Genetic Algorithms

Seminar „KE und Lernen in Spielen“

Bearbeitet von: Felix Becher

Leiter des Seminars: Prof. Dr. Johannes Fürnkranz

Gliederung

- Evolutionstheorie
- Genetische Algorithmen
- Vor- und Nachteile
- Beispiele
- Diskussion

Evolutionstheorie

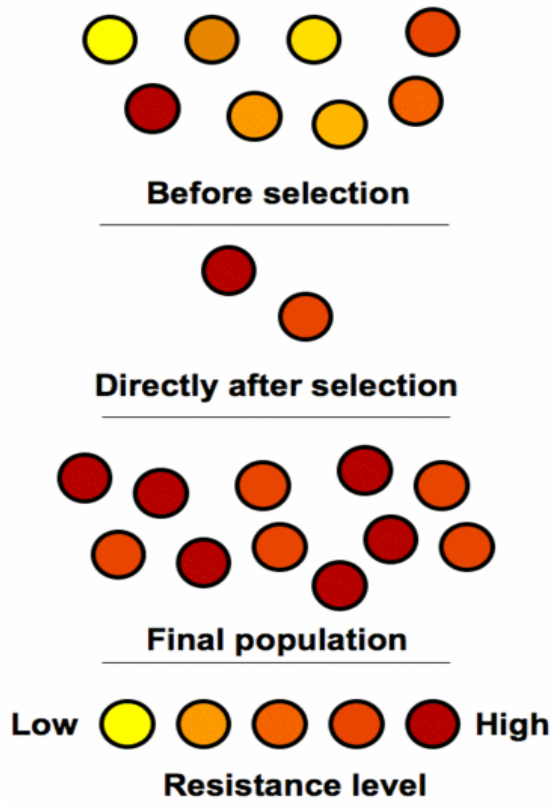
- Jeder Organismus hat einzigartige Attribute, die an die Nachkommen weitergegeben werden können
- Die Nachkommen sind einzigartig und besitzen Eigenschaften der Eltern
- „Selective breeding“ (Zuchtwahl) wird benutzt um Änderungen in nachfolgenden Generationen zu beeinflussen
- Dank dem Druck der Natur entwickeln sich Individuen mit der Zeit

Evolutionärer Druck

- Umwelt
  - Um zu Überleben müssen Individuen etwas leisten bspw. Nahrung oder Wasser suchen
- Wettbewerb
  - Individuen einer Spezies stehen im ständigen Wettbewerb miteinander (bspw. um einen Partner zu finden)
- Rivalität
  - Verschiedene Spezies beeinflussen sich gegenseitig
    - *direkt* durch Konfrontation oder
    - *indirekt* durch Wettbewerb um Ressourcen

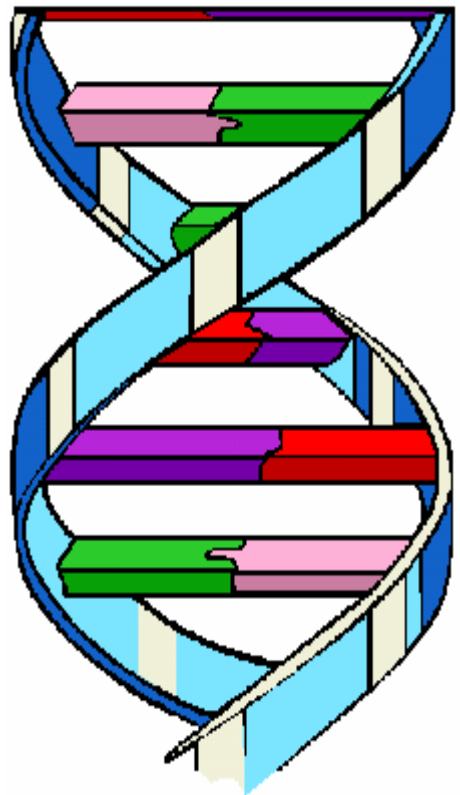
Natürliche Selektion (Natural Selection)

- Individuen, die im Wettbewerb besser abschneiden haben mehr Möglichkeiten zur Fortpflanzung. Dadurch haben die Nachkommen wiederum bessere Chancen zu überleben.



Genetics

- Genom (das Erbgut)
  - Das Genom enthält die Informationen, die zur Entwicklung (Ontogenese) der Bau- und Leistungsmerkmale eines Lebewesens oder eines Virus notwendig sind.
- Allel
  - ist eine der möglichen Ausprägungen eines Gens, das sich an einem bestimmten Ort (Locus) auf einem Chromosom befindet.
- Genotyp
  - ist eine Sequenz der Allele und bezieht sich auf die vollständige Kombination aller Allele aller Loci eines Organismus.
- Beziehung zw. Genom/Genotyp ist ähnlich der Beziehung Klasse/Instanz

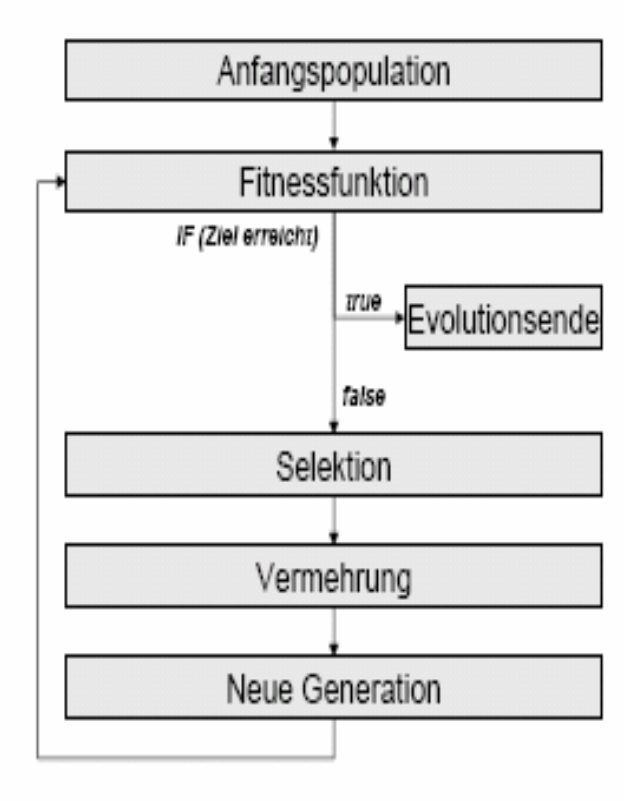


Genetic Algorithms

- Was sind sie?
  - Evolutionäre Algorithmen die Operationen wie Mutation, Kombination und Selection benutzen
- Mögliche Anwendungsgebiete?
  - Suchprobleme
  - Optimierunsprobleme
  - Machine learning
  - Adaptive rule-bases

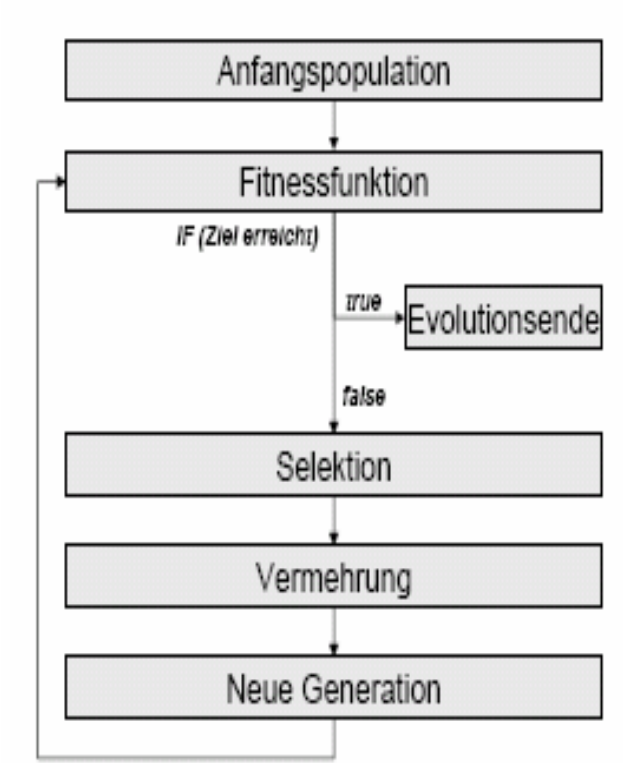
Struktur eines genetischen Algorithmus

1. Erschaffe einige Lösungen (Individuen)
2. Bewerte die Lösungen
    1. Falls gut genug, dann Ende
    2. Sonst
3. Entnehme die Besten
4. Erschaffe Nachkommen
5. Gehe zu 2



Genetic Algorithmic Process

- Mögliche Lösungen der Probleme sind verschlüsselt bspw. als bit strings
- Durch Crossover und Mutation werden aus alten neue Individuen erschaffen
- Die Fitness Funktion entscheidet darüber welche Individuen “überleben” sollen (survival of the fittest)



Initialisierung (Anfangspopulation)

- Eine geeignete Kodierung der Variablen der zu optimierenden Funktion ist notwendig (bspw. Bit string oder Array)
- Die erste Population muss den Suchraum möglichst gut abdecken
- Sie wird durch zufällige Werte erzeugt (z. B durch Mersenne-Twister Algorithmus)
- Je größer die Probe, um so besser (schneller) die Ergebnisse
- Ergebnisse die durch andere Verfahren gewonnen wurden, können (müssen) verwendet werden (seeding)

Evaluation

- Jedes Individuum der Population ist eine zulässige Lösung des Problems (Genotyp)
- Die fitness Funktion
  - bewertet die Lösungen im Hinblick auf das zu lösende Problem
  - verarbeitet die kodierten Werte zu einer Zahl
  - Je besser die Funktion umso besser das Ergebniss (im schlechtesten Fall kommt die Evolution nicht voran)
  - Kann sehr einfach sein (problemabhängig)

Selektion

- Idee: die ausgewählten Individuen sollen sich fortpflanzen.
- Fortpflanzung kann sexuel (mehrere Vorfahren) oder asexuell (1 Vorfahre) erfolgen
- Macht nur dann Sinn wenn man “bessere” Individuen präferiert (elitism)
- Problem: je strenger die Anforderungen an die Elite sind um so schneller schreitet zwar die Evolution voran, *aber* um so schneller wird auch der Lösungsraum “eingeschnürrt”.

Evolution - Crossover

- Sexuelle Fortpflanzung
  - Der genetische Code beider Vorfahren wird kombiniert
  - vorteilhafter, wenn keiner der Vorfahren stark bevorzugt wird
- Asexuelle Fortpflanzung
  - bringt einen der Lösung nicht näher. Hohe Mutationsrate ist nötig!

Evolution - Crossover

- Angenommen Individuen werden durch zwei Strings repräsentiert

  A = 11000 und B = 00111

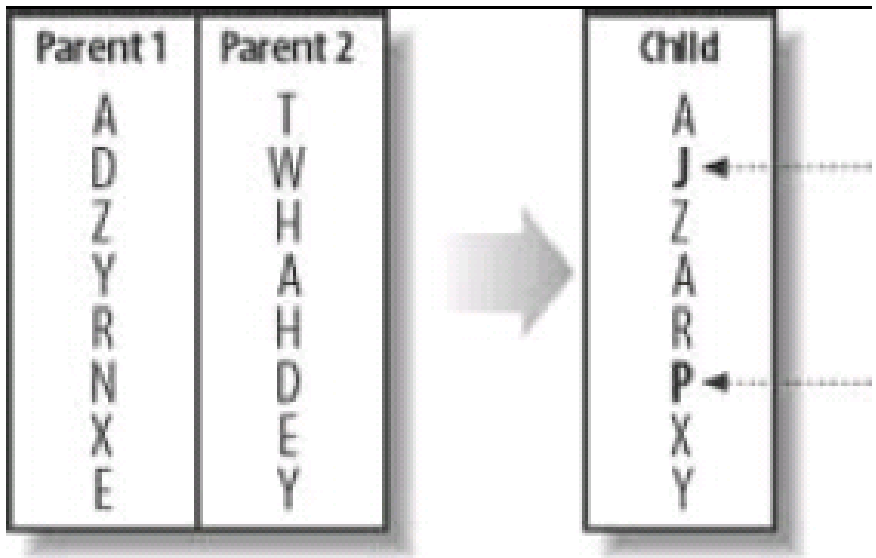
- Durch Zufall wird eine Grenze gewählt (z. B an der zweiten Position)
- Zwei neue Individuen werden erzeugt

  A’ =11111 and B’ = 00000

- Die neuen Individuen(Lösungen) können wiederum (falls sie die Selektion überstehen) gepaart werden

Mutation

- Mutation ist wichtig um die (biologische) Vielfalt zu bewahren
- Kann sich sowohl auf Genom-Niveau wie auch auf Genotype-Niveau abspielen
- Bei GA's mutiert meistens nur das Gen und nicht das Genom. Die Struktur bleibt also stabil.



Operatoren

- Crossover
  - single point crossover
  - two point crossover
  - uniform crossover
- Mutation
  - Gene Mutation
  - Structural Mutation
    - inversion (swap)
    - Insertion
- Selection (Fitness Scaling)
  - Random walk
  - Ranking
  - Tournament
  - Roulette Wheel
- Replacement
  - Entscheidet darüber ob/wie neue Individuen in die Population aufgenommen werden
  - Mögliche Strategien:
    - Vorfahren werden durch Nachkommen ersetzt
    - neue (starke) Individuen ersetzen alte und schwache
    - Älteste Individuen werden durch neuen ersetzt
    - Neuen Individuen ersetzen andere in ihrer Nähe

Konvergenzverhalten

- Manchmal ist ein Gen viel stärker als andere und dominiert dadurch (genetic drift)
- Falls dass Optimum erreicht ist, so ist es wünschenswert anderenfalls kann es passieren dass das Optimum nie gefunden wird
- Je kleiner die Menge der Lösungen um so höher ist die Wahrscheinlichkeiten für “genetic drift”
- Höhere Mutationswahrscheinlichkeiten sind ein probates Mittel dagegen

Konvergenzverhalten

- Stärkere Rolle des Zufalls ist gut bzw. schlecht
- Um “genetic drift” zu vermeiden
  - betreibt man sub-populationen
  - The Island Principle
  - Crowding und Sharing
- Lansame Konvergenz
  - Man züchtet Nachkommen aus Vorfahren die Nachbarn sind oder aber aus Vorfahren die erwünscht sind (restricted mating)
  - Erhöhung des Elitism-Niveaus

Stärken und Schwächen

- Funktioniert gut bei der Suche nach einem globalen Optimum, da die Suche im ganzen Lösungsraum erfolgt
- Kann erweitert werden durch genetische Operatoren. Gibt dem Entwickler die Möglichkeit sein Wissen über das Problem an GA weiter zu geben um die Qualität der Lösung oder die Geschwindigkeit zu optimieren
- Einfache Implementierung für komplexe Probleme

Stärken und Schwächen

- Der Algorithmus ist gegenüber gleichgültig gegenüber speziellen Eigenschaften des jeweiligen Problems
- Nicht effizient bei lokaler Optimierung (fine tuning). Alternative: greedy hill-climbing method
- Keine Vorhersage über die Dauer der jeweiligen Suche möglich. Problematisch, wenn schnelle Lösungen erforderlich sind
- Sogar mit einafchen Genotypen stößt man schnell an Grenzen des Möglichen (Speicher und Rechengeschwindigkeit). Möglichkeiten zur Optimierung von Lösungen in interaktive Anwendungen sind dadurch eingeschränkt.

Beispiel – Höhensuche

von Sebastian Schulz und Bastian Koell

- Codelänge: 2 integer, x und y Position
- Fitness: 1 integer, Höhe auf der Karte
- Mutation: Keine
- Selektion: Je höher desto besser
- Crossover: Aus 2 Individuen, indem dem Kind eine zufällige Position im durch die Eltern aufgespannten Rechteck zugewiesen wird

Beispiel – Höhensuche

von Sebastian Schulz und Bastian Koell

```
public void doTheEvolution(…….){
	for (int i=0; i<numGenerations; i++){
		for (int q=0;q<numCreatures;q++){
			int fitness=getFitnessAt(creatures[q].getX(), creatures[q].getY());
			myCreature.setFitness(fitness);
		}
		Arrays.sort(creatures);
		for (int q=0;q<numCreatures/2;q++){
			int papax=creatures[..].getX();
			int papay=creatures[..].getY();
			int mamax=creatures[..].getX();
			int mamay=creatures[..].getY();
			creatures[q]=new Creature(
				(int) (Math.min(papax,mamax)+Math.abs(mamax-papax)*Math.random()),
				(int) (Math.min(papay,mamay)+Math.abs(mamay-papay)*Math.random())
			);
		}
	}
}
```

Beispiel - Kenny vs. Jason

von Sebastian Schulz und Bastian Koell

- Das Beispiel:
  - Kenny ist ein GA, Jason kämpft nach festen Wahrscheinlichkeiten seiner Waffen
  - Jeder von beiden hat 3 Angriffsattacken und 3 Verteidigungen.
  - Jede Verteidigung ist spezifisch gegen einen Angriff

Beispiel - Kenny vs. Jason

von Sebastian Schulz und Bastian Koell

Kenny

Angriff	Schaden	Verteidigung	Schaden
Blitz-Attacke	10	Augen zu machen	5
Psi-Kontrolle	14	Ohren zu machen	7
Gabel-Attacke	16	**** - Abwehr	8

Jason

Angriff	Schaden	Verteidigung	Schaden
Böseguck en	10	Blitzschild	5
Schreien	14	Talkshow	7
****	16	Verstecke n	8

Beispiel - Kenny vs. Jason

von Sebastian Schulz und Bastian Koell

- Spielablauf:
  - Ein Kenny kämpft gegen einen Jason (Papier-Stein-Schere).
  - Insgesamt finden 1000 Kämpfe pro Generation statt.
- Getötete Kennys:
  - Getötete Kennys werden durch neue, rekombinierte/mutierte ersetzt
  - Jason bekommt volle Gesundheit
- Getötete Jasons:
  - Getötete Jasons werden geheilt
  - Kennys werden geheilt
  - In diesem Fall passiert Kenny nichts, weil er sich als guter Kämpfer herausgestellt hat

Beispiel - Kenny vs. Jason

von Sebastian Schulz und Bastian Koell

- Genetischer Code:
  - Double[6], jedes Double enthält die Wahrscheinlichkeit für eine Aktion
- Fitnessfunktion:
  - Differenz der Lebensenergie zwischen Kenny und Jason
- Rekombination:
  - Rekombiniert jedes neue Gen aus dem Mittelwert zweier Eltern.
  - Insgesamt wird aus sechs Eltern rekombiniert.
- Mutation:
  - Die Mutationsrate ist relativ hoch gewählt, damit sich Kenny möglichst schnell auf Veränderungen seines Gegners einstellen kann.
  - Mutationsrate: 20% , Wertänderung auf einem Gen: 3%

Ergebnisse aus MyreKrig und UT

- MyreKrig
  - die Ameisen zeigten nach 20 Generation eindeutige verbesserungen in was Suchstrategien angeht
  - Trotzdem nicht mit besten ANTs vergleichbar
- UT Bot mit GA und GT implementiert
  - Allerdings eine sehr einfache Map
  - nach 50 Generationen überlegener als das Original
  - das Original untralg sowohl dem Menschen als auch BOT aus 32er Generation (andere sub-population)
  - BOT aus 82er Generation war erfolgreich gegen den Menschen

GA‘s in kommerziellen Spielen

- GA‘s ziemlich selten da sehr rechenintensiv
- „NeuralBot“ für Quake 2 - ist auf wenig Akzeptanz seitens der Spieler gestoßen
- Ein experimenteller Bot für Unreal im Rahmen einer Magisterarbeit an der University of Aalborg
- „"Yes, very cool project... but NeuralBot will never really understand the 3D world or the specic map. Jan „MrElusive“ Paul, IdSoftware
- „"In a shoot-em-up like Quake, the "smart" thing for a creature to do would be to run like hell when they saw you coming. After all, they've just seen you waste three of their pals. Trouble is, that's boring. They'd just go and hide in a corner and shoot you in the back the first chance they get... the game would be reviewed as unfair and bad gameplay... but hold on; that's real AI!“

GA‘s in kommerziellen Spielen

- Weitere Beispiele
  - Schach und Gpotello
  - Half-Life
  - Creatures
  - Virtual Petz
  - Black & White

Verwandte Ideen

- Genetic Programming
- Evolutionary Strategies
- Evolutionary Programming

Quellen

- J. Holm, J.D Nielsen: *Genetic Programming – Applied to a Real Time Game Domain*, 2002
- Ingibjörg Asta Runarsdottir, Kenneth Vittrup, Morten Zinck, Thomas Winterberg, Kasper Orum Nielsen, Frederik Dannemare, Peter Sönder: *GeneAnt 2003*, 2003
- Jason Jones: *Benefits of Genetic Algorithms in Simulations for Game Designers*, 2003
- Sebastian Schulz und Bastian Koell: *Genetische Algorithmen*, 2005
- Ross Graham, Hugh McCabe and Stephen Sheridan: *Pathfinding in Computer Games*
- Viviane Gal, Cécile Le Prado, Stéphane Natkin, Liliana Vega: *Writing for video games*
- Aliza Gold: *Academic AI and Video games: a case study of incorporating innovative academic research into a video game prototype*, 2005

Quellen - Bücher

- David M. Bourg, Glenn Seemann: *AI for Game Developers*, O'Reilly, 2004
- Alex J. Champandard: *AI Game Development*, New Riders Publishing, 2003
- Zbigniew Michlewicz: „Genetic Algorithms + Data Structures = Evolution Programs“, 3rd revised and extended ed., Springer Verlag, 1996
- Steve Rabin. *AI Game Programming Wisdom*. Charles River Media, 2002.
- D. E. Goldberg. *Genetich Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publisching Company, Inc., 1989

Danke für Ihre Aufmerksamkeit!