INF FAKULTÄT FÜR INFORMATIK

Evolutionäre Algorithmen

Kapitel 9: Parallele Optimierung und Evolutionary Game Theory

Sanaz Mostaghim

Intelligente Systeme

Outline

1. **Parallelisierung evolutionärer Algorithmen**
   - Was kann man parallelisieren?
   - Inselmodell und Migration
   - Zellulare evolutionäre Algorithmen
   - Mühlenbeins Ansatz

2. **Evolutionäre Algorithmen zur Verhaltenssimulation**

Parallelisierung evolutionärer Algorithmen

EA: recht **teures Optimierungsverfahren**, da oft mit

- großer Population (tausende bis zehntausend Individuen)
- großer Zahl an Generationen (einige hundert)

gearbeitet werden muss für hinreichende Lösungsgüte

Vorteil: etwas höhere Lösungsgüte i.V.z. anderen Verfahren

Nachteil: unangenehm lange Laufzeit

Möglicher Lösungsansatz: **Parallelisierung**

Verteilung der notwendigen Operationen auf mehrere Prozessoren

Fragen:

- **Welche Schritte kann man parallelisieren?**
- **Welche vorteilhaften Techniken kann man bei der Parallelisierung zusätzlich anwenden?**

Was kann man parallelisieren?

Erzeugen der Anfangspopulation:

- meist problemlos, da i.A. Chromosomen der Anfangspopulation zufällig und unabhängig voneinander
- Versuch, Duplikate zu vermeiden, kann Parallelisierung behindern

  Parallelisierung dieses Schrittes eher wenig bedeutsam (Anfangspopulation wird nur einmal erzeugt)

Bewertung der Chromosomen:

- problemlos, da Chromosomen i.A. unabhängig voneinander bewertet (Fitness hängt nur vom Chromosom selbst ab)
- auch Gefangenendilemma: bearbeite Paarungen parallel

Berechnung der Ränge oder (relativen) Fitnesswerte:

- Bewertungen müssten hierfür zusammengeführt werden

Was kann man parallelisieren?

Selektion: ob Auswahlschritt parallelisierbar ist, hängt sehr stark vom verwendeten Selektionsverfahren ab

- **Erwartungswertmodell** und **Elitismus:**
  erfordern beide globale Betrachtung der Population und sind daher nur schwer parallelisierbar
- **Glücksrad**- und **Rangauswahl:**
  nachdem relativen Fitnesswerte bzw. Ränge bestimmt (was schwer parallelisierbar ist), ist Auswahl leicht parallelisierbar
- **Turnierauswahl:**
  ideal für Parallelisierung, besonders bei kleinen Turniergrößen, da keine globale Information notwendig (Vergleich der Fitnesswerte beschränkt sich auf Individuen des Turniers)

Was kann man parallelisieren?

Anwendung genetischer Operatoren
leicht zu parallelisieren, da jeweils nur ein (Mutation) oder zwei Chromosomen (Crossover) betroffen (zusammen mit Turnierauswahl: Steady-State-EA sehr gut parallelisierbar)

Abbruchbedingung:
einfacher Test, ob bestimmte Generationenzahl erreicht, bereitet bei Parallelisierung keine Probleme
Abbruchkriterien wie

- bestes Individuum der Population hat bestimmte Mindestgüte oder
- über bestimmte Anzahl von Generationen hat sich bestes Individuum nicht/kaum verbessert

dagegen für Parallelisierung weniger geeignet, da globale Betrachtung der Population erforderlich

Inselmodell

wenn z.B. schwer zu parallelisierendes Selektionsverfahren verwendet wird:

Parallelisierung erreichbar durch parallelle Berechnung mehrerer unabhängiger Populationen

jede Population bewohnt eine Insel, daher **Inselmodell**.

reines Inselmodell $\equiv$ mehrfache serielle Ausführung des gleichen EAs

liefert meist etwas schlechtere Ergebnisse als einzelner Lauf mit entsprechend größerer Population

Migration

zwischen Inselpopulationen können zu festgelegten Zeitpunkten (*nicht* in jeder Generation) Individuen ausgetauscht werden

Migration (Wanderung)

normalerweise keine direkte Rekombination von Chromosomen verschiedener Inseln

erst nach Migration: Rekombination genetischer Information einer Insel mit einer anderen Insel

Steuerung der Migration zwischen Inseln

Zufallsmodell:

zufälliges Bestimmen beider Inseln, zwischen denen Individuen ausgetauscht werden

beliebige Inseln können Individuen austauschen

Netzwerkmodell:

Inseln werden in Graphen angeordnet

Individuen wandern zwischen Inseln nur entlang der Kanten

Kanten werden zufällig bestimmt

Wettbewerb zwischen den Inseln

EAs, die auf Inseln angewandt werden, unterscheiden sich (in Verfahren und/oder Parametern)

Populationsgröße einer Insel wird entsprechend ihrer durchschnittlichen Fitness der Individuen erhöht oder erniedrigt

jedoch: Mindestpopulationsgröße, die nicht unterschritten wird

Zellulare evolutionäre Algorithmen

auch: "isolation by distance"

Prozessoren werden in (rechtwinkligen) Gitter angeordnet

Gitter bedeckt gewöhnlich Oberfläche eines Torus

Selektion und Crossover auf im Gitter benachbarte (durch Kanten verbundene), Mutation auf einzelne Prozessoren beschränkt

Beispiel: jeder Prozessor verwaltet ein Chromosom

- **Selektion:** Prozessor wählt bestes Chromosom seiner (vier) Nachbarprozessoren oder eines dieser Chromosomen zufällig nach ihrer Fitness
- **Crossover:** Prozessor führt Crossover mit gewähltem und eigenem Chromosom durch oder mutiert sein Chromosom (er behält besseren der beiden Nachkommen bzw. von Elter und Kind)

Bildung von Gruppen benachbarter Prozessoren, die ähnliche Chromosomen verwalten

mildert die oft zerstörende Wirkung des Crossover

Mühlenbeins Ansatz

Kombination von EAs **mit Zufallsaufstieg**

- jedes Individuum führt lokalen Zufallsaufstieg durch, d.h.
  - bei vorteilhafter Mutation wird Elter ersetzt
  - bei nachteiliger Mutation bleibt Elter erhalten
- Zufallsaufstieg leicht parallelisierbar
- Individuen suchen sich Crossover-Partner in ihrer Nachbarschaft (dazu: Definition des Abstandes zweier Individuen benötigt — vergleiche Nischentechniken, z.B. *power law sharing*)
- Nachkommen führen lokalen Zufallsaufstieg durch
- Individuen der nächsten Generation: nach **„lokalem“ Eliteprinzip** ausgewählt

$\Rightarrow$ übernehme beiden besten Individuen unter Eltern und optimierten Nachkommen

Outline

1. Parallelisierung evolutionärer Algorithmen

2. **Evolutionäre Algorithmen zur Verhaltenssimulation**
   - Das Gefangenendilemma
   - Genetischer Algorithmus
   - Erweiterungen

Evolutionäre Algorithmen zur Verhaltenssimulation

bisher: Verwendung von EAs um (numerische oder diskrete) Optimierungsprobleme zu lösen

jetzt: Verwendung von EAs um Verhalten zu simulieren (Populationsdynamik) und Verhaltensstrategien zu finden

Grundlage: Spieltheorie

- dient der Analyse sozialer und wirtschaftlicher Situationen
- Modellierung von Handlungen als Spielzüge in festgelegtem Rahmen
- wichtigste theoretische Grundlage der Wirtschaftswissenschaften

Allgemeiner Ansatz:

- kodiere Verhaltensstrategie eines Akteurs in Chromosom
- lasse Akteure miteinander interagieren und bewerte ihren Erfolg
- Akteure vermehren sich oder sterben aus, je nach erzieltem Erfolg

Das Gefangenendilemma

bekanntestes Problem der Spieltheorie ist das **Gefangenendilemma** (engl.: prisoner's dilemma)

- 2 Personen haben Banküberfall begangen und werden verhaftet
- Beweise reichen nicht aus, um sie in Indizienprozess wegen Banküberfall zu verurteilen
- Beweise reichen jedoch aus, um sie wegen eines geringfügigeren Deliktes (z.B. unerlaubter Waffenbesitz) zu verurteilen (Strafmaß: 1 Jahr Gefängnis)

  Angebot des Staatsanwaltes: Kronzeugenregelung
  - gesteht einer der beiden, wird er Kronzeuge und nicht verurteilt
  - der andere wird mit voller Härte bestraft (10 Jahre Gefängnis)
  - Problem: gestehen beide, gilt Kronzeugenregelung nicht
    da sie beide geständig sind, erhalten sie mildernde Umstände(Strafe: je 5 Jahre Gefängnis)

Das Gefangenendilemma

Analyse des Gefangenendilemmas durch **Auszahlungsmatrix**:

A \ B	schweigt	gesteht
schweigt	**−1** **−1**	**0** **−10**
gesteht	**−10** **0**	**−5** **−5**

Kooperation (beide schweigen) ist insgesamt am günstigsten

Aber: doppeltes Geständnis ist **Nash-Gleichgewicht**:
keine der beiden Seiten kann ihre Auszahlung erhöhen, wenn nur sie ihre Aktion ändert (jede Auszahlungsmatrix hat mind. ein Nash-Gleichgewicht [Nash, 1950])

Allg. Auszahlungsmatrix des Gefangenendilemmas

A \ B	cooperate	defect
cooperate	R, R	T, S
defect	S, T	P, P

R: Reward for mutual cooperation **P**: Punishment for mutual defectio
T: Temptation to defect **S**: Sucker's payoff

- genaue Werte für **R**, **P**, **T** und **S** sind nicht wichtig
- es muss aber gelten $\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}$ und $2\,\mathbf{R} > \mathbf{T} + \mathbf{S}$
  2. Bedingung nicht erfüllt $\Rightarrow$ wechselweises Ausbeuten besser

Das Gefangenendilemma

Viele Alltagssituation: beschreibbar mit Gefangenendilemma

aber: obwohl doppelter Defekt = Nash-Gleichgewicht, auch anderes (kooperatives) Verhalten

Fragestellung (nach [Axelrod, 1980]):
Unter welchen Bedingungen entsteht Kooperation in einer Welt von Egoisten ohne zentrale Autorität?

Antwort von [Hobbes, 1651] (Leviathan):

- **Gar nicht!** Ehe staatliche Ordnung existierte, wurde Naturzustand dominiert von egoistischen Individuen, die so rücksichtslos gegeneinander wetteiferten, dass das Leben „solitary, poor, nasty, brutish, and short“ war.
- **aber:** Auf internationaler Ebene gibt es *de facto* keine zentrale Autorität, aber dennoch (wirtschaftliche und politische) Kooperation von Staaten.

Das Gefangenendilemma

Ansatz von [Axelrod, 1980]: **iteriertes Gefangenendilemma**. (Gefangenendilemma von 2 Spielern mehrfach hintereinander gespielt, wobei sie vergangenen Züge des jeweils anderen Spielers kennen)

Idee dieses Ansatzes:

- wird Gefangenendilemma nur *einmal* gespielt, ist es am günstigsten, Nash-Gleichgewicht zu wählen
- wird es *mehrfach* gespielt, kann ein Spieler auf unkooperatives Verhalten des anderen reagieren
  (Möglichkeit der *Vergeltung* für erlittene Nachteile)

Fragestellungen:

1. **Entsteht im iterierten Gefangenendilemma Kooperation?**
2. **Was ist die beste Strategie im iterierten Gefangendilemma?**

Das Gefangenendilemma

[Axelrod, 1980] legte folgende **Auszahlungsmatrix** fest:

A \ B	cooperate	defect
cooperate	3, 3	5, 0
defect	0, 5	1, 1

(kleinsten ganzen Zahlen ≥ 0, die Bedingungen erfüllen)

- Wissenschaftler verschiedener Disziplinen (Psychologie, Sozial- und Politikwissenschaften, Wirtschaftswissenschaften, Mathematik) wurden eingeladen, Programme zu schreiben, die das iterierte Gefangenendilemma spielen
- Programm kann sich eigene und gegnerische Züge merken

Turniere

Zur Beantwortung beider Fragen führte Axelrod 2 Turniere durch:

1. **Turnier**:
   - 14 Programme plus ein Zufallsspieler (Fortran)
   - Rundenturnier mit 200 Spielen je Paarung
   - Sieger: A. Rapoport mit Tit-for-Tat (Wie du mir, so ich dir)

   Programme und Ergebnisse des ersten Turniers wurden veröffentlicht

   zu einem zweiten Turnier eingeladen

   Idee: Ergebnisanalyse für ggf. bessere Programme

2. **Turnier**:
   - 62 Programme plus ein Zufallsspieler (Fortran und Basic)
   - Rundenturnier mit 200 Spielen je Paarung
   - Sieger: A. Rapoport mit Tit-for-Tat (Wie du mir, so ich dir)

Tit-for-Tat

Spielstrategie von **Tit-for-Tat** ist *sehr* einfach:

- kooperiere im ersten Spiel (spiele C)
- mache in allen folgenden Spielen den Zug des Gegners aus dem direkt vorangehenden Spiel

beachte: reines Tit-for-Tat ist nicht unbedingt beste Strategie, wenn gegen *einzelne* andere Strategien gespielt wird

- nur wenn es in Population Individuen gibt, mit denen Tit-for-Tat kooperieren kann, schneidet es insgesamt sehr gut ab
- **Problem** von Tit-for-Tat: Es ist **anfällig für Fehler** — spielt Tit-for-Tat gegen Tit-for-Tat und spielt einer der beiden Spieler „aus Versehen“ Defekt, so kommt es zu wechselseitigen Vergeltungsschlägen

Eine wichtige Alternative ist **Tit-for-Two-Tat**:
schlage erst nach zweimaligem Defekt des Gegners zurück

Genetischer Algorithmus

Kodierung der Spielstrategien: [Axelrod, 1987]

betrachte alle Spielverläufe der Länge 3 ($2^6 = 64$ Möglichkeiten)

speichere für jeden Spielverlauf den im nächsten Spiel auszuführenden Zug (C – cooperate, D – defect, in 1 Bit):

		1. Spiel	2. Spiel	3. Spiel	
1. Bit:	Antwort auf	(C,C),	(C,C),	(C,C):	C
2. Bit:	Antwort auf	(C,C),	(C,C),	(C,D):	D
3. Bit:	Antwort auf	(C,C),	(C,C),	(D,C):	C
⋮		⋮		⋮	⋮
64. Bit:	Antwort auf	(D,D),	(D,D),	(D,D):	D

(1. und 2. Element jedes Paares: eigener bzw. gegnerischer Zug)

Zusätzlich: 6 Bit zur Kodierung des Spielverlaufs vorm 1. Zug

jedes Chromosom hat 70 binäre Gene (jeweils C oder D)

Genetischer Algorithmus: Ablauf

initialisiere Anfangspopulation mit zufälligen Bitfolgen (70 Bit)

aus aktueller Population: wähle zufällig Paare von Individuen

sie spielen 200-mal Gefangendilemma gegeneinander

für die ersten 3 Spiele: nutze (ein Teil des) im Chromosom abgespeicherten Anfangsspielverlaufs, um Zug zu bestimmen (fehlende/zu kurze Historie wird ersetzt/aufgefüllt)

jedes Individuum spielt gegen gleiche Anzahl von Gegnern (aus Rechenzeitgründen – 1987! – kein volles Rundenturnier)

Auswahl von Individuen für nächste Generation:

überdurchschnittliches Ergebnis	($x \geq \mu + \sigma$):	2 Kinde
durchschnittliches Ergebnis	($\mu - \sigma < x < \mu + \sigma$):	1 Kinde
unterdurchschnittliches Ergebnis	($\mu - \sigma \geq x$):	0 Kinde

genetische Operatoren: Binärmutation, 1-Punkt-Crossover

Genetischer Algorithmus: Ergebnis

sich ergebende Strategien sind **Tit-for-Tat** sehr ähnlich
[Axelrod, 1987] identifizierte folgende allgemeine Muster:
Don't rock the boat: kooperiere nach drei Kooperationen
(C,C), (C,C), (C,C) $\rightarrow$ C
Be provokable: spiele Defekt nach plötzlichem Defekt des Gegners
(C,C), (C,C), (C,D) $\rightarrow$ D
Accept an apology: kooperiere nach wechselseitiger Ausbeutung
(C,C), (C,D), (D,C) $\rightarrow$ C
Forget: (sei nicht nachtragend:) kooperiere nachdem Kooperation nach Ausbeutung wiederhergestellt (auch ohne Vergeltung)
(C,C), (C,D), (C,C) $\rightarrow$ C
Accept a rut: (rut *fig.* ausgefahrenes Gleis, alter Trott)
spiele Defekt nach dreimaligem Defekt des Gegners
(D,D), (D,D), (D,D) $\rightarrow$ D

Das Gefangenendilemma: Erweiterungen

- in Praxis: Auswirkungen von Handlungen nicht immer perfekt beobachtbar
- nicht genauer Zug des Gegners, sondern nur W'keiten sind bekannt
- oft ≥ 2 Akteure beteiligt: Mehr-Personen-Gefangenendilemma

- Berücksichtigung längerer Spielverläufe (mehr als drei Spiele)
- Hinzunahme einer Zufallskomponente für die Wahl des Spielzugs: W'keiten für Wahl von C und D statt eines festen Zuges
- Beschreibung der Spielstrategie durch Moore-Automaten oder allgemeine Programme, die dann in GA verändert werden

Literatur zur Lehrveranstaltung I


Axelrod, R. (1980).
More effective choice in the prisoner's dilemma.
Journal of Conflict Resolution, 24:379–403.


Axelrod, R. (1987).
The evolution of strategies in the iterated prisoner's dilemma.
In Davis, L., editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Morgan Kaufman, Los Altos, CA, USA.


Hobbes, T. (1651).
Leviathan.


Nash, J. F. (1950).
Non-cooperative games.
PhD thesis, Princeton University.