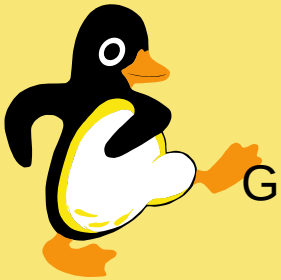


PGAPy: Genetische Algorithmen mit Python

Dr. Ralf Schlatterbeck
Open Source Consulting

Email: office@runtux.com
Web: http://www.runtux.com
Tel. +43/650/621 40 17

Inhalt

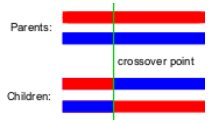
Einführung in Genetische Algorithmen

- Ort (Locus) des Chromosoms: Gen
  verschiedene Werte dafür: Allele
- Genotyp vs. Phänotyp:
  Chromosom codiert ein Individuum
- Survival of the fittest: Bewertung
- Genetische Operatoren: Gene neu gemischt
- Generationen: Neue Individuen ersetzen alte,
  verschiedene „Replacement Strategien“ möglich
- Populationsgröße

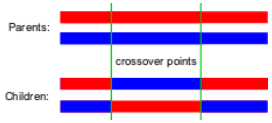
Gen-Typen

- Genetik: Aminosäuren (3 Möglichkeiten)
- Genetische Algorithmen: Traditionell bits
- Integer für Integer-Probleme (Scheduling, . . .)
- Real für diverse kontinuierliche Probleme
  (Modellierung von Oberflächen)
- Characters für linguistische Probleme
- Selbstdefinierte Typen

Genetische Operatoren: Crossover



Single-Point Crossover



Two-Point Crossover

- Crossover erzeugt aus zwei Genen ein neues
- Optional: Beide Varianten „leben“ weiter
- Uniform Crossover: Allele werden mit einer bestimmten Wahrscheinlichkeit zwischen den Genen ausgetauscht

Bilder Quelle: Wikipedia

Genetische Operatoren: Mutation

- Analog zur biologischen Mutation: Allele werden zufällig verändert
- Erhält die Diversität der Population
- Diversität verhindert lokale Minima/Maxima
- Für Integer, Real, oder Character-Gene: Wertebereichangabe
- Spezielle Mutation für spezielle Probleme, z. B. Index-Vertauschung für Travelling Salesman

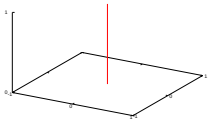
Bewertungsfunktion: Genotyp wird Phänotyp

- Auch „Fitness-Funktion“
- Baut aus dem Chromosom (Genotyp) das Individuum (Phänotyp) als Datenstruktur
- Bewertet die Fitness des Individuums
- Genetischer Algorithmus kann minimieren oder maximieren
- Falls genetische Darstellung ungültige Individuen erzeugen kann, muss auch dafür eine Bewertung gegeben werden

Bewertungsfunktion: Genotyp wird Phänotyp

- Faustregel (bei Minimieren):
  - Ungültige Lösung: Multiplikation,
  - Schlechte vs. gute Lösung: Addition
- „Stetige“ Bewertungsfunktion ist hilfreich

→ beware of „needle in a haystack“ fitness functions!



Termination des Algorithmus

- Optimierungsziel erreicht
  (Wert der Bewertungsfunktions)
- Max. Anzahl der Generationen
- Population nicht mehr divers
- Mehrere Generationen keine Verbesserung
- Anwendungsspezifische Kriterien
- Kombination aus obigen Varianten

PGAPy Initialisierung

```
from pga import *
class Rucksack (PGA) :
    def __init__ (self, bucket_size, **kw) :
        self.kw          = kw
        self.keys        = sorted (kw.keys ())
        self.bucket_size = bucket_size
        length           = len (kw)
        PGA.__init__ \
            ( self, int, length
            , init          = [(0, length - 1)] * length
            , maximize      = False
            , print_options = \
                [ PGA_REPORT_STRING
                , PGA_REPORT_WORST
                ]
            )
```


Bewertungsfunktion: Rucksack-Problem

```
def evaluate (self, p, pop) :
    bad         = 1
    n           = 0
    blen        = [0] * len (self)
    self.bucket = [[] for k in range (len (self))]
    for i in range (len (self)) :
        idx = self.get_allele (p, pop, i)
        key = self.keys [i]
        blen        [idx] += self.kw [key]
        self.bucket [idx].append (key)
    for b in blen :
        if b > self.bucket_size :
            bad *= b * b
        if b :
            n += 1
    return n * bad
```


PGAPy: Overriding methods

```
def print_string (self, file, p, pop) :
    self.evaluate (p, pop)
    for i in range (len (self)) :
        names = self.bucket [i]
        if names :
            print >> file, "%3d: %s" \
                % (i, ', '.join (names))
    PGA.print_string (self, file, p, pop)
...
pg = Rucksack \
  (300, file1=27,file2=250,file3=250,file4=50,file5=13)
pg.run ()
The Best Evaluation: 2.000000e+00.
The Best String:
  0: file1, file3, file5
  2: file2, file4
#    0: [    0], [    2], [    0], [    2], [    0]
```

Sudoku

```
def evaluate (self, p, pop) :
    puzzle = Puzzle (verbose = False, solvemax = 50)
    count  = 0
    for x in range (9) :
        for y in range (9) :
            val    = self.get_allele (p, pop, 9 * x + y)
            count += val != 0
            puzzle.set (x, y, val)
    puzzle.solve ()
    solvecount = puzzle.solvecount
    if not solvecount  : return 1000 * count * count
    if solvecount == 1 : return count
    return 1000 - count + solvecount
```


PGAPy: Python Wrapper für pgapack

- Parallel Genetic Algorithm Library (pgapack) by D. Levine, Mathematics and Computer Science Division, Argonne National Laboratory
- Lots of bells and whistles for experimentation
- Schnelle Prototypen
- . . . bis zu ganzen Anwendungen
- Download, siehe mein Projekt auf SourceForge: pgapy.sourceforge.net

Have Fun!

3						5		
						2	6	
			3		8			
							9	1
4			1					
	3	7						
8					6			
		6	4	8	5	7		
					9			2