EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

Evolutionäre Algorithmen

Übungsblatt 1, SS2011

Abgabe: 26.04.11

Aufgabe 1 (Biologische Evolution - 5 Punkte)

Erklären Sie die Begriffe Population, Individuum, Selektion, Mutation, Rekombination, Reproduktion und Fitness kurz in ihrer biologischen Bedeutung. Wie könnten die Prinzipien der biologischen Evolution in einem Optimierungsalgorithmus umgesetzt werden? Führen Sie dies an einem selbstgewählten konkreten Optimierungsproblem aus!

Aufgabe 2 (Anwendungsbeispiele - 6 Punkte)

Bitte geben Sie für die folgenden Methoden jeweils eine geeignete und praxisrelevante Problemstellung an und begründen Sie Ihre Entscheidung.

- Genetische Algorithmen
- Classifier systems
- Genetic programming
- Evolutionstrategien
- Evolutionäres programmieren
- Simulated annealing

Aufgabe 3 (Programmieraufgabe: Binäre Codierung - 9 Punkte)

Programmieren Sie eine Java-Klasse `BinaryChromosome` für die Codierung von Vektoren als binäre Strings! Die Klasse soll in der Lage sein, `nComponents` Binärzahlen mit je `nBits` Bits darzustellen. Dabei soll es (intern) möglich sein, gezielt auf einzelne Bits zuzugreifen.

Überlegen Sie zuerst, wie man in Java Binärstrings reperäsentieren könnte. Nennen Sie mindestens zwei Möglichkeiten (Sie dürfen auch Kommentarzeilen im Quelltext verwenden), und entscheiden Sie sich für eine (mit Begründung). Implementieren Sie anschließend die Klasse, die folgendes Gerüst haben soll

```
public class BinaryChromosome {
  public BinaryChromosome(int nComponents, int nBits, Random rand)
  public BinaryChromosome(BinaryChromosome b)
  public int[] decode()
  public int[] decodeGray()
  public String toString()
}
```

Der erste Konstruktor initialisiert das Objekt unter Zuhilfenahme von `rand` zufällig, der zweite Konstruktor ist ein Copykonstruktor. `decode()` liefert `nComponents` Zahlen zwischen 0 und $2^{nBits} - 1$ zurück, ebenso `decodeGray()`, wobei hier der Gray-Code verwendet wird. `toString()` wandelt den Bitstring in eine Zeichenkette um.

Hinweis: Die Umwandlung von Gray-Code zu Standard-Binärcode funktioniert folgendermaßen: Sei $g = g_1 g_2 \dots g_n$ ein Gray-codierter Bitstring, wobei g_1 das höchstwertige Bit ist. Dann ist durch

$$b_k := \sum_{j=1}^{k} g_j \pmod 2 \text{ für alle } 1 \leq k \leq n$$

eine Binärzahl $b = b_1 \dots b_n$ definiert. Hierbei bedeutet $\pmod 2$ den Rest bei der Division durch 2.

Allgemeiner Hinweis: Abgabe per E-mail (`andreas.jahn@uni-tuebingen.de`) ist erwünscht, besonders für die Programmieraufgaben! Bitte die Klassen vorher testen, da Syntaxfehler zu Punktabzug führen. Die Klassen sollten ohne irgendwelche Extras mit dem JDK 1.6 laufen. Bitte grundsätzlich keine grafischen Oberflächen programmieren (kostet immer mehr Zeit als man plant!), wer es trotzdem nicht lassen kann, sollte nur das AWT und Swing verwenden.