Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
Fachbereich Informatik, Mathematik und Naturwissenschaften

Evolutionäre Algorithmen

Zeitreihenanalyse mit Gene Expression Programming

Leipzig, August 2007

Vorgelegt von:	Matthias Jauernig (06INM)
Lehrveranstaltung:	Evolutionäre Algorithmen
Semester:	Sommersemester 2007
Verantwortlicher Professor:	Prof. Dr. rer.nat. Karsten Weicker

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Zielsetzung

Die Analyse von Zeitreihen ist ein diffiziles Problem mit hoher praktischer Bedeutung in der heutigen Ökologie und Ökonomie. Schließlich erlaubt die Kenntnis der Gesetzmäßigkeiten hinter einer Zeitreihe Prognosen für die Zukunft und kann somit z. B. bei Spekulationen an der Börse bares Geld wert sein. Da verwundert es nicht, dass in der Vergangenheit viele Verfahren entwickelt wurden um Zeitreihen vorhersagbar zu machen. Die Ergebnisse hielten dabei Einzug in praktische Anwendungen und doch bleibt noch viel Raum für weitere Entwicklungen, die sich auch in aktuellen Forschungen niederschlagen.

Diese Arbeit beschäftigt sich primär mit *Gene Expression Programming* (GEP, s. [1]), einen erst 2001 in [3] vorgestellten evolutionären Algorithmus, der noch Gegenstand aktueller Entwicklungen ist. Anhand dessen und einer in [5] behandelten Erweiterung soll untersucht werden, in wie weit das Problem der Zeitreihenanalyse gelöst werden kann. Weiterhin werden die Ergebnisse mit der Anwendung eines einfachen vorwärts gerichteten *künstlichen neuronalen Netzes* (KNN) verglichen und ausgewertet.

In Abschnitt 2 wird zunächst das Problem der Zeitreihenanalyse beschrieben, ebenso werden Vergleichsmaße für die einzelnen Verfahren fest gelegt und die zu untersuchenden Zeitreihen vorgestellt. Abschnitt 3 stellt GEP als aktuelle Entwicklung eines evolutionären Algorithmus vor, dabei werden zudem Parametereinstellungen festgelegt und Hypothesentests zu einigen ausgewählten Parametern durchgeführt. Abschnitt 4 beschreibt die in [5, 6] vorgestellte Erweiterung zu GEP, auch hier werden wieder Parametereinstellungen angegeben und Hypothesentests zu speziellen Parametern durchgeführt. Abschnitt 5 stellt das verwendete KNN vor, wobei mit einer kompakten Form der Priorisierung von GEP Rechnung getragen wird. In Abschnitt 6 werden die beschriebenen Verfahren miteinander verglichen, bevor Abschnitt 7 mit Schlussbemerkungen und einem abschließenden Fazit der Arbeit einen Rahmen gibt.

2 Problem der Zeitreihenanalyse

2.1 Problembeschreibung

Eine *Zeitreihe* ist eine Folge von Messwerten, die von der Zeit abhängt und über diese variiert. Die Messwerte werden dabei zu diskreten Zeitpunkten abgegriffen, die Zeit τ

zwischen 2 Messungen soll dabei hier als konstant angesehen werden, die Zeitpunkte sind also äquidistant. Typische Zeitreihen setzen sich aus regelhaften und zufälligen Ursachen zusammen, wobei die zufälligen Einflüsse als „Störungen“ oder auch „Rauschen“ bezeichnet werden. Mathematisch beschrieben werden kann eine Zeitreihe durch:

$$x_i = f\left(x_{i-\tau}, x_{i-2\tau}, \ldots, x_{i-(d-1)\tau}\right)$$

- $\tau \ldots$ Zeit zwischen 2 Messungen
- $d \ldots$ Dimension des Problems, d. h. Anzahl der berücksichtigten letzten Zeitreihenwerte
- $f \ldots$ Funktion, welche die Zeitreihe beschreibt

Die *Zeitreihenanalyse* hat nun zur Aufgabe eine Funktion f zu finden, die zu einer Anzahl d von Parametern (Zahlen $x_{i-d} \ldots x_{i-1}$ der Zeitreihe) den aktuellen Wert x_i voraussagt. Weiterhin soll damit eine Gesetzmäßigkeit gefunden werden mit der es möglich ist, von den bekannten Daten auf zukünftige Entwicklungen der Messwerte zu schließen und Voraussagen über diese zu treffen. Neben der Vorhersage des nächsten Wertes kann es sich dabei auch um längerfristige Prognosen handeln. Neben traditionellen mathematischen Methoden der Statistik können für die Analyse bestehender Zeitreihen auch evolutionäre Algorithmen und künstliche neuronale Netze eingesetzt werden, was den Untersuchungsschwerpunkt dieser Arbeit darstellt.

2.2 Vergleichsmaße

Als Vergleichsmaße der behandelten Methoden sollen zwei unterschiedliche Verfahren zum Einsatz kommen.

Das erste Maß besteht im *durchschnittlichen absoluten Fehler* (*MAE, Mean Absolute Error*), der wie folgt definiert ist:

$$MAE = \frac{1}{N-p} \cdot \sum_{i=p+1}^{N} |\tilde{x_i} - x_i|$$

- $N \ldots$ Anzahl der vorhandenen Messwerte
- $p \ldots$ Anzahl zu verwendender Parameter $i-p \ldots i-1$ zur Berechnung von x_i
- $\tilde{x_i} \ldots$ Vom Modell gelieferter i-ter Wert
- $x_i \ldots$ Von der Zeitreihe vorgegebener i-ter Wert

Beim zweiten Maß handelt es sich um *NMSE* (*Normalized Mean Square Error*), der in [6] verwendet wird und hier zu Vergleichszwecken Anwendung finden soll. NMSE ist definiert

als:

$$NMSE = \frac{1}{\sigma^2}\left(\frac{1}{N-p}\cdot\sum_{i=p+1}^{N}(x_i-\tilde{x_i})^2\right) = \frac{\sum_{i=p+1}^{N}(x_i-\tilde{x_i})^2}{\sum_{i=p+1}^{N}(x_i-\overline{x})^2}$$

σ^2 ... Varianz der vorgegebenen Messwerte
N ... Anzahl der vorhandenen Messwerte
p ... Anzahl zu verwendender Parameter $i-p \ldots i-1$ zur Berechnung von x_i
$\tilde{x_i}$ ... Vom Modell gelieferter i-ter Wert
x_i ... Von der Zeitreihe vorgegebener i-ter Wert
$\overline{x}$ ... Mittelwert über die vorgegebenen Messwerte, ergibt sich aus $\frac{1}{N-p}\cdot\sum_{i=p+1}^{N}x_i$

Zudem werden als weitere Vergleichskriterien zwischen den Methoden der MAE und NMSE über die verbrauchte Rechenzeit betrachtet und somit versucht, eine Aussage über die Praktikabilität der verwendeten Algorithmen zu treffen.

Die zur Verfügung stehenden Werte einer Zeitreihe werden außerdem in *Trainingsdaten* und *Testdaten* unterteilt. Die Trainingsdaten dienen dem Anlernen des KNN bzw. der GEP-Algorithmen, mit den Testdaten soll bestimmt werden, ob die jeweiligen Lösungen die nächsten für die Algorithmen noch unbekannten Punkte der Zeitreihe vorhersagen können. Dabei wird zwischen einer 1-Punkt-Vorhersage und einer Mehrpunkt-Vorhersage unterschieden. Die Güte der betrachteten Algorithmen wird sowohl mittels MAE als auch NMSE über die Trainingsdaten berechnet und eine Evaluation der Testdaten mittels MAE und prozentualem Fehler bei einer 1-Punkt- und 5-Punkt-Vorhersage durchgeführt.

2.3 Analysierte Zeitreihen

2.3.1 Furnas Zeitreihe

- *Beschreibung*: Monatlicher Wasserstand des Rio Grande am Furnas-Staudamm (Brasilien) in der Zeit von 1931-1978.
- *Quelle*: `http://www.stats.uwo.ca/faculty/aim/epubs/mhsets/monthly/furnas.dat`
- *Datenpunkte*: 576
- *Visualisierung*:

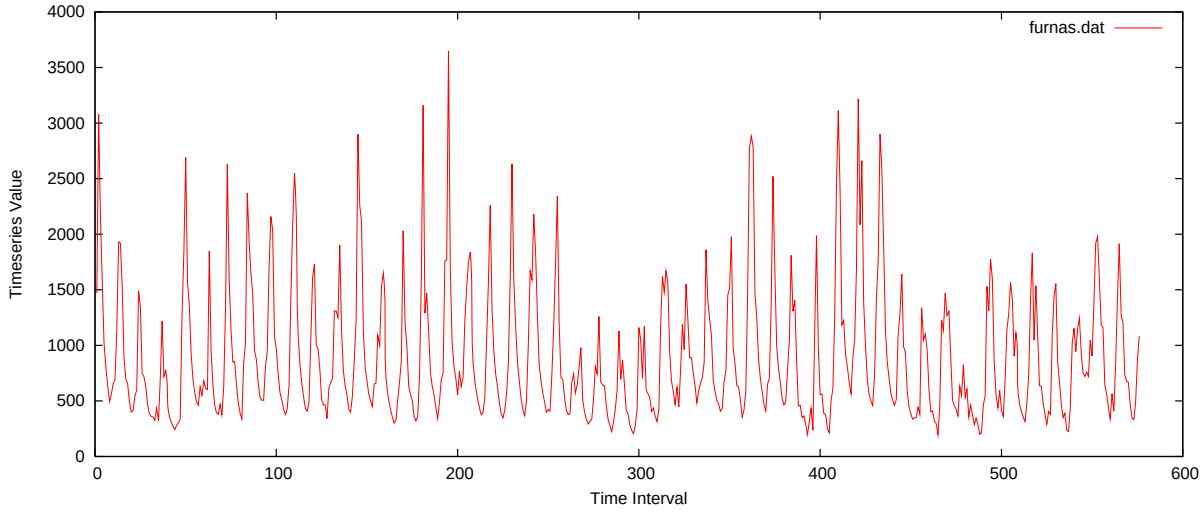

Abbildung 1: Furnas Zeitreihe

2.3.2 Erdbeben Zeitreihe

- *Beschreibung*: Anzahl von Erdbeben pro Jahr mit einer Stärke von 7.0 oder mehr in der Zeit von 1900-1998.
- *Quelle*: `http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/data/earthq.dat`
- *Datenpunkte*: 99
- *Visualisierung*:

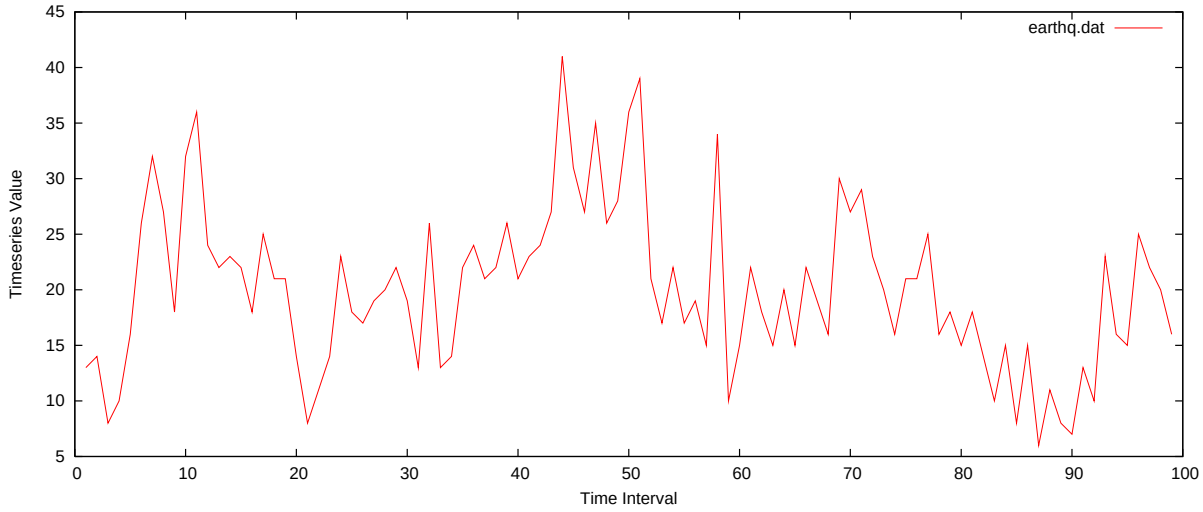

Abbildung 2: Erdbeben Zeitreihe

2.3.3 Sonnenflecken Zeitreihe

- *Beschreibung*: Jährliche Anzahl von Sonnenflecken in der Zeit von 1700-1988.
- *Quelle*:
  `http://www.stats.uwo.ca/faculty/aim/epubs/mhsets/annual/sunspots.1`
- *Datenpunkte*: 289
- *Visualisierung*:

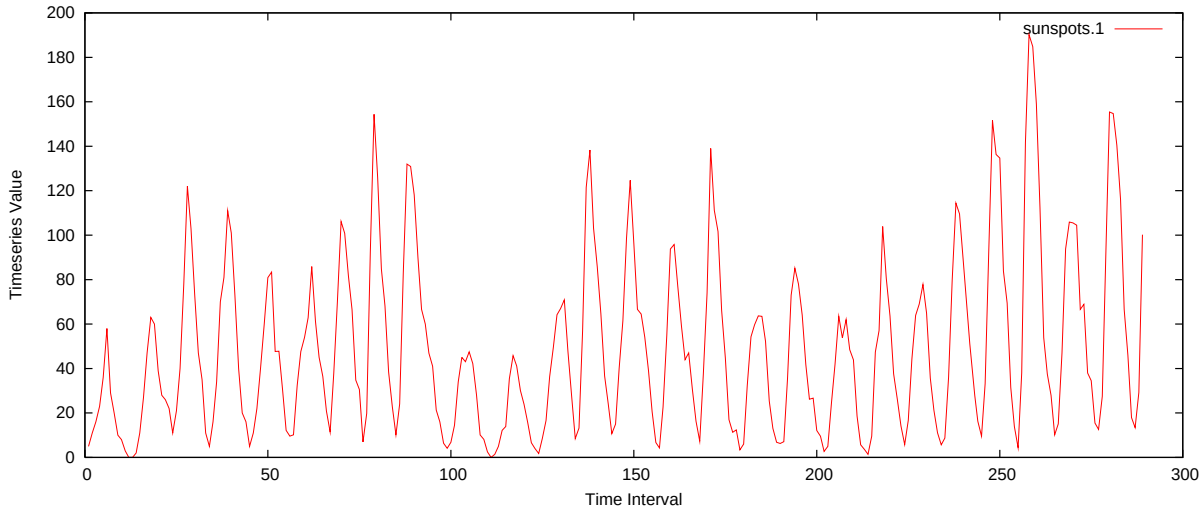

Abbildung 3: Sonnenflecken Zeitreihe

2.3.4 Umsatz von US-Haushalten Zeitreihe

- *Beschreibung*: Jährliche Umsätze von US-Haushalten in Tausend Dollar in der Zeit von 1965-1975.
- *Quelle*: `http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/data/abraham14.dat`
- *Datenpunkte*: 132
- *Visualisierung*:

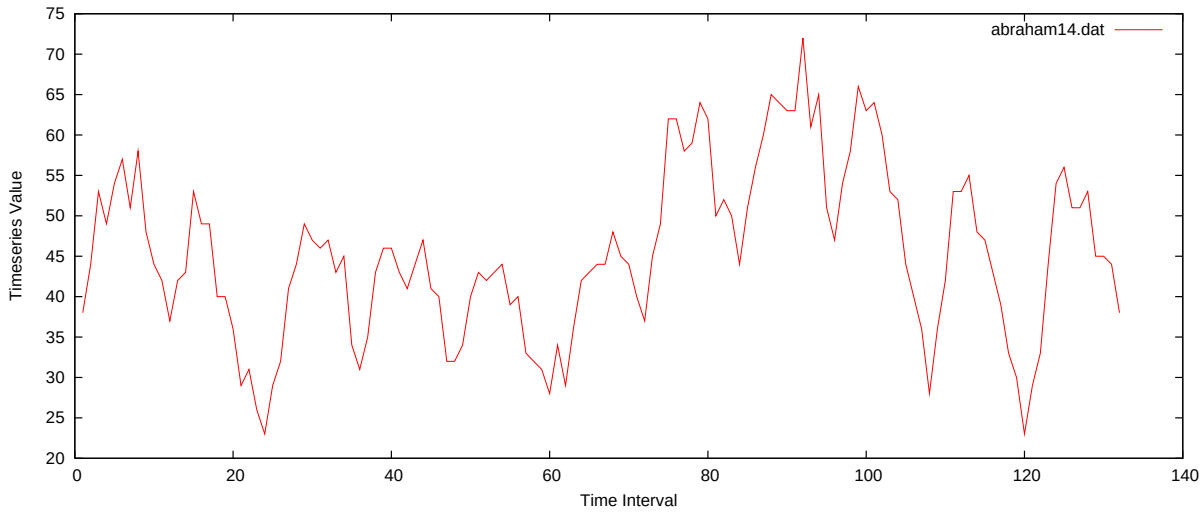

Abbildung 4: Umsätze von US-Haushalten Zeitreihe

3 Gene Expression Programming

Gene Expression Programming (*GEP*) wurde im Jahre 2001 von Candida Ferreira vorgestellt (s. Homepage unter [1]). Bei GEP handelt es sich nach [3] um eine Weiterentwicklung *genetischer Algorithmen* (*GAs*) und des *genetischen Programmierens* (*GP*). Ziel ist dabei wie bei GP die Evolution ausführbaren Codes, der zur Lösung eines vorgegebenen Problems eingesetzt wird. Wie auch bei GAs werden dabei Populationen von Individuen verwendet, die einer simulierten Evolution mit unterschiedlichen genetischen Operatoren unterworfen sind und von denen eine Güte berechnet werden kann, welche ausschlaggebend ist für den Fortbestand eines Individuums in der nächsten Generation. Der grundsätzliche Unterschied von GEP zu GAs und GP besteht im Aufbau der Individuen. Bei

GAs handelt es sich dabei um sogenannte Chromosomen, dies sind Zeichenketten fester Länge. Bei GP werden Bäume, sogenannte *parse trees*, unterschiedlicher Größe benutzt. Bei GEP kommen Chromosomen fester Länge als symbolische Zeichenketten zum Einsatz (*Genotypen*), die in *Ausdrucksbäume* (engl. *expression trees*) übersetzt werden (*Phänotypen*), deren Inorder-Traversierung der Code-Ausführung entspricht. Dies erlaubt eine erhöhte Ausdrucksmächtigkeit im Vergleich zu GAs und eine flexiblere Anwendung genetischer Operatoren im Vergleich zu GP. Details der Vorzüge von GEP werden hier nicht weiter verfolgt und können z. B. in [3] nachgelesen werden.

3.1 Datentypen

Prinzipiell findet eine Unterscheidung in *Genotypen* als Zeichenketten fester Länge und *Phänotypen* als Ausdrucksbäume variabler Größe statt. Genotypen werden zur Code-Generierung und -Ausführung in Phänotypen übersetzt, wobei es mit Genotypen fester Länge möglich ist Phänotypen unterschiedlicher Größe zu erzeugen, was eine einflussvolle Eigenschaft dieser beiden Datentypen ist und zu einer hohen Flexibilität bei der Code-Erzeugung führt. Im Gegensatz zu GP werden genetische Operationen auf den Genotypen und nicht auf den Ausdrucksbäumen ausgeführt, wodurch die syntaktische Korrektheit der Bäume leichter gewährleistet werden kann, wie noch zu sehen sein wird.

3.1.1 Genotypen

Generell ist ein Genotyp (Synonym „Chromosom“) eine sequentielle Aneinanderreihung von Symbolen (Zeichen, Gensymbole), bei denen zwischen *Funktionssymbolen* (*FS*) und *Terminalsymbolen* (*TS*) unterschieden wird. FS haben dabei eine bestimmte *Arität n*, bei TS kann es sich z. B. um die übergebenen Parameter an das zu evolvierende Programm handeln, ihnen lässt sich ein Wert zuweisen und mit einer Arität von 0 können sie als Spezialfall von FS angesehen werden. Der logische Aufbau eines Genotyps folgt einer hierarchischen Unterteilung in einzelne Gene, Gen-Kopf und Gen-Rest und schließlich Gensymbole (FS und TS). Ein Genotyp kann abhängig vom Problem aus mehreren *Genen* bestehen, die einzige Vorgabe ist dabei, dass alle Gene eines Genotyps dieselbe Länge bzgl. der Anzahl von Gensymbolen haben. Ein Gen besteht weiterhin aus einem *Gen-Kopf* und einem *Gen-Rest* (engl. *head* und *tail*). Kopf und Rest beinhalten dann die einzelnen *Gensymbole*, wobei im Gen-Kopf sowohl FS als auch TS in beliebiger Reihenfolge auftreten

dürfen, im Gen-Rest jedoch nur TS erlaubt sind (*Bedingung 1*). Die Längen von Kopf und Rest müssen dabei in einem fest vorgegebenem Verhältnis zueinander stehen (*Bedingung 2*). Sei die Länge des Gen-Kopfes h, die Länge des Gen-Restes t und die höchste Arität eines FS im Gen n. Dann muss folgende Gleichung erfüllt sein:

$$t = h \cdot (n - 1) + 1$$

Beide Bedingungen führen dazu, dass bei der Überführung von Genotypen in Phänotypen stets syntaktisch korrekte Ausdrucksbäume entstehen, d. h. der Baum derart abgeschlossen ist, dass in den Blättern nur TS vorhanden sind und alle inneren Knoten (bestehend aus FS) mit Söhnen besetzt sind.

Abbildung 5 verdeutlicht die Struktur eines Genotyps am Beispiel eines Chromosoms mit 3 Genen und eine Genkopf-Länge von 2.

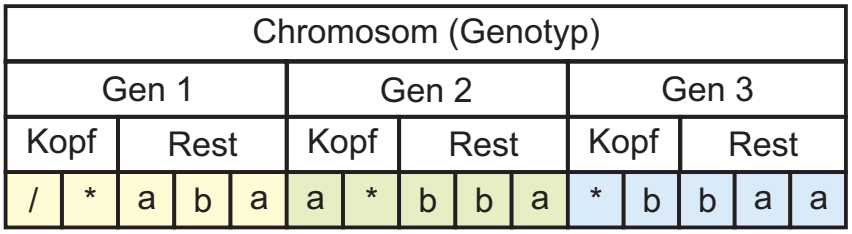

Abbildung 5: Beispiel eines Chromosoms aus 3 Genen

3.1.2 Phänotypen

Phänotypen entstehen durch die Überführung aus Genotypen und stellen Ausdrucksbäume (*expression trees*, *ETs*) dar, durch deren Inorder-Traversierung der in den Baumknoten enthaltene Code ausgeführt wird. Ein ET besitzt eine allgemeine Struktur, innere Knoten stellen im Genotyp enthaltene FS dar, die Anzahl der Söhne eines Knotens ist dabei durch die Arität des FS gegeben. Blattknoten bestehen ausschließlich aus TS und schließen den Baum nach unten hin ab. Unter diesen Bedingungen gilt ein ET als syntaktisch korrekt.

3.1.3 Datentyp-Überführungen

Jeder Genotyp lässt sich eindeutig in einen Phänotyp überführen. Die Übersetzung von Phänotypen in Genotypen ist zwar auch möglich, wird in der praktischen Anwendung des Algorithmus aber nicht benötigt und soll somit hier nicht weiter beschrieben werden.

Zunächst wird jedes Gen eines Genotyps in einen Ausdrucksbaum umgewandelt. Bei n Genen erhält man so n ETs (*Sub-ETs*), die im Anschluss durch eine vom Benutzer festgelegte *Verbindungsfunktion* (engl. *linking function*) miteinander kombiniert werden und so einen einzigen ET ergeben. Oft findet dabei lediglich der Additionsoperator Verwendung. Abbildung 6 zeigt den Pänotyp des zu Abbildung 5 gehörenden Genotyps auf. Die 3 zu den jeweiligen Genen gehörenden Sub-ETs werden mit dem Additionsoperator verbunden und ergeben somit einen einzigen Ausdrucksbaum.

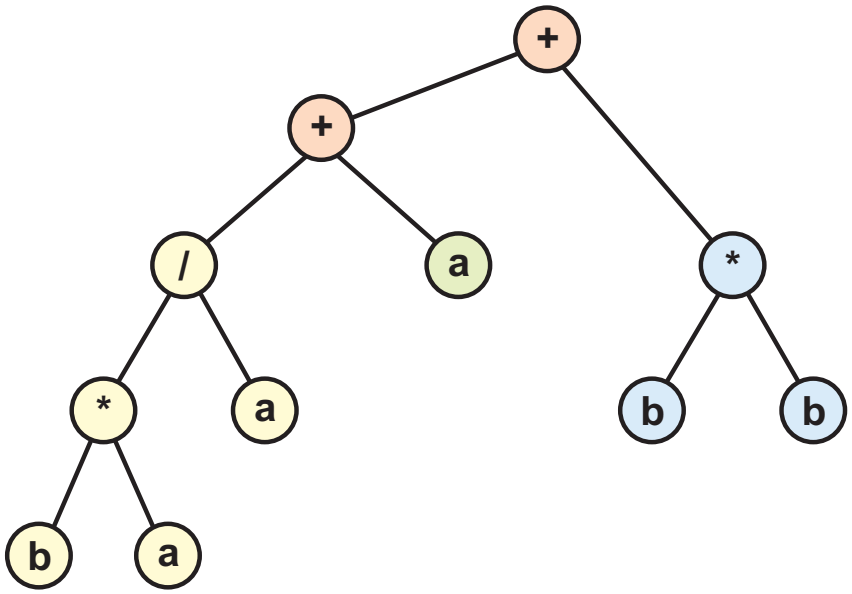

Abbildung 6: Phänotyp zum Genotyp aus Abbildung 5

Zur Transformation eines Gens in einen Sub-ET wird das Gen zeichenweise abgearbeitet, beginnend beim ersten Gensymbol von links gs_1. Implementiert werden kann die Übersetzung z. B. durch das Führen einer Liste l, welche „offene“ Gensymbole enthält, die durch Besetzen ihrer Söhne mit weiteren Gensymbolen geschlossen werden, also von der Liste entfernt werden. Ist l leer, so terminiert der Algorithmus. Zu Beginn der Übersetzung enthält l nur das erste Symbol gs_1 des Gens. Dieses wird entnommen und entsprechend seiner Arität n werden die nächsten n Symbole des Gens $gs_2 \dots gs_{1+n}$ betrachtet und sowohl als Söhne von gs_1 gesetzt als auch in ihrer vorgegebenen Reihenfolge zu l hinzugefügt. Durch die in Abschnitt 3.1.1 genannten Bedingungen ist dabei gewährleistet, dass die Länge des Gens nicht überschritten wird und sich somit syntaktisch korrekte Sub-ETs ergeben.

Abbildung 7 veranschaulicht die Wirkung dieses Algorithmus auf den Aufbau des Sub-ET bei der Transformation von Gen 1. Zusammenhängende Teilfolgen des Gens bilden die

einzelnen Ebenen des Teil-Ausdrucksbaum, dessen Errichtung damit einer ebenenweisen Konstruktion entspricht.

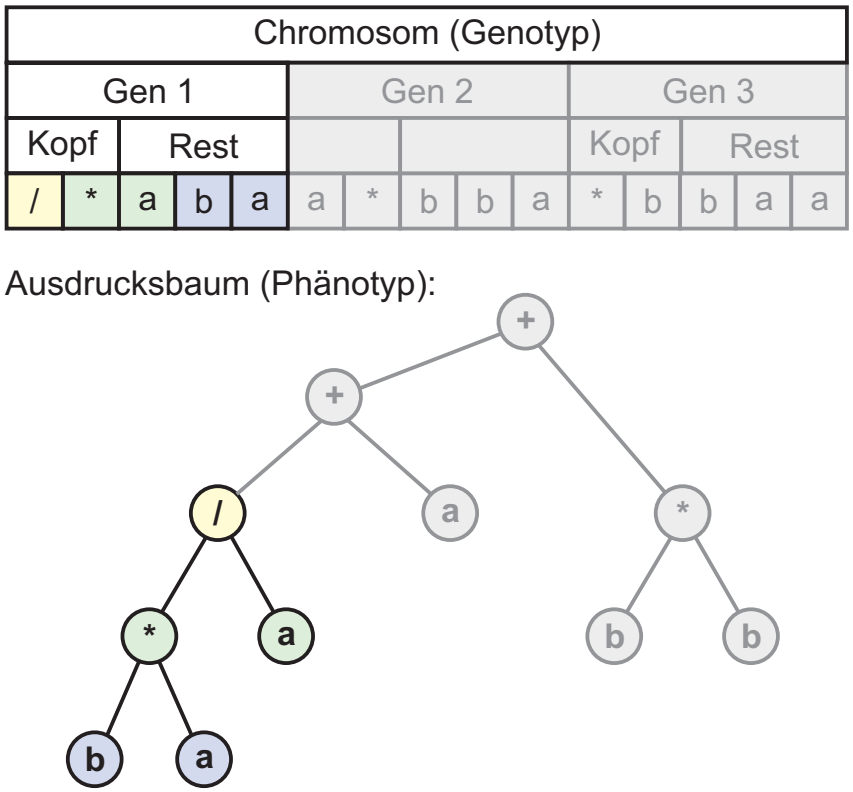

Abbildung 7: Transformation eines Gens in einen Sub-ET

Durch diesen Algorithmus ist erkennbar, dass sich bei der Überführung eines Gens der Länge k ein Sub-ET mit *höchstens* k Knoten ergibt. Ist die geführte Liste l leer, so terminiert der Algorithmus mit einem Sub-ET, der ebenso viele Knoten enthält wie Symbole im Gen betrachtet wurden. Dieser Gen-Teil wird auch als *kodierender Teil* (engl. *coding part*, *CP*) bezeichnet, da dessen Symbole im Ergebnis-Code wieder zu finden sind. Die außerhalb des CP liegenden Gensymbole haben in der aktuellen Generation keinen Einfluss auf den erstellten Code, können aber zum Tragen kommen, sobald Symbole im CP durch genetische Operatoren verändert werden. In den Extremfällen besteht der generierte Sub-ET nur aus einem Gensymbol (genau dann, wenn gs_1 ein Terminalsymbol ist) oder aus allen Symbolen des Gens. Dadurch ist es mit GEP möglich, aus Genotypen *fester* Länge Phänotypen *variabler* Größe zu erzeugen, wodurch sich ein flexibles Zusammenspiel dieser beiden Datentypen ergibt.

3.2 Algorithmus-Überblick

Die allgemeine Struktur des GEP-Algorithmus ergibt sich aus Abbildung 8.

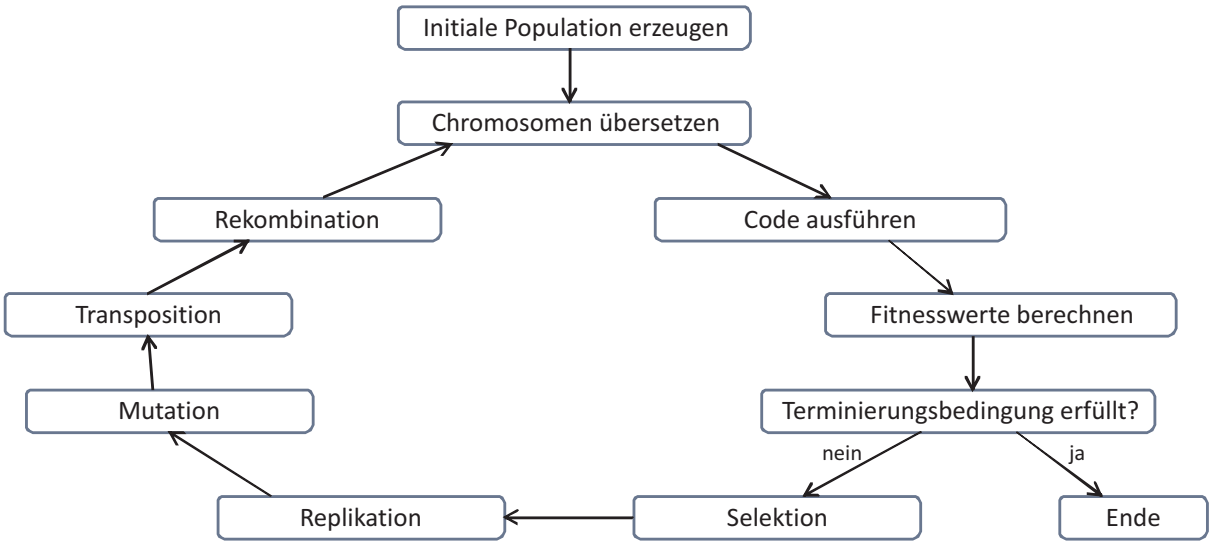

Abbildung 8: Flowchart des GEP-Algorithmus

Die initiale Population wird durch zufällige Generierung von Genotypen aus den vorgegebenen Parametern zu Gen-Länge und Gen-Anzahl sowie dem zu nutzenden Pool an Funktions- und Terminalsymbolen erzeugt.
Zur Code-Ausführung werden die Genotypen der aktuellen Generation in Phänotypen übersetzt und die entsprechenden Ausdrucksbäume inorder abgelaufen. Dadurch ergibt sich bei Belegung der Terminalsymbole mit Werten (z. B. Parameterwerten) an der Wurzel ein Ergebnis, welches in die Berechnung der Fitness des Individuums einfließt.
Die Terminierungsbedingung kann das Erreichen eines vorgegebenen Fitness- bzw. Fehlerwertes oder die Abarbeitung einer vorgegebenen Anzahl von Generationen sein.

3.3 Selektion und Replikation

3.3.1 Selektionsmethode

Als Selektionsmethode wird ein *fitnessproportionales rouletterad-basiertes Schema* in Kombination mit Replikation des besten Individuums (*simple elitism*) vorgeschlagen. Für n

Individuen einer Generation wird das Rouletterad $n-1$ mal gedreht. Das jeweils selektierte Individuum wird geklont und hält in die nächste Generation von Individuen Einzug. Das beste Individuum der aktuellen Generation wird automatisch in die nächste Generation übernommen. Dadurch ergeben sich in der nächsten Generation wieder n Individuen, womit die Populationsgröße stets bewahrt bleibt. Unter Berücksichtigung des Glücks beim Drehen des Rouletterads sowie der Fitness eines Individuums kann dieses sein Genmaterial in die nächste Generation übertragen.

Die Original-Ausarbeitungen zu GEP geben sich zu uneindeutig und zu knapp bezüglich der Verteilung von Chromosomen auf dem Rouletterad in Abhängigkeit von ihrer Fitness. Fitnessproportionalität erfordert ein Maximierungsproblem und eine Unter- sowie Obergrenze für Fitnesswerte, die durch die angegebene Fitnessfunktion nicht gegeben werden. Ebenfalls wird nicht beschrieben, wie mit Chromosomen umgegangen wird, deren Auswertung zu einem *NaN*-Ergebnis (*Not a Number*) oder zu *Infinity* führen, die also „nicht lebendig“ sind (z. B. aufgrund einer Division durch 0). Eine Alternative zur automatischen Zuordnung des absolut schlechtesten Fitnesswertes ist beispielsweise ein Wert, der einem Teil der schlechtesten Fitness der aktuellen Generation entspricht und womit ein solches Chromosom die Chance erhält in die nächste Generation zu gelangen. Dies könnte z. B. dann nützlich sein, wenn ein an sich gutes Chromosom durch Mutation ein „schlechtes“ Gensymbol erhalten hat. Das ansonsten „gute“ Genmaterial bekäme die Chance weiter gegeben zu werden und in der nächsten Generation wieder zum Tragen zu kommen.

Zur Vermeidung der beschriebenen Probleme wurde bei der vorliegenden Umsetzung ein *rangbasiertes Schemas* unter Beibehaltung der roulette-basierten Selektionsmethode verwendet. Dabei werden zunächst die n Individuen aufsteigend nach besser werdender Güte in einem Feld angeordnet. Der Anteil a_i am Rouletterad eines Individuums i ergibt sich dann wie folgt:

$$a_i = a_{i-1} + i^2, \quad a_0 = 1,\ 1 \leq i \leq n-1$$

Anstelle des Summands i^2 ist auch nur i denkbar, allerdings ergab sich hier in Tests ein zu geringer Selektionsdruck, der zu einem häufigen Verwerfen guter Individuen führte. Die Größe des Rouletterads ergibt sich aus a_{n-1}. Es ist erkennbar, dass mit dieser Methode alle Individuen die Chance erhalten in die nächste Generation kopiert zu werden, wobei besseren Individuen abhängig von ihrem Rang in der aktuellen Generation eine größere Wahrscheinlichkeit zukommt. „Nicht lebendige“ Individuen erhalten die schlechtesten Ränge, allerdings bleibt auch hier die Chance auf Replikation gewahrt.

3.3.2 Verwendete Fitnessfunktion

Durch die Verwendung eines rangbasierten Schemas spielt die tatsächlich zum Einsatz kommende Fitnessfunktion keine herausragende Rolle, sie muss nur die Güte der Individuen entsprechend ihrer Ränge widerspiegeln. Dementsprechend wurde in dieser Arbeit nur der durchschnittliche absolute Fehler E über alle N Trainingsdaten als Fitness eines Individuums verwendet, womit ein Minimierungsproblem gegeben ist:

$$E = \frac{1}{N-p} \cdot \sum_{i=p+1}^{N} |\tilde{x_i} - x_i|$$

$p \ldots$ Anzahl zu verwendender Parameter $i-p \ldots i-1$ zur Berechnung von x_i
$\tilde{x_i} \ldots$ Ist-Ergebnis für den i-ten Wert
$x_i \ldots$ Soll-Ergebnis des i-ten Wertes

$\tilde{x_i}$ ergibt sich aus der Übersetzung des aktuellen Individuums in einen Phänotyp, womit ein Syntaxbaum erhalten wird. Durch Setzen der Trainingsdaten $i-p \ldots i-1$ als Werte für die Terminalsymbole (welche die Parameter des Algorithmus darstellen) und Inorder-Traversierung des Baumes kommt es zu einer Code-Ausführung, die in der Wurzel mit dem gewünschten Ergebnis $\tilde{x_i}$ endet.

3.4 Genetische Operatoren

Allen genetischen Operatoren ist gemein, dass sie ausschließlich auf den Genotypen arbeiten und diese in ihrem Aufbau ändern, indem Genteile ausgetauscht oder verändert werden. Als Einschränkung muss dabei die Struktur der Gene erhalten bleiben um die Überführbarkeit in korrekte Ausdrucksbäume zu gewährleisten. Candida Ferreira gibt u. a. in [2] als Klassen genetischer Operatoren die Mutation, Rekombination und Transposition an, die im Nachfolgenden kurz vorgestellt werden. Die Ausführungen in den Originalarbeiten sind leider nicht eindeutig, sodass sich viele Fragen auftaten, die meist nur durch viel Probieren zumindest teilweise beantwortet werden konnten. Schlussendlich war es aufgrund dessen nicht möglich, alle Ergebnisse zu reproduzieren und Sicherheit darüber zu erlangen, die Umsetzung äquivalent zur Autorin durchgeführt zu haben.

3.4.1 Mutation

Bei GEP kommt nur eine Mutations-Art zum Einsatz, die ein Symbol zufällig gegen ein anderes aus dem Pool zu verwendender Gensymbole austauscht (*1-Punkt-Mutation*). Dazu werden alle Genotypen symbolweise betrachtet. Soll das Symbol gemäß der Mutationsrate mutiert werden, so ist auf die Wahrung der Korrektheit des Genotyps zu achten. D. h. im Gen-Kopf kann ein Austausch in Funktions- oder Terminalsymbole erfolgen, im Gen-Rest dürfen hingegen keine Funktionssymbole vorkommen und es ist damit lediglich aus den Terminalsymbolen ein Objekt auszuwählen.

Eine offene Frage bei der Mutation ist die Wahl von Funktions- und Terminalsymbolen (FS und TS). Wenn der Pool an Symbolen beispielsweise aus 4 FS und 12 TS besteht, so ist die Wahrscheinlichkeit ein TS zu wählen dreimal größer als die der Wahl eines FS. Dies führt allerdings zu einer unausgewogenen Evolution. Wie die Wahl tatsächlich durchgeführt wird verschweigen alle GEP-Artikel. Gelöst wurde dies in der vorliegenden Arbeit, indem zunächst festgelegt wird, ob in ein Funktions- oder Terminalsymbol mutiert werden soll. Dann wird aus dem jeweiligen Symbol-Pool ein Symbol zufällig ausgewählt. Somit wird mit der gleichen Wahrscheinlichkeit in ein FS oder TS mutiert.

3.4.2 Rekombination

Standardmäßig gibt es 3 Rekombinationsarten: 1-Punkt-, 2-Punkt- und Gen-Rekombination. Bei der Anwendung der Rekombination werden alle Genotypen betrachtet und es werden die jeweiligen Rekombinationsraten angesetzt. Soll ein Genotyp einer Rekombination unterworfen werden, so wird zufällig ein anderer Genotyp ausgewählt und die beiden Individuen werden miteinander rekombiniert. Diese Vorgehensweise wird in den Originalartikeln nicht eindeutig beschrieben, hat sich in eigenen Tests jedoch als praktikabel herausgestellt. Alle Rekombinationsarten erfordern, dass die involvierten Individuen die gleiche Struktur bzgl. der Anzahl an Genen und der Länge der Gen-Köpfe besitzen. Im Standard-GEP ist dies stets der Fall, bedeutet jedoch eine Einschränkung bei der in Abschnitt 4 vorgestellten Erweiterung.

Bei der *1-Punkt-Rekombination* wird zufällig ein Punkt der Genotypen gewählt und das gesamte Genmaterial ab diesem Punkt wird zwischen den selektierten Genotypen ausgetauscht (1-Punkt-Crossover).
Bei der *2-Punkt-Rekombination* werden zwei Punkte der Genotypen gewählt und das

gesamte Genmaterial zwischen einschließlich diesen Punkten wird ausgetauscht (2-Punkt-Crossover).
Bei der *Gen-Rekombination* wird zufällig ein Gen gewählt, welches zwischen den beiden Genotypen ausgetauscht wird.

3.4.3 Transposition

Transposition beschreibt das Kopieren von Genmaterial innerhalb eines Individuums. Dazu werden die Genotypen der Reihe nach betrachtet und an jedem Genotyp werden die verschiedenen Transpositionsraten angesetzt. Unterschieden wird zwischen 3 Transpositionsarten: IS-, RIS- und Gen-Transposition.

Mit Durchführung der *IS-Transposition* (IS = *insertion sequence*) werden Genteile (Transposons) von einer beliebigen Stelle des Genotyps an eine andere Stelle kopiert. Die Einfügestelle muss sich im Kopf eines Gens befinden, dabei darf es sich nicht um das erste Symbol handeln. Der eingefügte Genteil darf nicht über den Genkopf hinaus ragen, da ansonsten die Struktur des Gens nicht gewährleistet werden kann. Hinter der Einfügestelle befindliche Gensymbole im Genkopf werden nicht ersetzt, sondern nach rechts verschoben, bis das Ende des Genkopfes erreicht ist. Die Länge des Transposons wird zufällig gewählt und darf eine vom Benutzer festgelegte obere Schranke nicht überschreiten. Das Transposon wird durch die Operation kopiert, d. h. es kommt nach erfolgter IS-Transposition sowohl am ursprünglichen Ort als auch am Einfügeort im Gen vor.

Die *RIS-Transposition* (RIS = *root insertion sequence*) ähnelt der IS-Transposition, nur dass es sich bei RIS-Transposons um Genteile handelt, die mit einem Funktionssymbol beginnen. Dazu wird ein zufälliger Punkt im Kopf eines Gens gewählt und es wird absteigend bis zum Genkopf-Ende nach dem nächsten FS gesucht. Wird kein FS gefunden, so terminiert die Operation. Ansonsten wird ein weiteres Gen ausgewählt und das Transposon wird an die erste Position dieses Gens kopiert. Die Länge des Transposons wird wiederum durch eine vom Benutzer festgelegte obere Schranke limitiert und Gensymbole ab der Einfügestelle im Genkopf werden nach rechts verschoben.

Bei der *Gen-Transposition* erfolgt entgegen der anderen Transpositionsarten keine Dopplung von Genteilen durch Kopie. Stattdessen werden zwei Gene zufällig gewählt und gegeneinander ausgetauscht. Es ist zu beachten, dass dieser Operator keinen weitergehenden Effekt auf Genotypen hat, deren Gene bei der Übersetzung in einen Phänotyp mittels einer kommutativen Funktion miteinander verbunden werden.

3.5 Probleme

Abgesehen von der schwierigen originalgetreuen Umsetzbarkeit des beschriebenen GEP-Systems gibt es noch weitere Probleme bzw. Faktoren, welche im GEP-Algorithmus selbst liegen und die Güte der evolvierten Problemlösung schmälern. So ist der zu verwendende *Funktionssatz* wie auch viele andere Parameter explizit vom Benutzer zu wählen, der damit einen großen Einfluss auf die Evolution hat und über entsprechendes Hintergrundwissen verfügen muss. Das größte Problem hinsichtlich der Zeitreihenanalyse stellt aber die ausschließliche Generierung und Verwendung *impliziter Konstanten* dar. So können feine Parameter-Zusammenhänge nicht durch angepasste Konstanten modelliert werden, sondern sind mit Hilfe von Funktionen zu beschreiben. Dadurch ist GEP nicht in der Lage, einen fein abgestimmten Algorithmus zu entwickeln.

3.6 Parametereinstellungen

Standardmäßg sollen die in Tabelle 1 angegebenen Parameter verwendet werden. Sie entsprechen grob den Vorschlägen, die für Parameter in der textuellen Beschreibung und bei unterschiedlichen Problemen in [3] gegeben werden und wurden durch Tests an das gegebene Problem angepasst.

Parameter	Wert
Generationen	1000
Populationsgröße	50
Funktionssatz	+ - * /
Parameterzahl	12
Trainingsteil	90%
Genkopf-Länge	10
Gene pro Chromosom	3
Verbindungsfunktion	+
Selektionsmethode	Rangbasiertes Roulette
Mutationsrate	0.03
1-Punkt Rekombinationsrate	0.3
2-Punkt Rekombinationsrate	0.3
Gen Rekombinationsrate	0.1
IS Transpositionsrate	0.1
RIS Transpositionsrate	0.1
Gen Transpositionsrate	0.1
max. IS Transposon-Länge	3
max. RIS Transposon-Länge	3

Tabelle 1: Parametereinstellungen für GEP

3.7 Hypothesentests

In den folgenden Unterabschnitten werden beispielhaft einzelne Tests zur Wahl günstiger Parameter für GEP durchgeführt. Dabei ist bekannt, dass für eine umfassende Betrachtung alle möglichen Parameterkombinationen betrachtet werden müssten, da Abhängigkeiten zwischen einzelnen Parametern bestehen. Weiterhin hängen die Parameter von der untersuchten Zeitreihe ab und können für andere Zeitreihen teilweise abweichen. Da diese Problematik nicht in vertretbarer Zeit zu behandeln war und diese Ausarbeitung den Fokus auf den Vergleich unterschiedlicher Verfahren legt, wurde diese Zielsetzung nicht weiter verfolgt. Als behandelte Zeitreihe kommt die in Abschnitt 2.3.1 vorgestellte *Furnas Zeitreihe* mit 576 Datenpunkten zum Einsatz, die sowohl periodische als auch zufällige Einflüsse beinhaltet.

Die nachfolgenden Tests wurden unter den folgenden Rahmenbedingungen durchgeführt: als Fehlermaß kam der in Abschnitt 2.2 vorgestellte MAE zum Einsatz. Unter einer bestimmten Parametereinstellung wurden 25 Testläufe durchgeführt. Das Ergebnis ergab sich dann aus dem arithmetischen Mittel des MAE der jeweils besten Genotypen aller durchgeführten Läufe. Bei Variieren eines Parameters wurden die anderen Parameter wie in Tabelle 1 angegeben gewählt.

3.7.1 Einfluss der Mutationsrate

Hypothese: Es wird vermutet, dass eine zu kleine Mutationsrate zu einer langsamen Evolution führt bzw. zu einer Stagnation auf einem Punkt. Eine zu hohe Mutationsrate sollte alle Genotypen zu schnell verändern, was unerwünscht ist und die besten Genotypen zerstört. Die Güte dürfte in diesem Fall starke Varianzen zwischen einzelnen Generationen aufweisen. Entsprechend der Originalarbeiten zu GEP wird vermutet, dass eine „gute“ Mutationsrate im Bereich von 0.05 zu finden ist.

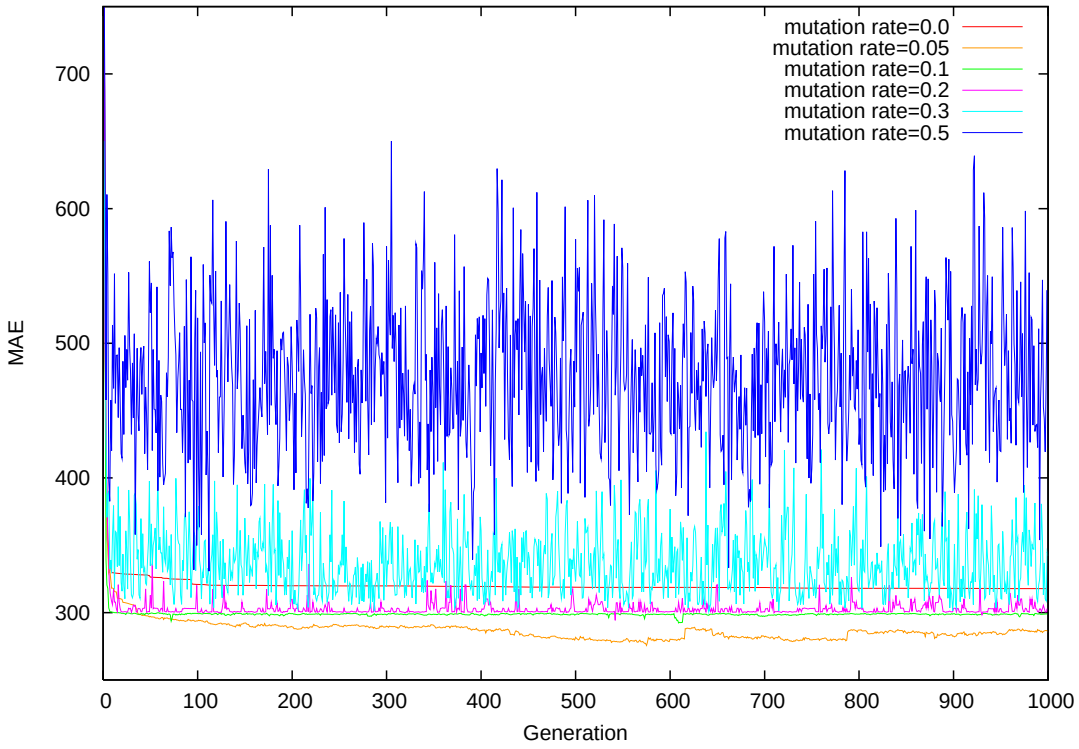

Abbildung 9: Einfluss von Mutationsraten auf die Evolution bei GEP

Auswertung: Abbildung 9 bestätigt die ursprüngliche Vermutung. Bei einer Mutationsrate

von 0.0 stagniert die Evolution, bei zu hohen Mutationsraten (hier ab 0.2) werden zu viele Gensymbole auf einmal mutiert, sodass sich keine zielgerichtete Evolution ergibt. Der optimale Bereich liegt demnach zwischen 0.0 und 0.1, ein zweiter Versuch in diesem Intervall soll Klarheit schaffen und wird in Abbildung 10 dargestellt.

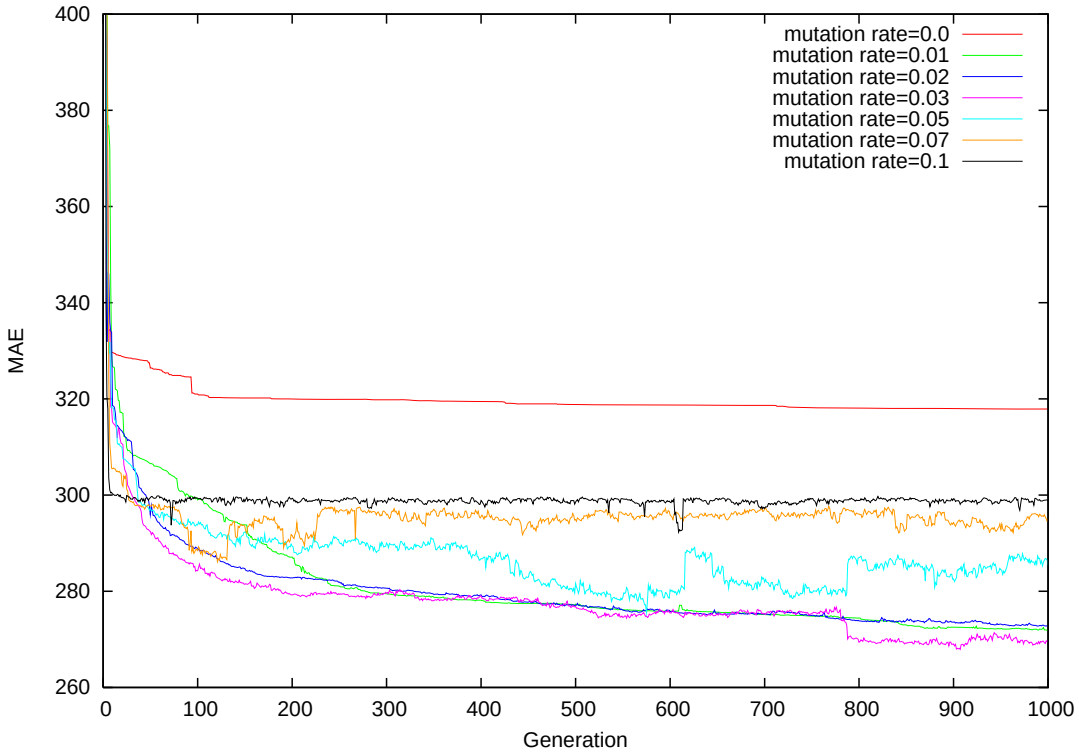

Abbildung 10: Einfluss abgestufterer Mutationsraten auf die Evolution bei GEP

Auswertung: Die beste Evolution findet mit Mutationsraten im Bereich 0.1 bis 0.3 statt und liegt damit unter den in den Originalartikeln zu GEP angegebenen Werten. Kleinere und größere Mutationsraten sind aus o. a. Gründen eher hinderlich für die Evolution.

3.7.2 Einfluss der Populationsgröße

Hypothese: Eine zu kleine Population wird dazu führen, dass die Güteentwicklung stark variiert, da die Veränderung weniger Genotypen bereits einen großen Teil der Gesamtpopulation umfasst. Große Populationen bedeuten eine Glättung der Güteentwicklung, zu große Populationen könnten aber dazu führen, dass die Güte zu lange in einem lokalen Optimum verharrt und temporäre Verschlechterungen nicht akzeptiert werden, da noch

genug andere Genotypen mit einer besseren alten Güte existieren. Dies könnte unter Umständen den Fortschritt der Evolution hemmen. Auf jeden Fall steigt die Laufzeit mit steigender Populationsgröße linear an.

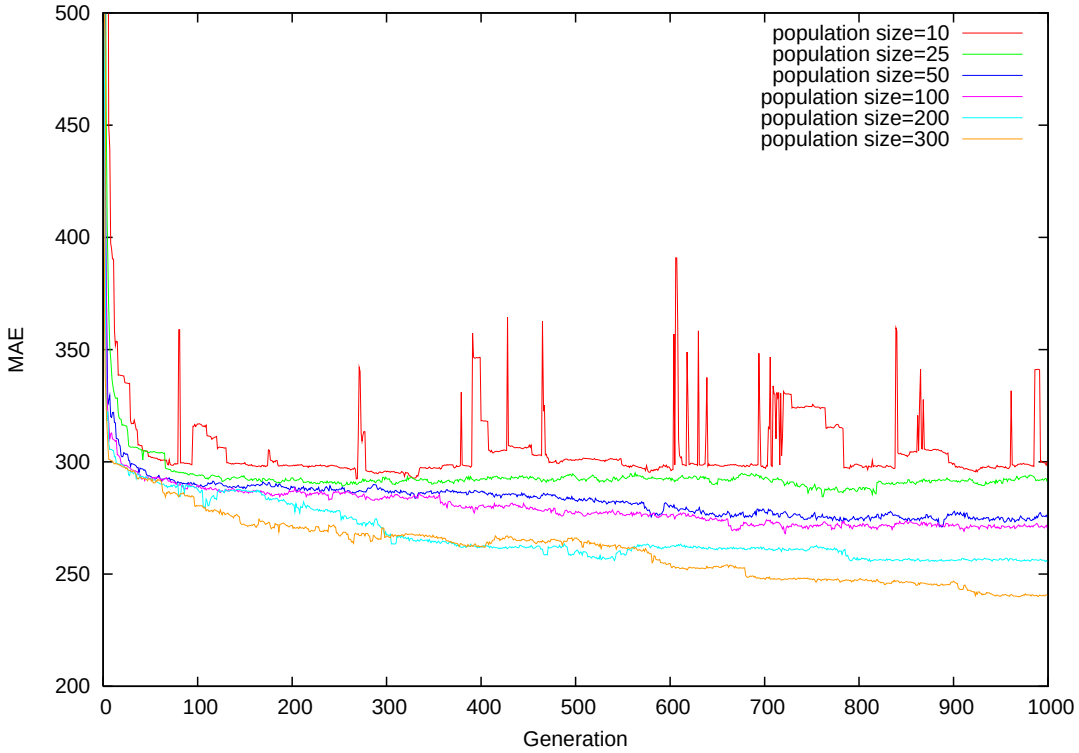

Abbildung 11: Einfluss der Populationsgröße auf die Evolution

Auswertung: Abbildung 11 zeigt wie erwartet starke Schwankungen zwischen einzelnen Generationen bei einer kleinen Population von 10 Individuen. Auch mit 25 Individuen ist dieses Problem noch in abgeschwächter Form vorhanden. Größere Populationen führen in einem beschränkten Rahmen offensichtlich zu einer besseren Güteentwicklung, da mehr genetisches Material zur Verfügung steht, aus dem neue Individuen gebildet werden können. Eine Hemmung der Evolution ist dabei nicht zu erkennen. Eine Populationsgröße von 50 stellt dennoch eine gute Wahl dar, da dies einen guten Kompromiss zwischen Güteentwicklung und Laufzeit darstellt. Diese fällt bei einer Population mit 300 Individuen ca. 6 mal so hoch aus und rechtfertigt nur in Ausnahmefällen die evolutionären Vorteile.

3.7.3 Einfluss der Genkopf-Länge

Hypothese: Es wird sich der Fall ergeben, dass bei einer zu kleinen Gen-Länge kein passender Algorithmus evolviert werden kann und die Evolution stagniert. Zu große Genlängen stellen theoretisch kein Hindernis dar, weil damit variable kleinere Phänotypen erzeugt werden können. In der Praxis könnte es für GEP jedoch schwierig sein, einen Algorithmus genau der richtigen Länge zu konstruieren.

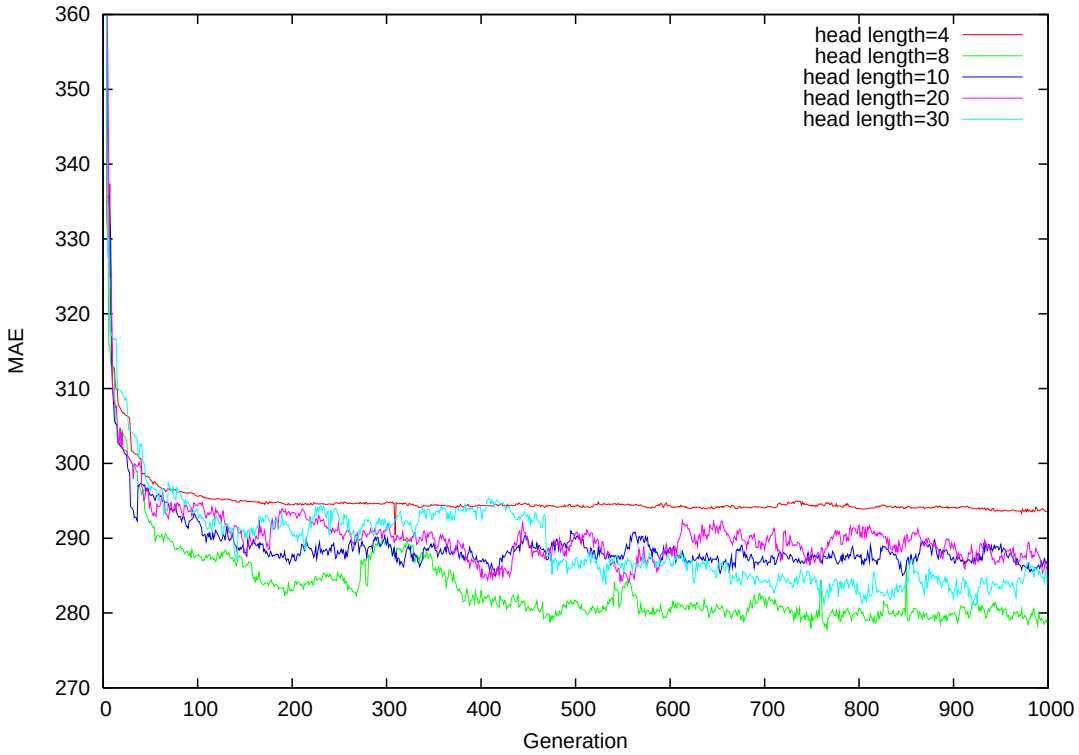

Abbildung 12: Einfluss der Genkopf-Länge auf die Evolution

Auswertung: Aus Abbildung 12 geht hervor, dass eine Genkopf-Länge von 4 für die Evolution zu klein ist. In diesem Fall verbleibt der MAE auf einem hohen Wert. Darüber hinaus liefert eine Länge von 8 für den Genkopf hier das beste Ergebnis, allerdings sind die Schwankungen zu den anderen Längen gering. Selbst sehr lange Genköpfe mit 20 oder 30 Gensymbolen liefern vergleichbare Ergebnisse, d. h. GEP ist auch hier in der Lage konkurrenzfähige Algorithmen zu evolvieren. Es muss wiederum beachtet werden, dass die Laufzeit mit der Genkopf-Länge linear ansteigt und somit eine eher kleine Genkopf-Länge von 8–10 zu wählen ist.

4 Erweitertes Gene Expression Programming

Die Schwächen des Standard-GEP bei der Zeitreihenmodellierung nahmen weitere Forscher zum Anlass, um neue verbesserte Systeme basierend auf GEP zu entwickeln. Ein System stellt das in [5, 6] vorgestellte *EGIPSYS* (**E**nhanced **G**ene Express**I**on **P**rogramming for **SY**mbolic regression problem**S**) dar, das speziell auf Probleme der symbolischen Regression und Zeitreihenanalyse abgestimmt ist. Die dabei eingebrachten Modifikationen von GEP werden in den nachfolgenden Unterabschnitten präsentiert.

4.1 Konstantensymbole

In EGIPSYS wird neben Terminal- und Funktionssymbolen eine dritte Art von Gensymbolen eingeführt, die *Konstantensymbole*. Konstantensymbole sind ein Spezialfall von Terminalsymbolen mit dem Unterschied, dass sie nicht den variablen Parameter eines Algorithmus als Wert enthalten, sondern eine zuvor festgelegte Konstante, die unabhängig von Parameteränderungen ist und somit explizit in die Berechnung mit einfließt.

Die Nutzung von Konstanten kann über einen vom Benutzer einstellbaren Parameter reguliert werden. Über diesen gibt er an, mit welcher Wahrscheinlichkeit ein Gensymbol als eine Konstante generiert bzw. in eine Konstante mutiert werden soll. Zwei weitere Parameter geben an, in welchem Bereich eine Konstante bei ihrer Erzeugung liegen kann. Der Initialwert wird dann als Zufallszahl in diesem Bereich ermittelt und der Konstanten zugewiesen.

4.2 Modifizierte Mutation

Der Mutationsoperator des GEP wurde unter den genetischen Operatoren am stärksten modifiziert, vor allem in Hinblick auf den Umgang mit Konstanten. Zum einen kann in Abhängigkeit der Mutationsrate und der Konstanten-Nutzungsrate ein beliebiges Gensymbol in eine Konstante mit einem zufälligen Wert aus einem beschränkten Wertebereich umgewandelt werden. Wird der Mutationsoperator auf ein Konstantensymbol angewendet, so sind zwei Operationen möglich. Entweder wird das Konstantensymbol gegen ein anderes zufälliges Gensymbol ausgetauscht oder es wird ein zufälliger kleiner Betrag zum Wert der Konstante addiert. Die Wahrscheinlichkeit jedes Ereignisses liegt bei jeweils 50%. Soll ein

Betrag zur Konstante addiert werden, so wird mit gleicher Wahrscheinlichkeit zufällig entschieden, ob der Betrag positiv oder negativ ist. Dann wird zufällig ein Wert ermittelt, der nicht größer als 10% des Konstantenwertes ist und dieser wird addiert bzw. subtrahiert.

4.3 Lokale Suche

Der lokale Suchoperator wird nach allen anderen genetischen Operationen am Ende einer Generation durchgeführt und hat die Funktion, den Wert der Konstanten eines Individuums so anzupassen, dass sich für dieses eine bessere Güte ergibt. Nach Anwendung des Operators ergibt sich auf jeden Fall eine Güte, die gleich oder besser ist als der vorherige Gütewert. Mit der lokalen Suche ergibt sich somit eine Fein-Abstimmung der Konstanten, welche zu einer lokalen Optimierung der Gütewerte führt.

Der Suchoperator wird in 2 Schritten angewendet. Im ersten Schritt werden solange 10% zum Konstantenwert hinzu addiert, bis sich keine Güteverbesserung mehr ergibt oder bis eine maximale Zahl von 10 Durchläufen erreicht ist. Sollte sich im ersten Schritt eine Verschlechterung der Güte einstellen, so wird dieselbe Prozedur unter Anwendung von Werte-Subtraktion durchgeführt. Wurden weniger als 10 Durchläufe in diesem ersten Schritt durchgeführt, so befindet sich ein besserer Konstantenwert zwischen den beiden zuletzt ermittelten Werten und es wird versucht diesen im zweiten Schritt zu ermitteln. Dazu wird wiederum in 10 Durchläufen jeweils der Mittelwert zwischen den beiden begrenzenden Konstantenwerten berechnet und die Güte dafür ermittelt. Je nach Verbesserung oder Verschlechterung der Güte wird der untere oder obere begrenzende Konstantenwert auf den Mittelwert gesetzt, sodass sich eine immer engere Eingrenzung des „optimalen“ Wertebereichs der Konstanten ergibt.

Der Suchoperator ist sehr rechenaufwendig und dadurch nur mit viel Vorsicht einzusetzen. Bei der Anwendung werden alle Genotypen und für jeden Genotyp alle Gensymbole sequentiell durchlaufen. Bei Konstantensymbolen wird der lokale Suchoperator unter Berücksichtigung einer Nutzungsrate angewendet. Eine geringe Nutzungsrate von 1–5% hat sich in Tests als guter Kompromiss zwichen Werte-Adaption und Rechenzeit herausgestellt. Der Einfluss der lokalen Suche sollte allerdings nicht überschätzt werden, wie auch beim entsprechenden Hypothesentest in Abschnitt 4.7.3 zu sehen sein wird.

4.4 Variable Chromosomen

Im Standard-GEP wird die Länge des Gen-Kopfes als vom Benutzer vorgegebener Parameter festgelegt. EGIPSYS versucht durch Chromosomen unterschiedlicher Länge dieses System zu flexibilisieren, da es im normalen GEP ein offenes Problem darstellt, eine optimale Gen-Länge zu finden. Im Allgemeinen erfordern kompliziertere Probleme längere Gene (s. [2]).

In EGIPSYS werden mit der initialen Population Chromosomen erzeugt, die Gene jeweils gleicher Länge enthalten, unter den Chromosomen variiert die Gen-Länge allerdings. Dazu sind vom Benutzer zwei begrenzende Gen-Längen einzugeben. Zwischen diesen beiden Grenzen werden 50% der Chromosomen gleichverteilt erzeugt, die anderen 50% erhalten Gene mit zufälligen Längen aus demselben Bereich.

Die Nutzung verschieden langer Chromosomen hat Auswirkungen auf die Rekombination. Alle 3 Rekombinationsarten dürfen hier nur noch zwischen solchen Chromosomen durchgeführt werden, deren Gene dieselbe Länge besitzen.

In eigenen Tests stellte sich die Nutzung dieser Methode unterschiedlich langer Chromosomen als nicht vorteilhaft im Vergleich zu Genotypen fester Länge heraus. Der Grund liegt in der Zufälligkeit der initialen Population. So können Chromosomen einer im Hinblick auf die Lösung des vorliegenden Problems schlechten Länge zufällig so mit Gensymbolen belegt werden, dass sie gegenüber Chromosomen guter Länge bevorteilt sind. In den ersten Generationen kommt es dann zu einem Aussterben temporär schlechter Chromosomen, womit sich eine Länge durchsetzt und dann im gesamten evolutionären Prozess verwendet wird. Chromosomen variabler Länge haben somit keine positiven Auswirkungen auf die Evolution, auch ist ihre Rolle dadurch gehemmt, dass bei der Übersetzung von Genotypen fester Länge Phänotypen variabler Größe erzeugt werden. Dadurch ist es zwar nötig eine hinreichende Größe für Genotypen festzulegen, die tatsächlich benötigte Länge wird dann allerdings selbstständig im evolutionären Prozess ermittelt.

4.5 Selektion

4.5.1 Alternative Selektionsmethode

In Ergänzung zum rouletterad-basierten Selektionsschema von GEP (s. Abschnitt 3.3) wird für EGIPSYS eine *Turnierselektion* vorgeschlagen. Dabei werden bei n Individuen

n Turniere durchgeführt. An jedem Turnier nimmt ein vom Benutzer festzulegender Prozentsatz der Individuen teil. Das Individuum mit der besten Fitness gewinnt und wird in die nächste Generation kopiert. Die Turnierselektion führt insbesondere bei hohen Turniergrößen zu einem starken Selektionsdruck, bei dem das stärkste Individuum oft kopiert wird. Außerdem hat das schwächste Individuum keine Chance auf die Weitergabe seines Genmaterials.

4.5.2 Fitnessfunktion

Wie auch bei der in Abschnitt 3.3 vorgestellten Selektion spielt die tatsächliche Fitnessfunktion bei EGIPSYS eine untergeordnete Rolle, da die Turnierselektion keine quantitativen Relativitäten zwischen Individuen berücksichtigt, sondern nur den absoluten Sieger des Turniers ermittelt. Dementsprechend wird hier dieselbe Funktion E verwendet, die auch für GEP in Abschnitt 3.3.2 definiert wurde.

In [5] werden zwei Fitnessfunktionen vorgeschlagen, die zwei zusätzliche Parameter erfordern und anscheinend nur einer verbesserten Visualisierung der Fitnesswerte dienen. Auch aufgrund der Vergleichbarkeit des hier implementierten Systems zu GEP und KNN wird auf diese Fitnessfunktionen nicht zurück gegriffen.

4.6 Parametereinstellungen

Die in EGIPSYS verwendeten Parameter entsprechen einem erweiterten Satz der Parameter von GEP. Demzufolge sind viele Parameter entsprechend ihrer GEP-Vorschläge übernommen worden. Tabelle 2 gibt die zum Einsatz kommenden Parameter an.

Parameter	Wert
Generationen	1000
Populationsgröße	50
Funktionssatz	+ - * /
Parameterzahl	12
Trainingsteil	90%
min. Genkopf-Länge	8
max. Genkopf-Länge	12
Gene pro Chromosom	3
Verbindungsfunktion	+
Konstantennutzungsrate	0.33
Min. initialer Konstantenwert	0
Max. initialer Konstantenwert	100
Rate für lokale Suche	0.01
Selektionsmethode	Turnierselektion
Turniergröße	10%
Mutationsrate	0.02
1-Punkt Rekombinationsrate	0.3
2-Punkt Rekombinationsrate	0.3
Gen Rekombinationsrate	0.1
IS Transpositionsrate	0.1
RIS Transpositionsrate	0.1
Gen Transpositionsrate	0.1
max. IS Transposon-Länge	3
max. RIS Transposon-Länge	3

Tabelle 2: Parametereinstellungen für erweitertes GEP

4.7 Hypothesentests

4.7.1 Einfluss der Mutationsrate

Hypothese: Hier soll untersucht werden, ob sich wie vermutet die Ergebnisse der Wahl der Mutationsrate von GEP in Abschnitt 3.7.1 auch auf das erweiterte GEP übertragen lassen.

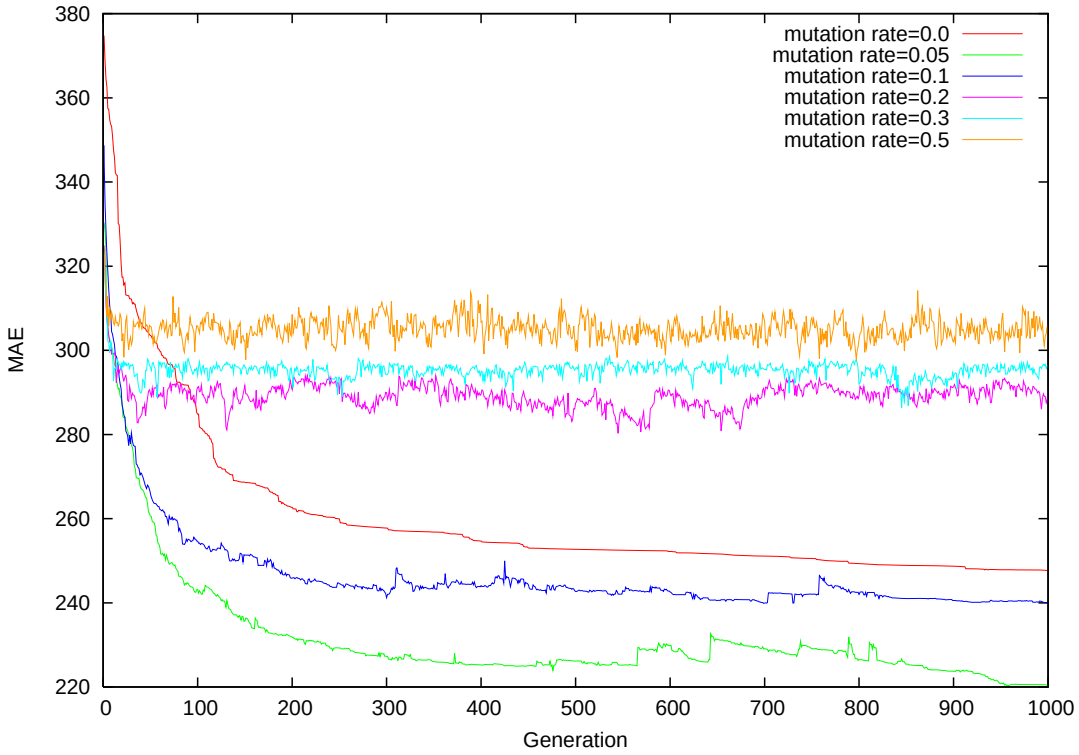

Abbildung 13: Einfluss von Mutationsraten auf die Evolution bei GEP

Auswertung: Abbildung 13 bestätigt die Vermutung, dass wiederum der Bereich von 0.0 bis 0.1 für eine optimale Mutationsrate in Frage kommt. Bei zu geringer Mutationsrate schreitet die Evolution langsam voran, im Gegensatz zu GEP in Abbildung 9 findet aber bei einer Mutationsrate von 0.0 noch eine sichtbare Verbesserung in höheren Generationen statt. Erklärt werden kann dies mit den übrigen genetischen Operatoren und der lokalen Optimierung von Konstantenwerten, was zu einer Güteverbesserung führt. Mutationsraten größer als 0.1 führen auch wie bei GEP zu einer nicht zielgerichteten Evolution, da zu viele Symbole eines Chromosoms in einem Schritt verändert werden. Abbildung 14 stellt noch einmal eine feinere Untersuchung der Mutationsrate im Bereich von 0.0 bis 0.1 dar.

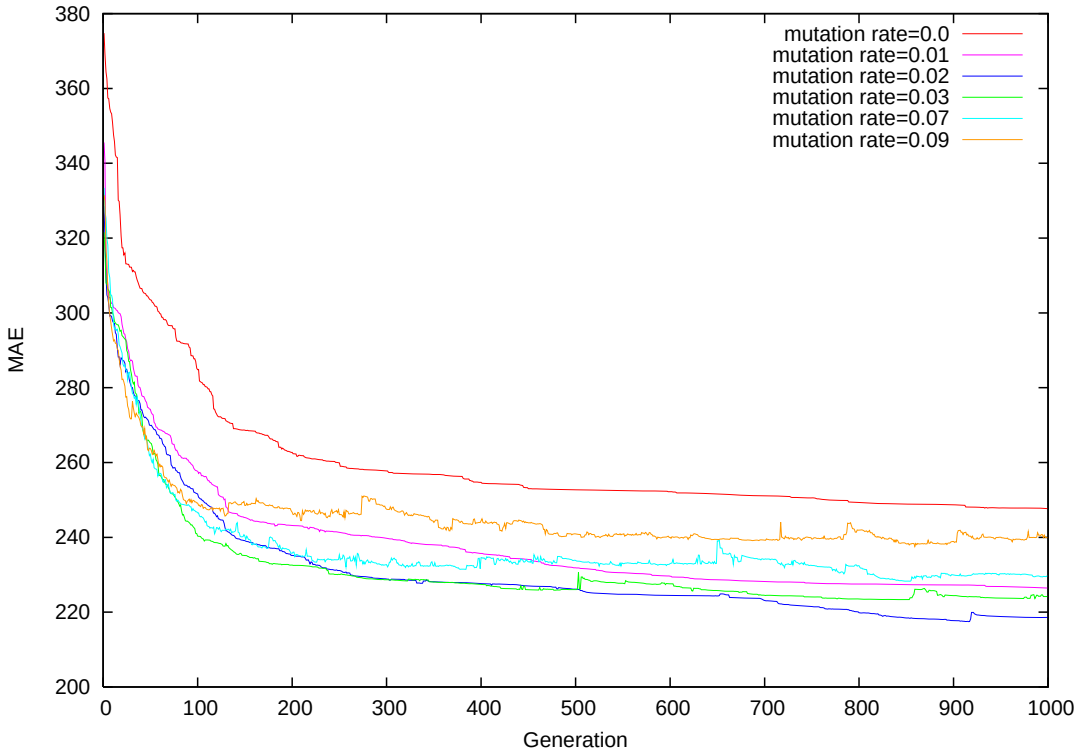

Abbildung 14: Einfluss abgestufterer Mutationsraten auf die Evolution bei GEP

Auswertung: Es ist erkennbar, dass ein optimaler Wert für die Mutationsrate bei ca. 0.02 liegt und damit dem Optimum für GEP in Abschnitt 3.7.1 nahe kommt.

4.7.2 Einfluss der Selektionsmethode

Hypothese: Hier werden rangbasierte Rouletteselektion und Turnierselektion mit ihrem Einfluss auf die Evolution verglichen. Eine Prognose, welche Methode besser funktionieren wird, ist dabei zunächst nicht möglich.

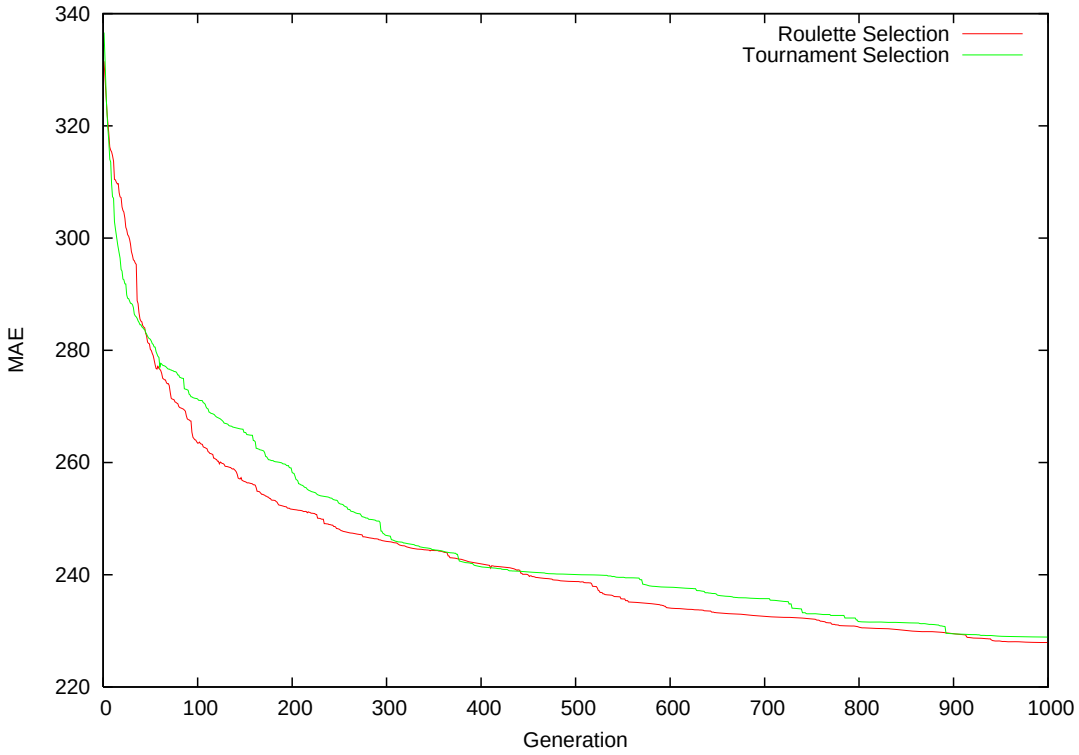

Abbildung 15: Einfluss der Selektionsmethode auf die Evolution

Auswertung: Abbildung 15 lässt keinen aussagekräftigen Unterschied zwischen beiden Selektionsarten erkennen. Da die rangbasierte Rouletteselektion mit einem höheren Aufwand verbunden ist, wird hier die einfache Turnierselektion favorisiert.

4.7.3 Einfluss der Rate für die lokale Suche von Konstantenwerten

Hypothese: Hier wird untersucht, welche Vorteile die Wahl einer größeren Rate zur Anpassung der Konstantenwerte mittels lokaler Suche mit sich bringt. Es wird vermutet, dass eine höhere Rate zu einer besseren Evolution führt, da Konstanten feiner abgestimmt werden.

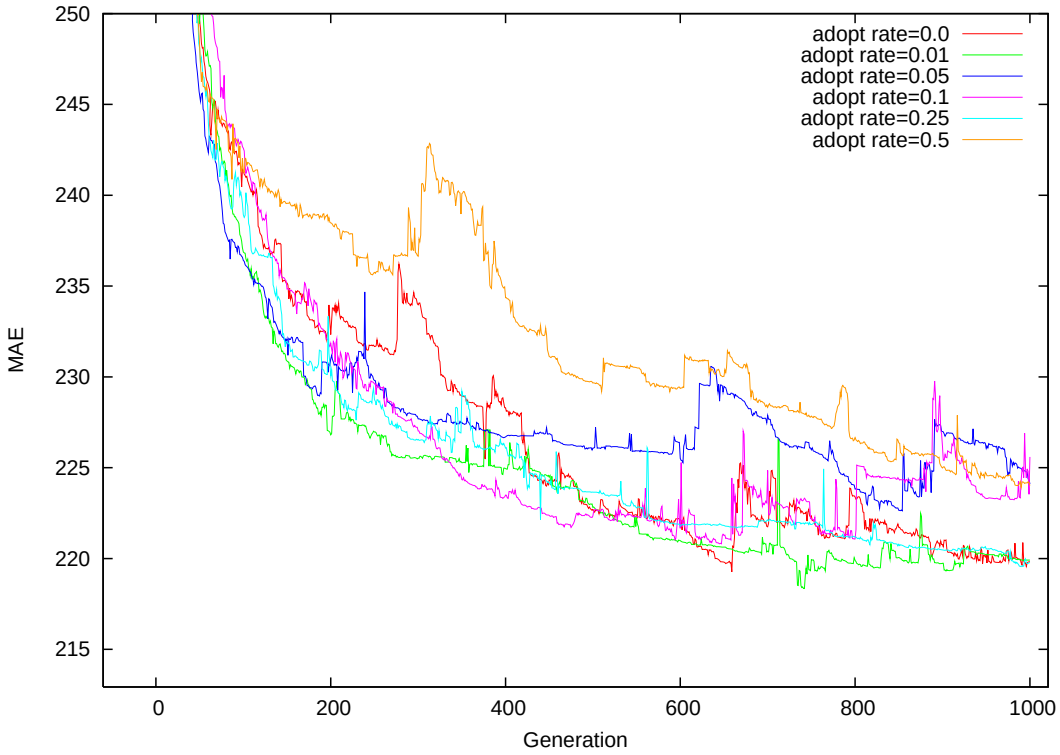

Abbildung 16: Einfluss der Rate für die lokale Suche auf die Evolution

Auswertung: Abbildung 16 stellt die größte Überraschung der durchgeführten Testläufe dar. Entgegen der ursprünglichen Annahme führt eine höhere Rate für die Anpassung von Konstanten nicht sichtbar zu einer besseren Evolution. Im Gegenteil liefert ein Deaktivieren der lokalen Suche mittels Suchrate von 0.0 sogar mit die besten Ergebnisse! Eine plausible Erklärung dafür zu finden gestaltet sich als schwierig. Durch statistische Tests hat sich heraus gestellt, dass eine durchschnittliche Verminderung des MAE durch die lokale Suche mit einer Suchrate von 1.0 bei nur ca. 1.74% liegt. Das Fortschreiten der Evolution ist daher nur sekundär geprägt von der lokalen Konstantensuche, primären Einfluss haben hingegen die genetischen Operatoren. Bei der Suche nach einer optimalen Lösung des Zeitreihenproblems stellt ein MAE von ca. 220 nach 1000 Generationen kein zufrieden stellendes Ergebnis dar, sodass die Feinabstimmung von Konstanten hier noch keine Vorteile mit sich bringt. Der Einfluss dürfte umso größer werden, je stärker sich die Evolution einer optimalen Lösung nähert. Dieser Punkt wird bei der hier durchgeführten Betrachtung allerdings nie erreicht. Eine hohe Rate für die lokale Suche von Konstantenwerten kann die hohen Geschwindigkeitseinbußen daher hier nicht rechtfertigen, sodass die Deaktivierung der Suche oder höchstens die Wahl einer kleinen Suchrate bis 0.05 empfohlen wird.

4.7.4 Einfluss der Genanzahl

Hypothese: Es werden Chromosomen mit unterschiedlicher Gen-Anzahl erzeugt und deren Einfluss auf die Evolution betrachtet. Die Vermutung liegt in einer schlechten Evolution bei Nutzung nur eines Gens und besserer Evolution für 2 oder 3 Gene. Bei mehr Genen sollte der Algorithmus Probleme haben eine geeignete Problemlösung zu finden und sich die höhere Genanzahl zunutze zu machen.

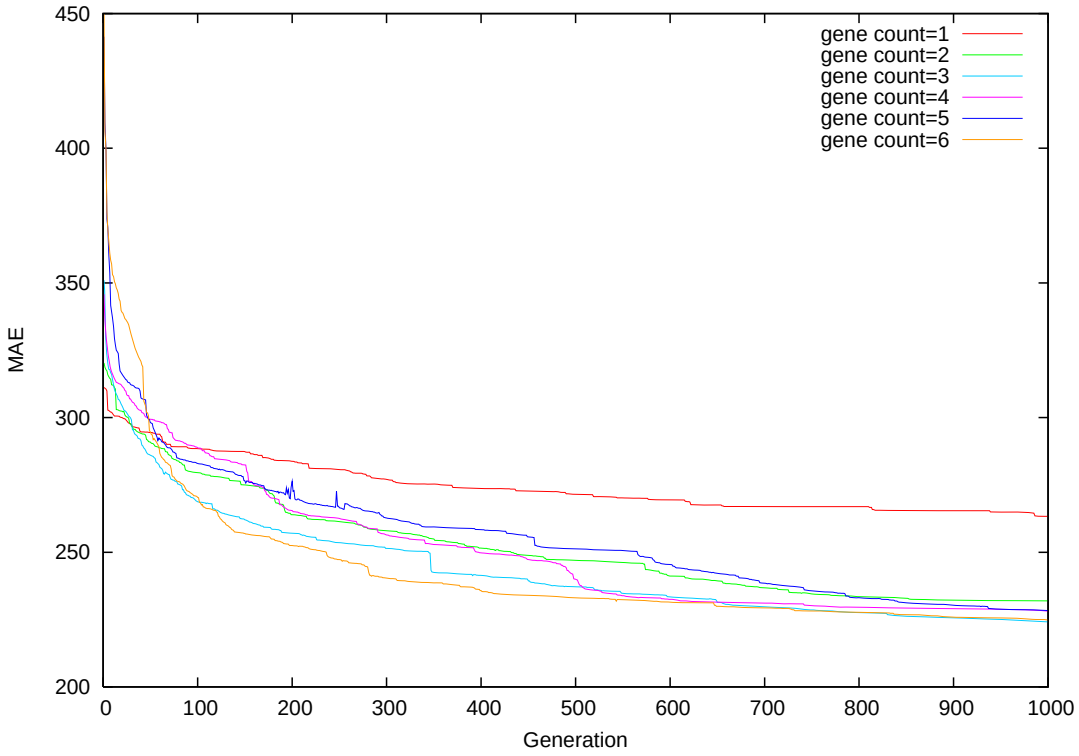

Abbildung 17: Einfluss der Genanzahl auf die Evolution

Auswertung: Aus Abbildung 17 wird ersichtlich, dass wie vermutet die Nutzung nur eines Gens zu einer signifikant schlechteren Evolution im Vergleich mit Multi-Gen-Chromosomen führt. Die Anzahl der Gene scheint davon abgesehen kaum eine Rolle zu spielen. So sind die Unterschiede bei der Nutzung von 2 bis 6 Genen pro Chromosom gering und lassen keinen Trend oder gar Gesetzmäßigkeit erkennen. So ist nach 400 Generationen eine Genanzahl von 5 scheinbar eine schlechte Wahl, wohingegen 6 Gene gute Resultate liefern. In den ersten 50 Generationen kommt es mit wenigen Genen erwartungsgemäß zu einem schnelleren Abfallen des MAE, mit vielen Genen wird einige Zeit benötigt, um eine gute Lösung zu evolvieren. Aufgrund der linear ansteigenden Laufzeit ist die Genanzahl eher klein zu wählen, 2 bis 3 Gene pro Chromosom sind hier eine gute Wahl.

5 Künstliche Neuronale Netze

Künstliche Neuronale Netze (*KNN*) sind allgemein ungerichtete Graphen, deren Knoten aus künstlichen Neuronen bestehen, die über primitive Funktionen verfügen. Die Kanten sind zumeist gewichtet, ihre Werte simulieren biologische Synapsenverbindungen, die durch häufige Benutzung verstärkt und durch geringe Nutzung geschwächt werden. Es gibt zahlreiche Netz-Modelle, die sich für bestimmte Klassen von Problemen eignen. Probleme, die mit KNN behandelt werden entspringen einer Vielzahl von Bereichen, u. a. der Mustererkennung, der Bildanalyse, der Optimierung und der Robotik.

5.1 Netzmodell

Als häufig verwendetes und an viele Probleme leicht anpassbares Modell neuronaler Netze kommt hier ein *Feedforward-Netz* zum Einsatz. Dieses ist in mehrere Schichten von Neuronen unterteilt, wobei jedes Neuron der Schicht i mit jedem Neuron der Schicht $i+1$ verbunden ist und der Signalfluss ausschließlich inkrementell durch die Schichten stattfindet. Die erste Schicht ist die Eingabeschicht, an die Problemgrößen angelegt werden und die letzte Schicht ist die Ausgabeschicht, an der Ergebnisse abgelesen werden können. Alle weiteren Schichten gelten als verdeckte Schichten und besitzen keine Schnittstelle nach außen, womit ein KNN grundsätzlich als black box anzusehen ist.

Beim verwendeten Feedforward-Netz wird zwischen 2 Phasen unterschieden. In der *Trainingsphase* wird das KNN mit dem *Backpropagation*-Algorithmus angelernt, indem ihm bekannte Paare von Ein- und Ausgaben präsentiert werden und sich das Netz diesen Daten anpasst. In der *Arbeitsphase* können dem Netz nach dem Training Eingaben präsentiert werden und es besteht die Hoffnung, dass auf noch unbekannte Eingaben anhand der erlernten Daten Ausgaben ermittelt werden, die eine zufrieden stellende Lösung des zugrunde liegenden Problems darstellen.

Backpropagation als Lernmethode ist ein *Gradientenabstiegsverfahren*. Dem KNN werden dabei Paare zusammen gehörender Ein- und Ausgaben präsentiert. In einem ersten Schritt werden die Eingabedaten an die erste Schicht des KNN angelegt und die Signale werden durch das Netz propagiert, bis die Ausgabeschicht erreicht ist.

In einem zweiten Schritt wird zunächst an der Ausgabeschicht der Fehler der tatsächlichen Ausgabe zur erwarteten Ausgabe berechnet. Dies geschieht über folgende Fehlerfunktion

E:

$$E = \sum_{i,\ N_i \in A} (\tilde{y_i} - y_i)^2$$

$A \ldots$ Ausgabeschicht
$N_i \ldots$ Neuron i
$\tilde{y_i} \ldots$ Soll-Output von N_i
$y_i \ldots$ Ist-Output von N_i

Anschließend wird der Gradient des Fehlers bzgl. der Gewichte zwischen den letzten beiden Schichten berechnet. Dieser gibt den maximalen Anstieg des Fehlers in einem Punkt wieder. Die Gewichte zwischen vorletzter und letzter Schicht werden dann entgegen der Gradientenrichtung angepasst, wodurch der Fehler für das präsentierte Trainingsdatum verringert wird. Der Fehler wird dann weiter in Rückwärtsrichtung durch das Netz propagiert, bis die erste Schicht erreicht ist und alle Gewichte angepasst wurden. Durch mehrmaliges Ausführen dieses Verfahrens auf vorhandene Trainingsdaten besteht die Hoffnung, dass ein möglichst gutes (globales) Fehlerminimum erreicht wird, welches sowohl für die Trainingsdaten als auch für bisher unbekannte Datensätze ein gutes Ergebnis darstellt.

Eine umfassende Darstellung des Netzmodells ist nicht Gegenstand dieser Ausarbeitung und kann z. B. in [7, 8] nachgelesen werden.

5.2 Netzaufbau

Das in dieser Arbeit verwendete Feedforward-Netz besteht strukturell aus 3 Schichten: Ein- und Ausgabeschicht sowie eine verdeckte Schicht. Die Anzahl der Neuronen der Eingabeschicht entspricht der Anzahl p der Werte $x_{i-p}, \ldots, x_{i-1}$ der Zeitreihe, die als Parameter in die Berechnung des Wertes x_i mit einfließen sollen. Die Eingaben werden mit Hilfe eines vom Benutzer fest zu legenden Maximalwertes auf 1 normiert und so dem Netz präsentiert. Die Ausgabeschicht besteht aus nur einem Neuron, welches den vom Netz ermittelten Wert x_i auf 1 normiert ausgibt und noch auf den tatsächlichen Wertebereich abgebildet werden muss. Abbildung 18 stellt den Netzaufbau noch einmal grafisch dar.

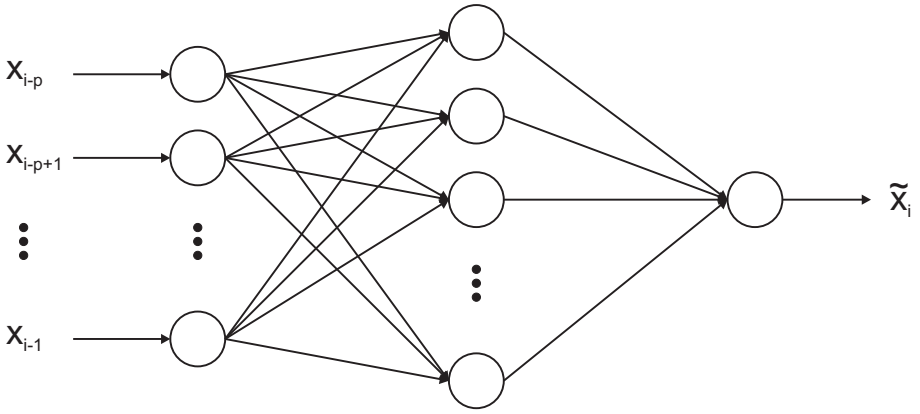

Abbildung 18: Aufbau des verwendeten KNN

5.3 Parametereinstellungen

Die angegebenen Parameter haben sich in Tests als praktikabel herausgestellt, wobei eine bessere Anpassung sicher möglich wäre. Dies ist allerdings nicht Gegenstand dieser Arbeit, die sich auf GEP und nicht auf KNN konzentriert.

Parameter	Wert
Schichten	3
Verdeckte Schichten	1
Neuronen der Eingabeschicht	12
Neuronen der verdeckten Schicht	30
Neuronen der Ausgabeschicht	1
Lernrate	1.6
Faktor des Impulsterms	0.0

Tabelle 3: Parametereinstellungen für das KNN

6 Methoden-Vergleich

Um die Methoden untereinander zu vergleichen, wurden stets dieselben Fehlermaße MAE und NMSE (s. Abschnitt 2.2) verwendet, deren Betrachtung über die Zeit erfolgte. Hierzu wurden bei allen Verfahren die in den Abschnitten 3.6, 4.6 und 5.3 festgelegten Parameter verwendet und für jede Methode 25 Trainingsläufe über jeweils 300 Sekunden (= 5

Minuten) durchgeführt. Nach jeder Sekunde wurde bei den GEP-Verfahren der Wert des besten Genotyps ausgewertet, bei dem KNN wurde der aktuelle Lernstatus heran gezogen. Über alle Durchläufe wurde das arithmetische Mittel von MAE und NMSE gebildet und für die Auswertungen grafisch aufbereitet. Diese Methodik wurde verwendet, um die Vergleiche losgelöst von abstrakten Zeiteinheiten wie einer Generation bei GEP oder einer Epoche bei KNN durchzuführen und mit real verbrauchter Rechenzeit ein praktisch relevantes Maß zu verwenden. Alle Tests wurden auf demselben Rechner unter vergleichbaren Bedingungen (keine zyklisch laufenden Prozesse, keine weitere nutzerbeeinflusste Rechenaktivität, stand-alone Rechner) durchgeführt.

In den Tabellen der Voraussagefehler gilt die folgende Symbolik: `MAE1`, `MAE5`, `PERC1` und `PERC5` stellen den absoluten und prozentualen Fehler über eine 1-Punkt- respektive 5-Punkt-Vorhersage im Vergleich zur Entwicklung der Original-Zeitreihe dar.

In den folgenden Unterabschnitten findet ein Vergleich der Methoden für die 4 in Abschnitt 2.3 vorgestellten Zeitreihen statt. Dabei wird bei der Furnas-Zeitreihe auf einige Details wie die Darstellung der Anpassung der jeweiligen Modelle an die Zeitreihe und die Angabe der evolvierten Algorithmen bei den GEP-Verfahren eingegangen. Bei den anderen Zeitreihen werden diese ausführlichen Informationen nicht mit angeführt.

6.1 Furnas Zeitreihe

In Abbildung 19 wird der MAE über 300 Sekunden bei der Bearbeitung der Furnas-Zeitreihe dargestellt. Hier lässt sich ein klarer Vorteil des KNN gegenüber GEP und EGIPSYS erkennen, der im Vergleich zu GEP bei ca. 47% liegt. EGIPSYS hebt sich dabei mit einem 15% besseren MAE noch einmal stark vom Standard-GEP ab, konnte durch die Verwendung expliziter Konstanten hier also für eine bessere Evolution sorgen.

Abbildung 20 veranschaulicht den NMSE der einzelnen Verfahren über die Zeit. Das KNN ist den anderen Verfahren deutlich überlegen, der NMSE ist hier um 72% besser als bei GEP. GEP und EGIPSYS liegen in etwa gleich auf, wobei EGIPSYS einen leichten Vorteil besitzt. Der NMSE beträgt hier am Ende der Evolution 0.49124. In [6] wurde für dieselbe Zeitreihe bei EGIPSYS ein NMSE von 0.3445 mit ähnlichen Parametern ermittelt. Dieses Ergebnis konnte hier nicht nachvollzogen werden und das „Warum“ steht als offene Frage im Raum. Am Wahrscheinlichsten sind Abweichungen in Details der Implementierung, da diese weder bei GEP noch bei EGIPSYS eindeutig und reproduzierbar beschrieben sind.

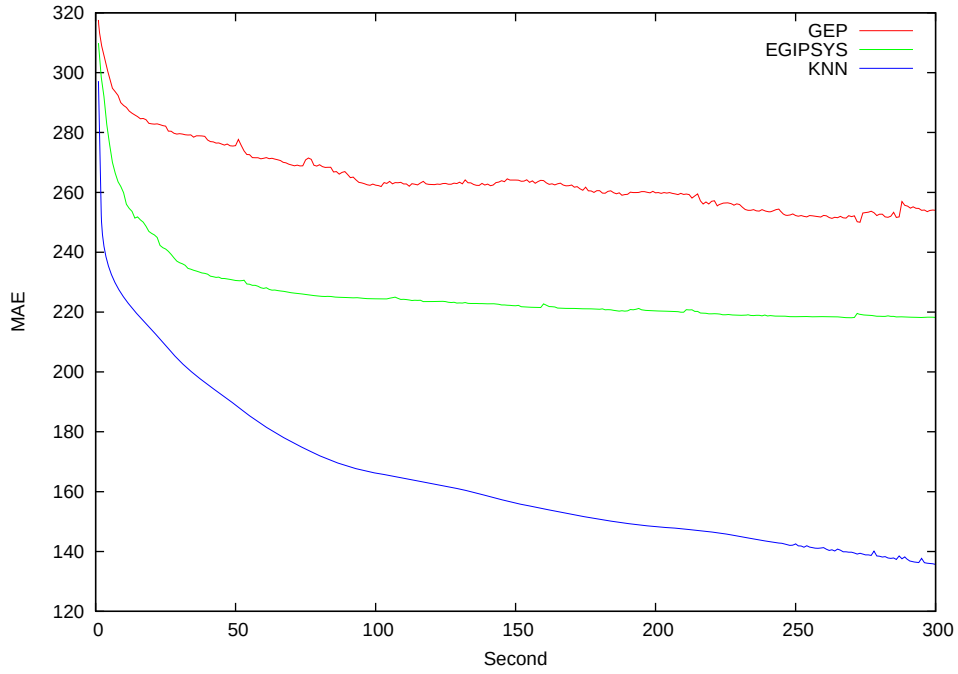

Abbildung 19: Vergleichsgrafik des MAE, Furnas-Zeitreihe

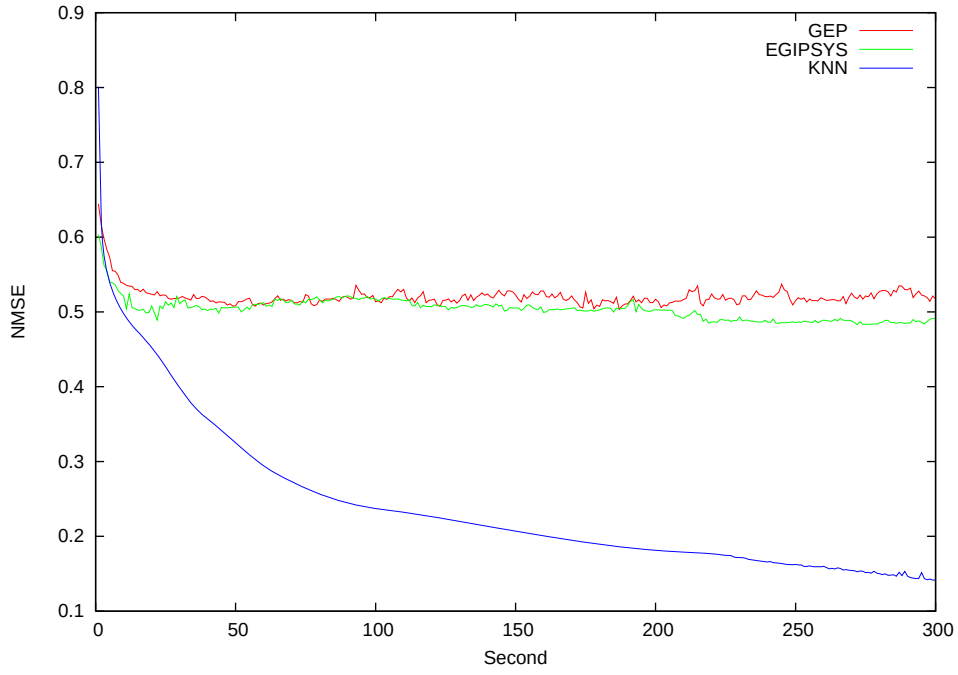

Abbildung 20: Vergleichsgrafik des NMSE, Furnas-Zeitreihe

Der durch *GEP* evolvierte Ausdruck des besten Genotyps lautet wie folgt:

```
x[i] = (((((((x[i-12]+x[i-11])+x[i-11])/x[i-4])*((x[i-10]+x[i-11])+
```

```
(x[i-11]+x[i-10]))))/x[i-4]) + (((((x[i-12]+x[i-11])+x[i-11])/x[i-4])*
((x[i-10]+x[i-11])+(x[i-11]+x[i-10])))/x[i-4])) + (x[i-1]-(x[i-1]/
((((x[i-2]+x[i-2])+x[i-12])+(x[i-10]+x[i-9]))/x[i-1])))
```

Mathematisch vereinfacht führt das zu der folgenden Gleichung:

$$x_i = 4 \cdot \frac{(x_{i-12} + 2 \cdot x_{i-11}) * (x_{i-10} + x_{i-11})}{x_{i-4}^2} + x_{i-1} - \frac{x_{i-1}^2}{2 \cdot x_{i-2} + x_{i-12} + x_{i-10} + x_{i-9}}$$

Die somit beschriebene Zeitreihe wird im Vergleich zur Original-Furnas-Zeitreihe in Abbildung 21 dargestellt. Auf den ersten Blick fällt auf, dass der Algorithmus scheinbar in der Lage war die Zyklizität der Zeitreihe zu erlernen, da in der Voraussage dieselbe Periode erscheint. Abbildung 22 zeigt einen detaillierteren Ausschnitt. Erkennbar ist die Nicht-Angepasstheit an kleine Details, was hauptsächlich auf das Fehlen expliziter Konstanten zurück geführt werden kann.

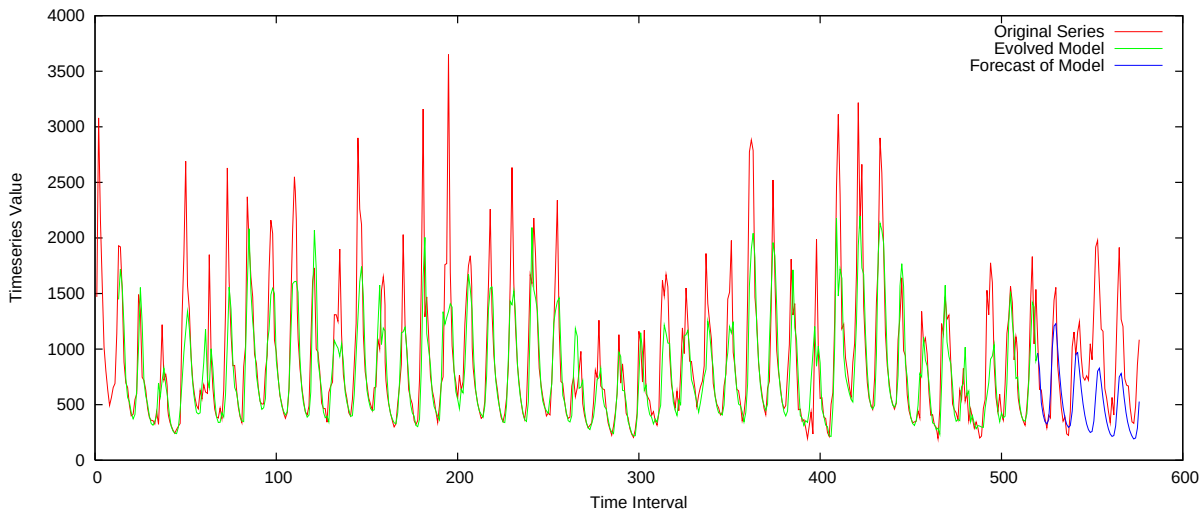

Abbildung 21: Anpassung an Furnas-Zeitreihe mittels GEP

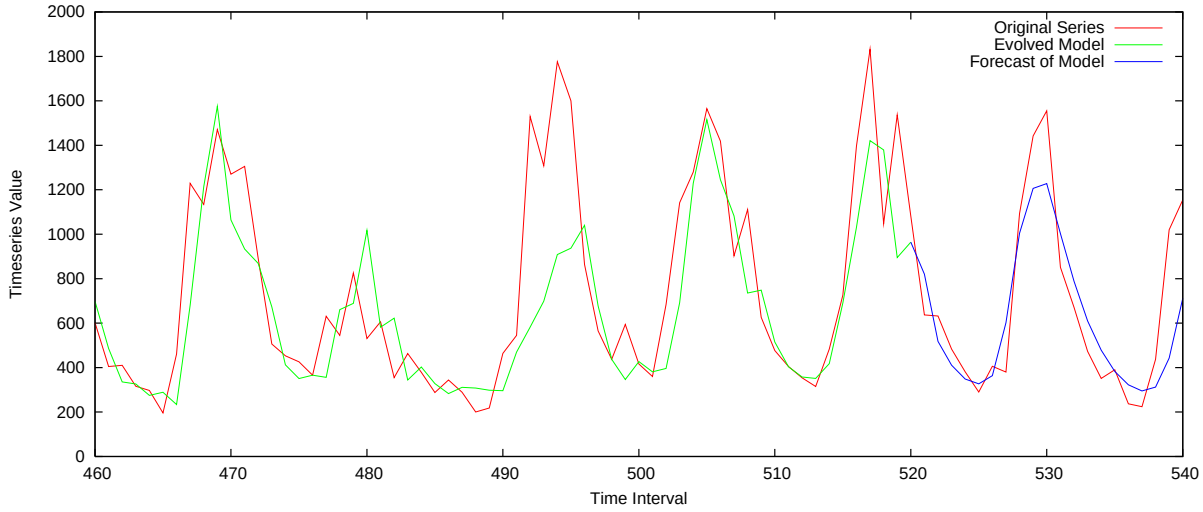

Abbildung 22: Detail-Ansicht von Abbildung 21

Der durch *EGIPSYS* (*erweitertes GEP*) evolvierte Ausdruck des besten Genotyps lautet wie folgt (Konstantenwerte auf 2 Dezimalstellen gerundet):

```
x[i] = (((((358.03+(x[i-12]-x[i-1]))/7.37)-76.15) +
 (((((x[i-10]-x[i-1])-x[i-1])+x[i-11])/100.69)*12.01)) +
 (x[i-1]-((x[i-2]+(x[i-3]-38.76))/49.6)))
```

Mathematisch vereinfacht führt das zu der folgenden Gleichung:

$$x_i = \frac{358.03 + x_{i-12} - x_{i-1}}{7.37} - 76.15 + \frac{12.01 * (x_{i-11} + x_{i-10} - 2 \cdot x_{i-1})}{100.69} + x_{i-1} - \frac{x_{i-2} + x_{i-3} - 38.76}{49.6}$$

Abbildung 23 zeigt die so beschriebene Zeitreihe im direkten Vergleich mit der vorgegebenen Furnas-Zeitreihe. Auch hier konnte die grundsätzliche Periodizität der Zeitreihe erlernt werden. Im Vergleich zum „normalen“ GEP konnte eine bessere Anpassung an die Original-Zeitreihe erreicht werden, was sich auch im geringeren MAE widerspiegelt. Abbildung 24 zeigt einen Ausschnitt der Zeitreihe im Detail. Erkennbar ist trotz der Verwendung expliziter Konstanten eine hohe Abweichung im Detail, die zwar geringer ausfällt als bei GEP, für eine genaue Abbildung der Zeitreihe allerdings immernoch zu groß ist.

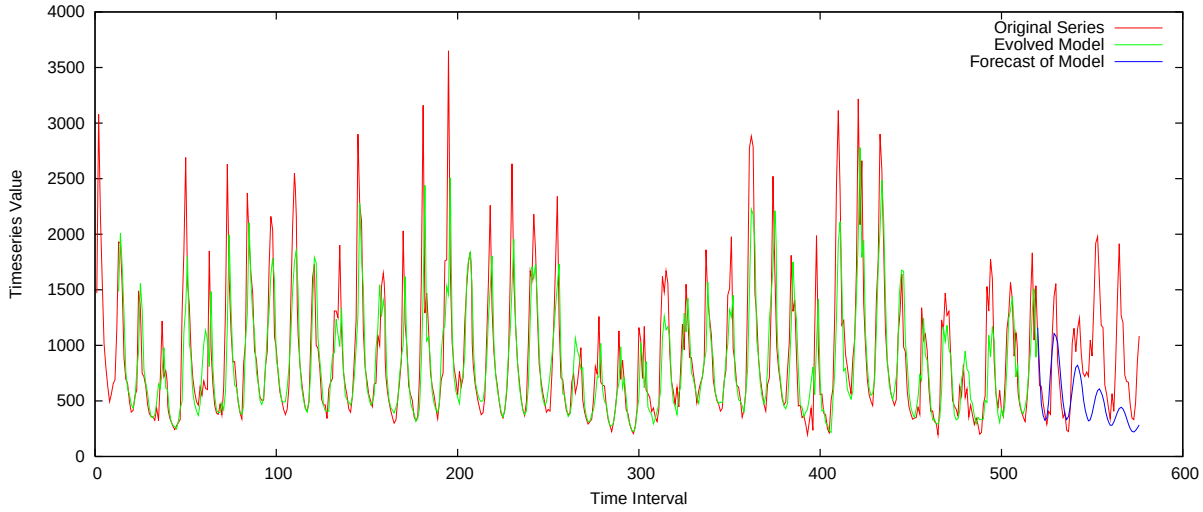

Abbildung 23: Anpassung an Furnas-Zeitreihe mittels EGIPSYS

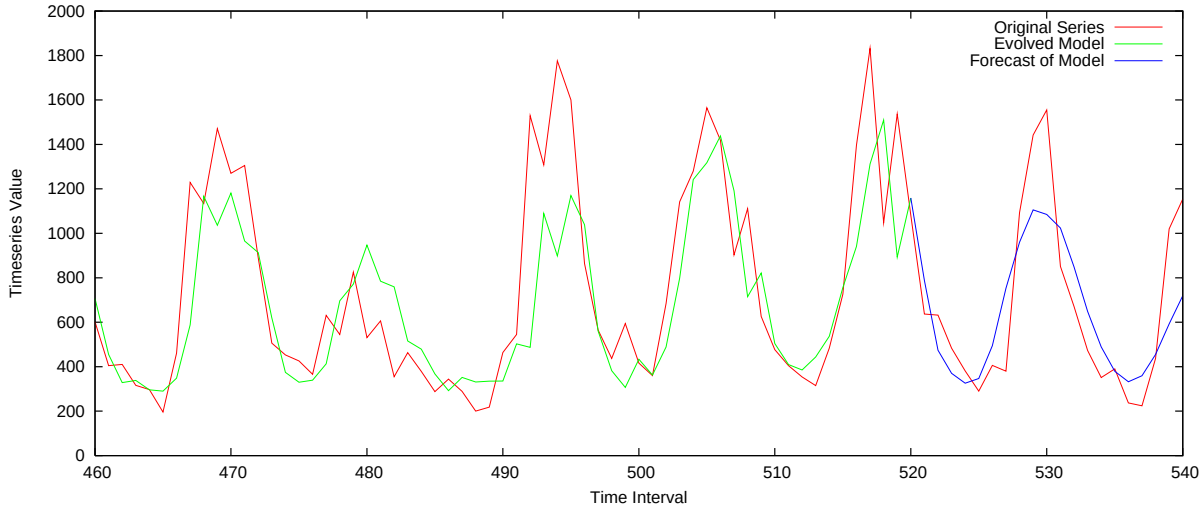

Abbildung 24: Detail-Ansicht von Abbildung 23

Die Anpassung an die Furnas-Zeitreihe mittels *KNN* ist in Abbildung 25 zu sehen. Eine explizite mathematische Beschreibung der Zeitreihe wie bei GEP und EGIPSYS ist dabei nicht möglich, da alle trainierten Werte nur implizit durch die Gewichte des Netzes gegeben sind. Im Vergleich zu GEP und erweitertem GEP ist eine bessere Anpassung an die Zeitreihe erkennbar, die in Abbildung 26 wieder als Ausschnitt dargestellt wird. Auch wenn das KNN einen geringeren MAE als die beiden anderen Methoden aufweist, so war es doch nicht in der Lage alle kleinen Details wiederzugeben. In Hinblick auf eine gute

Generalisierbarkeit und den Einfluss von Störungen in die Original-Zeitreihe stellt dies jedoch keinen Nachteil, sondern eher eine Art Glättung der Original-Zeitreihe dar, was sich auf Vorhersagen unbekannter Werte positiv auswirken kann.

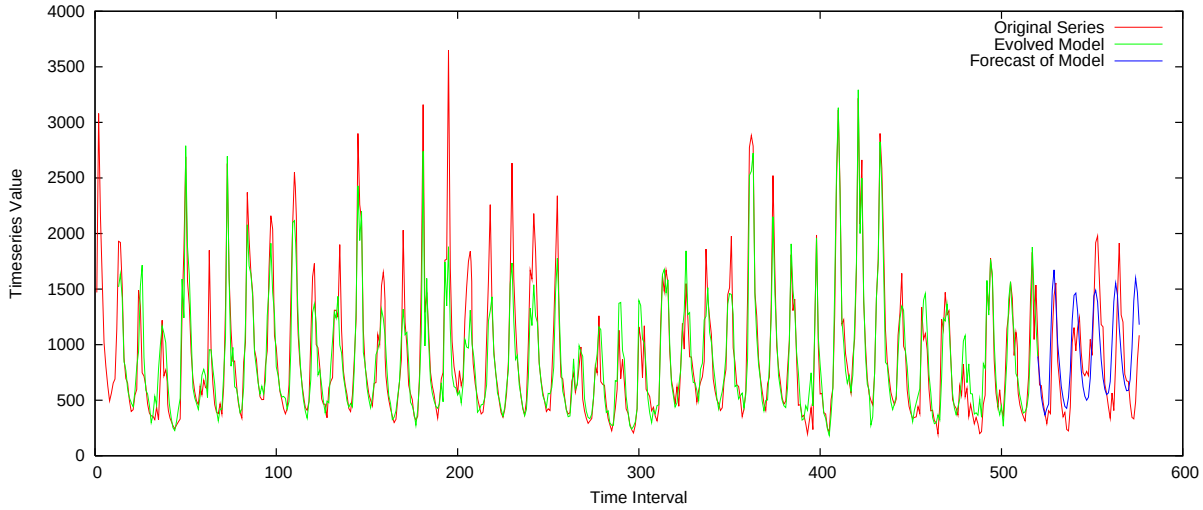

Abbildung 25: Anpassung an Furnas-Zeitreihe mittels KNN

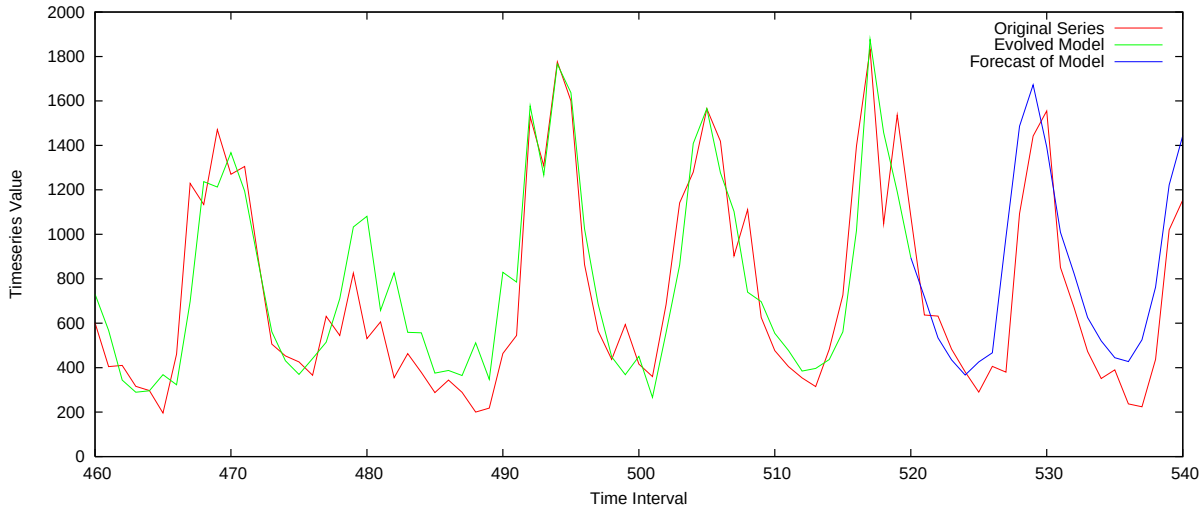

Abbildung 26: Detail-Ansicht von Abbildung 25

Tabelle 4 stellt für die einzelnen Methoden die erzeugten *Fehler in der Voraussage* des Testdatensatzes dar.

	MAE1	MAE5	PERC1	PERC5
GEP	183.59	87.76	28.82%	16.58%
EGIPSYS	150.72	106.72	23.66%	21.22%
KNN	113.55	88.6	12.8%	17.79%

Tabelle 4: Ermittelte Voraussagefehler für die Furnas-Zeitreihe

6.2 Erdbeben Zeitreihe

Abbildung 27 liefert den Vergleich der drei verwendeten Verfahren in Hinblick auf den MAE für die Erdbeben-Zeitreihe. Dem KNN ist es problemlos gelungen sich der Zeitreihe bis auf einen minimalen MAE anzupassen, der Unterschied zu den anderen Methoden liegt bei über 99%. GEP und EGIPSYS liegen gleich auf, wobei die Evolution auf einem recht hohen Level bei einem MAE von ca. 4 stagniert. Problematisch scheint für beide Verfahren die fehlende Periodizität der Zeitreihe zu sein, die wenig offensichtliche Gesetzmäßigkeiten erkennen lässt.

Abbildung 28 visualisiert den NMSE der Erdbeben-Zeitreihe. Auch hier konvergiert der NMSE bei dem KNN schnell gegen 0, während er sich bei GEP auf einem Wert von ca. 1.6 und bei EGIPSYS auf einem Wert von ca. 1 befindet. Der Unterschied im NMSE von GEP zu EGIPSYS beträgt demnach ca. 35% und von GEP zum KNN über 99%.

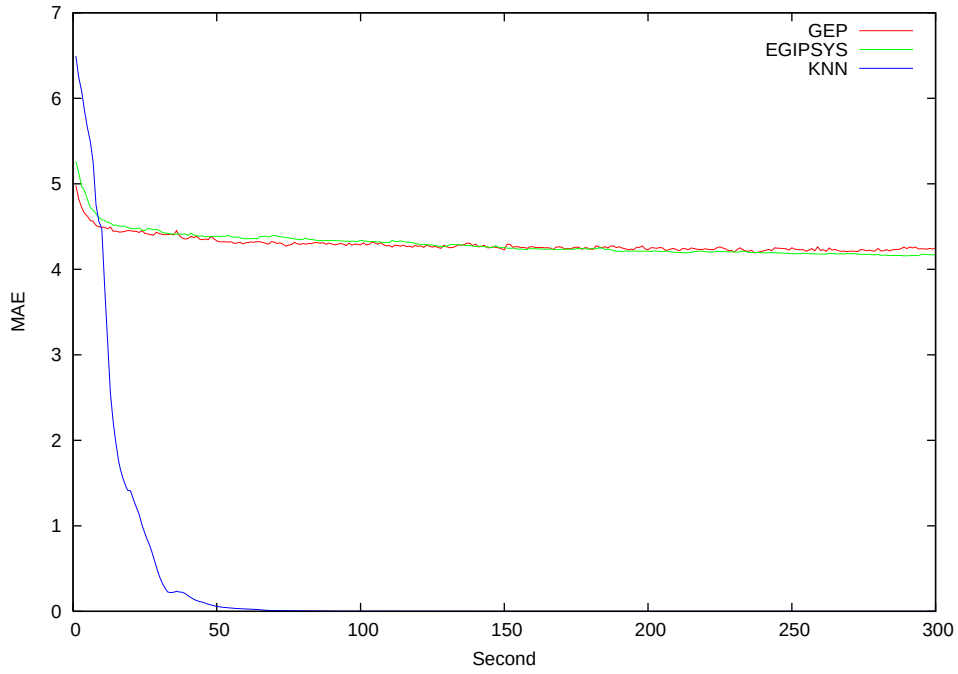

Abbildung 27: Vergleichsgrafik des MAE, Erdbeben-Zeitreihe

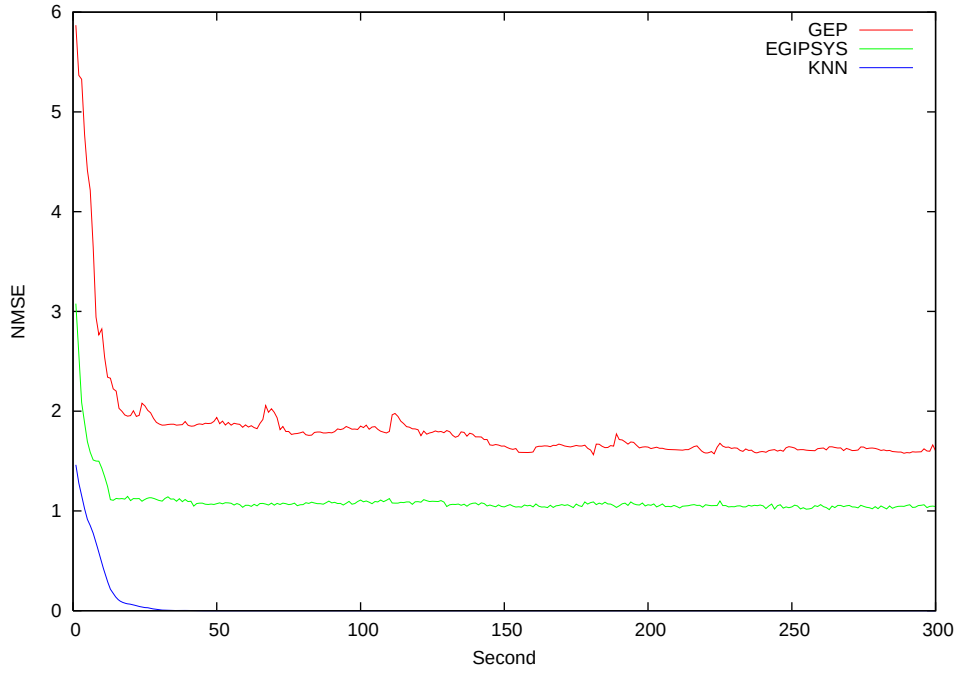

Abbildung 28: Vergleichsgrafik des NMSE, Erdbeben-Zeitreihe

Tabelle 5 stellt für die einzelnen Methoden die erzeugten Fehler in der Voraussage des Testdatensatzes dar.

	MAE1	MAE5	PERC1	PERC5
GEP	2.43	6.24	34.77%	41.2%
EGIPSYS	2.69	3.91	38.46%	25.2%
KNN	0.00000064	5.73	0.0000079%	39.29%

Tabelle 5: Ermittelte Voraussagefehler für die Erdbeben-Zeitreihe

6.3 Sonnenflecken Zeitreihe

Abbildung 29 verdeutlicht den MAE bei der Sonnenflecken-Zeitreihe und lässt einen klaren Vorteil des KNN vor EGIPSYS und GEP erkennen. Mit einem MAE in Höhe von 31% des Wertes von GEP ist das KNN unangefochten. EGIPSYS erreicht immerhin einen MAE in Höhe von 81% verglichen mit GEP.

In Abbildung 30 wird der NMSE aller 3 Verfahren dargestellt. Das KNN besitzt dabei am Ende einen NMSE von 0.023, der damit nur ca. 11% des NMSE der anderen beiden Methoden beträgt. Während der Evolution ist GEP gegenüber EGIPSYS zunächst bevorteilt, am Ende liegen beide Verfahren aber in etwa gleich auf.

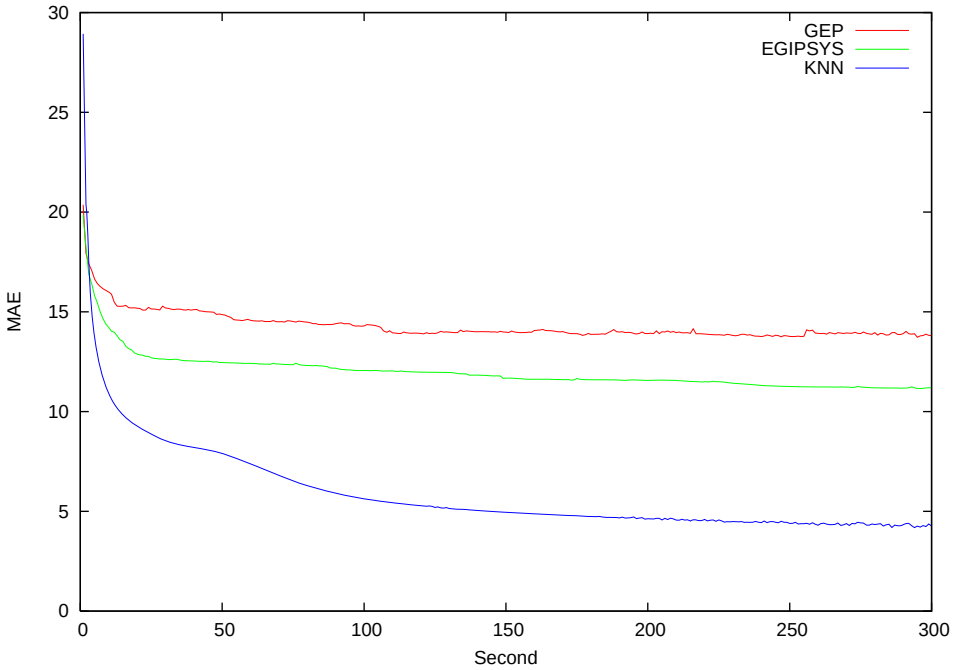

Abbildung 29: Vergleichsgrafik des MAE, Sonnenflecken-Zeitreihe

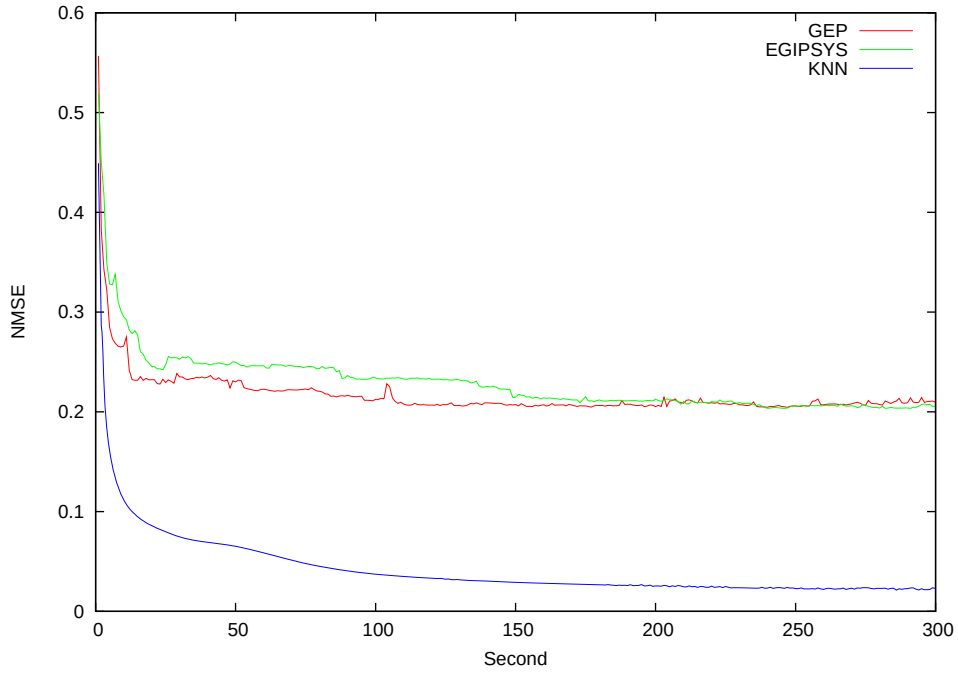

Abbildung 30: Vergleichsgrafik des NMSE, Sonnenflecken-Zeitreihe

Tabelle 6 stellt für die einzelnen Methoden die erzeugten Fehler in der Voraussage des Testdatensatzes dar.

	MAE1	MAE5	PERC1	PERC5
GEP	8.3	20.65	15.41%	127.54%
EGIPSYS	42.79	18.11	79.38%	87.67%
KNN	1.72	23.56	3.19%	153.65%

Tabelle 6: Ermittelte Voraussagefehler für die Sonnenflecken-Zeitreihe

6.4 Umsatz von US-Haushalten Zeitreihe

In Abbildung 31 wird der MAE über 300 Sekunden für alle verwendeten Verfahren dargestellt. Das KNN liefert eine stetige Minimierung des Fehlers, wodurch sich am Ende ein MAE in Höhe von ca. 13% im Vergleich zu GEP ergibt. EGIPSYS evolviert relativ schnell eine Lösung, die zu Beginn sogar das KNN übertrifft. Allerdings verbleibt die Evolution dann im Großen und Ganzen bei dieser Lösung. Die Evolution bei GEP schreitet langsamer voran, erreicht am Ende aber annähernd das Niveau von EGIPSYS.

Abbildung 32 verdeutlicht den NMSE, wobei die Ergebnisse grob den Resultaten der MAE-Untersuchung entsprechen. Das KNN besitzt am Ende einen NMSE in Höhe von 2% verglichen mit GEP. GEP und EGIPSYS liegen nach den 300 Sekunden annähernd gleich auf.

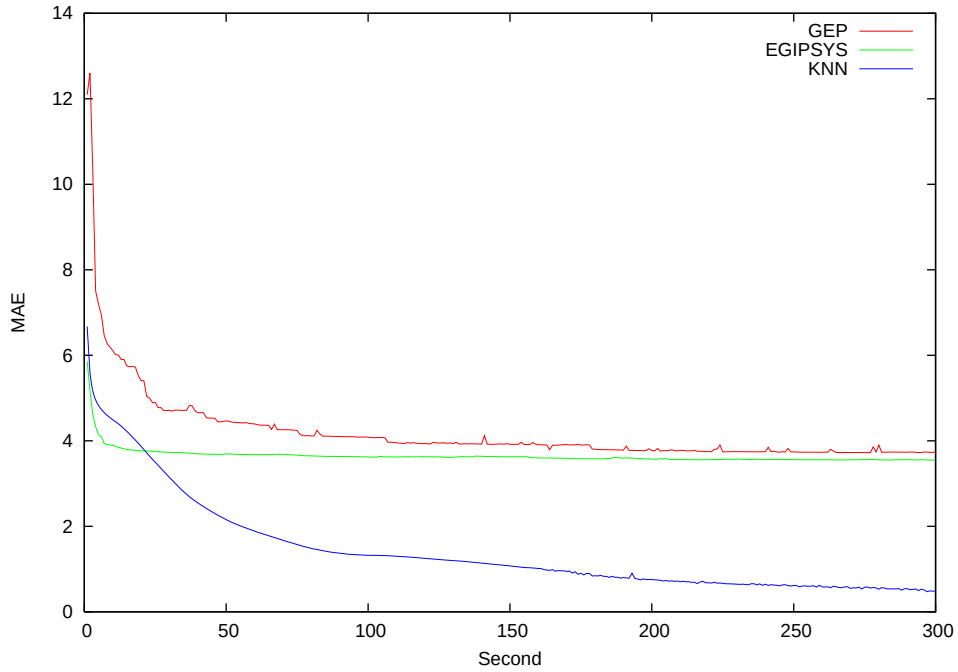

Abbildung 31: Vergleichsgrafik des MAE, Zeitreihe 'Umsatz von US-Haushalten'

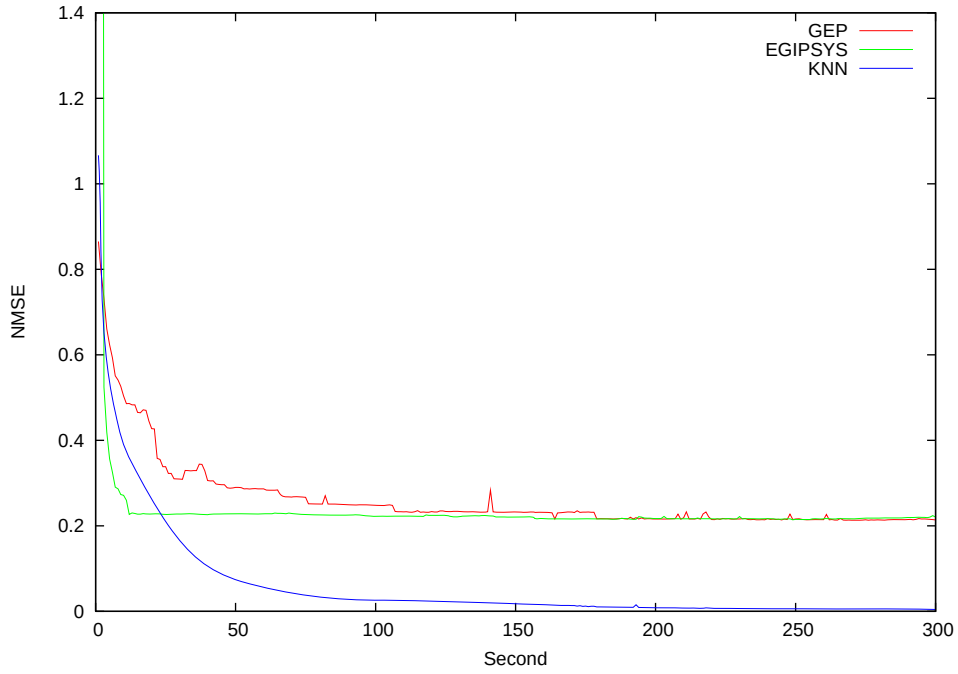

Abbildung 32: Vergleichsgrafik des NMSE, Zeitreihe 'Umsatz von US-Haushalten'

Tabelle 7 stellt für die einzelnen Methoden die erzeugten Fehler in der Voraussage des Testdatensatzes dar.

	MAE1	MAE5	PERC1	PERC5
GEP	7.45	3.19	32.39%	10.26%
EGIPSYS	6.88	3.62	29.91%	10.88%
KNN	0.33	3.68	1.09%	11.09%

Tabelle 7: Ermittelte Voraussagefehler für die Zeitreihe 'Umsatz von US-Haushalten'

7 Zusammenfassung

In der vorliegenden Arbeit wurde GEP als evolutionärer Algorithmus zur Lösung von Problemen betrachtet, der wie genetisches Programmieren (GP) in der Lage ist Algorithmen zu evolvieren. Im Gegensatz zu GP arbeiten die genetischen Operatoren von GEP dabei auf Genotypen fester Länge, die erst durch eine gesonderte Übersetzung in ausführbare Ausdrucksbäume umgewandelt werden. Die Fähigkeiten von GEP wurden dabei anhand des Problems der Zeitreihenanalyse untersucht, das in Abschnitt 2 vorgestellt wurde. GEP selbst mit seinen Datentypen und genetischen Operatoren beinhaltete Abschnitt 3, hierbei wurden für ausgewählte Parameter Hypothesentests durchgeführt und die Ergebnisse ausgewertet (Abschnitt 3.7). In Abschnitt 4 wurde mit EGIPSYS eine Erweiterung zu GEP vorgestellt, die beispielsweise die Verwendung expliziter Konstanten erlaubt. Auch hier wurden Hypothesentests durchgeführt. Als dritte Vergleichsmethode nach GEP und EGIPSYS wurden in Abschnitt 5 künstliche neuronale Netze (KNN) vorgestellt. Für alle drei Verfahren wurden in den einzelnen Abschnitten Parametereinstellungen angegeben, die dann in Abschnitt 6 verwendet wurden um die Methoden miteinander zu vergleichen. An vier festgelegten Zeitreihen wurden dabei Untersuchungen bzgl. der Entwicklung von Fehlermaßen nach der verbrauchten Rechenzeit durchgeführt.

Die Ergebnisse der vergleichenden Untersuchungen haben dabei gezeigt, dass KNN bei der Anpassung an die vorgegebene Zeitreihe klar die besten Ergebnisse liefern konnten. Als einziger störender Faktor ist das Fehlen einer expliziten mathematischen Formel zu nennen, da Wissen in KNN nur implizit durch die Gewichte gegeben ist. EGIPSYS war in der Lage Algorithmen zu evolvieren, welche die Zeitreihe gegenüber Standard-GEP besser abbilden konnten. Trotzdem sind sowohl bei EGIPSYS als auch bei GEP die Fehler so groß, dass von einer genauen Anpassung keine Rede sein kann. Vielmehr können grobe Strukturen und Periodizitäten erlernt werden, kleine Details bleiben dabei aber außen vor. Hier hat ein KNN mit seinem zielgerichteten fehlerminimierenden Lernprozess klare Vorteile, sowohl was benötigte Rechenzeit als auch geliefertes Resultat angeht. Teilweise konnten veröffentlichte Ergebnisse von GEP und EGIPSYS nicht nachvollzogen werden, was am ehesten mit einer unterschiedlichen Detail-Implementierung erklärt werden kann. Die Original-Arbeiten zu GEP und EGIPSYS geben sich an dieser Stelle zu wage und ungenau, als dass Sicherheit darüber erlangt werden könnte, dass die gesamte Implementierung den Originalen entspricht. Somit gelten die hier dargestellten Ergebnisse auch nur für die verwendete Implementierung, eine Übertragbarkeit auf die Originalartikel zu GEP ist dabei ausgeschlossen.

Allein aus der Anpassung der vorgegebenen Zeitreihe geht das KNN als klarer Sieger hervor. Auch bei einer 1-Punkt-Vorhersage ist es in der Lage die besten Ergebnisse zu liefern. Allerdings sind die produzierten Fehler einer 5-Punkt-Vorhersage bei GEP oder EGIPSYS kleiner als beim KNN und dies relativiert die genaue Anpassung an die Trainingsdaten wieder. EGIPSYS und GEP können hier teilweise bessere Ergebnisse liefern, wobei beide Verfahren näherungsweise gleich auf liegen, mit leichten Vorteilen für EGIPSYS. Ob dies nur einen Zufall darstellt oder die Anwendbarkeit der Verfahren realitätsnah widerspiegelt kann an dieser Stelle nicht abschließend gesagt werden. Hierzu müssten groß angelegte Vorhersage-Tests durchgeführt und ausgewertet werden, was über den Inhalt dieser Arbeit hinaus geht.

A Testumgebung

A.1 Rahmenbedingungen

Bei den durchgeführten Versuchen kam die folgende Rechnerumgebung zum Einsatz:

- Hardware: AMD Athlon XP 2400+, 768 MB RAM
- Software: Windows XP SP2 / Debian Linux 3.1, Java 1.6

A.2 Implementierung in GUINNEA

GEP, erweitertes GEP und das KNN wurden in dem vom Autor entwickelten *GUINNEA* (***G**raphical **U**ser **I**nterface for **N**eural **N**etworks and **E**volutionary **A**lgorithms*) umgesetzt. Die Idee für GUINNEA entstand bereits vor der Bearbeitung dieses Projekts und in dessen Rahmen wurde dann eine erste prototypische Umsetzung in Java mit derzeit ca. 7000 Zeilen reinem Quelltext erarbeitet. Das Ziel von GUINNEA ist die Vorgabe eines Rahmens für die Implementierung und das Testen von allgemeinen Algorithmen, im Speziellen von künstlichen neuronalen Netzen und evolutionären Algorithmen. Dafür wird ein Algorithmengraph aufgebaut, dessen Knoten Ein- und Ausgabeschnittstellen oder abzuarbeitende Algorithmen sein können. Die Kanten entsprechen Datentransformationen zwischen den einzelnen Knoten. Derzeit ist dies noch eine einfache Liste, in Zukunft ist aber ein allgemeiner Graph mit verschiedenen Schnittstellen zu den Knoten geplant. Alle Graph-Elemente werden als Plugins realisiert, womit GUINNEA leicht erweiterbar ist und zum schnellen Prototyping genutzt werden kann. Das Beziehen der Software ist derzeit nur per CVS über die Sourceforge-Projektseite unter [4] möglich, bis die Entwicklung einen Stand erreicht hat, bei dem es sich lohnt ein Softwarepaket heraus zu geben.

Das KNN und GEP wurden als GUINNEA-Plugin entwickelt, erweitertes GEP als Spezialfall davon. Abbildung 33 zeigt die prototypische Oberfläche mit den Parametrisierungsmöglichkeiten zu GEP.

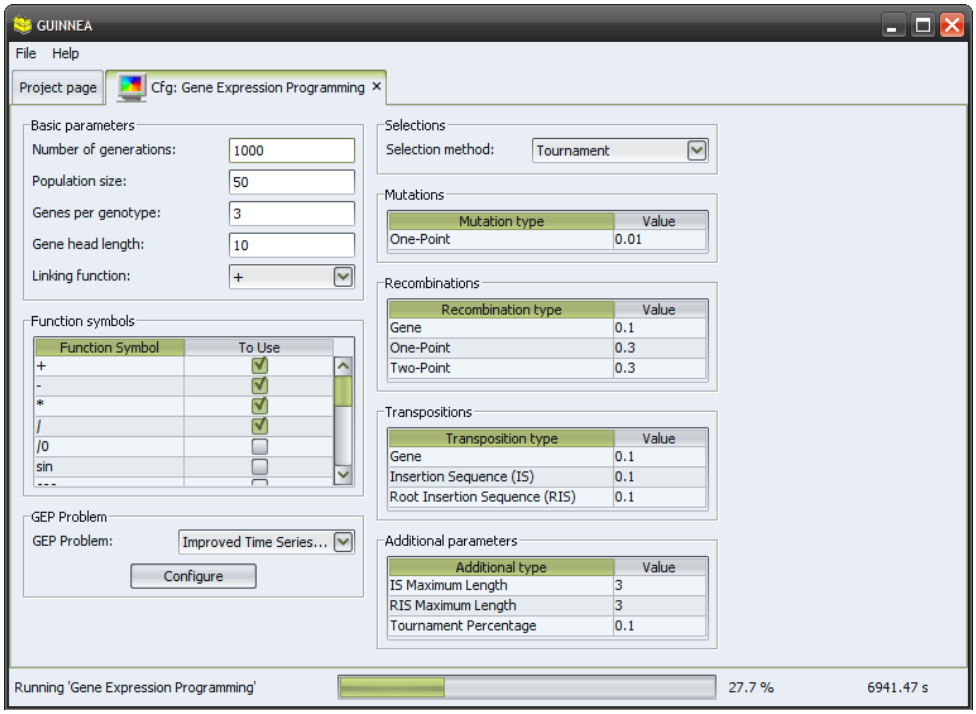

Abbildung 33: Screenshot von GUINNEA

Um mit den entwickelten Plugins Hypothesentests und Methodenvergleiche durchführen zu können, wurden die entsprechenden Module angepasst und die Ergebnisse zur Auswertung mittels *gnuplot* in Dateien abgespeichert.

Literatur

[1] Ferreira, Candida: *Gene Expression Programming Homepage.* http://www.gene-expression-programming.com, letzter Zugriff: 02.08.2007

[2] Ferreira, Candida: Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. In: *Complex Systems* (2001). http://www.gene-expression-programming.com/webpapers/GEP.pdf

[3] Ferreira, Candida: Gene Expression Programming in Problem Solving. (2001). http://www.gene-expression-programming.com/webpapers/GEPtutorial.pdf

[4] Jauernig, Matthias: *GUINNEA Project Page.* http://sourceforge.net/projects/guinnea, letzter Zugriff: 13.08.2007

[5] Lopes, Heitor S. ; Weinert, Wagner R.: EGIPSYS: An Enhanced Gene Expression Programming Approach For Symbolic Regression Problems. In: *International Journal of Applied Mathematics and Computer Science* 14 (2004), 375–384. http://matwbn.icm.edu.pl/ksiazki/amc/amc14/amc1437.pdf

[6] Lopes, Heitor S. ; Weinert, Wagner R.: A Gene Expression Programming System For Time Series Modeling. In: *Proceedings of XXV Iberian Latin AmericanCongress on Computational Methods in Engineering (CILAMCE)* (2004). http://www.cpgei.cefetpr.br/~hslopes/publicacoes/2004/cilamce2004.pdf

[7] Patterson, Dan: *Künstliche neuronale Netze: das Lehrbuch.* Prentice Hall Verlag GmbH, 1996. – ISBN 3–8272–9531–9

[8] Zell, Andreas: *Simulation Neuronaler Netze.* Addison-Wesley (Deutschland) GmbH, 1994. – ISBN 3–89319–554–8