13 ist der Anteil der erfolgreichen Simulationen (AWS), normiert auf die Gesamtzahl der Simulationen, pro Meßpunkt T aufgetragen, sowohl für BPTT als auch für ES. Es zeigt sich, daß die mit dem BPTT-Algorithmus trainierten Netzwerke nicht in der Lage sind, Mustersequenzen, die länger als 36 Zeittakte sind, zu verarbeiten Im Gegensatz dazu ist es mit evolutionär trainierten Netzen möglich, bei dieser sehr eingeschränkten Problemstellung, wesentlich längere Mustersequenzen, bis zu einer Länge von $T = 200$, richtig zu klassifizieren.

Um bei BPTT zu testen, ob die im Gradienten unterrepräsentierte Zeitinformation für die eingeschränkte Leistungsfähigkeit verantwortlich ist, wurde in einer weiteren Messung dem BPTT zu jedem Zeitpunkt der Fehler eingespeist. Gibt man dem BPTT zusätzliche Informationen durch Berechnung des Fehlers zu jedem Zeitpunkt und speist dies in die Gradientenberechnung ein, so ist auch der BPTT-Algorithmus in der Lage dieses Problem für beliebige Werte von T zu lösen.

4.3 Automaton-Problem

Das Automaton-Problem stellt eine Klassifikationsaufgabe dar, bei der verschiedene Restriktionen des Latching-Problems aufgehoben sind. Zum einen ist die Anzahl der Neuronen größer und alle Gewichte sind trainierbar. Andererseits unterliegt der Teil der Eingabe, der Rauschen darstellt, nicht mehr der Restriktion, daß das Netzwerk trotz der Störung für immer in dem stabilen Attraktor des Fixpunktes des Ausgabeknotens verbleibt. Das vollvernetzte Netz besteht aus 6 verborgenen Neuronen, einem Ausgabe- und einem Eingabeneuron und besitzt damit 56 trainierbare Gewichte.

Alle Gewichte wurden aus dem Intervall $[-0.5, 0.5]$ mit gleichverteilter Wahrscheinlichkeit initialisiert. Wie bei dem Latching-Problem dient als Transferfunktion der verborgenen Neuronen und des Ausgabeneurons der Tangens-Hyperbolicus.

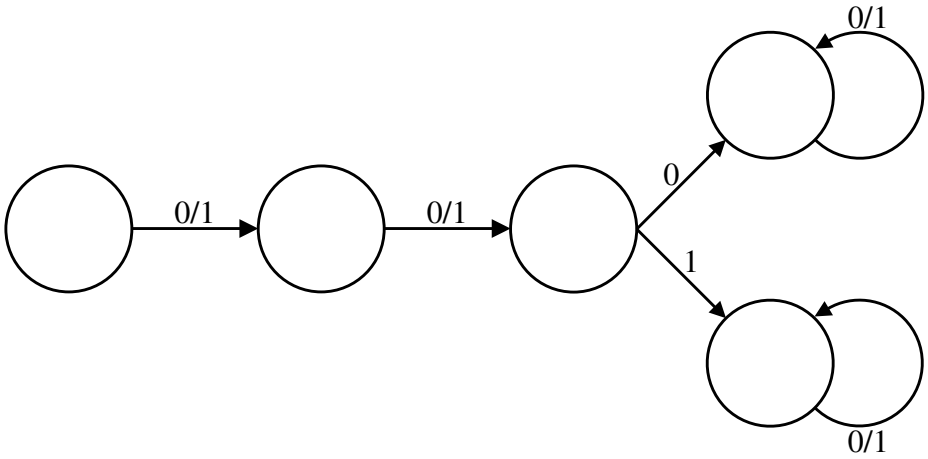

Bild 14: Endlicher Automat mit 5 Zuständen

Das Eingabemuster besteht aus einer binären Sequenz der Länge T, dadurch ist das Rauschen auf den Maximalwert von Eins begrenzt. Die Zugehörigkeit zu einer Klasse wird dadurch definiert, daß das Eingabeneuron $e(t)$ zum Zeitpunkt $t_L = 3$ entweder die Aktivität 0 für die Klasse 1, oder 1 für die Klasse 2 annimmt.

$$e(t) = \begin{cases} & 0 \text{ oder } 1 \text{ zufällig} & : \; 0 < t < t_L \\ 0 \in \text{Klasse 1}, 1 \in \text{Klasse 2} & & : \; t = t_L \\ & 0 \text{ oder } 1 \text{ zufällig} & : \; t_L < t \leq T \end{cases} \tag{20}$$

Alle Mustersequenzen können durch einen endlichen Automaten mit 5 Zuständen, wie in Bild 14 gezeigt, erzeugt werden. Trainings- und Testmenge bestanden jeweils aus 500 Mustern. Als Sollausgabe wurde für die Klasse 1 $a_{soll,1}(T) = 1.0$ und für die Klasse 2 $a_{soll,2}(T) = -1.0$ gewählt. Auch bei diesem Problem wurde nur der Fehler am Ende einer Sequenz berechnet und in den entsprechenden Algorithmus berücksichtigt. Ebenso galt ein Muster als richtig klassifiziert, wenn der quadratische Fehler am Ende der Sequenz $F(T) \leq 0.2$ war.

4.3.1 Konvergenzverhalten des ES bei unterschiedlichen Parametereinstellungen

Ein RNN stellt ein nichtlineares dynamisches System dar, das unter Umständen sehr sensitiv auf Änderungen der Gewichtsmatrix mit einer unerwünschten Variabilität der Ausgabe reagieren kann. Deshalb wurde bei diesem Testbeispiel speziell das Konvergenzverhalten bei unterschiedlichen Parametereinstellungen betrachtet.

Für diese Untersuchung wurde eine ES mit *Letzte-Elimination* verwendet. Als Trainings- und Testmenge fanden Mustersequenzen der Länge $T = 15$ Einsatz. Untersucht wurde zum einen der Einfluß der Rekombination und der Mutation. Außerdem wurde der Selektionsdruck durch Variation der Populationsgröße und des Parameters n für die Anzahl der zu eliminierenden Individuen verändert. Damit verbunden war die Untersuchung des Einflusses des Elitismus mit einem oder mehreren Individuen, die unverändert in die nächste Generation übernommen wurden.

Es wurde zunächst die Wirkung der Mutation auf das Konvergenzverhalten untersucht. Dazu wurde eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ und drei unterschiedlichen Einstellungen der Mutations-Parameter getestet. Der Rekombinationsgrenzwert war $\alpha_r = 0.5$, die Varianz $\sigma_r = 1.0$ und das beste Individuum wurde unverändert in die nächste Generation übernommen. Alle Mutationsparameter waren global für die gesamte Population eingestellt und wurden innerhalb der maximalen Generationenanzahl auf die Hälfte ihres Anfangswertes reduziert. Zum Vergleich wurden die Parameter so eingestellt, daß die Mutationsrate bei der ersten Messung sehr gering war und in den weiteren Messungen vergrößert wurde, wobei eine der verwendeten Parametereinstellung empirisch optimiert wurde.

In Tabelle 5 sind die Mutationsparameter absteigend nach der Mutationsrate und die Ergebnisse der Messungen, d.h. Generationenanzahl, und die prozentuale Er-

kennungsrate aufgeführt. Bild 15 zeigt in Abhängigkeit von der Generationenanzahl

Tabelle 5: Verhalten einer ES bei Änderung der Mutationsparameter

α_m	σ_m	Generationen	Erkennung in (%)
1.5	0.015	80	0.0
0.5	0.15	120	100.0
0.05	0.5	170	75.0

den über die Population summierten und gemittelten MSE pro Muster ($PMSE$) für die jeweilige Mutationsparameter-Einstellung. In Bild 16 ist die Ähnlichkeit der Population (AP) in Abhängigkeit der Generationenanzahl dargestellt. Die Ähnlichkeit ergibt sich als mittlerer euklidischer Abstand der Objektvariablenvektoren aller Individuen zu dem Objektvariablenvektor des besten Individuums. Zudem fand eine Normierung des Ähnlichkeitswertes in Bezug auf Gewichts- und Individuenanzahl statt. Sie ist ein indirektes Maß für die Verteilung der Population im Suchraum.

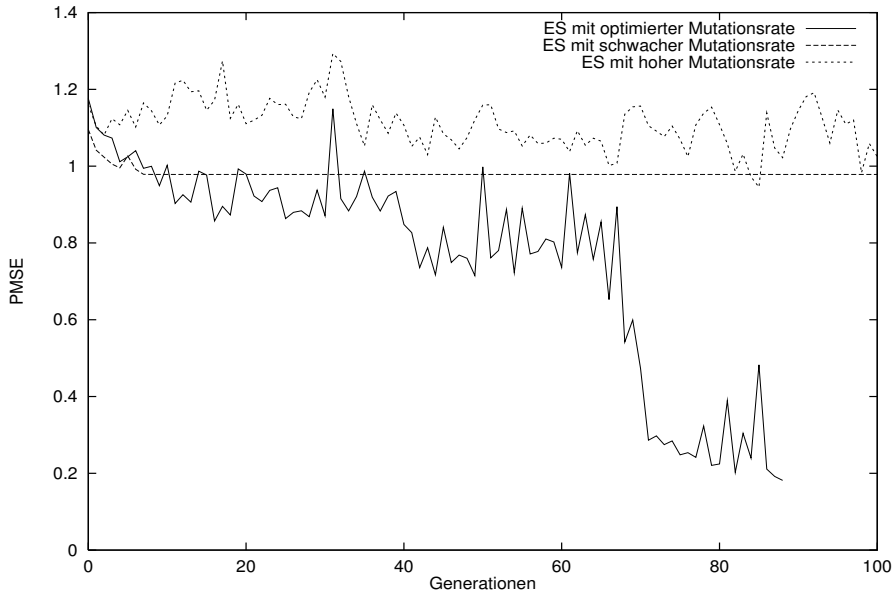

Bild 15: ($PMSE$) bei unterschiedlichen Mutationsraten

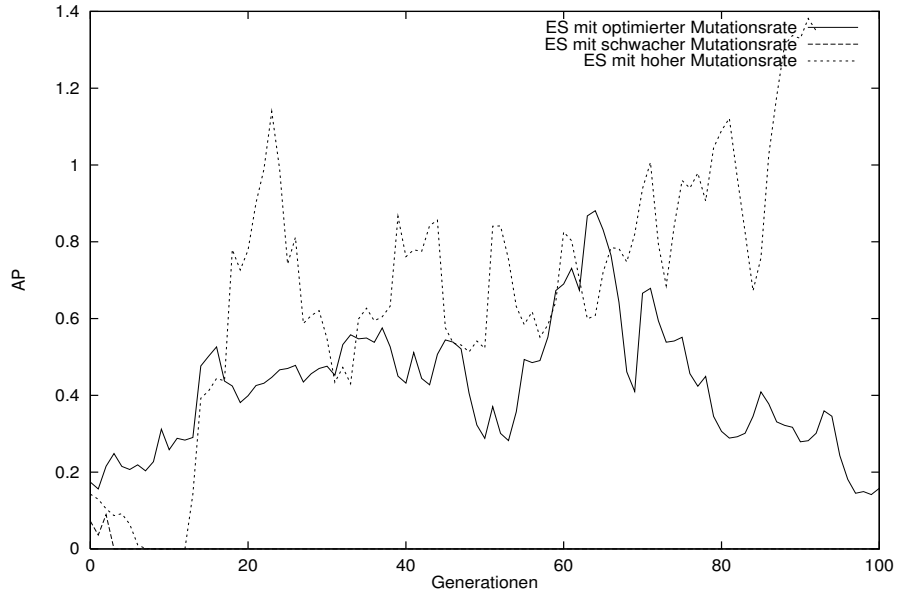

Bild 16: Ähnlichkeit der Population bei unterschiedlichen Mutationsraten

Wie die Ergebnisse in Tabelle 5 zeigen, ist sowohl bei einer zu groß als auch bei einer zu gering gewählten Mutationsrate keine Lösung des Problemes möglich. Bild 15 zeigt, daß bei zu groß gewählter Mutationsrate der $PMSE$ erheblich variiert, im Laufe der Optimierung aber nicht abnimmt. Die Ähnlichkeit der Population variiert stark und steigt an, wie in Bild 16 zu sehen ist. Dies bedeutet, daß der Algorithmus ungerichtet und rein stochastisch den Raum absucht und in der angegeben Generationenanzahl keine Lösung findet.

Ein anderes Verhalten zeigte sich bei zu geringer Mutationsrate. Hierbei sinkt der $PMSE$ nur um einen geringen Betrag, und bleibt dann im Laufe der weiteren Optimierung konstant. Deutlicher wird das Verhalten bei zu geringer Mutationsrate in Bezug auf die Ähnlichkeit der Individuen der Population, da auch hier die Kurve sehr schnell absinkt. Erklärbar ist dies durch die für die Aufrechterhaltung einer notwendigen Verteilungsbreite der Individuen im Suchraum nicht ausreichenden Mutationsrate. Die Bevorzugung der besten Individuen bewirkt, daß die Verteilung immer schmaler wird und die Population kontrahiert und der Algorithmus in ein

lokales Minimum konvergiert.

Die Variante bei der die Mutationsparameter empirisch optimiert wurden zeigte anderes Verhalten. Der $PMSE$ sinkt beständig, um dann schnell kleiner zu werden. Die Ähnlichkeit der Individuen, d.h. die Verteilung der Population im Suchraum, steigt hier erst an, um dann allmählich kleiner zu werden. Bei dieser Variante ist die Verteilung im Suchraum und die anschließende Konzentration auf ein Gebiet im Suchraum durch die gewählte Parametereinstellung gegeben.

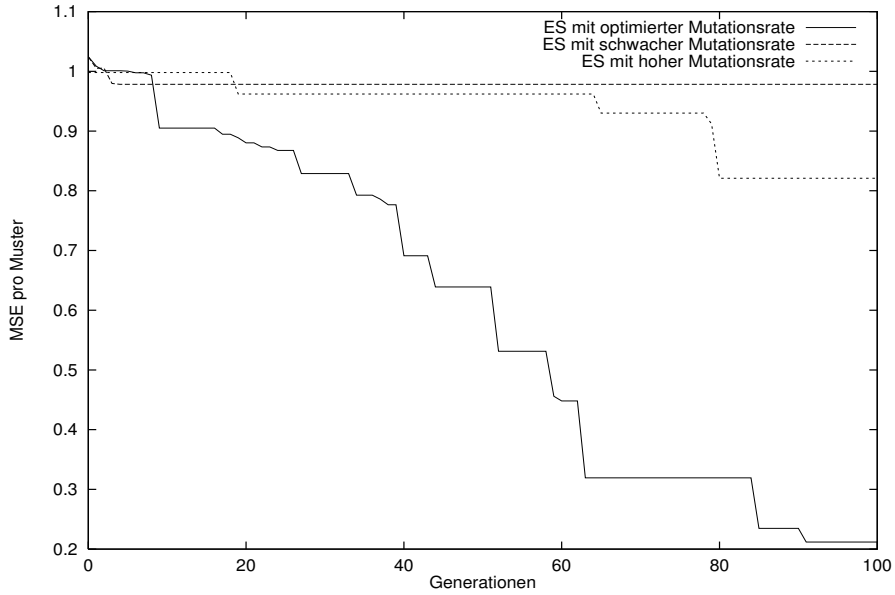

Bild 17: MSE pro Muster des besten Individuums bei unterschiedlichen Mutationsraten

Der MSE pro Muster des jeweils besten Individuums pro Generation ist in Bild 17 dargestellt. Bei der zu groß gewählten Mutationsrate nimmt der Fehler aufgrund der ungerichteten, stochastischen Suche nur zufällig ab und bleibt ansonsten konstant wegen des Elitismus. Die Variante mit zu kleiner Mutationsrate zeigt eine zunächst geringfügige Verbesserung, um im weiteren Verlauf konstant zu bleiben. Dies resultiert aus der frühen Kontraktion der Population auf einen sehr kleines Gebiet des Suchraumes. Bei der Variante mit der adaptierten Parametereinstellung wird der

Fehler in Stufen sukzessive reduziert. Auch hier ist zu beobachten, daß speziell gegen Ende der Optimierung der Fehler über eine größere Generationenzahl konstant bleibt.

Anschließend wurde die Wirkung der Rekombination untersucht, indem zum einen die Wahrscheinlichkeiten zur Rekombination und zum anderen die Mutationsparameter verändert wurden. Eingesetzt wurde wie oben eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$. Die Parameter der Rekombination wurden so variiert, daß einerseits zwei mögliche Extrema und andererseits eine optimierte Parametereinstellung einem Test unterzogen werden konnten. Das eine Extrem bestand darin, daß keine Rekombination benutzt wurde. Im anderen Extrem waren alle Nachkommen durch die Rekombination generiert.

Um die Abhängigkeit der Rekombination von der Mutation zu untersuchen, sind die Mutationsparameter ebenfalls variiert worden. Die Mutationsparameter waren zum einen die im vorangegangenen Test als optimal bestimmten Werte $\alpha_m = 0.5$, $\sigma_m = 0.15$ und $\alpha_m = 0.5$, $\sigma_m = 0.25$. Im Laufe der Optimierung wurden diese Parameter wieder auf die Hälfte reduziert. Im Elitismus wurde nur das beste Individuum unverändert in die nächste Generation übernommen. Alle Mutationsparameter waren global für die gesamte Population festgelegt.

In Tabelle 6 sind Generationenanzahl und prozentuale Erkennungsrate in Abhängigkeit von den Rekombinationsparametern und den Mutationsparametern σ_m aufgeführt. Die Ergebnisse aus Tabelle 6 zeigen, daß die Wirkung der Rekombination auch abhängig von der Mutationsrate ist. Eine Lösung wird unabhängig von dem gewählten Wert für σ_m nur von den Varianten mit optimierten Parameterwerten erreicht.

Weitere Detailuntersuchungen zeigten, daß die Varianten ein unterschiedliches Konvergenzverhalten aufweisen. Die erhöhte Mutationsrate führt bei der optimierten Einstellung der Rekombination zu einem punktuellem Auseinandertreiben der Po-

Tabelle 6: Verhalten einer ES bei Änderung der Parameter der Rekombination

α_r	σ_r	σ_m	Generationen	Erkennung in (%)
1.0	0.0001	0.15	80	75.0
1.0	0.0001	0.25	210	35.0
0.5	1.0	0.15	130	100.0
0.5	1.0	0.25	190	100.0
0.0	0.0	0.15	110	0.0
0.0	0.0	0.25	180	0.0

pulation. Dies bedeutet, daß die Verteilung der Individuen zu bestimmten Zeitpunkten plötzlich viel größer wird. Das Konvergenzverhalten ist bei optimierten Rekombinations- und Mutationsparametern wesentlich glatter, ebenso die Verteilung der Population. Im Gegensatz zu der Variante mit den optimierten Einstellungen, ist die Fehlerreduktion bei der Variante mit erhöhter Mutationsrate wesentlich sprunghafter.

Dieses Verhalten wird besonders deutlich falls alle Nachkommen durch Rekombination entstehen. Bei der geringeren Mutationsrate wird die Verteilung der Population immer schmaler, d.h. dieser Algorithmus konvergiert vorzeitig in ein lokales Optimum. Bei der höheren Mutationsrate bleibt die Verteilung der Population nahezu konstant, weshalb der Algorithmus keine glatte Konvergenz aufweist, sondern den Fehler sprunghaft verringert. Im Fall der Variante ohne Rekombination zeigt sich, daß die Verteilungsbreite der Population bei geringerer Mutationsrate schnell schmal wird. Allerdings erfolgt dies nicht so schnell wie bei den Algorithmen mit Rekombination.

Insgesamt muß die Rekombination unter Berücksichtigung der Mutationsrate bewertet werden. Primär sorgt die Rekombination für einen Austausch von Information

zwischen den Individuen. Einerseits kann eine Rekombination in Verbund mit Mutation die Verteilung einer Population nahezu konstant halten. Andererseits kann sie diese ebenfalls schmaler machen. Bei geeigneter Parameterabstimmung sorgt die Rekombination für eine Konvergenz zur optimalen Lösung, indem die Verteilung der Population richtig gesteuert wird. Dies bedeutet, daß die Verteilung der Population zu Beginn breit und im Laufe der Optimierung schmaler wird. Damit kann eine Konzentration auf den das globale Optimum umgebenden Suchraum stattfinden. Der Austausch von Informationen führt so zu einer gleichmäßigeren Konvergenz.

Der Selektionsdruck hat mitentscheidenden Einfluß auf das Konvergenzverhalten von ES. Gesteuert wird der Selektionsdruck sowohl durch die Populationsgröße, die Anzahl der für die Vermehrung zugelassenen Individuen und deren Auswahl, als auch durch den Elitismus. Im folgenden wird zuerst der Elitismus und nachfolgend die Einstellung von Populationsgröße und Anzahl der zur Nachkommenserzeugung zugelassenen Individuen untersucht.

Für die Untersuchung des Elitismus wurde wie oben eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ benutzt. Die Parameter der Mutation und der Rekombination waren zum einen die optimierten Werte $\alpha_m = 0.5$, $\sigma_m = 0.15$ aus den obigen Tests und zum anderen $\alpha_r = 0.5$, $\sigma_r = 1.0$. Auch hierbei wurden die Mutationsparameter im Laufe der Optimierung auf die Hälfte reduziert. Als Varianten wurde die Anzahl der unverändert in die nächste Generation übernommenen Individuen zu $n_E = 0$ oder $n_E = 3$ gewählt. Bei einem Wert für n_E von 0 zeigte es sich, daß der Algorithmus nicht in der Lage war eine Lösung zu finden. Sowohl der $PMSE$, als auch der beste MSE pro Generation schwankten stark. Damit variierte auch die Ähnlichkeit der Population um einen Wert.

Wurde die Mutationsrate reduziert, indem σ_m auf 0.07 und α_m auf 0.6 verkleinert wurden, gelang es dem Algorithmus eine korrekt Lösung zu finden. Dieses jedoch erst

nach 250 Generationen. Die Variante mit $n_E = 3$ konnte das Problem nicht lösen, da die Verteilung der Population zu früh zu schmal wurde und selbst eine Vergrößerung der Mutationsrate dies nicht verhindern konnte. Wurde die Mutionsrate sehr groß, so schwankte die Ähnlichkeit der Population um einen Mittelwert herum und wurde nicht schmaler. An dieser Stelle wurde die Variante mit $n_E = 1$ in den Vergleich mit einbezogen, da diese Variante in den vorangehenden Tests bereits untersucht wurde. Im Vergleich zu den anderen Varianten zeigte es sich, daß der Elitismus mit $n_E = 1$ die Konvergenz beschleunigt.

Bei der weiteren Untersuchung des Selektionsdruckes wurde der Parameter, der die zur Erzeugung von Nachkommen benutzten Anzahl von Individuen angibt, verändert. Es wurden die folgenden beiden Extrema untersucht. Zum einen waren in der Variante mit $n = 0$ alle Individuen zugelassen – d.h. es wurde mit Ausnahme des Elitismus kein Selektionsdruck ausgeübt – und zum anderen waren mit $n = 22$ nur sehr wenige Individuen zugelassen, dies bedeutet einen sehr starken Selektionsdruck.

Beide Varianten konnten aus jeweils unterschiedlichen Gründen keine Lösung finden. Im Falle von $n = 0$ wurde die Verteilung der Population nicht schmaler, und es konnte auch keine Mutationsrate gefunden werden, die eine Kontraktion der Verteilung ermöglichte. Somit suchte der Algorithmus nur stochastisch den Raum ab. Bei starkem Selektionsdruck wurde die Verteilung der Population hingegen sehr schnell sehr schmal, wodurch der Algorithmus vorzeitig in ein lokales Optimum konvergierte. Auch hier konnte dies durch Parameteränderung, speziell Vergrößerung der Mutationsrate, nicht verhindert werden. Das bedeutet, daß keine passende Abstimmung zu den vorgegebenen Parametern gefunden werden konnte.

Desweiteren wurde die Wirkung der Populationsgröße auf Konvergenzgeschwindigkeit und -verhalten untersucht. Dabei besteht allerdings die Problematik, daß die Änderung der Populationsgröße weitere Parameteränderung bedingt, damit eine Konvergenz eintritt. Der Einfluß der Populationsgröße kann also nicht isoliert be-

trachtet werden. Es wurde der oben beschriebene Algorithmus genutzt, jedoch mit empirisch voroptimierten Parametern der Mutation und der Rekombination. Die Populationsgröße betrug zum einen 25 mit $n = 10$ und zum anderen 100 mit $n = 35$.

Insgesamt wurden alle Parameter derart eingestellt, daß eine vollständige Lösung gefunden wurde.

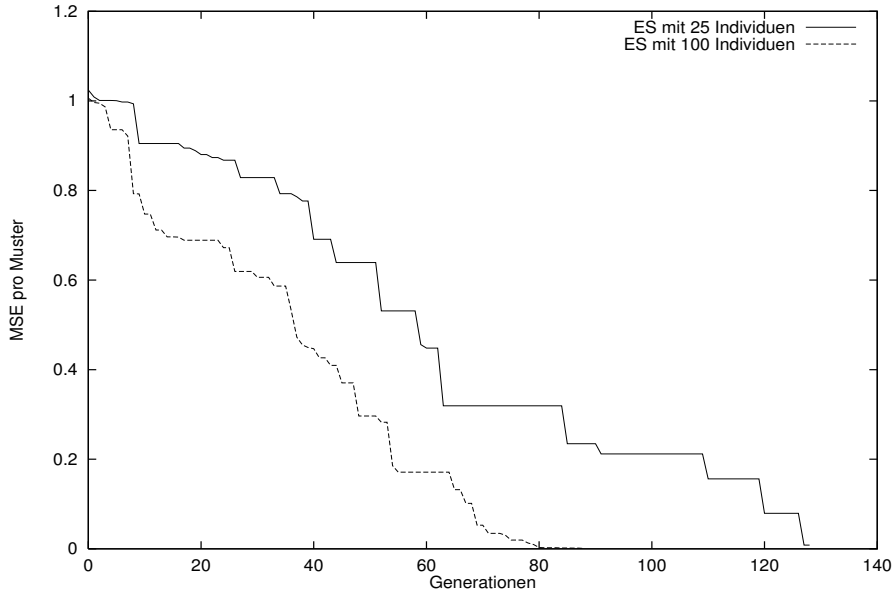

Bild 18: MSE pro Muster des besten Individuums bei unterschiedlichen Populationsgrößen

Im Bild 18 ist der MSE pro Muster des besten Individuums pro Generation für beide Populationsanzahlen dargestellt. Bild 19 zeigt die Ähnlichkeit der Population gemessen am besten Individuum pro Generation für beide Populationsgrößen. Im Vergleich der beiden Varianten zeigt der Algorithmus mit größerer Individuenanzahl eine gleichmäßigere Konvergenz, sowohl bei der Fehlerkurve als auch bei der Ähnlichkeit der Population. Insbesondere ist die Fehlerminimierung kontinuierlicher und nicht durch so lange Zeiten ohne Änderung des jeweils am besten Individuums gemessenen Fehlers gekennzeichnet. Dieses Verhalten kann dadurch erklärt werden, daß der Algorithmus durch die größere Anzahl der Individuen, speziell gegen Ende

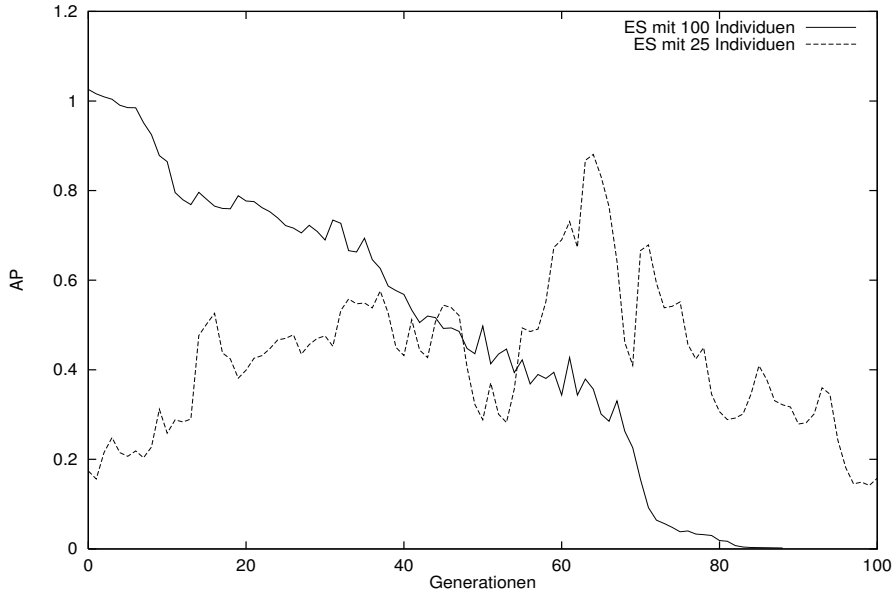

Bild 19: Ähnlichkeit der Populationen bei unterschiedlichen Populationsgrößen

der Optimierung, die Umgebung des Optimums besser abtasten kann. Gemessen an der Generationenanzahl konvergiert diese Variante mit 80 Generationen schneller, als der Algorithmus mit weniger Individuen, welcher 130 Generationen benötigt. Betrachtet man allerdings die Anzahl der insgesamt getesteten Individuen, so sind es bei der ersten Variante 8000, während es bei der zweiten Variante 3250 sind. Folglich ergibt sich für die Variante mit der größeren Population eine Verdreifachung des Rechenaufwandes.

Das Fazit dieser Untersuchungen ist, daß alle Strategieparameter passend aufeinander abgestimmt sein müssen, da jeder Parameter in komplexer Weise Einfluß auf die Einstellungen der anderen Parameter hat. Insbesondere die Strategieparameter, die die Mutation steuern, haben einen entscheidenden Einfluß auf Konvergenzverhalten bzw. Konvergenzgeschwindigkeit. Als ein guter Kompromiß zwischen Aufwand, d.h. benötigter Rechenzeit, und einer Konvergenz in einer begrenzten Generationenanzahl, kann die ES, in der Form als (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ angesehen werden.

4.3.2 Vergleich BPTT und ES

Zum Vergleich wurde einerseits BPTT-Algorithmus mit einer Lernschrittweite von $\eta = 0.01$ und einer maximalen Anzahl von 200 Epochen eingesetzt. Ein Trainingsdurchlauf wurde als erfolgreich gewertet, wenn alle Muster der Testmenge korrekt klassifiziert wurden. Andererseits wurde das Netzwerk durch eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letze-Elimination* mit $n = 10$ trainiert. Der Mutationsgrenzwert war $\alpha_m = 0.5$ und die Varianz $\sigma_m = 0.15$. Der Rekombinationsgrenzwert war $\alpha_r = 0.5$ und die Varianz $\sigma_r = 1.0$. Im Zuge des Elitismus wurde das beste Individuum unverändert in die nächste Generation übernommen. Als alternatives Abbruchkriterium diente die korrekte Erkennung aller Testmuster, wobei das Training nach 300 Generationen abgebrochen wurde. Im Bild 20 ist der Anteil der

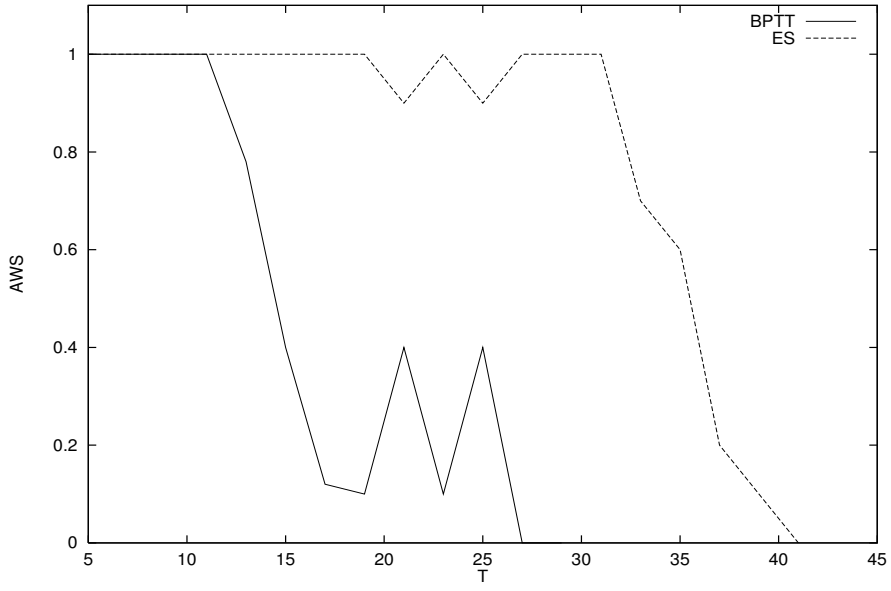

Bild 20: AWS in Abhängigkeit von der Sequenzlängen T

erfolgreichen Simulationen (AWS), normiert auf die Gesamtzahl der Simulationen pro Meßpunkt T aufgetragen, sowohl für BPTT als auch für ES. Wie dort zu sehen ist, kann ES im Vergleich zu BPTT die notwendigen Informationen über längere Zeiträume verarbeiten. Der BPTT-Algorithmus ist nicht mehr in der Lage, Muster-

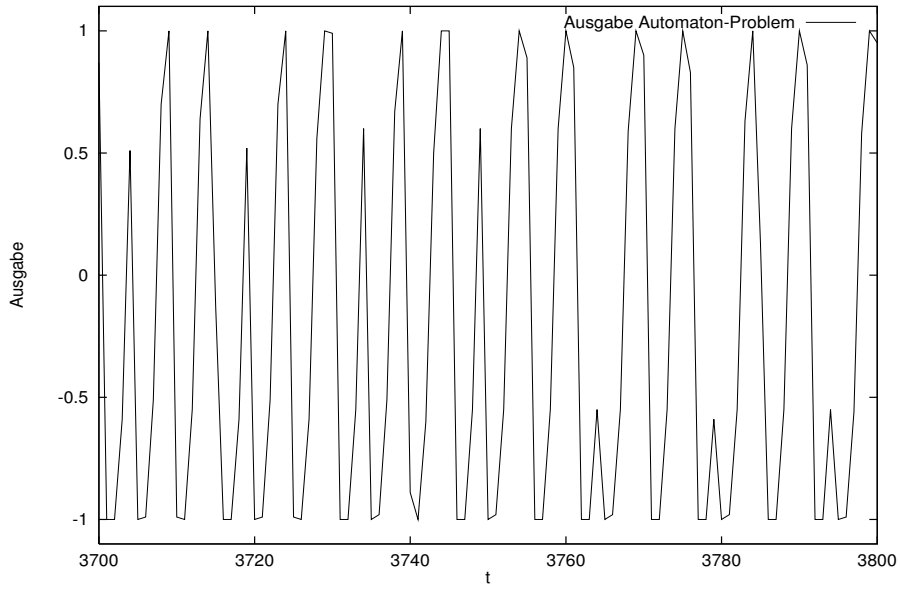

Bild 21: Exemplarische Ausgabe des RNN

sequenzen die länger als 27 Zeittakte sind, richtig zu klassifizieren. ES leistet dies bis zu 41 Zeittakte.

In Bild 21 ist die Ausgabe des Netzes über einen Zeitraum von 100 Zeittakten, bei einer Sequenzlänge von $T = 20$ dargestellt. Wie dort zu sehen ist, verbleibt das Netz nicht in einem stabilen Zustand, unabhängig vom Trainingsalgorithmus. Stattdessen lernt das Netz zum Zeitpunkt T in dem richtigen Zustand zu sein, während es in der übrigen Zeit unterschiedliche Zustände annimmt. Das Lösungsverhalten des BPTT-Algorithmus unterscheidet sich bei diesem Problem von dem der ES.

Bild 22 zeigt den MSE pro Muster in Abhängigkeit der Anzahl der Generationen bzw. der Epochen mit $T = 15$. Der BPTT-Algorithmus reduziert den Fehler im Mittel zunächst geringfügig, wobei der Fehler dann stärker abfällt, da der Algorithmus ein lokales Minimum gefunden hat, um dann sofort wieder anzusteigen. Der beim Training mit BPTT wieder ansteigende MSE pro Muster wird dabei durch die Gewichtsänderung nach jedem angelegten Muster verursacht. Dann wird innerhalb weniger Epochen der Fehler sehr klein. Dies läßt auf ein ausgeprägtes globales

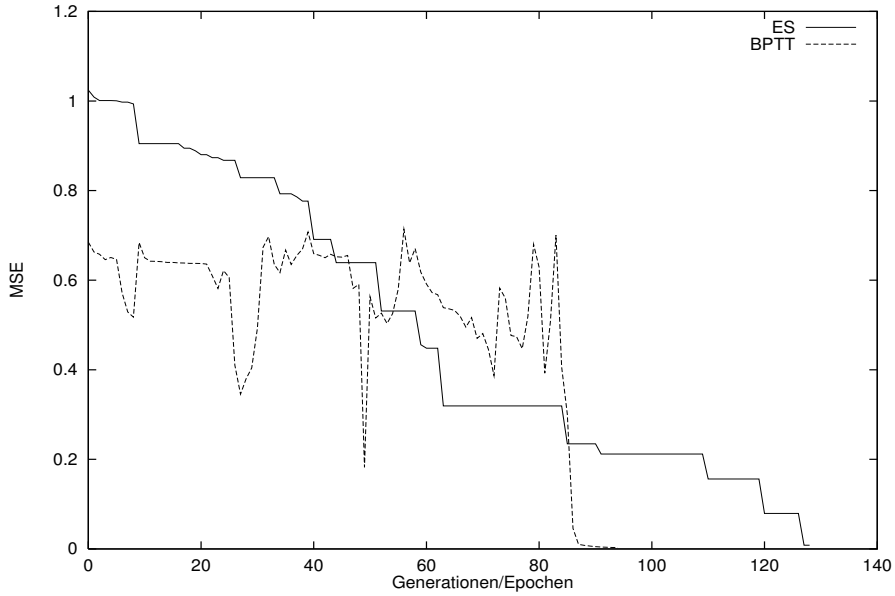

Bild 22: Vergleich des Verlaufes der Fehlerminimierung von BPTT und ES

Optimum schließen, welches zwar von einigen lokalen Extrema, aber ansonsten von einer verhältnismäßig flachen Ebene umgeben ist. Die ES dagegen minimiert den Fehler monoton.

Da die Veränderungen der Gewichte durch die Mutation zufällig und nicht in irgendeiner Form gerichtet ist, kann der Algorithmus die Topologie der Fehlerlandschaft nicht direkt ausnutzen, wie dies der BPTT-Algorithmus durch den Gradienten kann. Zum Testen eines optimumverdächtigen Gebietes muß bei ES zunächst eine Subpopulation in diesem Gebiet angesiedelt werden, die dieses absuchen kann. Erweist sich die Suche als fruchtbar, so wird ein immer größerer Anteil der Gesamtpopulation an dieser Stelle konzentriert. Werden jedoch innerhalb dieser Zeitspanne an anderen Stellen gleich gute oder bessere Optima gefunden, so wird dort ebenfalls gesucht und eventuell die Suche an dem ersten Optimum durch die genetischen Operatoren wieder eingeschränkt. Damit wird verhindert, daß sich zu große Teil der Population zu schnell in einem Optimum konzentrieren. Diese im Vergleich zu BPTT langsame Vorgehensweise zur genaueren Evaluierung eines optimumverdächtigen Bereiches

des Suchraumes kommt in Bild 22 zum Ausdruck.

Um zu untersuchen, ob eine durch ES gefundene Lösung nicht doch durch eine rein stochastische Suche gefunden wird, wurde der folgende Test durchgeführt. Zu diesem Zweck wurden 20000 Netze (wesentlich mehr als durch ES während des gesamten Trainings getestet wurden) zufällig initialisiert und ihre Erkennungsrate gemessen. Das Intervall aus dem die initialen Gewichte entnommen wurden, war mit dem Lösungsintervall $[-3.0, 3.0]$ identisch. Die Trainingsmenge bestand aus Mustersequenzen der Länge $T = 15$. Es zeigte sich, daß das beste Netz nur auf eine Erkennungsrate von 75% kam. Verglichen mit dem Training durch eine ES, in dem eine vollständige Lösung gelang und nur 3250 Netze getestet wurden, zeigt sich, daß diese Verfahren reinen stochastischen Lernverfahren überlegen sind.

5 Untersuchungen zur Spracherkennung

5.1 Sprachdaten und Merkmale

In weiteren Messungen wurde die Fähigkeit der ES zum Training von RNN für die sprecherunabhängige Einzelworterkennung untersucht. Das verwendete Vokabular bestand aus sechs Wörtern, den Zahlwörtern *Null*, *Eins*, *Zwei*, *Drei*, *Neun* und dem Kommandowort *Nein*. Diese Worte wurden ausgewählt, da sie aufgrund ihrer phonetischen Ähnlichkeit eine hohe Verwechslungsträchtigkeit aufweisen. In Tabelle 7 sind die Wörter sowie ihre phonetische Transkription wiedergegeben.

Tabelle 7: Wörter des Vokabulars und ihre phonetische Transkription

Vokabulareintrag	Transkription
Null	n U l
Eins	a I n s
Zwei	ts v a I
Drei	d r a I
Neun	n O Y n
Nein	n a I n

Das Datenmaterial bestand aus Aufnahmen von insgesamt 100 Sprechern, die zur Unterdrückung von Störgeräuschen in einer Sprecherkabine aufgezeichnet wurden. Dabei handelte es sich um 50 weibliche und 50 männliche Sprecher mit einer Altersverteilung zwischen 20–60 Jahren, die der demographischen Struktur der deutschen Bevölkerung entspricht. Die Datenmenge wurde in eine Trainings- und eine Testmenge unterteilt, indem sie unter Beachtung einer demographischen Ausgewogenheit der Trainingsmenge halbiert wurde.

Die Aufnahmen liegen mit 8 kHz abgetastet und mit 16 Bit quantisiert vor.

Durch eine digitale Filterung wurden die Signale auf Telefonbandbreite (300–3400 Hz) bandpaßgefiltert. Darauffolgend wurde eine zusätzliche Preemphase-Filterung mit $H(z) = 1.0 - 0.925z^{-1}$ durchgeführt. Anschließend wurde die Kurzzeit-Autokorrelationsfunktion aus, mit einem Hamming-Fenster gewichteten, Blöcken von jeweils 256 Signalwerten mit einem beidseitigen Überlapp von 80 Werten berechnet. Daraus wurden mittels des Durbin-Algorithmus die Koeffizienten eines optimalen linearen Prädiktors mit einer Ordnung von 12 berechnet und diese in die selbe Anzahl von Cepstral-Koeffizienten transformiert. Die pro Wort resultierende Folge von 12-komponentigen Merkmalsvektoren diente als Eingabemuster.

Zunächst fand ein Experiment mit zeitnormierten Daten statt, die aus den Originaldaten gewonnen wurden. Dazu wurden alle Merkmalsvektorfolgen der Wörter auf die selbe Anzahl von Vektoren gebracht [Rei91, E$^+$90]. Dazu wurden die Merkmalsvektorfolgen in Segmente aufgeteilt, wobei die Anzahl der Segmente der gewünschten Anzahl der Vektoren entspricht. Diese Segmentation erfolgt derart, daß innerhalb eines Segmentes die Summe der Abweichungen der Merkmalsvektoren von dem Mittelwertvektor des Segmentes minimal ist. Als reduzierte Merkmalsvektorfolge wurden die Mittelwertvektoren der Segmente genutzt. Zusätzlich wurde die Anzahl der Cepstral-Koeffizienten eines Vektors auf 8 beschränkt. Es wurden Daten mit jeweils 8, 20 und 30 Vektoren erzeugt.

5.2 Messung mit minimalem Vokabular

Zunächst wurde mit den zeitnormierten Daten bei einem noch weiter eingeschränkten Wortschatz, nämlich den Wörtern *Zwei* und *Drei*, Untersuchungen durchgeführt. Diese beiden Wörter sind sich phonetisch sehr ähnlich und unterscheiden sich im besonderen nur in ihrem Beginn. Damit sind diese Wörter ideal geeignet, um an ihnen die Leistungsfähigkeit von Algorithmen in Bezug auf die Erfassung von Zeitabhängigkeiten zu untersuchen. Für dieses Experiment wurde ein RNN mit 8

Eingabe-, 2 Ausgabe- und einem Schwellwertneuron und der Fermi-Funktion als Transferfunktion der informationsverarbeitenden Neuronen genutzt. Die Anzahl der verborgenen Neuronen betrug mit zunehmender Vektoranzahl 12, 16 und 20.

Die Sollausgabe entsprach einer (1-aus-2)-Codierung. Als erkannt galt ein Wort, wenn das ihm zugeordnete Ausgabeneuron die größte Aktivität hatte. Alle Merkmalsvektoren wurden einzeln hintereinander angelegt und jeweils die Ausgabe berechnet. Dabei wurde nach dem Ende eines Wortes das Netz zurückgesetzt, um nicht von der Abfolge der Wörter abhängig zu sein bzw. keine Effekte durch die Wirkung vergangener Muster zu haben. Der Netzwerksfehler wurde nur am Ende einer Sequenz berechnet. Beim BPTT-Algorithmus wurde er entsprechend zurückpropagiert. Bei den ES wurde er hingegen über alle Merkmalsvektorfolgen summiert und als Bewertungsfunktion benutzt. Die Netzwerke wurden zum einen mit BPTT mit einer Lernrate η von 0.15 und einer maximalen Anzahl von 500 Epochen trainiert. Zum anderen wurden die Netzwerke durch eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ trainiert. Der Mutationgrenzwert war $\alpha_m = 1.5$ und die Varianz $\sigma_m = 0.25$. Der Rekombinationsgrenzwert war $\alpha_r = 0.8$ und die Varianz $\sigma_r = 1.0$. Das beste Netz wurde unverändert in die nächste Generation übernommen.

In Tabelle 8 sind die Ergebnisse der Messungen, prozentuale Erkennungsraten und Epochen- bzw. Generationenanzahl, geordnet nach Algorithmus und Vektorenanzahl aufgelistet. Eine zusätzliche Variante der ES mit 100 Individuen, im folgenden mit ES-100 bezeichnet, ist in der letzten Zeile dargestellt, wird jedoch erst später betrachtet. Die besseren Ergebnisse zeigen, daß für diese Datenmenge die ES im Gegensatz zu BPTT die Netze so trainieren können, daß sie längere Zeitabhängigkeiten verarbeiten können. Allerdings ist selbst bei den evolutionären Algorithmen die Erkennungsleistung, speziell bei den längeren Mustersequenzen, nicht befriedigend.

Die Ursache für dieses Verhalten der ES ist vermutlich in der gesteigerten Komple-

Tabelle 8: Erkennungsraten bei zeitnormierten Daten der Wörter *Zwei* und *Drei*

Algorithmus	Vektorenanzahl	Trainingsiterationen	Erkennungsrate
BPTT	8	100	91.0
BPTT	20	150	72.0
BPTT	30	170	61.0
ES	8	110	91.0
ES	20	130	86.0
ES	30	210	72.0
ES-100	20	4500	91.0

xität der Fehlerebene zu sehen, die aus der größeren Gewichtsanzahl und der viel höheren Anzahl der zu verarbeitenden Merkmale resultiert. Der Lernvorgang der ES kann als ungleichmäßige Abtastung des Suchraumes interpretiert werden. Gesetzt den Fall, die komplexe Struktur der Fehlerlandschaft erfordert zur Lösung eine engmaschige Abtastung – und sei es nur in der Nähe des globalen Optimums – dann stellt eine zu geringe Population eine Verletzung des „Abtasttheorems“ dar. Eng lokalisierte Optima, deren Umgebung steil abfällt können nicht gefunden werden.

Um diese Hypothese zu testen, wurde zunächst am Automaton-Problem mit $T = 60$ eine Messung mit auf 100 Individuen vergrößerter Population durchgeführt. Dabei erwies sich, daß diesmal, im Gegensatz zu den Tests mit einer Population von 25 Individuen, eine vollständige Lösung gefunden wurde. Nach diesem Experiment wurde dieser Test auch an den zeitnormierten Daten mit Sequenzlängen von 20 Vektoren durchgeführt, indem die Population auf 100 Individuen vervierfacht wurde. D.h. es wurde eine (100/2,100)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 55$ trainiert. Der Mutationgrenzwert war $\alpha_m = 1.0$ und die Varianz $\sigma_m = 0.15$. Der Rekombinationsgrenzwert war $\alpha_r = 0.5$, die Varianz $\sigma_r = 1.0$. Das beste Netz wurde unverändert in die nächste Generation übernommen. Wie

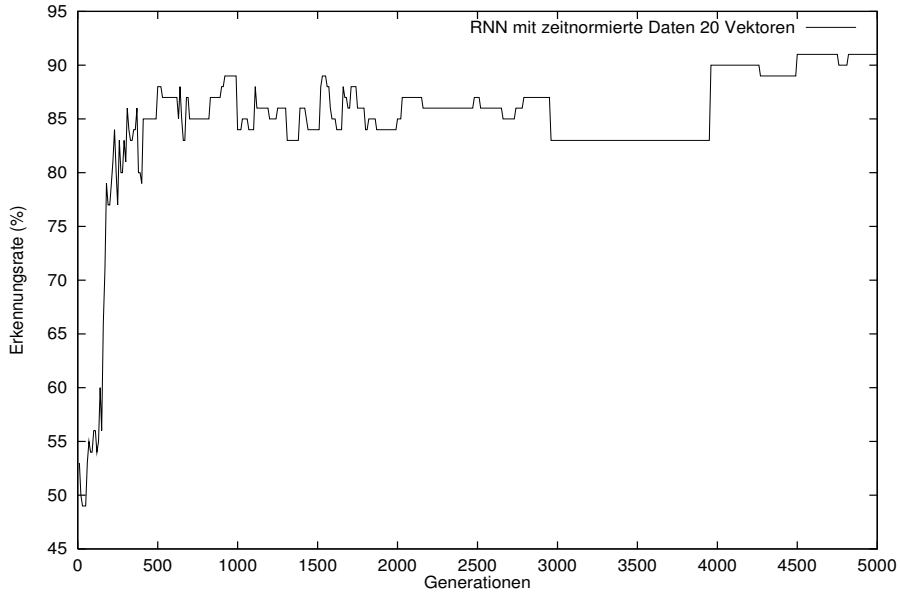

Bild 23: Erkennungsrate (%) in Abhängigkeit der Generationenanzahl

bei der Variante ES-100 in der letzten Zeile der Tabelle 8 dargestellt, ist nun die Erkennungsleistung wieder auf dem Niveau der Erkennung bei einer Sequenzlänge von 8 Vektoren. Damit zeigt sich auch bei diesem Experiment das zuvor beschriebene Verhalten der ES. Dabei ist der Rechenaufwand in Anbetracht der benötigten Anzahl von 4500 Generationen und damit 450000 getesteten Netzen sehr groß.

In Bild 23 ist prozentuale Erkennungsrate in Abhängigkeit von der Generationenanzahl dargestellt. Wie dort zu sehen ist, steigt die Erkennungsrate zu Beginn stark an, um dann lange Zeit nahezu konstant zu bleiben und erst zum Schluß nochmals anzuwachsen. Die globale Parameterabstimmung erwies sich dabei als problematisch, da erst nach einer großen Anzahl von Generationen Aussagen über die Konvergenzeigenschaften zu treffen waren, d.h. ob die Strategie eine hohe Erkennung erzielen würde. Aus diesem Grund konnten die Parameter nicht gezielt optimiert werden, weshalb der beschriebene Parametersatz wahrscheinlich eine suboptimale Einstellung darstellt.

Im anschließenden Test war das Datenmaterial nicht mehr zeitnormiert. Da die

Worte nur in den ersten 20–30 Vektoren unterschiedlich sind, muß der Trainingsalgorithmus in der Lage sein, Zeitdauern von etwa 30–40 Zeittakten zu überblicken. Es wurde ein neuronales Netz mit 25 verborgenen Neuronen und ansonsten gleicher Parametereinstellung wie bei der Messung mit zeitnormierten Daten verwandt. Das Netzwerk wurde zum einen mittels BPTT-Algorithmus mit einer Lernrate η von 0.15 und einer maximalen Anzahl von 600 Epochen trainiert. Zum anderen wurde das Netzwerk durch eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ trainiert. Der Mutationsgrenzwert war $\alpha_m = 0.5$ und die Varianz $\sigma_m = 0.1$. Der Rekombinationsgrenzwert war $\alpha_r = 0.3$, die Varianz $\sigma_r = 1.0$. Das beste Netz wurde unverändert in die nächste Generation übernommen.

Die Populationsgröße wurde aus Aufwandsgründen nicht erhöht, zumal das Datenmaterial eine andere Fehlerlandschaft bewirkt als bei den normierten Daten und das Problem damit unter Umständen lernbar bleibt. Insbesondere zeigen die Experimente am Automaton-Problem, daß bei diesen Sequenzlängen ES noch eine Lösung finden können.

Tabelle 9: Erkennungsraten bei nicht zeitnormierten Daten der Wörter *Zwei* und *Drei*

Algorithmus	Trainingsiterationen	Erkennungsrate
BPTT	100	58.0
EA	850	68.0

Die niedrige Erkennungsrate von 58 Prozent deutet darauf hin, daß die Information zur Klassifikation der Wörter über einen Zeitraum verteilt war, der mit dem BPTT-Algorithmus nicht erfaßt werden konnte. Die ES ist etwas besser und liegt 10 Prozentpunkte über dem Ergebnis des Gradientenverfahrens. Prinzipiell deutet sich jedoch eine bessere Leistungsfähigkeit im Gegensatz zum BPTT an. Diese jedoch mit größeren Populationen zu verifizieren ist aus Aufwandsgründen nicht möglich.

5.3 Messungen mit erweitertem Vokabular

Das bei diesen Messungen verwandte Vokabular bestand aus den oben beschriebenen sechs Wörtern. Die Daten waren zeitnormiert. Wie in den vorangegangenen Messungen wurden die Netze derart trainiert, daß der Fehler erst am Ende einer Mustersequenz errechnet und betrachtet wurde. Für die Ermittlung der Fehlerinformation erst am Ende einer Mustersequenz war für die ES keine Parameterabstimmung zu erreichen, die zu einem akzeptablen Ergebnis führte. Die ES konvergierten stets in ein lokales Optimum. Vermutlich war die Population für diese Datenstruktur zu klein. Ein Test mit größeren Populationen war aus Aufwandsgründen nicht durchführbar, da bereits mit der kleinen Population und ohne zusätzliche Parametertests Rechenzeiten von bis zu 4 Wochen nötig waren.

In den folgenden Messungen wurde der Fehler hingegen bei jedem Merkmalsvektor berechnet und in die Gradientenberechnung, bzw. in die Qualitätsfunktion einbezogen. Damit wird zusätzliche Information eingebracht. Diese sorgt in der Gradientenberechnung dafür, daß nicht „irrelevante“ Information bezüglich der Fehlerlandschaft extrahiert werden, was zur Konvergenz in ein lokales Extremum führen würde. Die Fragestellung war, ob ES auch bei dieser Datenkonstellation in der Lage sind, bessere Erkennungsleistungen als BPTT zu erbringen.

Es wurde ein RNN mit bei zunehmender Merkmalsvektoranzahl auf 30, 35 und 40 steigender Anzahl an verborgenen Neuronen benutzt. Ansonsten wurde die gleiche Parametereinstellung wie bei der Messung mit dem minimalen Vokabular verwandt. Alternativ wurden die Netzwerke durch eine (25/2,25)-Strategie mit Elitismus und Ersetzungsschema *Letzte-Elimination* mit $n = 10$ trainiert. Der Mutationsgrenzwert war $\alpha_m = 0.5$ und die Varianz $\sigma_m = 0.1$. Der Rekombinationsgrenzwert war $\alpha_r = 0.7$, die Varianz $\sigma_r = 1.0$. Das beste Netz wurde unverändert in die nächste Generation übernommen.

Tabelle 10: Erkennungsraten bei zeitnormierten Daten von 6 Wörter

Algorithmus	Segmente	Trainingsiterationen	Erkennungsrate (%)
BPTT	8	210	88.3
BPTT	20	320	92.2
BPTT	30	520	93.6
ES	8	1280	88.1
ES	20	1830	91.8
ES	30	2210	93.4

In Tabelle 10 sind die prozentualen Erkennungsraten und die Epochen, bzw. die Generationenanzahl für dieses Experiment aufgelistet. Die Ergebnisse zeigen, daß ES bei dieser Problemstellung nicht bessere, sondern etwas geringere Leistungen als BPTT erbringen.

Die im Gegensatz zu den vorherigen Experimenten besseren Leistungen von BPTT beruhen auf der Tatsache, daß durch die Fehlerinformation der Zeitraum, den der Algorithmus überblicken kann, vergrößert wird. Der Gradient erhält zu jedem Zeitpunkt Information über die Fehlerlandschaft und kann dadurch, selbst wenn die langreichweitigen Abhängigkeiten im Gegensatz zu den kurzreichweitigen im Gradienten unterrepräsentiert werden, aus der resultierenden Gesamtinformation Gewichtsänderungen extrahieren, die im Laufe des Trainings in ein hinreichend gutes Optimum führen.

Im Sinne der obigen Betrachtung der ES als Abtastung scheint die Fehlerlandschaft nicht mehr so komplex zu sein, wodurch auch eine geringere Populationsgröße zu einer Lösung führt. Jedoch ist dadurch nicht ausgeschlossen, daß mit einer größeren Populationsanzahl eine Steigerung der Erkennungsleistung über jene von BPTT nicht möglich wäre. Dies konnte hier jedoch aus Aufwandsgründen nicht getestet

werden. Somit kann die Frage ob ES auch in dieser Datenkonstellation in der Lage sind, bessere Erkennungsleistungen als BPTT zu erbringen, nicht abschließend beantwortet werden. Insgesamt zeigt sich auch bei diesen Experimenten, daß Rechenzeiten für ES im Vergleich zu denen des BPTT deutlich länger sind.

5.4 Abschätzung und Vergleich des Rechenaufwandes von ES und BPTT

Zur Betrachtung des Aufwands werden die zu vergleichenden Algorithmen, ES und BPTT, in zwei Teilschritte zerlegt. Dies ist zum einen der für beide Algorithmen in der Grundstruktur gleiche *Evaluations-Anteil*, in dem die Ausgabe aller Muster und der Fehler errechnet wird. Zum anderern ist es der jeweils unterschiedliche *Adaptions-Anteil*, in dem die Gewichte verändert werden. Betrachtet man den *Evaluations-Anteil*, muß dieser bei ES mit einer der aktuellen Populationsgröße in einer Generation entsprechenden Häufigkeit durchgeführt werden, während dies in einer Epoche des BPTT nur einmal geschieht. Im *Adaptions-Anteil* von BPTT wird die Gradientenberechnung mit Fehlerrückführung durchgeführt. Der *Adaptions-Anteil* von ES ist im wesentlichen durch die zweistufigen Zufallsprozesse gekennzeichnet, in denen Mutationen und Rekombinationen ausgeführt werden.

Angemerkt sei, daß innerhalb einer Epoche von BPTT die Gewichte einmal pro Muster geändert werden. Bei ES geschieht dies nur einmal pro Generation und Netz. Ein direkter Vergleich bzw. eine Bewertung dieses Sachverhaltes ist nicht möglich, da im Hinblick auf die Verringerung des Fehlers auf der Testmenge keine Aussage über die Qualität einer Gewichtsänderung folgt. Weil der *Adaptions-Anteil* von ES aufgrund der Zufallsprozesse nur schwierig numerisch exakt zu erfassen ist, wurden beide Verfahren empirisch im Hinblick auf die benötigte Rechenzeit miteinander verglichen.

Der Vergleich wurde anhand von Testläufen des Automaton-Problems mit $T = 15$ durchgeführt. Die maximale Generationen- bzw. Epochenanzahl war auf 10 festgelegt. Es wurden die Konfigurationen der Verfahren benutzt, welche auch beim direkten Vergleich der beiden Algorithmen in Bezug auf ihre Leistungsfähigkeit Anwendung fanden. Die Einstellung einer Compileroption gestattete die Erfassung der Zeitinformationen während des Programmlaufes. Um den Einfluß momentaner Belastungen des Computers zu minimieren, wurde dieser Test mehrmals wiederholt und die gemessenen Werte gemittelt.

In Tabelle 11 sind die Algorithmen, sowie die spezifischen Werte ihrer Anteile und deren Summe aufgelistet. Insgesamt bedeutet dies, im Falle gleicher Epochen- ,

Tabelle 11: Messergebnisse der zeitvergleichenden Messungen an BPTT und ES

Algorithmus	Zeittakte für *Evaluation*	Zeittakte für *Adaptions*	Summe
BPTT	13.3	10.9	23.2
ES	312.3	562.6	874.9

wie Generationenanzahl und einer Populationsgröße von 25 Individuen, einen Geschwindigkeitsverhältnis von 1:37. Konkret bedeutet dies im Falle der Tests mit sechs Wörtern und einer Vektoranzahl von 30 eine Rechenzeit der ES von bis zu 8 Wochen. Im Vergleich dazu benötigt BPTT auf einem Computer mit folgenden Leistungsmerkmalen etwa einen Tag. Der Rechner hatte eine Leistung nach SPEC [1] von 3.14 CINT95 und 7.50 CFP95. Aus diesem Grund konnten sowohl die Populations- , als auch die Netzgrößen nicht sehr groß gewählt werden, da ansonsten die benötigte Rechenzeit zu lang geworden wäre.

[1] „Standard Performance Evaluation Corporation“, eine Organisation, die sich mit der herstellerübergreifenden Messung von praxisbezogenen Rechenleistungsdaten beschäftigt. Ausführliche Erläuterungen zu den Messmethoden und Resultaten für viele Systeme sind im InterNet unter http://www.specben.org/ verfügbar

6 Zusammenfassung

Im Rahmen der vorliegenden Diplomarbeit wurde die Leistungsfähigkeit von evolutionären Algorithmen zum Training von RNN untersucht und mit gradientenbasierten Trainingsalgorithmen verglichen. Die Zielsetzung war dabei im besonderen die Prüfung der Verwendbarkeit in der Sprachverarbeitung, speziell der Spracherkennung.

Zunächst wurde anhand eines Prädiktionsproblems die prinzipielle Leistungsfähigkeit von EA untersucht, indem ein MLP mit unterschiedlichen evolutionären Algorithmen trainiert wurde. Verschiedene Varianten von GA und ES sind an diesem Beispiel getestet und miteinander verglichen worden. Im Rahmen der Untersuchungen an GA stellte sich heraus, daß eine Mindestgenauigkeit der Quantisierung zur Lösung erforderlich ist. Es zeigt sich, daß die Genauigkeit der Approximation mit abnehmendem Quantisierungsfehler besser wird. Damit ist eine Behandlung dieses Problems mit grob quantisierten Gewichten nachteilig. Demgegenüber profitiert ES sowohl in der Approximationsgenauigkeit, als auch in der Konvergenzgeschwindigkeit von der direkten Darstellung der Objektvariablen als reelle Zahlen.

Weiterhin zeigte sich bei ES, daß die Genauigkeit einer Lösung auch von der Populationsgröße abhängig ist, da mit wachsender Populationsgröße der Parameterraum besser abgetastet werden kann. Im Vergleich mit ES benötigten GA längere Konvergenzzeiten und bedingten zudem aufgrund der Codierung und Decodierung einen höheren Rechenaufwand als ES, so daß die Untersuchungen an RNN nur mit ES durchgeführt wurden.

Zunächst wurde mit dem Latching-Problem eine, in der Komplexität eng begrenzte, Klassifikationsaufgabe mit Zeitabhängigkeiten untersucht. Die zur Verfügung gestellte Information war bei diesem Beispiel sehr gering, da der Fehler nur am Ende einer Mustersequenz berechnet wurde. Es stellte sich heraus, daß selbst bei dieser

sehr einfachen Aufgabenstellung die gradientenbasierten Verfahren nach dem Überschreiten einer maximalen Mustersequenzlänge T keine Lösung finden konnten. Im Gegensatz dazu war ES in der Lage, das Problem für alle gemessenen Variationen des Parameters T zu lösen. Erst wenn während des Trainings dem Gradientenverfahren zusätzliche Informationen durch Fehlereinspeisung zur Verfügung gestellt wurde, hatte der BPTT-Algorithmus die selbe Leistungsfähigkeit.

Als weiteres Experiment mit Zeitabhängigkeiten wurde das Automaton-Problem untersucht, welches mittels eines RNN gelöst werden sollte. Bei diesem Problem wurde besonderer Wert auf die Untersuchung des Konvergenzverhaltens bei Änderungen der Parameter von ES gelegt. Die Untersuchungen ergaben, daß die einzelnen Parameter in komplexer Weise miteinander interagieren und nur eine gute Abstimmung aller Parameter aufeinander eine befriedigende Leistung in Bezug auf Konvergenzgeschwindigkeit und Klassifikationsergebnis erbringt. Wie bei dem Latching-Problem wurde der Fehler nur am Ende einer Mustersequenz berechnet. Dies bewirkt, daß der BPTT-Algorithmus bereits bei Sequenzlängen von $T = 27$ nicht mehr in der Lage ist, die Zeitabhängigkeiten in dem Gradienten zu repräsentieren. Mit ES dagegen konnten RNN trainiert werden, die in der Lage sind, Sequenzlängen bis zu $T = 41$ richtig zu klassifizieren. Die Untersuchungen bestätigen, daß der beschränkende Faktor in erster Linie der Trainingsalgorithmus und nicht das Netzwerksparadigma ist.

Die Simulationsexperimente mit zeitnormierten Sprachdaten zeigen, daß mit ES prinzipiell höhere Erkennungsleistungen als mit dem gradientenbasierten Algorithmus des BPTT erzielt werden können. Jedoch nimmt schon bei der Klassifikation der Zahlwörter *Zwei* und *Drei* die Klassifikationsleistung mit zunehmender Sequenzlänge ab. Es erfordert eine drastische Vergrößerung der Populationsgröße, um zumindest gleich gute Ergebnisse zu erzielen. Zusätzliche Tests am Automaton-Problem stützen diese Aussage. Jedoch steigt der Rechenaufwand durch Vergrößerung der Populationsgröße so stark an, daß bei nicht zeitnormierten Sprachdaten ES mit adäquater Populationsgröße nicht mehr simulierbar waren.

In den Untersuchungen an dem Vokabular mit sechs Wörtern wurde der Fehler für jeden anliegenden Merkmalsvektor berechnet und im Gradienten bzw. zur Bewertung bei ES im Training verwendet. In diesen Messungen erbringen beide Algorithmen nahezu identische Klassifikationsergebnisse.

Insgesamt verhindert der drastisch ansteigende Rechenaufwand bei den Sprachdaten die Verarbeitung von größeren Vokabularien und langen Wörtern durch ES. Aus der Beschränkung der Populationsgröße durch die vorhandene Rechnerkapazität resultierte eine nichtoptimale Anpassung von Selektionsdruck, Mutationsrate und Populationsverteilung im Suchraum. Insbesondere erweist sich die globale Anpassung der Strategieparameter bei den vergrößerten Populationen als problematisch. Weitere Untersuchungen an ES mit Strategien zur Selbstadaption dieser Parameter bieten sich daher für zukünftige Forschung an.

Literatur

[B+94] Y. Bengio et al. Learning Long-Term Dependencies with Gradient is Difficult. In *IEEE Trans. on Neural Networks*, volume 5, pages 157–166, 1994.

[B+97] Th. Bäck et al. Evolutionary Computation: Comments on the History and Current State. In *IEEE Trans. on Evolutionara Computation*, volume 1, No. 1, pages 3–17, 1997.

[Bä92] T. Bäck. The Interaction of Mutation Rate, Selection, and Self-Adaption within a Genetic Algorithm. In *Parallel Problem Solving from Nature*, volume 2, 1992.

[Bä93] T. Bäck. Optimal Mutation Rates in Genetic Search. In *Genetic Algorithms: Proceedings of the 5th International Conference*. Morgan Kaufmann, 1993.

[BH93] T. Bäck and U. Hammel. Modelloptimierung mit evolutionären Algorithmen. Technical report, Universität Dortmund, 1993.

[Bis95] C.M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press Oxford, 1995.

[Bra95] R. Brause. *Neuronale Netze*. B.G. Teubner Stuttgart, 1995.

[Dar56] Charles Darwin. On the Origins of Species by Means of Natural Selection, 1856.

[E+90] S.A. Euler et al. Statistical Segmentation and Word Modeling Techniques in Isolated Word Recognition. In *Proc. ICASSP*, pages 745–748, 1990.

[FM92] S. Forrest and M. Mitchel. What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and their Explanation. Technical report, Santa Fe Institute, 1992.

[FM93] Stephanie Forrest and Melanie Mitchell. Relative Building-Block Fitness and the Building-Block Hypothesis. In *Foundations of Genetic Algorithms*, volume 2. Morgan Kaufmann, 1993.

[G+96] C. Lee Giles et al. Learning Long-Term Dependencies in NARX Recurrent Neural Networks. In *IEEE Trans. on Neural Networks*, volume 7, No. 6, pages 1329–1338, November 1996.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[H+94] John H. Holland et al. When Will a Genetic Algorithm Outperform Hill Climbimg. In *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, 1994.

[HB92] F. Hoffmeister and T. Bäck. Genetic Algorithms and Evolution Strategies: Similarities and Differences. Sys-1/92, Universität Dortmund, 1992.

[Hol75] John Holland. *Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Systems.* The University Press of Michigan Press, Ann Arbor, 1975.

[KS86] A. Kunick and W.-H. Steeb. *Chaos in dynamischen Systemen.* Bibliographisches Institut Wissenschaftsverlag, 1986.

[Pea89] B.A. Pearlmutter. Learning State Space Trajectories in Recurrent Neural Networks. In *Neural Computation*, volume 1, pages 263–269, 1989.

[Pin89] F.J. Pineda. Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation. In *Neural Computation*, volume 1, pages 161–172, 1989.

[Rec73] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biololgischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[Rec92] Ingo Rechenberg. Evolutionsstrategie. unveröffentlichtes Manuskript, 1992.

[Rei91] Herbert Reininger. Nichtlineare Signalverarbeitung mit neuronalenNetzwerken. In *Kleinheubacher Berichte 34*, pages 601–605, 1991.

[Rei94] Herbert Reininger. *Stochastische und neuronale Konzepte zur automatischen Spracherkennung*. Th. Hector, 1994. Forum Phoneticum 60.

[RM86] D.E. Rumelhart and J.L. McClelland. Parallel Distributed Processing, Exploration in the Microstructure of Cognition, 1986.

[Roj93] Rojas. *Theorie der neuronalen Netze*. Springer-Verlag, 1993.

[Wer90] J.P. Werbos. Backpropagation Through Time: What It Does and How to Do It. In *Proc. IEEE*, volume 78 No.10, pages 1550–1560, 1990.

[Wü94] Harald Wüst. Automatische Spracherkennung mit vollvernetzten rekurrenten neuronalen Netzen. Universität Frankfurt am Main, 1994. Diplomarbeit.

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Abkürzungen

RNN	:	Rekurrentes neuronales Netz
MLP	:	Mehrschichtiges Perzeptron
BPTT	:	Backpropagation-Through-Time
EA	:	Evolutionäre Algorithmen
ES	:	Evolutionäre Strategien
GA	:	Genetische Algorithmen
MSE	:	Mean-Square-Error pro Muster
PMSE	:	Populations-MSE
AWS	:	Anteil der erfolgreichen Simulationen
AP	:	Ähnlichkeit der Population gemessen am jeweils besten Individuum

Danksagung

Herr Dozent Dr. Herbert Reininger danke ich für die Aufnahme in die Arbeitsgruppe, der kritischen Durchsicht dieser Arbeit und Anregungen zu dieser. Herr Diplom-Physiker Klaus Kasper danke ich für die kritischen Fragen zu dieser Arbeit und diversen Hilfestellungen. Insbesondere möchte ich den Herren Diplom-Physiker Holger Schalk und Diplom-Physiker Ulrich Balss für ihre aufopfernde Hilfe bei der Korrektur dieser Arbeit danken. Insgesamt möchte ich allen die diese Arbeit zu korrigieren halfen besonders danken. Den Herren Diplom-Physiker Harald Wüst und speziell Diedrich *‚Packman‘* Bossecker danke ich für die vielen Hilfestellungen bei der gradientenbasierten Methode des Back-Propagation-Through-Time und letzterem auch für die Hilfestellungen bei der awk-Skriptprogrammierung. Im besonderen möchte ich Herr Diplom-Physiker Marcus *‚safe often, safe early‘* Prätzas danken für die schwierige und sehr zeitaufwendige Systemadministration und damit für ein funktionsfähiges Rechner-Netzwerk, und das er immer ein offenes Ohr für meine Probleme mit den Rechnern und Programmen hatte. Als das ultimative Debugging-Tool erwieß sich in hartnäckigen Fällen immer Herr Diplom-Physiker Ulrich Balss, wofür ihm hiermit gedankt sei. Für viel Spass und Frühstück möchte ich den Herren Diplom-Physiker Martin *‚copy -r‘* Schlothauer, Diplom-Physiker Suat Suna, Holger Gruber und nochmals Marcus Prätzas, Holger Schalk und Diedrich Bossecker danken. Und last but not least möchte ich meinen Eltern Winfried und Rita Wolkenhauer und meiner Lebensgefährtin Doris Dessert für die großartige Hilfe und Unterstützung danken, die sie mir während dieser Zeit gaben.