Algorithmische Anwendungen

WS 06/07

Evolutionäre Algorithmen

erstellt von:
Sven Nissel, 11042398, ai716@gm.fh- koeln.de
Christian Fehmer, 11042419, ai696@gm.fh- koeln.de
Grupp e D_gelb_Ala0607
am 23.01.07

Fachhochschule Köln
Campus Gummersbach
Allgemeine Informatik

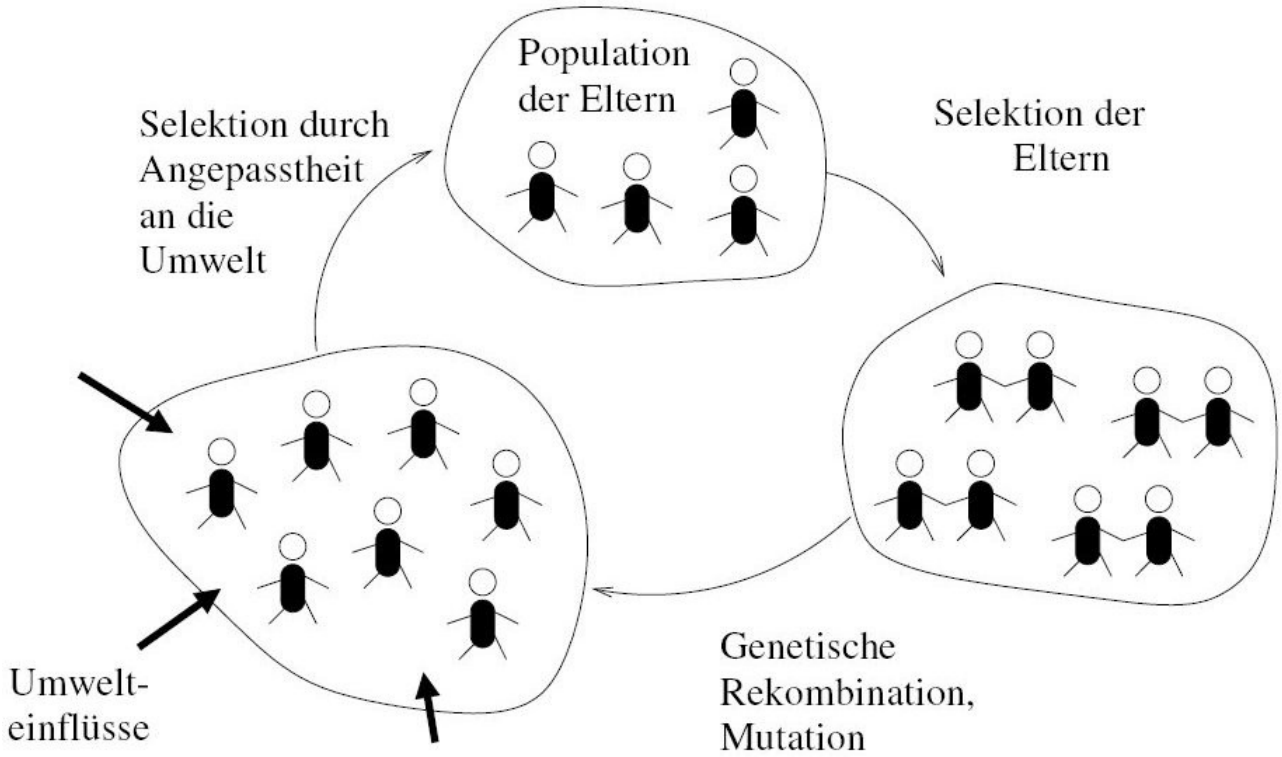
Inhaltsverzeichnis

1 Einführung

Unter dem Begriff "Algorithmus" versteht man eine Verarbeitungsvorschrift. Diese Vorschrift ist fest vorgegeben sodass sie von einem Computer abgearbeitet werden kann.
Mit Evolutionären Algorithmen wird eine Gruppe von Algorithmen bezeichnet, die bei der Lösungssuche den Evolutionsprozess simulieren. Evolutionärer Algorithmen sind bereits seit den sechziger Jahren bekannt.

Um einen Evolutionsprozess zu simulieren benötigt man drei Verfahren der Evolution:

1. Mutation
   Durch ungerichtete Veränderung wird der Algorithmus und/oder deren Parameter angepasst. Der Grad der Mutation gibt die Stärke der Veränderung an.
2. Rekombination/Vererbung
   Ein Algorithmus entsteht immer aus seinen „Eltern“ und Kombiniert deren Eigenschaften. Im Unterschied zur Natur könnten auch mehr als zwei „Eltern“ als Vorlage dienen.
3. Selektion
   Erfolglose Lösungsansätze werden nicht weiterverfolgt. Wird aber ein Lösungsansatz zu früh abgebrochen können evt. auch nur lokale Maxima gefunden werden.



1.1 Vorteile Evolutionärer Algorithmen

- Durch die zufällige Mutation gibt es evt. mehrere Lösungsmöglichkeiten
- Das Problem muss vom Algorithmus nicht verstanden werden, da er „try and error„ mögliche Lösungen ausprobiert. Dies ist gerade bei sehr komplexen oder ungelösten Problemen von Vorteil
- Die Suche nach Lösungen läuft Parallel. Dadurch können Evolutionäre Algorithmen leichter für stark verteilte Systeme konzipiert werden

1.2 Nachteile Evolutionärer Algorithmen

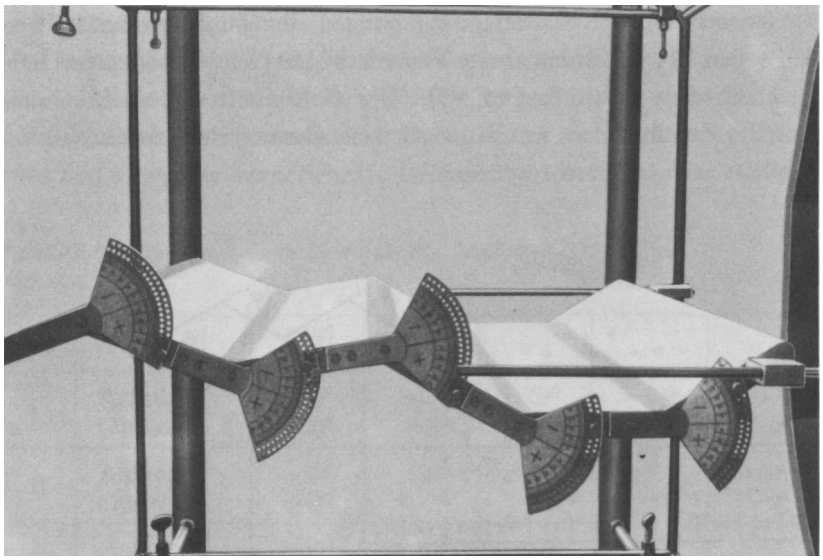
- Lange Laufzeiten, da viele Lösungsansätze ausprobiert und verworfen werden.
- Findet evt. nur lokale Maxima bzw. Minima eines Problems.
- Gibt es für ein Problem schon ein Optimierungsverfahren, dann sind Evolutionäre Algorithmen oftmals langsamer und ineffizienter

2 Ursprung

Seit den 50er Jahren haben sich Wissenschaftler mit der Evolutionären Programmierung auseinander gesetzt. Die Anfänge bildeten hier z.B. Friedman [friedman56] über Selektion, oder Friedberg [friedberg58], [friedberg59] über eine lernende Maschine.

Praktische Ansätze wurden erstmals von Rechenberg 1973 in seinen Buch über die Evolutionsstrategie [rechenberg73] erarbeitet. Bis dahin wurden zwar versucht die Evolution im Computer oder mathematischen Modellen nachzubilden, aber ein praktischer Nutzen wurde darin nicht gesehen.

Rechenberg setzte Mutation und Selektion ein um z.B. den Strömungswiederstand eines Flügel zu optimieren. Da die Rechenkapazitäten zu dieser Zeit nicht ausreichten um einen Windkanal zu simulieren, musste Rechenberg eine flexible Gelenkplatte bauen um verschiedene Mutationen aus zu probieren.



Dabei wurden zufällig die Winkel der einzelnen Segmente verändert und danach der Strömungswiederstand gemessen. Dieser Versuchsaufbau zeigt den hohen Aufwand der Evolutionsstrategie. Jeder Veränderung muss gemessen werden um sie anschließend bewerten zu können. Das man mit dieser Methode ein Optimum findet scheint im ersten Augenblick reiner Zufall. Am Ende seiner Experimente konnte Rechenberg sogar teilweise bessere Ergebnisse liefern, als bis dahin bekannt gewesen war.

3 Anwendungsgebiete

Grundsätzlich können alle Probleme (z.B. in der Informatik) die zu komplex sind oder für die es kein Optimierungsverfahren gibt mit Evolutionären Algorithmen untersucht werden. Doch oftmals schränkt die Rechenleistung die Möglichkeiten ein. Gerade bei großen Problemen mit vielen Parametern ist die nötige Rechenleistung riesig.

Ein sehr interessantes Anwendungsgebiet sind Ampelnetze. Schon wenige Ampelschaltungen können starke Wechselwirkungen aufeinander haben und sind mit herkömmlichen Algorithmen nur unzureichend gut zu lösen. Um ein Ampelnetz optimal zu steuern, müssten unzählige Parameter berücksichtigt werden:

- Wetter
- Geschwindigkeitsbegrenzungen
- Anfahrtsverzögerungen
- Bremswege
- Verkehrsaufkommen
- Störungen durch z.B. Unfällen
- Entfernung zwischen Ampeln
- ...

Ein Evolutionärer Algorithmus würde nach einer Ampelschaltung suchen, die möglichst optimal ist, ohne die Eingabeparameter zu berücksichtigen. Er würde nur die Auswirkungen überprüfen und damit eine Selektion durchführen. Bei einer starken Mutation sollte man solch einen Algorithmus nicht in der Praxis einsetzen, da es zu unvorhersehbaren Problemen kommen kann. In der einer Simulation könnte aber ein solcher Algorithmus neue Lösungsideen finden.

4 Qualitäts- und Tauglichkeitsdichte

Ziel der Evolutionsstrategie und auch jedes Mathematischen Ansatzes ist es die Qualität bzw. die Tauglichkeit einer Konstruktion zu erhöhen.

Versucht z.B. ein Ingenieur eine Maschine zu optimieren wird er versuchen am Ende eine Maschine zu erfinden, die das gegebene Problem am besten löst. Damit der Ingenieur entscheiden kann welche Entwicklung die bessere ist, muss eine Wertigkeit eingeführt werden. Die Maschine muss also im gesamten bewertet werden.

Zum optimieren einer Maschine stehen dem Ingenieur gewisse Bauelemente zur Verfügung z.B. Stäbe, Platten, Rohre, Spulen usw. Dabei ändert er nur die einige charakteristischen Merkmale der Grundelemente, wobei die Funktionsidee der einzelnen Elemente erhalten bleibt. Es werden geometrische Abmessungen, physikalische Zustandsgrößen, Stoffeigenschaften und ähnliches abgewangelt. [rechenberg73]

4.1 Parameter

Die Merkmale der Grundelemente einer Konstruktion nennt man in der Programmierung Parameter. Also müssen die Parameter (Merkmale) so lange abgeändert werden bis die maximale Tauglichkeitsdichte (Fitnesswert) gefunden wird. Damit diese Vorgehensweise erfolgreich sein kann und nicht vollkommen den Zufall überlassen ist, muss sichergestellt sein, dass eine unendlich kleine Änderung im Parameterraum eine unendlich kleine Änderung der Tauglichkeitsdichte zur Folge hat. Hat eine kleine Änderung eines Parameters mal große und mal keine Auswirkungen auf die Tauglichkeitsdichte, so ist es unmöglich sinnvoll im Parameterraum zu selektieren.

5 Varianten der Evolutionsstrategien

5.1 (1 + 1) – Evolutionsstrategie

Bei der (1 + 1) – Evolutionsstrategie wird eine Generation kopiert. Die daraus entstandenen Kinder werden zufällig verändert (mutiert). Die Kinder werden mit ihren Vätern (jedes Kind hat nur einen Vater) verglichen und das schlechtere Objekt wird gelöscht.

Man kann mit dieser Methode nur lokale Maxima finden. Dafür können die Nachfolgegenerationen nie schlechter sein als ihre Vorgänger.

Da man mit der (1 + 1) – Evolutionsstrategie nur in monoton steigenden Funktionen globale Maxima finden kann, gibt es oftmals schon eine Mathematisch schnellere Lösung. [evocomp]

5.2 (μ + λ) – Evolutionsstrategie

Bei der (μ + λ) – Evolutionsstrategie werden aus μ Vätern λ Kinder generiert. Dabei wird genauso wie bei der (1 + 1) – Evolutionsstrategie einfach die Väter kopiert und mutiert . Wenn λ > μ, dann findet eine Mehrfachauswahl der Väter statt.

Aus der Menge der Eltern und Kindern werden die μ besten Objekte ausgewählt.

Mit dieser Methode findet man wie bei der (1 + 1) – Strategie nur lokale Optima. Dieses Verfahren ist aber schneller, da die besten Objekte aus der Gesamtmenge übernommen werden. [evocomp]

5.3 (μ , λ) – Evolutionsstrategie

Die (μ , λ) – Evolutionsstrategie wurde von Hans- Paul Schwefel eingeführt. Hierbei werden auch aus μ Eltern λ Kinder erzeugt. Doch bei der Selektion nur die μ besten Kinder ausgewählt und die Väter grundsätzlich vergessen.

Wenn λ nur geringfügig größer als μ ist, sind sehr wahrscheinlich auch viele Kindergenerationen schlechter als ihre Väter Ist λ wesentlich größer als μ, so ist es sehr wahrscheinlich, dass viele Kindergenerationen besser sind als ihre Väter.

Durch dieses Verfahren ist es auch möglich globale Maxima zu finden. Die Größe der Qualitätsschwankungen in den Nachfolgegenerationen kann über die Mutation und das Verhältnis zwischen Anzahl der Väter und Anzahl der Kinder gesteuert werden.

Das Nachkommen schlechter als ihre Vorfahren sein können ist ein klarer Nachteil dieses Verfahrens. [evocomp]

5.4 (μ / ρ # λ) – Evolutionsstrategie

Das (μ / ρ # λ) – Verfahren bildet die Evolution am realistischsten ab. Dabei werden mit λ mal p Eltern λ Kinder erstellt. Es werden also aus der Menge μ immer p Eltern herausgenommen um ein Kind zu erzeugen. Dies wir λ mal wiederholt.

Für das Erzeugen der Kinder gibt es zwei Möglichkeiten.

1. Bilden von Durchschnittswerten des Parameter Arrays

   Beispiel:

```
1 ElternTeil[0] = {5,9,2}
2 ElternTeil[1] = {1,4,8}
3 ElternTeil[2] = {4,4,4}
4 Kind = {(5+1+4)/3, (9+4+4)/3, (2+8+4)/3}
```

2. Die Eltern Parameter werden mithilfe einer Maske Vertauscht und aus der Ergebnismenge wird zufällig ein Array gewählt.

   Beispiel:

```
1 ElternTeil[0] = {5,9,2}
2 ElternTeil[1] = {1,4,8}
3 ElternTeil[2] = {4,4,4}
4 Maske_für_0 = {0,2,1}
5 ElternTeil[0] = {5,4,8}
6 ElternTeil[1] = {1,4,2}
7 ElternTeil[2] = {4,9,4}
8 SelectRandom(ElternTeil)
```

   Hierbei ist zu beachten, dass auch mehrere Vertauschungsmasken eingesetzt werden könnten. Im Beispiel wurde nur das Elternteil 0 mit 1 und 2 vertauscht. Aber nicht 1 mit 2.

Mit dem (μ / ρ # λ) – Verfahren können einmal die durchschnittlich besten Parameter gefunden werden, oder es können die Parameter aus den besten Eltern miteinander kombiniert werden. Hierbei ist für den Selektionsdruck wieder abhängig von der Größe von λ. [evocomp]

6 Anwendung auf das Problem traffic control

6.1 Problemstellung

Aufgabe des Evolutionären Algorithmus soll ein ein Straßennetz mit verschiedenen Ampeln bzw Kreuzungstypen zu optimieren.

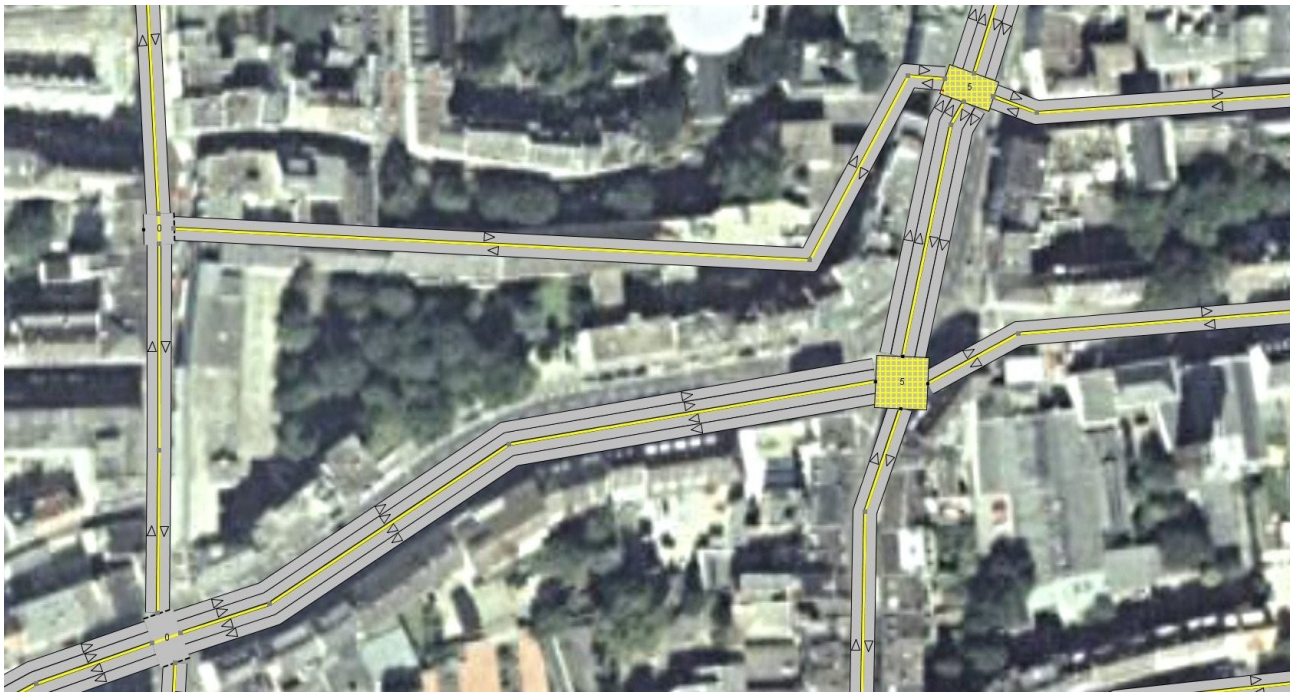


Abbildung 1: vier Kreuzungen in Köln

Gegeben ist ein Straßennetz mit sieben Kreuzungen. Es gibt mehrere Ein- und Ausgänge die aus denen unterschiedlich viele Autos rein oder raus fahren.

Auf den Kreuzungen sollen entweder Ampeln aufgestellt werden oder rechts vor links Kreuzung installiert werden.

Am Ende soll eine Kombination an Ampeln und/oder Rechts vor Links Kreuzungen gefunden werden, die für den geringsten Stau und schnellsten Durchfluss von Autos sorgen.

Um einen Evolutionären Algorithmus umsetzen zu können, muss man sich im klaren sein, was in seiner Problemstellung das Kind, die Generation und die Population ist.

In diesem Fall sind die Kreuzungen die Gene des Kindes. Jedes Gen besteht aus zwei Basenpaaren (Kreuzungstyp und Schatzeit).

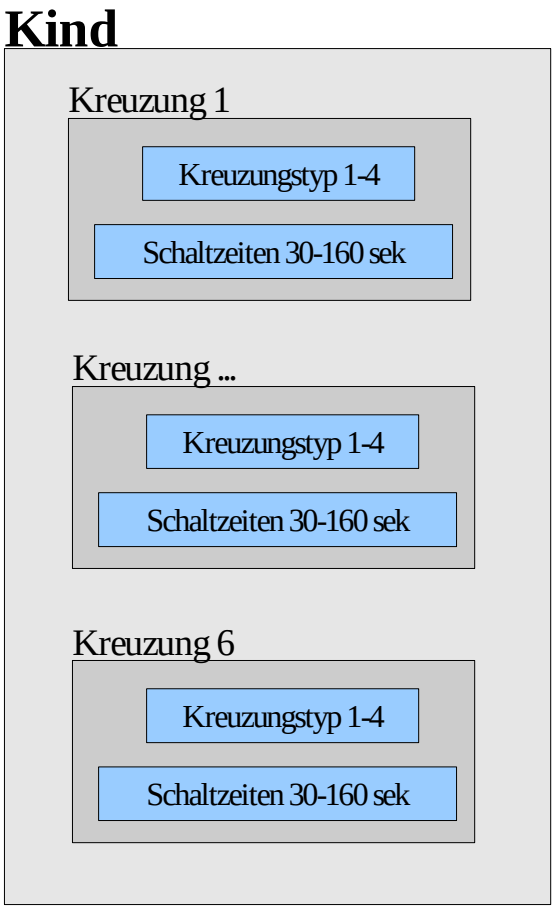


Abbildung 2: Kind mit Parametern

Eine Generation beinhaltet eine begrenzte Anzahl an Kindern um den Rechenaufwand in Grenzen zu halten.

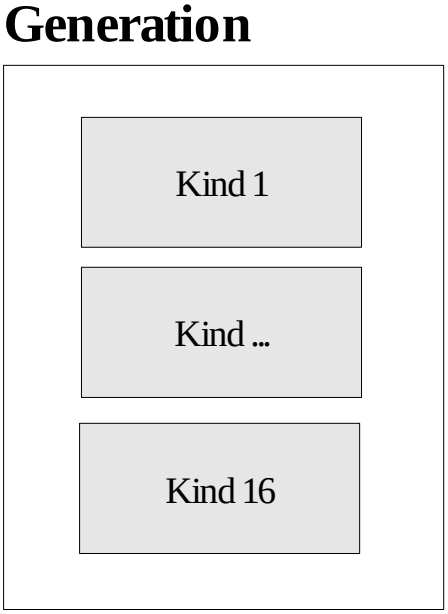


Abbildung 3: Generation mit Kindern

In einer Population können unendlich viele Generationen enthalten sein. Das Beste Kind aus einer Population ist der Beste gefundene Wert.

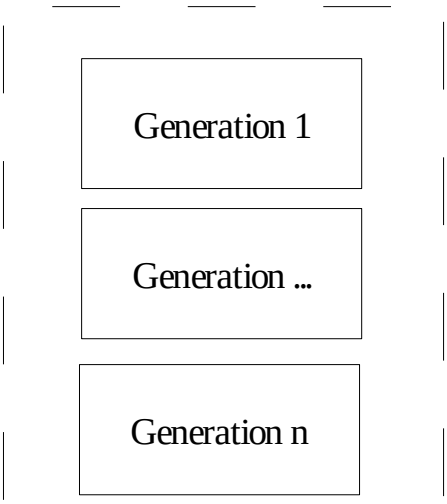


Abbildung 4: Population mit n Elementen

7 Simulations- und Evolutions- Umgebung

Wie schon Anfangs erwähnt, werden die Evolutionären Algorithmen anhand eines Ampelnetz getestet. Dafür werden fünf Komponenten benötigt:

1. Simulation
2. Darstellung der Simulation
3. Bewertung der Simulation bzw. deren Messwerte
4. Evolutionäres Selektieren und Kombinieren
5. Darstellung der erstellten Generationen

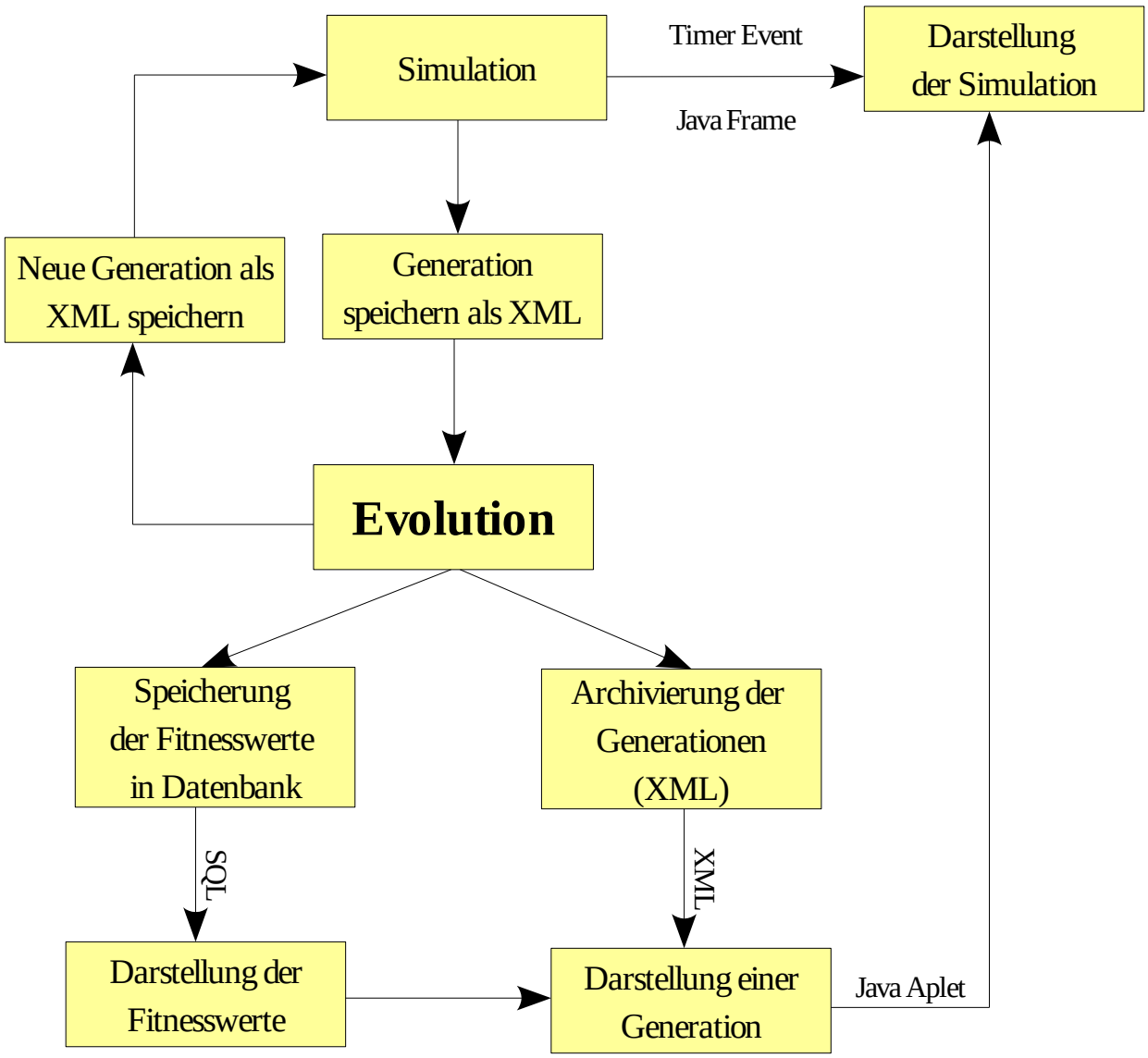
Dieses Paket von Programmteilen hat den Namen

plain **e**volutionary **a**lgorithm for **n**aive **u**rban **t**raffic **s**imulations

zu deutsch: einfacher evolutionärer Algorithmus für naive Stadtverkehrssimulationen.

7.1 Aufbau

Damit ein reibungsloser Ablauf zwischen den Komponenten gewährleistet ist, wurden klare Schnittstellen definiert. Diese Schnittstellen ermöglichten unabhängig voneinander entwickelbare Komponenten. Teilweise war die Trennung der Programmteile schwieriger, wie z.B. bei der Simulation und deren Darstellung. Andere Teile wie die Darstellung der Bewertung konnten ganz klar getrennt werden.



Der Datenaustausch über die Komponentengrenzen hinaus wurde mithilfe von Java Interfaces, XML und SQL umgesetzt.

7.1.1 Simulation

Die Simulation ist auf Grundlage von [vissim] weiterentwickelt worden.

Die Simulation stellt hauptsächlich das Fahrverhalten von Kraftfahrzeugen nach. Dabei wurde besonderen Wert auf Gleichmäßigkeit und vorausschauendes Fahren gelegt. Die Fahrzeuge beschleunigen gleichmäßig bis zur Höchstgeschwindigkeit. Die Fahrzeuge bremsen frühzeitig ab und beschleunigen erst wenn der Abstand zu vorherigen Fahrzeug groß genug ist.

Die Simulation musste an vielen „Ecken“ korrigiert, erweitert und angepasst werden um die Kommunikation mit anderen Komponenten gewährleisten zu können.

7.1.2 Darstellung der Simulation

Die Darstellung ist eng mit der Simulation verknüpft. Da das die Grundlage des Projekts nur zur Darstellung gedacht war konnten nicht alle Darstellungskomponenten entfernt werden. Doch das deaktivieren der meisten Grafischen Ausgaben konnte das Programm ca. um den Faktor 50 beschleunigen. Die Darstellung des Straßennetzes wurde auf eigene Bedürfnisse angepasst. So wird auch ein Stadtplan im Hintergrund geladen um so ein besseres Verständnis für die Simulation zu bekommen. Die srollbarkeit von Karten verursachte große Änderungen im Quellcode und machten die Ausgabe etwas langsamer.

Damit die Simulationen auch im Browser angezeigt werden können musste das Java Applet komplett umgeschrieben werden. Änderungen am Uhrsprungsprogramm hatten die Verwendung des „alten“ Applets unmöglich gemacht.

7.1.3 Evolution

Die Evolutions Komponente speichert die Generationen als XML – Dateien. Dies ist die Schnittstelle zu der Simulation und Simulationsdarstellung. Die Parameter in den XML – Dateien werden kombiniert, mutiert und selektiert. Die Ergebnisse einer Simulation werden als so genanter Fitnesswert in eine MySql Datenbank gespeichert. Dies ist die Schnittstelle zur Komponente die für die Darstellung der Fitnesswerte zuständig ist.

7.1.4 Darstellung der Fitnesswerte

Die Fittnesswerte werden mithilfe von PHP aus der Datenbank geladen und als HTML Seite angezeigt. Dazu werden Graphen und Tabellen erzeugt um eine Übersicht über die Ergebnisse zu gelangen. Diese Komponente zeigt von der ungefähren Rechenzeitnutzung der Generationen über Generationsübersichten bis hin zur Darstellung einzelner Parameter einer Generation alles über den Simulationsablauf und Evolutionsporzess.

7.1.5 Darstellung einer Generation

Um eine Generation im Webfrontend anzeigen zu können wird die passende XML Datei geladen und dem Java Applet übergeben. Dies zeigt ähnlich wie im Java Fenster den kompletten Ablauf der Simulation an.

7.2 Funktionsweise

Die wichtigsten Elemente des Programms sind Simulation, Evolution und Auswertung. Deswegen wird auf andere Bereiche nicht eingegangen um den Fokus auf das wesentliche zu halten.

7.2.1 Simulation

Eine Simulation kann die Realität nur bedingt gut nachstellen. Das begrenzen auf Parameter und das Vereinfachen und Digitalisieren von reellen Situationen führt automatisch zu einer Verfälschung des Ergebnisses. Deswegen sollte großen Wert darauf gelegt werden, dass die Vereinfachungen nur zu geringen Verfälschungen führen. Zwar wäre eine „Realitätsnahe“ Simulation die beste Lösung, doch leider erlauben heutige Rechenkapazitäten diesen Rechenaufwand nicht. Deswegen bleibt es bei einen Modell der Situation.

Bei der Simulation wurde auf folgende Eigenschaften besonderen Wert gelegt:

1. **Gleichmäßiger aber nicht Gleicher Fluss an eingehenden Autos**

   Damit die Ampeln sich nicht auf einen immer gleich bleibenden eingehenden Autofluss einstellen können, werden die Zeitabstände für das Einfahren in die Stadt leicht variiert.

   Damit aber die Generationen vergleichbar bleiben, sind die Zeitabstände im Durchschnitt gleich.

2. **Unterschiedliche Spuren Anzahl**
3. **Unterschiedlicher Zustrom von unterschiedlichen Richtungen**
4. **Gleich bleibendes Verhalten der Autos**

   Damit die Generationen untereinander vergleichbar sind, ist das verhalten der Autos auf eine Situation immer gleich.

   Dies zeigt schon eine Schwäche unseres Modells auf. Niemand verhält sich in der gleichen Situation genau gleich. Doch den verschlafenden Autofahrer der einfach über rot fährt oder den Fahranfänger der nicht sauber anfahren kann zu simulieren erweist sich als sehr schwierig und als unendlichen Unterfangen.

5. **Verschiedene Kreuzungstypen**
   - Links vor Rechts
   - Links vor Rechts mit Abbiegespur
   - Ampeln werden im Uhrzeigersinn geschaltet
   - Ampelschaltung (Wegerecht und Senkrecht gleich behandelt)
6. **Verschiedene Ampelschaltzeiten**
7. **Beliebtheit von Straßen**

7.2.2 Evolution

Für die Nachstellung der Evolution wurde die $(\mu / \rho \# \lambda)$ – Evolutionsstrategie und im geringfügigen Maße die (1+1) – Evolutionsstrategie genutzt.

Grundsätzlich werden die Eltern mit einer zufällig erstellten „Gen“- Maske miteinander kombiniert. Diese Maske wird aus Performancegründen pro Generation nur einmal erstellt. Mit einer einstellbaren Wahrscheinlichkeit (Standard 0,5%) werden die „Kindergene“ Mutiert.

Die „Kindergene“ besthen aus folgenden Parametern:

6 Ampeln bestehend aus:

- 6 verschiedenen Ampelschaltzeiten
- 4 verschiedenen Kreizungstypen (2 ohne Schaltzeiten)

Damit gibt es:

$(6 Schaltzeiten \cdot 2 \; n)^{6} + (2 \; n)^{6} = 2.986.048$ verschiedene Kreuzungskombinationen

Bei einer Berechnungdauer von ca. 120 Sekunden pro Kind würde man 4147 Tage benötigen um alle Kombinationen auszurechnen.

In der Population „mutation- 05“ wurde schon ab der 20 Generation ein guter Fitnesswert gefunden. In Generation 87 wurde der beste der Population gefunden.

$20 \; n \cdot 48 \; r \cdot 120 \; n = 32 \; n$

$87 Generationen \cdot 48 \; r \cdot 120 Sekunden = 5{,}8 Tage$

7.2.3 Auswertung

Um aus den vielen Berechnungen eine Schlussfolgerung ziehen zu können, sind drei Werte einer Generation sehr wichtig:

1. Bester Wert (Kind)
2. Durchschnittswert
3. Varianz

Der beste Wert ist für die Suche nach der besten Ampleschaltung am wichtigsten. Selbst wenn alle anderen Kinder einer Generation „schlecht“ sind, kann der beste Wert einer Generation der Beste aller Generationen sein und somit der gesuchte Wert.

Der Durchschnittswert einer Generation gibt die Qualität einer Generation an. Die nachfolge Generation wird bei einen hohen Durchschnittswert sehr wahrscheinlich auch einen hohen oder höheren Durchschnittswert haben. Durch die Selektion erhöht oder hält sich in der Regel der Durchschnittswert.

Die Varianz gibt die Unterschiede in einer Generation an. Eine Generation mit vielen guten und schlechten Kindern hat eine hohe Varianz. Bei einer hohen Mutation ist die Varianz sehr wahrscheinlich auch ziemlich hoch.

Abbildung 5: „Mutation 05“ - 40 Generationen

Wie in Abbildung 5 zu sehen, steigt der Durchschnittswert (blau) an ohne merklich ab zufallen. Da ran erkennt man, das der Selektionsdruck hoch ist und kaum schlechtere Generationen zulässt. Ähnlich und fast parallel verhält sich die „Besten- “ Kurve (orange). Die Gene der besten Kinder bleiben größtenteils durch Kombination und Selektion erhalten. Somit verschlechtern sich auch hier die Werte nur wenig. Die Varianz zeigt deutlich die Dynamik in der Evolution. Bis sich eine Population angepasst hat, ist die Varianz sehr hoch. Danach bestimmt hauptsächlich der Mutationsgrad die Höhe der Varianz.

Abbildung 6: „Mutation 06“ - 96 Generationen

Bei längeren durchführen der Simulation ändern sich die Kurven in Abbildung 6 kaum noch. Zwar liegen ab Generation 20 viele der besten Werte über 200 (Fitnesswert), aber in der Generation 87 wurde der beste Wert mit 235 gefunden.

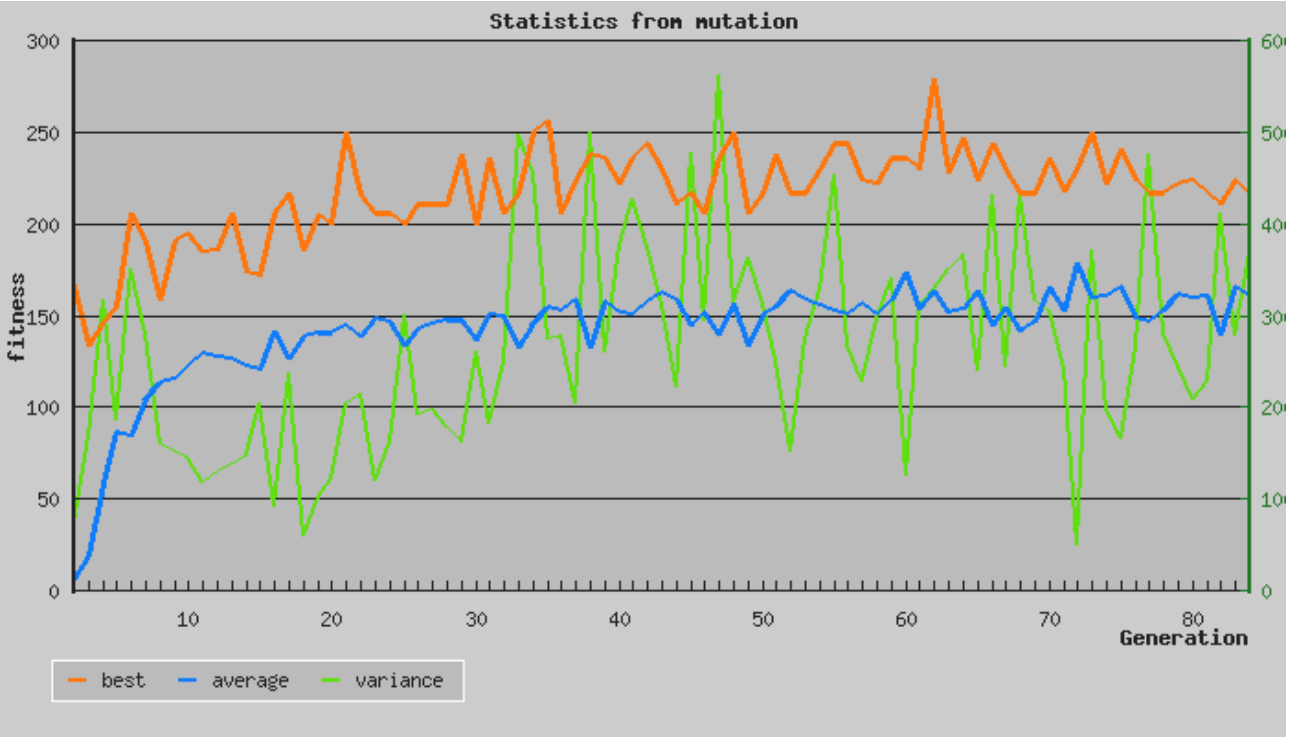


Abbildung 7: „Mutation“ - 84 Generationen

Bei einer hohen Mutationsrate wie in Abbildung 7 zu sehen, schwanken sich die Werte stärker. Trotzdem verschlechtern sich sie Werte kaum. Es werden sogar deutlich bessere Werte gefunden, die durch die zufällige Mutation entstanden sind.

8 Fazit

Beim Einsatz von Evolutionären Algorithmen kommt man in dem Beispiel des traffic control System schnell zu brauchbaren Ergebnissen. Auch wurden unerwartete Ampelschaltungen generiert, die zu Beginn der Entwicklung als schlecht eingeschätzt wurden.

Sind aber die Problemstellung Mathematisch bzw. durch einen Einfachen Algorithmus lösbar, so sind Evolutionäre Algorithmen zu langsam und geben häufig auch nur gute, aber nicht den besten Wert aus.

Da aber viele Probleme in der Praxis sehr Komplex sind, ist das Anwendungsgebiet für Evolutionäre Algorithmen sehr groß.

Das Gebiet der Evolutionäre Algorithmen ist sehr groß und für viele Probleme benutzbar. Es kann auf diesem Gebiet noch viel geforscht werden.

Wegen der Vielseitigkeit des Themas war dieses Projekt sehr interessant und lehrreich.

9 Glosar

Basenpaar Ein Basenpaar enthält einen Parameter eines Kindes.

Eltern Ein Elternteil ist ein Kind welches zur Kinder Erzeugung genutzt wird.

Fitnesswert Der Fitnesswert gibt die Quallität eines Kindes an. Die Grundlage für diesen Wert sind oftmals Messungen aus der Simulation.

Gen

Generation Eine Generation beinhaltet eine begrenzte Anzahl von Kindern bzw. Eltern. Ein Kind wird nicht in eine andere Generation kopiert. Eltern geben ihre Gene nur durch Kombination an ihre Kinder weiter.

Kind Ein Kind ist das Objekt, welches die Gene enthält. Es wird dann Kind genannt, wenn es gerade erzeugt wurde.

Meta- Heueristik Neben den Heueristiken gibt es problemunabhängige Lösungsverfahren, die so genannten Metaheueristiken deren Ziel es ist, globale Lösungen zu konstruieren, ohne in schlechten lokalen Optima zu enden. Metaheueristiken sind Vorgehensweisen zur Steuerung untergeordneter Heueristiken, die auf ein bestimmtes Problem abgestimmt sind. Dabei werden die Ergebnisse der untergeordneten Heueristiken iterativ ausgewertet. Auf diese Weise will man möglichst effizient Lösungen ermitteln, die beim globalen Optimum liegen. Zu diesen Metaheueristiken gehören neben Simulated Annealing, Tabu Search und Threshold Accepting auch die Evolutionsalgorithmen (auch Evolutionäre Algorithmen genannt). Hier unterscheidet man wiederum zwischen zwei unterschiedlichen Verfahrensweisen: den Evolutionsstrategien und den Genetischen Algorithmen. Der Einsatz dieser Verfahren ist allerdings erst ab einer gewissen Komplexität der Problemstellung sinnvoll.[evocomp]

Mutation Gibt an wie häufig ein Gen Zufällig verändert wird.

Population Eine Population beinhaltet eine Menge an Generationen. Eine Generation wird nicht in eine andere Population hineinkopiert.

Selektionsdruck Gibt an wie stark in einer Generation selektiert wird.

Tauglichkeitsdichte Anderer Begriff für Fifnesswert.

10 Quellen

[fogel66] Fogel, L.G. , Owens, A.J. & Walsh, M.J. (1966). Artificial intelligence through simulated evolution. New York: John Wiley and Sons

[friedman56] Friedman, G. J. (1956). Selective feedback computers for engineering synthesis aund nervous system analogy. Masters- /Diplomarbeit, University of California, Los Angeles.

[rechenberg73] Rechenberg, Ingo (1973). Evolutionsstrategie: Optimierung technischer Evolution. Stuttgart: frommann- holzbog

[vissim] Tom Fotherby (stand 12.12.2006) http://www.tomfotherby.com/Contents/Education/Project/VISSIM_applet.html

[evocomp] Alexander Müller (stand 12.12.2006) http://www.evocomp.de

10.1 Weitere Literatur

[1] http://www.fh- meschede.de/public/willms/ea/

[2] http://de.wikipedia.org/wiki/Evolution%C3%A4rer_Algorithmus

[3] http://www.heise.de/tr/artikel/79237

[friedberg58] Friedberg, R. M. (1958). A learning machine: Part I. IBM Journal of Research and Development, 2(1), 2- 13.

[friedberg59] Friedberg, R. M., Dunham, B. & North, J. H. (1959). Alearning machine: Part II. IBM Journal of Research and Development, 3(3), 282- 287.