Algorithmen und Datenstrukturen 2

Sommersemester 2007
12. Vorlesung

Peter F. Stadler

Universität Leipzig
Institut für Informatik
studla@bioinf.uni-leipzig.de

Genetische Algorithmen

Übersicht

- Wie funktionieren genetische Algorithmen?
- Für welche Aufgabenstellungen sind sie (vielleicht) anwendbar?
- Wo liegen die Schwierigkeiten?
- Beispiele aus verschiedenen Bereichen.

Entwicklungsbiologie

Lebewesen durchlaufen eine Entwicklung

- vom Niederen zum Höheren
- vom Einfachen zum Komplexen
- hin zu erfolgreicherem Handeln

Evolutionäre Entwicklung

- erfolgt ausschließlich durch Fortpflanzung über viele Generationen
- in jeder Generation die Besten bevorzugt

Evolution kann als Lernprozess verstanden werden:

> Evolution optimiert die Lösung der Probleme, die durch die Umwelt vorgegeben sind.

Vererbung

Bei der Fortpflanzung werden Eigenschaften der Eltern auf die Kinder vererbt.

Vererbung bedeutet

- eine (zufällige) Kombination der Eigenschaften der Eltern
- evtl. weitere zufällige Abweichungen durch Mutation.

Beobachtung:

Es treten praktisch nur sinnvolle Kombinationen und nur sehr wenige destruktive Mutationen auf.

Vergleich:

Versuchen Sie, ein C-Programm durch eine „zufällige Mutation“ so zu ändern, dass es besser wird!

Genetik (Minimalprogramm)

Erbinformation wird in mehreren *Genen* gespeichert.

Für uns:

- Gen ist Platzhalter für ein Zeichen aus einem gegebenen Wertevorrat.
- Das konkrete Zeichen heißt *Allel*.

Bei der Fortpflanzung werden mittels *Crossover* die Gene der Eltern kombiniert, um die Gene eines Kindes zu erhalten.

Beispiel: Elterngene: `abcdefg` und `ABCDEFG`

Crossover nach Position 4

Gene der Kinder: `abcdEFG` und `ABCDefg`

Erbgut von Lebewesen

Erbgut des Menschen

- 100.000 Gene bilden eine Folge aus
- 3.000.000.000 Symbolen

Mykoplasmen

- einfachste freilebende Lebewesen
- besitzen noch 500 bis 800 Gene

Ähnlichkeit des Erbgutes

Das Erbgut zweier Menschen stimmt zu 99,9% überein.

Auch die Ähnlichkeit zu anderen Organismen ist groß:

Schimpanse	98,7%
Maus	>95%
Fliege	50-60%
Fadenwurm	40%
Hefe	30-50%
Banane	ca. 15%

Quelle: Focus Nr. 16 / 2002

Algorithmische Sicht

- Gene beschreiben eine Menge von Parametern eines abstrakten Individuums.
- Es gibt eine Population aus vielen solchen Individuen.
- Individuen können sich nach einem vorgegebenen Schema vermehren (Crossover und Mutation).
- Jedes Individuum kann mittels einer Fitness-Funktion bewertet werden. Diese Funktion benutzt gerade die in den Genen gespeicherten Parameter.
- Selektion: Individuen können aus der Population entfernt werden (sterben) wegen hohen Alters oder mangelnder Fitness.

Problemlösen mittels GA

Für ein Problem soll aus einer großen Menge von möglichen Konfigurationen eine möglichst optimale Lösung gefunden werden.

- Die möglichen Konfigurationen werden durch Gene beschrieben.
- Die Fitness beschreibt die „Nähe“ zur optimalen Lösung.

Ziel ist es, in einer Population eine optimale Lösung zu züchten.

Erstes Beispiel

Gesucht sind 5 ganze Zahlen a, b, c, d, e, so dass die Funktion

$$F = |(a-b)| + |(b-c)| + |(c-d)| + |(d-e)| + |(e-a)|$$

minimiert wird. (Lösung: a = b = c = d = e).

Kodierung: Als einfachste Möglichkeit fassen wir in den Genen die freien Variablen zusammen: (a,b,c,d,e)

Fitness: $F \rightarrow \min$

Rekombination: einfaches Crossover an Position co:

- wähle $1 \leq co \leq 4$,
- kopiere die Allele 1 bis co vom ersten Elternteil und die Allele von co+1 bis 5 vom zweiten.

Mutation an einer Position um ± 1 ist möglich.

Details zu Beispiel 1

- Startpopulation aus 50 Individuen
- Fünf Gene (für jede Zahl ein Gen), Wertebereich [-50,50]
- Vermehrung mit einfachem Crossover, Mutation ±1 mit Wahrscheinlichkeit 1% bei jedem Gen
- Durch zufällige Eltern-Paare werden 50 Kinder erzeugt.
- Von den nun insgesamt 100 Individuen überleben die 50 mit der besten Fitness.

Bsp. 1: Generationen 0 und 1

Rang	Fit-ness	Generation 0					Fit-ness	Generation 1				
1	**24**	-33	-29	-26	-35	-38	**24**	-33	-29	-26	-35	-38
2	**74**	-4	6	22	0	-15	**74**	-4	6	22	0	-15
3	**78**	-21	-3	-16	-29	-42	**78**	-21	-3	-16	-29	-42
4	**82**	5	-8	-14	-17	24	**82**	5	-8	-14	-17	24
5	**90**	-3	10	18	17	41	**90**	-3	10	18	17	41
6	**96**	3	-34	-29	-33	-40	**94**	-38	-32	8	-14	-13
7	**112**	-38	-32	8	-14	-48	**96**	3	-34	-29	-33	-40
8	**114**	-28	2	5	14	-43	**102**	-12	-19	31	-2	-13
9	**118**	-5	4	-6	44	24	**112**	-38	-32	8	-14	-48
10	**120**	18	-1	-22	-7	38	**114**	-28	2	5	14	-43
...	**...**	...	...	...	...	...	**...**	...	...	...	...	...
49	**282**	-1	-33	34	-34	40	**166**	10	46	-6	-2	41
50	**346**	38	-49	12	43	-43	**166**	18	15	11	-40	43

Bsp. 1: Generationen 15 und 46

Rang	Fit-ness	Generation 15					Fit-ness	Generation 46				
1	**20**	-33	-29	-28	-35	-38	**2**	-33	-33	-32	-33	-33
2	**20**	-33	-29	-28	-35	-38	**2**	-33	-32	-32	-33	-33
3	**22**	-33	-29	-27	-35	-38	**4**	-33	-31	-31	-33	-33
4	**22**	-33	-29	-27	-35	-38	**4**	-33	-31	-31	-33	-33
5	**22**	-33	-29	-27	-35	-38	**4**	-33	-31	-31	-33	-33
6	**22**	-33	-29	-27	-35	-38	**4**	-33	-31	-31	-33	-33
7	**22**	-33	-29	-27	-35	-38	**4**	-33	-32	-31	-33	-33
8	**22**	-33	-28	-26	-35	-37	**4**	-33	-31	-31	-33	-33
9	**22**	-33	-29	-27	-35	-38	**4**	-33	-31	-31	-33	-33
10	**22**	-33	-29	-27	-35	-38	**4**	-33	-31	-31	-33	-33
...	**...**	...	...	...	...	...	**...**	...	...	...	...	...
49	**24**	-32	-29	-26	-35	-38	**4**	-33	-31	-31	-33	-33
50	**24**	-33	-29	-26	-35	-38	**4**	-33	-31	-31	-33	-33

Allgemeiner Genetischer Algorithmus I

Vorbereitung

- Fitness-Funktion problemangepasst festlegen
- Abbruchkriterium festlegen
- Genetische Kodierung: lernrelevante Parameter des Systems durch Gene eindeutig kodieren
- Genetische Operationen festlegen (Vermehrung, Mutation)
- Selektionskriterien festlegen

Allgemeiner Genetischer Algorithmus II

Durchführung

1. Initialisierung: Ausgangspopulation erzeugen
2. Rekombination: Nachkommen erzeugen mittels der genetischen Operationen
3. Fitness bestimmen
4. Selektion: Bilden der neuen Population durch Selektion (z.B. entsprechend Qualität, Alter)
5. Abbruch? Falls Abbruchkriterium erfüllt, dann STOP.
6. Gehe zu Schritt 2.

Selektionsmethoden

- Ein Kind wird jeweils von einem Elternteil erzeugt:

(μ+λ)-Strategie: Die Population besteht aus μ Individuen, daraus werden λ Kinder erzeugt, die μ fittesten Individuen überleben.

(μ,λ)-Strategie: Die Population besteht aus μ Individuen, daraus werden λ>μ Kinder erzeugt, die μ fittesten dieser Kinder überleben. [Sinnvoll ist z.B. eine (10+100)-Srategie.]

- Ein Kind wird jeweils von k Elternteilen erzeugt:

(μ/k+λ)-Strategie: Population aus μ Individuen, daraus λ Kinder, die μ fittesten überleben. [Im Beispiel wurde also eine (50/2+50)-Srategie benutzt.]

(μ/k,λ)-Strategie: analog

Fitness maximieren

- Fitness ist eine Funktion F von mehreren Variablen.
- Die Suche nach Individuen mit großer Fitness entspricht der Suche nach dem Maximum von F.

Es gibt verschiedene Verfahren für verschiedene Typen von Aufgaben:

- Gibt es ein oder mehrere lokale Maxima?
- Wie glatt ist die Funktion (z.B. differenzierbar)?
- Wie weit darf die Lösung vom globalen Maximum entfernt sein?

GA für Reihenfolge-Probleme

Untersucht werden sollen Fragestellungen, bei denen die Größe des Erfolges von der Reihenfolge der Arbeitsschritte abhängt.

Beispiele

- Traveling Salesman Problem: Zu gegebenen n Punkten in der Ebene ist eine möglichst kurze Rundreise gesucht.
- Anordnungsprobleme unter Verwendung möglichst vieler Teile: Rucksackproblem
- allgemeinstes Beispiel: Maximierung einer Funktion, die auf allen Permutationen einer Menge {1, 2,..., n} definiert ist.

Probleme beim Kreuzen

Wie sollen die Nachkommen bei Reihenfolge-Problemen erzeugt werden?

Rekombination: einfaches Crossover an Position co:

- wähle $1 \leq co < n$
- kopiere die Allele 1 bis co vom ersten Elternteil und die Allele von co+1 bis n vom zweiten

Elternteil 1: `(1,2,3,4,5,6,7,8)`

Elternteil 2: `(6,5,4,3,1,2,7,8)`

Crossover an co = 3:

Kind: `(1,2,3,3,1,2,7,8)`

$\Rightarrow$ kein gültiges Individuum

Crossover von Permutationen

1. Crossover-Bereich für die Eltern A und B festlegen
2. Für Kind 1 den nicht zu verändernden Bereich von A kopieren und
3. in den Crossover-Bereich die restlichen Elemente von A einfügen, und zwar in der Reihenfolge, wie sie in B vorkommen.
4. Kind 2 mit vertauschten Rollen von A und B erzeugen.

Beispiel

Elternteil A	1	2	3	4	5	6	7	8	Schritt 1
Elternteil B	8	6	4	2	7	5	3	1	
Kind 1	-	2	3	-	5	6	-	-	Schritt 2
Kind 2	8	-	-	2	-	-	3	1	
Kind 1	8	2	3	4	5	6	7	1	Schritt 3
Kind 2	8	4	5	2	6	7	3	1	

Crossover-Bereich

Crossover und Mutation

- Crossover ist nicht symmetrisch zwischen den Eltern.
- Vom ersten Elternteil wird der ausgewählte Bereich (wie immer) unverändert übernommen.
- Vom zweiten Teil wird kein Bereich übernommen, aber immerhin bleibt für die übernommenen Elemente die relative Ordnung erhalten.

Einfachste und natürlichste Mutation:

Wähle eine Teilliste aus und permutiere diese.

Das Traveling Salesman Problem (TSP)

Vorüberlegungen:

Wege können vorwärts und rückwärts durchlaufen werden, d.h. einem Teilweg ist zunächst (auch bei der Bestimmung der Fitness) keine Richtung zugeordnet.

$\Rightarrow$ Nachbarschaft in einer Permutation entscheidend, nicht die Reihenfolge

Anwendung: Mutation

Drehen eines Teilweges:

Wähle einen beliebig langen Teilweg aus und durchlaufe diesen rückwärts.

$\Rightarrow$ Änderung nur an zwei Elementarstücken – an den Enden des Teilweges

TSP Vorbereitung

- Genetische Kodierung: lernrelevante Parameter durch Gene kodieren
  Permutation der n Städte
- Fitness-Funktion problemangepasst festlegen
  Summe der Entfernungen von Stadt zu Stadt auf der Rundreise
- Genetische Operationen festlegen (Vermehrung, Mutation)
  - Crossover für Permutationen
  - Edge-2
- Selektionskriterien festlegen
  - Überleben der μ besten Individuen
  - Tournament set
- Abbruchkriterium festlegen
  - Unterschreiten eines festgelegten Limits (Rundreise „kurz genug“)
  - genau x Generationen
  - x Generationen keine Änderung am fittesten Individuum

TSP Recombination: Edge-2

Erklärung am Beispiel:

Rekombiniert werden sollen die Wege

Elternteil 1: **g d m h b j f i a k e c**

Elternteil 2: **c e k a g b h i j f m d**

Dazu werden die Nachbarschaften in folgender Kantenliste dargestellt:
(Anfang – Enden, doppelt auftretende Enden mit * markiert)

```
a - *k,g,i    e - *k,*c     i - h,j,a,f
b - *h,g,j    f - *j,m,i    j - *f,i,b
c - *e,d,g    g - a,b,c,d   k - *e,*a
d - *m,g,c    h - *b,i,m    m - *d,f,h
```

Edge-2 Algorithmus

1. Der Startknoten wird zufällig gewählt.
2. Der aktuell gewählte Knoten wird aus den rechten Seiten der Kantenliste überall entfernt.
3. Der nächste Knoten wird folgendermaßen gewählt:
   - Falls die Kantenliste zum aktuellen Knoten einen mit * markierten Knoten enthält, wird dieser gewählt. (Bei mehreren Knoten mit dieser Eigenschaft wird zufällig daraus gewählt.)
   - Sonst wird der unmarkierte Knoten mit der kürzesten verbliebenen Kantenliste gewählt. (Bei mehreren Knoten mit dieser Eigenschaft wird zufällig daraus gewählt.)
   - Sonst wird zufällig ein Knoten ausgewählt.
   - Ist kein Knoten mehr wählbar, sind wir fertig.
4. Weiter zu Schritt 2.

Edge-2 im Beispiel

Elternteil 1: **g d m h b j f i a k e c**

Elternteil 2: **c e k a g b h i j f m d**

Kantenliste:

```
a - *k,g,i    d - *m,g,c    g - a,b,c,d    j - *f,i,b
b - *h,g,j    e - *k,*c     h - *b,i,m     k - *e,*a
c - *e,d,g    f - *j,m,i    i - h,j,a,f    m - *d,f,h
```

Zufällige Auswahl: **a**

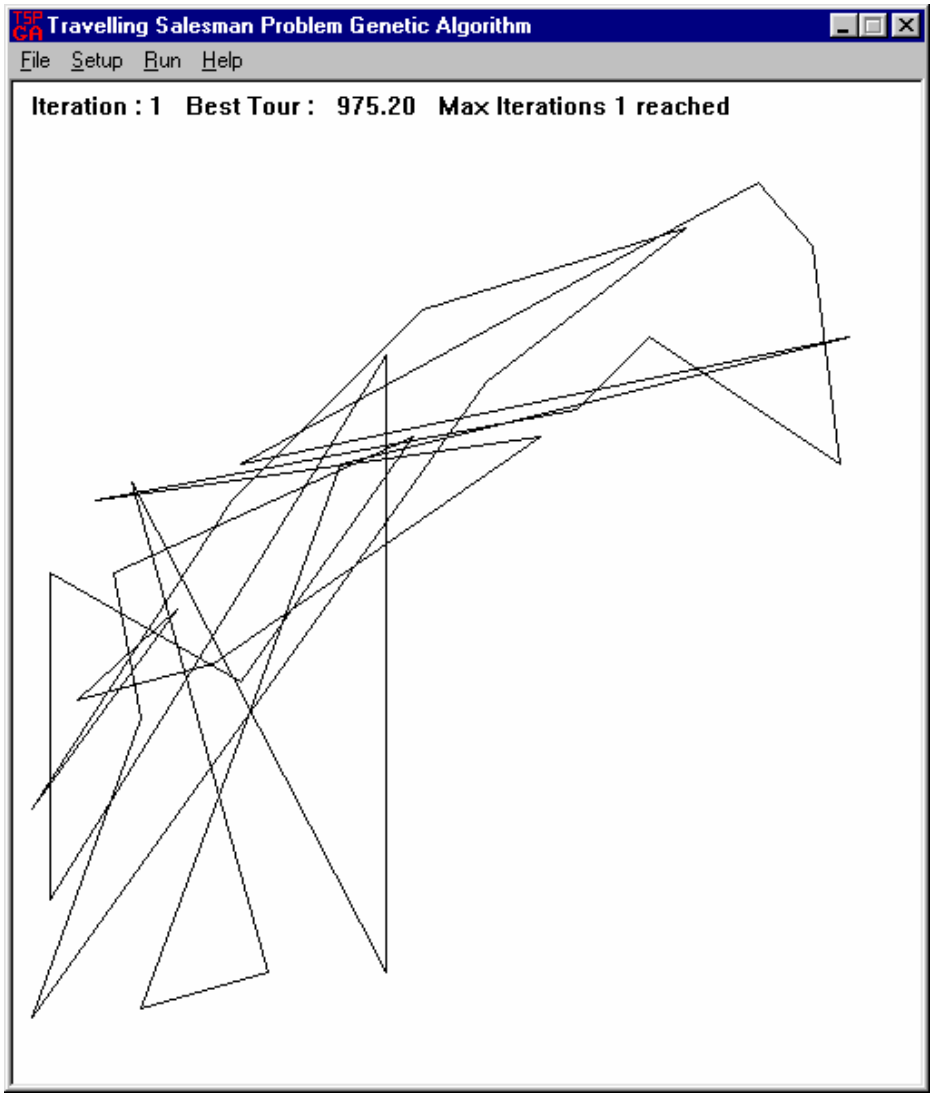
In Schritt 3 wird zunächst **k** gewählt, dann **e**, dann **c**. Danach wird von **d** und **g** (Länge der verbliebenen Kantenliste ist jeweils 2) zufällig **d** gewählt, dann **m** (eindeutig), **h** (zufällig aus **f** und **h**), **b** (eindeutig), **g** (eindeutig).

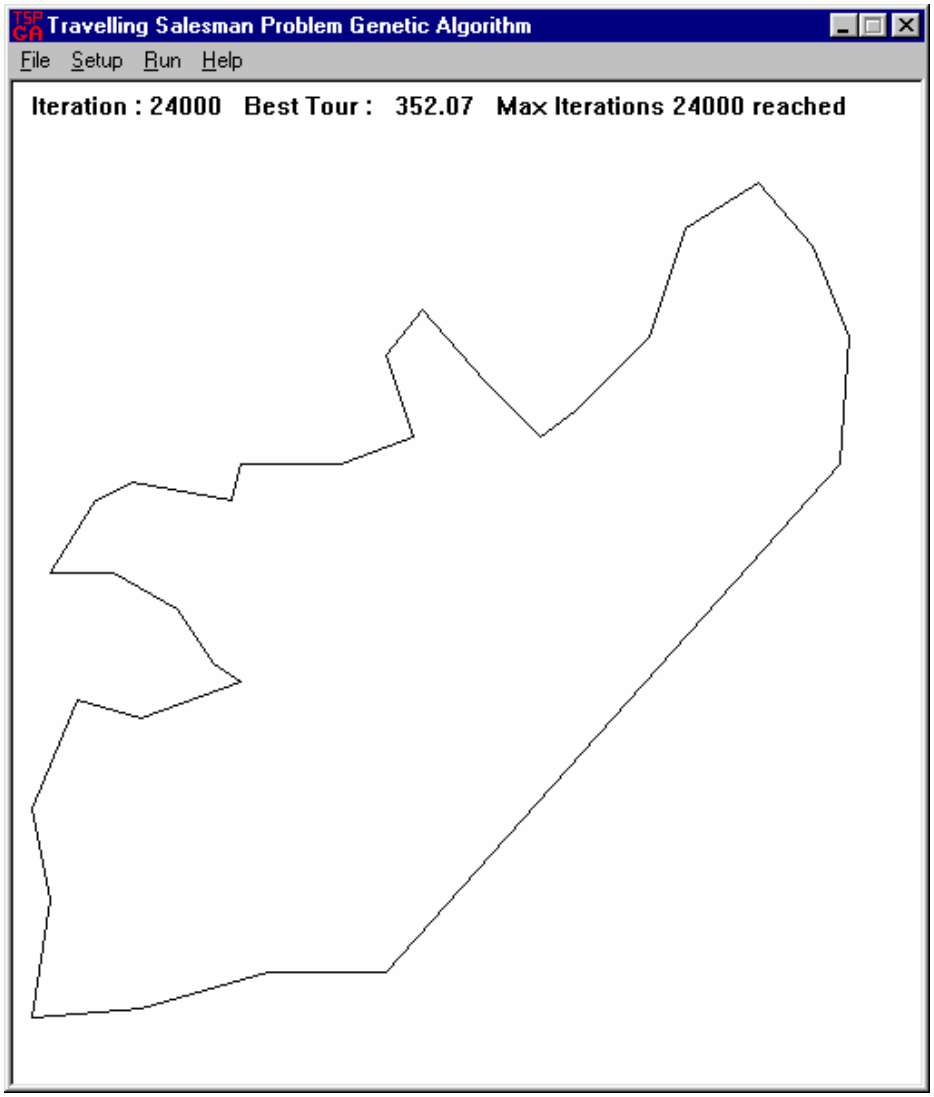
Jetzt wird **i** zufällig ausgewählt, es entsteht erstmalig eine neue Kante.

Danach **f** und **j**.

Ergebnis: **a k e c d m h b g i f j**

TSP Demo


Travelling Salesman Problem Genetic Algorithm
File Setup Run Help
Iteration : 1 Best Tour : 975.20 Max Iterations 1 reached


Travelling Salesman Problem Genetic Algorithm
File Setup Run Help
Iteration : 24000 Best Tour : 352.07 Max Iterations 24000 reached

Spieltheorie

Spieltheorie untersucht

- interaktive Strategien von Individuen,
- die entgegengesetzte oder konfligierende Interessen haben

Typisches Beispiel: Zwei-Personen-Spiele

Es gibt zwei Parteien,

- die sich gegenüberstehen und
- vom Gegner nur die Spielzüge sehen, aber nichts über seine Strategie wissen.

Ein weitere Beschränkung dazu sind Zwei-Personen-Nullsummenspiele, bei denen der eine Gegner gewinnt, was der andere verliert. In diesem Fall ist Kooperation ausgeschlossen, da nur einer von beiden gewinnen kann und somit der andere verlieren muss.

Das Gefangenendilemma (Prisoner's dilemma)

Zwei Gefangene sind verdächtig, gemeinsam eine Straftat begangen zu haben. Die Höchststrafe für das Verbrechen beträgt 5 Jahre. Der Richter macht jedem der beiden folgendes Angebot: Wenn Du auspackst, und somit Deinen Partner belastest, kommst du ohne Strafe davon und er muss die vollen 5 Jahre absitzen. Wenn ihr beide schweigt, haben wir genügend Indizienbeweise, um euch für 2 Jahre einzusperren, wenn ihr beide gesteht, müsst ihr 4 Jahre eures Lebens hier verbringen.

Die beiden Gefangenen haben keine Möglichkeit, sich über ihr Vorgehen abzustimmen. Wie werden sie sich entscheiden?

Jeder der Gefangenen hat also zwei Möglichkeiten: schweigen oder gestehen, oder (aus Sicht des anderen Gefangenen) kooperieren oder defektieren (cooperate/defect).

payoff-Matrix

A / B	schweigen	gestehen
schweigen	(-2,-2)	(-5,0)
gestehen	(0,-5)	(-4,-4)

A / B	cooperate	defect
cooperate	R=(3,3)	S=(0,5)
defect	T=(5,0)	P=(1,1)

Strafen für A bzw. B (links) umgerechnet in Gewinne (rechts) durch Addition von 5.

Die vier Kombinationen werden (aus Sicht von Spieler A) bezeichnet mit R (=reward) für beiderseitige Kooperation, P (=punishment), falls beide defektieren. Die Versuchung T (=temptation), den Gegner reinzulegen steht der Gefahr gegenüber, hereingelegt zu werden (S=sucker).

Alternative Formulierungen und Iteration

In schwierigen Zeiten sollst Du auf dem Schwarzmarkt Waren tauschen. Dazu sollen Du und Dein Tauschpartner zu einem bestimmten Zeitpunkt an verschiedenen Stellen die Ware hinterlegen. Natürlich ist Dein Gewinn am größten, wenn Du nichts hinterlegst und dafür die gewünschte Ware bekommst. (Das gleiche gilt aber leider auch für Deinen Partner). Du hast keine Möglichkeit herauszubekommen, ob Dein Partner die Ware hinterlegen wird oder nicht. Dieselbe Prozedur wiederholt sich Woche für Woche, wobei Du natürlich das bisherige Verhalten Deines Partners mit in Betracht ziehen solltest.

Unterschied: Diesmal wird das Vorgehen iteriert und die Spieler lernen sich kennen. Damit kann A auf die (vermutete) Strategie von B eingehen und umgekehrt.

Beispiel aus der Wirtschaft

Ein einfaches Beispiel für ein Nullsummenspiel, bei dem die Maximin-Strategie anwendbar ist, ist der Wettbewerb zwischen zwei Konkurrenten (hier: Mineralwasseranbieter), die vor der Entscheidung stehen, ihr Produkt für $1 oder $2 anzubieten. Falls der Preis $1 ist, werden auf dem Markt insgesamt 10.000 Flaschen verkauft, für $2 werden insgesamt 5.000 Flaschen verkauft. Die festen Kosten jeder Firma betragen $5.000. Falls beide Firmen ihr Produkt zum gleichen Preis anbieten, verkauft jeder die gleiche Anzahl Flaschen, d.h. der Gewinn wird geteilt und reicht dann für jede Firma aus, ihre Kosten zu decken. Wenn aber einer der beiden Konkurrenten zu einem geringeren Preis verkauft, werden alle Flaschen von ihm abgenommen und der andere Anbieter geht leer aus. In diesem Fall gewinnt die eine Firma $5.000, die andere verliert $5.000.

Nash Equilibrium

DEFINITION: If there is a set of strategies with the property that no player can benefit by changing her strategy while the other players keep their strategies unchanged, then that set of strategies and the corresponding payoffs constitute the Nash Equilibrium.

Im Mineralwasser-Beispiel: Wenn beide billig anbieten, haben wir ein Nash Equilibrium.

Im Gefangenendilemma: Wenn beide gestehen, haben wir ein Nash Equilibrium.

Bemerkung: Das Nash Equilibrium ist wichtig in der Ökonomie, 1994 Nobelpreises für Wirtschaftswissenschaften für John Nash, John C. Harsanyi und Reinhard Selten für ihre Beiträge zur Spieltheorie

Fragen für das iterierte Gefangenendilemma

- Kann sich Kooperation entwickeln?
- Zahlen sich "Gutmütigtkeit" oder "Vertrauen" aus?
- Gibt es eine "beste" Strategie für das iterierte Gefangenendilemma?
- Falls nicht, was sind allgemeine Eigenschaften "guter" Strategien?

Mögliche Strategien lassen sich grob in drei Gruppen einteilen:

Kooperativ: Versucht, den Gegner zum Kooperieren zu bewegen. Ziel: R (= 3 Punkte). Gefahr: Kann ausgenutzt werden, das führt zu S (=0 Punkte)

Aggressiv : Versucht, den Gegner zu übervorteilen. Ziel: T (= 5 Punkte) Gefahr: Gegner kann ebenfalls defektieren, das führt zu P (= 1 Punkt)

Blind: Spielt eine feste oder zufällige Strategie

Konkrete Beispiel-Strategien I

defect: agressiv, blind

defektiere bei jedem Zug

Defect ist eine sehr einfache Strategie, die blind immer defektiert aber damit gleichzeitig das Ziel verfolgt, T zu erreichen. Es ist leicht zu sehen, daß defect nicht geschlagen werden kann. Keine Strategie kann im direkten Vergleich mehr Punkte als defect erhalten, denn es sind für den Gegner nur die Ausgänge S (0 Punkte) oder P (1 Punkt) möglich.

cooperate: kooperativ, blind

kooperiere bei jedem Zug

Diese Strategie ist das Gegenstück zu defect, und hat entgegengesetzte Eigenschaften. Cooperate kann nicht gewinnen, denn es kann entweder R (=3 Punkte) oder S (0 Punkte) erreichen, in beiden Fällen also nicht mehr Punkte als der Gegner. Das ist bei kooperationswilligen Gegnern kein Nachteil, allerdings wird cooperate wegen der blinden Strategie sehr leicht "ausgebeutet".

random: blind

ermittle Zufallszahl $0<z<1$,
kooperiere, wenn $z < 0.5$,
defektiere sonst.

Random spielt mit gleicher Wahrscheinlichkeit defect oder cooperate. Keine Strategie, die eine Kooperation etablieren möchte, kann damit bei random Erfolg haben, wird also, wenn sie nicht blind spielt, nicht leichtfertig kooperieren. Damit ist es für random nicht möglich, auf Dauer R zu erreichen.

Konkrete Beispiel-Strategien II

per_kind: blind, agressiv

spiele periodisch [kooperieren, kooperieren, defektieren]
Diese Strategie versucht, den Gegner "in Sicherheit" zu wiegen und dann zuzuschlagen, in der Hoffnung, dass er gelegentliches defektieren hinnimmt und sich dann wieder auf Kooperation einlässt.

tit-for-tat: kooperativ

kooperiere im ersten Zug,
in jedem weiteren Zug spiele den Zug, den der Gegner bei letzten Mal benutzt hat.
Diese Strategie ist kooperationswillig, wehrt sich aber auch gegen Ausbeutungsversuche. Gleichzeitig ist sie nicht nachtragend, sondern beantwortet erneute Kooperationsbereitschaft mit Kooperation. Tit-for-tat kann nicht gewinnen, da es niemals unmotiviert defektiert, also nie versucht T (= 5 Punkte) zu erhalten. Andererseits kann es aber auch nicht mit mehr als 5 Punkten Abstand verlieren, weil es sich nur ein einziges Mal ausbeuten läßt.

pavlov: kooperativ

kooperiere im ersten Zug,
dann nur, wenn beide Spieler denselben Zug gemacht haben
Pavlov verfolgt einen ähnlichen Grundgedanken wie tit-for-tat, stellt jedoch strengere Anforderungen an die eigene Kooperationswilligkeit: Nur nach einer erfolgreichen Kooperation wird weiter kooperiert, d.h. die Strategie reagiert auf Ausbeutungsversuche mit Nichtkooperation und macht dann von sich aus keinen Versuch, eine Kooperation wieder zu etablieren.

Gute Strategien

Gute Strategien sollen folgende Eigenschaften haben:

- *freundlich* sein, d.h. nie als erster defektieren,
- *zurückschlagen* , d.h. eine Defektion nicht unbeantwortet lassen,
- *nachgiebig* sein, d.h. nach dem Zurückschlagen Kooperation wieder zulassen,
- *einfach* sein, d.h. das eigene Verhalten für den Gegner durchschaubar machen.

Die erfolgreichste Strategie bisher ist tit-for-tat (Wie du mir, so ich dir). Die einzige bekannte bessere Strategie ist die folgende, die nicht *einfach* ist und mehr Wissen über die Vergangenheit verlangt:

gradual:

- beim ersten Zug wird kooperiert,
- das erste Defektieren des Gegners wird mit einem Defektieren und anschließend zwei Kooperationen beantwortet,
- schließlich wird das n-te Defektieren des Gegners mit n Defektionen und zwei Kooperationen beantwortet

Gentik for Prisoners

Allgemeines Vorgehen: Gedächtnis von drei Generationen, d.h. die Entscheidung über den nächsten Zug fällt in Abhängigkeit der letzten drei Züge.

Für jeden vollständigen Zug gibt es vier Möglichkeiten, d.h. wir benötigen $4^3=64$ bit für eine Folge von 64 Cs und Ds als Antworten auf die 64 möglichen Vergangenheiten.

Zum Start benötigen wir noch die Angabe dreier hypothetischer Schritte, damit die Auswahl beginnen kann. Insgesamt 70 bit.

Genetische Operationen

Wir verwenden das übliche Vorgehen:

- Zufällige Initialisierung der Individuen
- Fitness wird ermittelt, indem die Individuen in einem Turnier gegeneinander spielen und nach Erfolg bewertet werden.
- Selektion: Einteilung der Individuen in drei Drittel entsprechend ihrer Fitness: Top-Individuen dürfen 2x heiraten, durchschnittliche Individuen 1x und Versager gar nicht.
- Vermehrung: Crossover und Mutation wie üblich. Die Elterngeneration stirbt danach völlig aus.

Resultate der Evolution

Globale Beobachtung: Die besten Individuen sind vergleichbar gut wie die besten „von Hand“ erzeugten heuristischen Strategien.

Erfolgreiche Individuen enthalten beispielsweise die folgenden Regeln:

- Nicht provozieren: Kooperation soll fortgesetzt werden.
  (CC)(CC)(CC) -> C
- Sich provozieren lassen: Falls man aus heiterem Himmel angegriffen wird:
  (CC)(CC)(CD) -> D
- Entschuldigungen annehmen: Nach Kooperationsangebot kooperieren
  (CD)(DC)(CC) -> C
- Vergessen: Nach zweimaliger Kooperation Vorgeschichte vergessen
  (DC)(CC)(CC) -> C
- Zweikampf annehmen: (DD)(DD)(DD) -> D