I. Boersch, Brandenburg

Genetisches Programmieren einfacher Roboterfähigkeiten

1 Zusammenfassung

Die Leistungsfähigkeit autonomer Systeme (AS) hängt wesentlich von ihrer Fähigkeit zur Adaption an wechselnde Umgebungs- oder interne Bedingungen ab. Der Ansatz „Genetisches Programmieren“, als Methode des maschinellen Lernens, wurde 1992 von J. Koza als Erweiterung der bekannten Genetischen Algorithmen und Evolutionsstrategien entwickelt. Die Grundidee besteht im Anwenden einer evolutionären Adaptionsleistung auf Computerprogramme, d.h. ein Computer entwickelt selbständig eine Population von ausführbarem Programm-Code. Die dabei entstehenden Code-Fragmente liegen miteinander im evolutionären Wettstreit und passen sich im Laufe der simulierten Evolution an eine vorgegebene Fitnessfunktion an.
Wir haben in einem Projekt die Leistungsfähigkeit dieses Ansatzes zum maschinellen Entdekken eines Bewegungsmusters in realen autonomen Systemen untersucht. Das von uns gewählte Testszenario besteht aus einem GP-System und einem realen Roboter mit zufälliger Morphologie. Der Beitrag stellt die Probleme und Ergebnisse dieses Experimentes dar.

2 Einleitung

Die natürliche Evolution erzeugte vielfältige Formen und Lebewesen, die hervorragend an spezielle ökologische Nischen in ihrer Umwelt angepaßt sind. Insbesondere sind in allen Lebewesen exzellente Steuermechanismen zur Kontrolle ihres eigen Körpers entwickelt worden. Trotz der komplexeren Morphologie eines Lebewesens im Vergleich mit heutigen Robotern zeichnen sich biologische Bewegungsabläufe durch Schnelligkeit, Genauigkeit, Eleganz und Robustheit aus. Die Steuerung von Körperbewegungen erfolgt auf der Ausführungsebene in der Regel unbewußt, entweder durch das Erlernen von Reiz-Reaktionsverbindungen während der Ontogenese oder durch Übernahme gelernter, oft praktizierter Vorgänge in das genetische Material (siehe auch Baldwin-Effekt in [8]).
Kann diese Idee der Evolution von Bewegungsmustern zur Entwicklung von einfachen Bewegungsfähigkeiten autonomer Roboter genutzt werden? Die Umsetzung hätte 2 wesentliche Vorteile:

1. Es können Robotermorphologien zielgerichtet bewegt werden, die aufgrund eines fehlenden oder sehr komplexen kinematischen Modells (viele Elemente, hohe Prozeß- oder Meßunschärfe, Nichtlinearitäten, instabile Totzeiten) konventionell nicht steuerbar sind
2. Zu einer gegebenen vielversprechenden Morphologie kann ein optimal angepaßter Bewegungsablauf entwickelt werden, der die Leistungsgrenzen des physisch Möglichen erreicht (z.B. ein Hochsprungroboter)

Wir untersuchten in diesem Experiment den ersten Aspekt.
Bei der Entwicklung von Bewegungsmustern für komplexe Roboter stellen sich 2 grundsätzliche Probleme:

1. Die unbekannte Komplexität der Lösung hat zur Folge, daß Optimierungsmethoden, die auf einer vordefinierten Struktur arbeiten (z.B. Künstliche Neuronale Netze, Genetische Algorithmen, Suchverfahren) den Suchraum zu stark einschränken.

2. Es sind apriori keine Beispiele für das zu lernende Verhalten vorhanden, so daß kein supervised learning (z.B. Klassifikatorlernen, Clusterbildung, Entscheidungsbäume) angewendet werden kann.

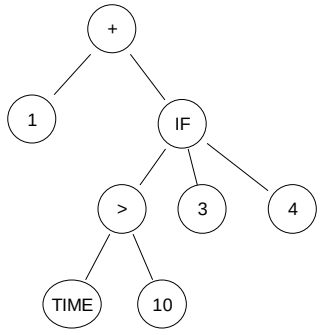
Wir treffen hier auf ein zentrales Problem der Informatik

„How can computers learn to solve problems without being explicitly programmed? In other words: How can computers be made to do what is needed to be done, without being told exactly how to do it?“ (zugesprochen Arthur Samuel, 50er Jahre)

3 Genetisches Programmieren von Bewegungsmustern

Genetisches Programmieren (GP) wurde von John R. Koza [7] Anfang der 90er Jahre als Erweiterung der Genetischen Algorithmen (GA) [4] zur Lösung derartiger Probleme vorgeschlagen. GP bietet gegenüber GA eine automatische Anpassung der Lösungskomplexität an die Komplexität des Problems [9].
Die Methode soll hier nur kurz mit ihren interessanten Aspekten vorgestellt werden, zur eingehenden Erläuterung sei auf Koza [7] direkt bzw. GP-Einführungen ([1], [5], Anhang 8.1) verwiesen.
GP transformiert eine Population von individuellen Programmen, jedes mit einem zugeordneten Fitnesswert, in eine neue Generation der Population. Die Transformation nutzt das Darwinsche Prinzip des Überlebens des fittesten Individuums und Analogien zu natürlichen genetischen Operatoren wie Crossover (geschlechtliche Fortpflanzung) und Mutation. Der Fitnesswert wird den Programmen durch einen Beobachter (Mensch oder Programm) zugewiesen, so daß der Fitnesswert das Reinforcement in bezug auf die gewünschten Zieleigenschaften darstellt. Die Repräsentation der Individuen (Programme) erfolgt durch S-Expressions (Prefix-Notation):

S-Expression:	(+ 1 (IF (> TIME 10) 3 4)	Parsetree (geordnete Zweige):
imperativ:	if (TIME > 10) return 1+3; else return 1+4;	



Die Selektion der Individuen für die nächste Generation erfolgt in der Regel mit einer Wahrscheinlichkeit proportional zum Fitnesswert oder durch Fitness-Auswahl aus einer zufälligen Gruppe (Tournament-Selection). GP realisiert somit eine gerichtete (im Sinne heuristische), aber zufällige Suche im Lösungsraum, so daß das Erreichen einer bestimmten Lösungsqualität in einer bestimmten Zeit nicht garantiert werden kann. Allerdings kann GP als Any-Time-Algorithmus zu jedem Zeitpunkt die bis dahin beste Lösung (best-so-far solution) liefern. Je länger der Algorithmus arbeitet, desto besser werden die Ergebnisse.
Bei Nutzung der GP-Methode zur Evolution von Bewegungsmustern auf realen Robotern treten folgende Probleme auf:

- Kopplung der Fitness-Bestimmung an die Realzeit erzeugt große Versuchszeiten
- Hohe Anzahl von Parametern: 13 numerische, 6 strategische Parameter [7, S.114]
- Unbeabsichtigte Koevolutionseffekte

4 Versuchsaufbau

4.1 Versuchsziel

Inspiriert durch ein ähnliches Experiment an der Universität Dortmund [3] untersuchen wir die Umsetzbarkeit der GP-Methode zum Erlernen eines Bewegungsmusters zur zielgerichteten Vorwärtsbewegung eines realen Roboters mit zufälliger Morphologie (sog. Servorium, Abb. 1). Die Lösung dieses Problems soll die Leistungsfähigkeit der GP-Methode an einem komplexen Problem demonstrieren, für das mit anderen Methoden aufgrund der unbekannten Lösungskomplexität schwerlich Lösungen gefunden werden können. Zu den Kriterien der Anwendbarkeit des Verfahrens („Beschreibung der ökologischen Nische des GP“) siehe Kapitel 6.

4.2 Technischer Aufbau

Das zu bewegende Servorium wurde durch Anwenden eines Konstruktionsschemas mit einer zufälligen Komponente (Würfeln) definiert und aufgebaut, so daß ein Roboter vorlag, der nicht explizit für einfache Steuerbarkeit entworfen wurde. Konstruktionselemente sind einfache Servomotoren, Kugelgelenke und Gewindestangen.

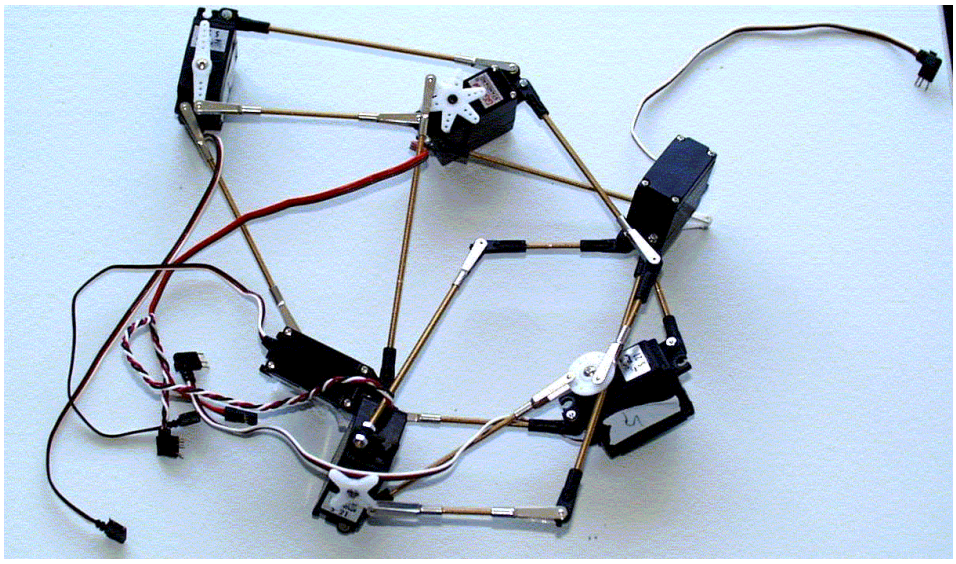


Abbildung 1: Servorium aus low-cost Bauteilen

Der Roboter schien aufgrund der relativ kräftigen Servomotoren das Potential zu einer gerichteten Vorwärtsbewegung zu besitzen und die Anwendung der GP-Methode sollte ein spezielles, nur für diese Morphologie gültiges Bewegungsmuster in der quasiunendlichen Menge möglicher Bewegungsmuster entdecken.

Zur Beobachtung der Fitness-Funktion boten sich 2 Varianten an:

- optische Erkennung mittels Markierung und Bildverarbeitung
- Odometrie eines zu ziehenden Wagens

Wir wählten aus pragmatischen Gründen die Odometrie-Variante und konstruierten aus LEGO-Technik ein Gefährt zur Aufnahme der Weglänge und Wegrichtung mit 2 Phasequadratur-Encodern (Abb. 2).

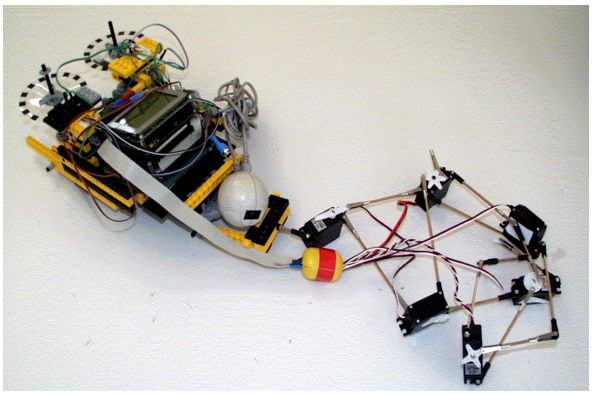


Abbildung 2: Servorium mit Odometriewagen und Steuereinheit

Zur Steuerung der Aktoren und Sensoren sowie zur Realisierung der GP-Kernels wurde ein angepaßtes Eyebot-Modul mit MC68332/25MHz/1MB [2] der Firma Jokerrobotics verwendet. Die Protokollierung des Lernverlaufs, sowie die Sicherung der Individuen zur späteren Verwendung erfolgte über eine serielle Schnittstelle auf einem PC. Abb. 3 stellt die gesamte Systemarchitektur dar.

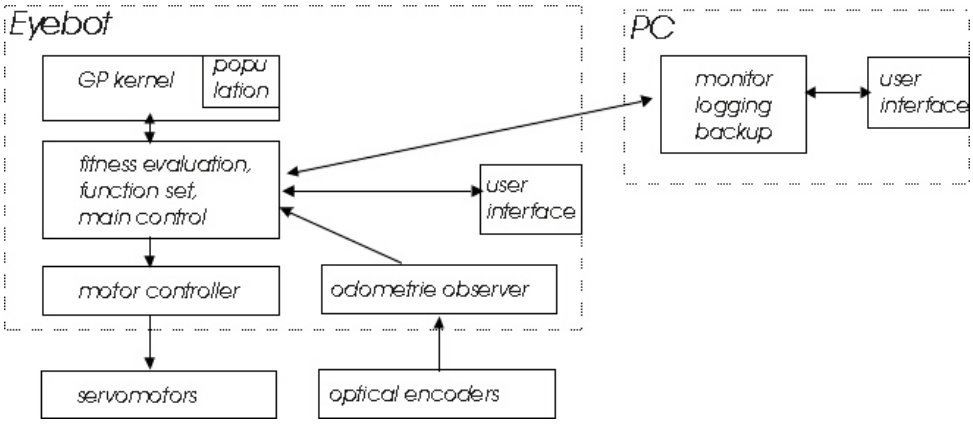


Abbildung 2: Systemarchitektur

Obwohl bis auf die Monitoringeinheit ein autonomer Betrieb vorgesehen war, war es nötig eine stationäre Stromversorgung zu nutzen, da sich der Stromverbrauch des Servoriums bei gegenseitiger Verspannung der Motoren auf über 4A erhöhte und somit nicht die Betriebsspannung des CPU-Moduls mit herkömmlichen Pb-Akkumulatoren gesichert werden konnte.

4.3 Anpassung der GP-Methode

Als GP-Kernel wurde das freiverfügbare objektorientierte GP-QUICK gewählt [10]. Die Anwendung der GP-Methode auf diese Problemstellung erforderte folgende Anpassungen:

Definition der Fitness-Funktion

Die Fitness-Funktion liefert der GP-Methode das Reinforcementsignal bezüglich der vom Nutzer definierten gewünschten Eigenschaften eines Lösungsmusters, in unserem Fall also eine Funktion der zurückgelegten Strecke und der Bewegungsrichtung. Die Bewertung eines individuellen Programms erfolgt dabei während einer vordefinierten Lebensdauer von 2 Sekunden, die für jedes Programm 2 mal wiederholt wird. Die zyklische Ausführung ermöglicht den Vergleich von Programmen unterschiedlicher Laufzeit und basiert auf der Annahme, daß ein sinnvolles Bewegungsmuster eine Periodenlänge kleiner 2 Sekunden vorweist. Außerdem werden somit Programme, die zyklisch ausführbar sind, d.h. die mit der Roboterkonfiguration, die sie selbst hinterlassen, besonders gut auskommen, durch die Evolution bevorzugt. Die Wahl der Wiederholungsanzahl (fitness cases), sowie der Ausführungszeit erwiesen sich als kritisch für den Verlauf der Experimente, da eine Testzeit von 4 Sekunden pro Individuum zu einer maximalen Generierungsrate von $900h^{-1}$ führt. Da sich die CPU während dieser Zeit im wesentlichen im Wartezustand befindet, wird also Zeit zur teuersten Ressource des Verfahrens.

Definition Funktionsset und der Terminale

Funktionen	
ADD, SUB, MUL	normale binäre arithmetische Operationen
DIV	geschützte Division
MSLEEP	Warten für x Millisekunden
SETSERVO1 .. SETSERVO6	Einstellen des Servomotors auf einen bestimmten Winkel
PROG4	Sequenz
Terminale	
CONST	Konstante im Intervall [-127,128]

Da die Terminale nur Werte enthalten, die zur Laufzeit des Individuums konstant bleiben, erübrigt sich die Aufnahme von Verzweigungsinstruktionen in das Funktionsset. Alle Funktionen im Funktionsset erfüllen die Closure-Annahme bezüglich des Datentypes ‚Byte', garantieren also die Abarbeitung für beliebige Parameter dieses Typs. Das Funktionsset erscheint ausreichend mächtig, um ein gewünschtes Bewegungsmuster auszudrücken, d.h. der Suchraum enthält wahrscheinlich eine Lösung (intuitive Annahme).

Einstellen der Parameter

Nach Koza sind beim GP-Verfahren 2 Hauptparameter (numerisch), 11 Nebenparameter (numerisch) und 6 Strategieparameter zu wählen, hier die wesentlichen:

Hauptparameter:

- Populationsgröße M=100
- max. Anzahl von Generationen bestimmt durch Versuchsdauer

Nebenparameter:

- Wahrscheinlichkeit Crossover: 0.8
- Wahrscheinlichkeit Reproduktion: 0.05
- Wahrscheinlichkeit Mutation: 0.15
- Wahrscheinlichkeit Permutation: 0
- maximale Baumtiefe: 20

Strategieparameter:

- Generierungsmethode: ramped half and half (50% grow, 50%full)
- Selektion: Tournament, Gruppengröße = 10
- Annealing Crossover: ja (das neue Individuum wird nur übernommen, wenn es eine bessere Fitness als seine Eltern erreicht)
- Annealing Mutation: ja (entsprechend)

Zur Aufteilung der Generierungsmethode in GP-Quick siehe Anhang 8.2

Definition der Versuchsdauer

Da sich GP-Methoden an ihrem Zeitverbrauch messen lassen müssen, stand für uns nicht die Anzahl von Individuen-Generierungen im Vordergrund, sondern die erreichte Qualität der Lösung nach einer fest definierten Versuchsdauer. Die Versuchsdauer wurde auf 8 Stunden festgelegt.

5 Experimente (1)

Phase 1

Geplant waren kontinuierliche Experimente mit veränderten Parametern (Populationsgröße, Generierungswahrscheinlichkeiten) und Vergleich der Fitnesskurven (Lernkurven). Es stellte sich jedoch heraus, daß bei keinem der ersten 5 Versuche eine signifikante Vorwärtsbewegung nach Ende des Versuches zu beobachten war. Statt dessen wurde eine Vielzahl von ‚pathologischen' Individuen beobachtet, die nur einen oder keinen Servobefehl enthielten und somit zu keiner sinnvollen Bewegung fähig waren. Das Fitnessniveau der gesamten Population blieb also konstant auf 0, so daß die heuristische Suche zum zufälligen Absuchen des Suchraumes degenerierte. Die Evolution sprang nicht an.

Phase 2

Um diesen Effekt zu verhindern, verkleinerten wird den Suchraum, indem offensichtlich ‚pathologische' Individuen keine Evaluierungszeit auf dem Roboter erhielten, sondern sofort mit der minimalen Fitness bewertet wurden. ‚Pathologisch' definierten wir als Programm, daß nicht mindestens 2 Servobefehle für mindestens 3 Servos enthielt.
Mit dieser Vorselektion wurden weitere 5 Versuche durchgeführt. Dabei entstanden stets nach wenigen Generationen in der ersten Stunde sehr bewegliche Populationen, die so gut wie keine ‚pathologischen' Individuen enthielten. Allerdings konnten auch hier nach 8 Stunden kaum erkennbare Vorwärtsbewegungen beobachtet werden.

Phase 3

Da die (zufällige) initiale Population offensichtlich einen großen Einfluß auf einen schnellen Erfolg der Methode ausübt, gingen wird dazu über, Populationen nach ca. einer Stunde subjektiv durch einen Beobachter zu beurteilen, der die vorherrschenden Bewegungen einer Population als vielversprechend oder nicht bewertet. Vielversprechende Populationen erhielten die volle Versuchsdauer von 8 Stunden, die anderen wurden abgebrochen und neu initialisiert. Dieses Vorgehen erwies sich als notwendig, denn aufgrund der Kopplung der Fitnessbestimmung an die Realzeit befinden wir uns nach einer Versuchsdauer immer noch in der ‚Initialisierungsphase' der Evolution, die enorm durch die initialen Individuen geprägt ist. Im Durchschnitt wurden 3-4 Populationen abgebrochen, um einen kompletten Versuch durchzuführen.
In 5 Versuchen nach diesem Schema konnte ein Individuum mit einer deutlich ausgeprägten Vorwärtsbewegung generiert werden. Die Endgeschwindigkeit dieses Programms bei unbegrenzter zyklischer Ausführung lag bei ca. 3 mh^{-1}. Das Programm übernahm die gesamte Population und konnte auch nach mehreren zusätzlichen Stunden nicht verbessert werden. Das Verhalten wurde auf Video dokumentiert.

6 Ergebnisse

Wir konnten zeigen, daß die GP-Methode prinzipiell in der Lage ist, komplexe Bewegungsmuster für Roboter mit zufälliger Morphologie zu generieren, die vom Anwender definierte Zieleigenschaften erfüllen. Folgende Probleme traten auf:

- Evolution kann die Initialisierungsphase durch die Kopplung der Fitnessbestimmung an die Realzeit nicht verlassen, somit dominierender Einfluß der initialen Population auf den Versuchsverlauf
- Unbeabsichtigte Koevolution führt zur Stagnation: Individuen wurden als fit bewertet, wenn das Individuum, das vorher den Roboter benutzte, diesen in einer günstigen, instabilen Lage hinterließ. Solange das vorbereitende Individuum in der Population verbreitet war, wiesen andere Programme eine gute, (aber nicht selbst verursachte) Fitness auf. Wenn das vorbereitende Programm aus der Population verdrängt wurde, verschlechterte sich rapide die

Fitness aller neu erzeugten Programme. Da die Fitness nur beim Generieren der Programme bestimmt wurde, dominierten die älteren gut bewerteten Programme die Population, ohne daß ihre Nachkommen sie verdrängen konnten. Die Evolution stagnierte. Der Effekt stellt einen möglichen Endpunkt der Evolution im frühen Stadium dar, in dem erst kleine Bewegungen entstanden sind.

Für die Anwendbarkeit der GP-Methode sprechen folgende Indikatoren:

- **fehlendes Prozeßmodell:** Der zu beeinflussende Prozeß ist nicht oder sehr schwierig modellierbar
- **unbekannte Komplexität:** Die Komplexität einer Lösung ist unbekannt oder schwer abschätzbar
- **Testbarkeit des Suchraumes:** Der Roboter kann seinen Bewegungsspielraum erforschen der (wie bei unserem Servorium) apriori keine Gefährdungen enthält oder durch Aktionsbeobachter auf ungefährliche Aktionen eingeschränkt ist (das Erlernen von Bewegungsmustern zur Landung eines Flugzeuges sind somit nicht möglich).
- **Wahrnehmbarkeit:** Das Reinforcement bezüglich der gewünschten Zieleigenschaften muß in relativ kurzer Zeit wahrnehmbar sein.
- **komplexe Lösungseigenschaften:** komplexe, auch widersprüchliche Eigenschaften einer gewünschten Lösung lassen sich in gewichteter Form in einer Fitnessfunktion vereinigen, z.B. schnelle Bewegung mit kleinem Energieverbrauch.

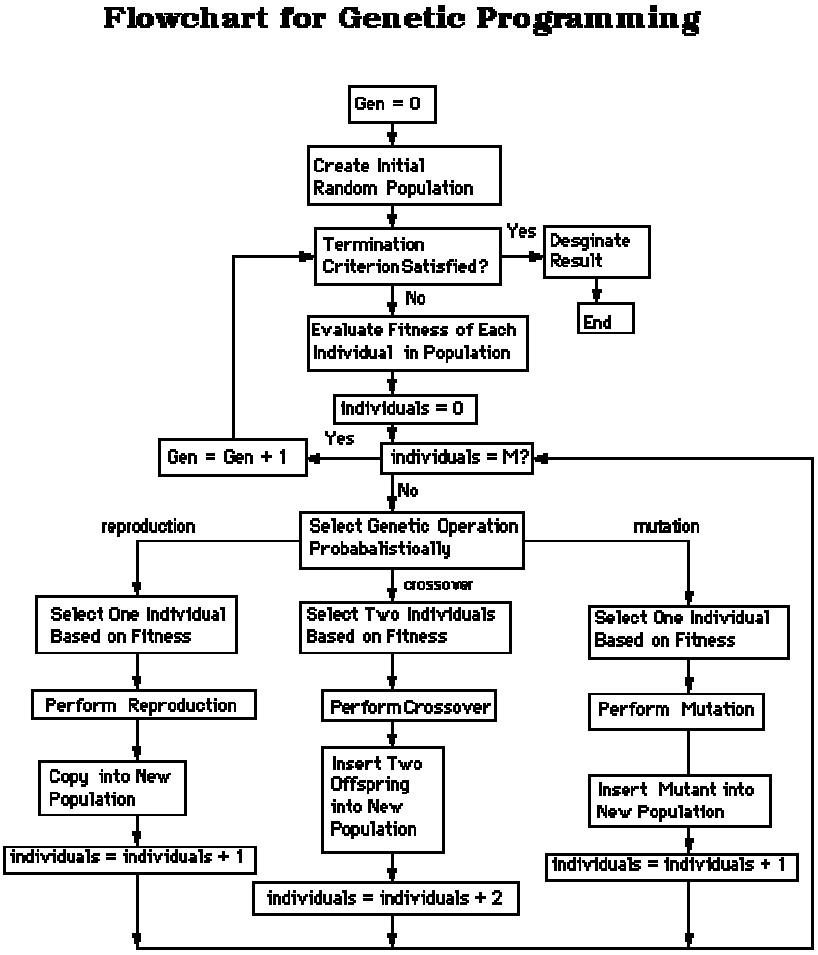
7 Ausblick

Mit dem beschriebenen Versuchsaufbau konnten wir die automatische Programmierung eines relativ schwierig zu findenden Bewegungsmusters mit Hilfe der GP-Methode demonstrieren. Allerdings konnte die eigentliche evolutionäre Optimierungsleistung der GP-Methode nicht genutzt werden, da die Evolution die Initialisierungsphase nicht verlassen konnte. Weitere Untersuchungen sind für folgende Fragen geplant:

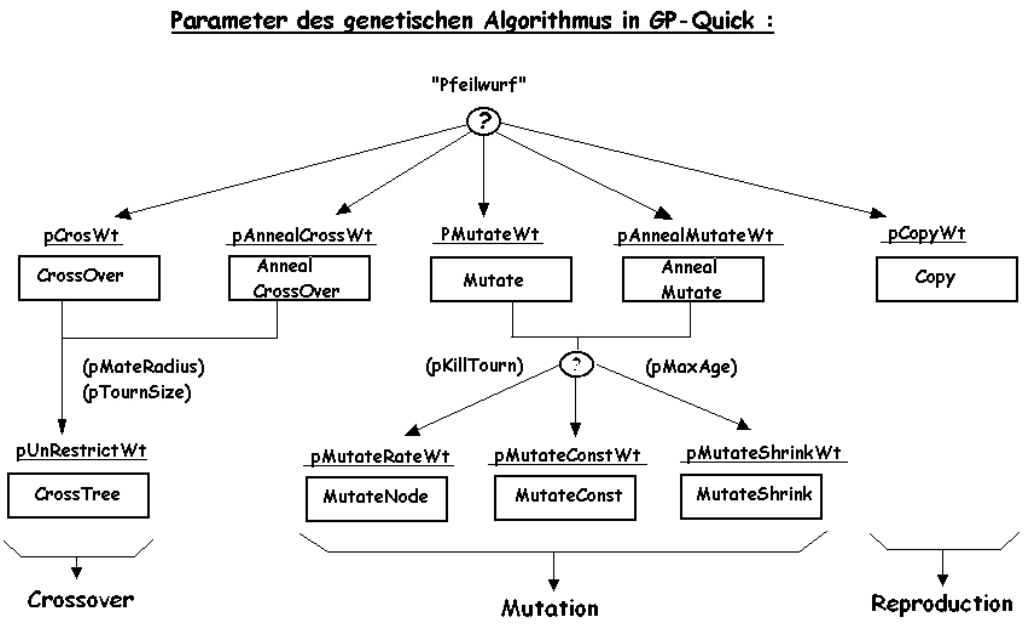
- Erhöhen der Generierungsrate durch Erlernen eines Prozeßmodells, mit dem Ziel der Entkopplung der Fitnessberechnung von der Realzeit (Adaptive Control, [6])
- Untersuchung der Initphase der GP-Methode: Wann ist es günstig, eine Population abzubrechen und neu zu initialisieren, wenn die zur Verfügung stehende Gesamtzeit (Anzahl der Generierungen) fest definiert ist?
- Übertragung des Versuchs auf eine sechsbeinige Rugwalker-Laufmaschine zum Erlernen eines Bewegungsmusters für eine schnelle, energiesparende Fortbewegung. Dieser Versuch sollte u.E. vielfältigere Lösungen als der hier beschriebene erzeugen, da die Motoren des Servoriums zwar kräftig, aber ohne Hebel in ihrer Wirkung stark beschränkt sind. Die Lösungsdichte im Suchraum der Laufmaschine ist voraussichtlich höher.

8 Anhang

8.1 GP-Flowchart nach [7]



8.2 GP-Quick-Parameter



Literatur:

1. Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D.: Genetic Programming – An Introduction, Morgan Kaufmann Publishers, San Francisco CA, dpunkt.verlag Heidelberg, 1998
2. Bräunl, Th.,:Eyebot – Online Documentation, http://www.ee.uwa.edu.au/~braunl/eyebot/
3. Dittrich, P., Buergel, A., Banzhaf , W.: Learning to Move a Robot with Random Morphology. In Evolutionary Robotics, First European Workshop, EvoRob98, Paris, France, April 1998, Proceedings, Phil Husbands and Jean-Arcady Meyer (eds.), LNCS 1468, pp. 165-178 Springer, Berlin, 1998
4. Holland, John H.: Adaption in Natural and Artificial systems, second edition, MIT Press, 1992, first edition 1975
5. Jaime, F.:The Genetic Programming Notebook, http://www.geneticprogramming.com
6. Kokar, M. M.: Learning Control – Methods, Needs and Architectures. In Antsaklis, P. J. and Passino K. M., editors, An Introduction to intelligent and autonomous control, pages 263-282, Kluwer Academic Publishers, Boston, 1993
7. Koza. J. R.: Genetic Programming, MIT Press, Cambridge MA, 1992
8. Mitchell, M., Forrest, S.: Genetic Algorithms ans Artificial Life. In Christopher G. Langton, editor, Artficial Life – An Overview, pages 267-289, MIT, Cambridge MA, 1995
9. Reynolds, C. W.: An evolved, Vision-Based Model of Obstacle Avoidance Behavior. In Cristopher G. Langton, editor, Artificial Life III, pages 327-346, Addison-Wesley Publishing Company, 1993
10. Singleton, A.: GPQUICK – A simple Genetic Programming system in C++, ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/gp-code/

Autorenangaben:

Dipl.-Inform. Ingo Boersch
FH Brandenburg
FB Informatik und Medien
Psf 2132
14737 Brandenburg
E-Mail: boersch@fh-brandenburg.de