EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

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

Evolutionäre Algorithmen

Übungsblatt 10, SS 2011

Abgabe: 05.07.11

Aufgabe 25 Implementierung einer Evolutionsstrategie - (14 Punkte)

In dieser Aufgabe soll die einfache (1+1)-Implementierung Ihrer Evolutionsstrategie aus Aufgabe 23 zu einer (μ, λ)-ES erweitert werden. Laden Sie sich dazu das neue Framework von der Übungsseite[1] herunter. Darin finden Sie das bereits bekannte Verzeichnis `resources`, das dieselben Testreihen enthält, die auch in Aufgabe 23 gegeben waren. Das `lib`-Verzeichnis stellt wesentliche Hilfsfunktionen bereit, die nicht bearbeitet werden müssen. Im Verzeichnis `src` finden Sie die bereits bekannte Klasse `Application`, die das Programm startet. Die für Sie wesentliche Klasse `ES` befindet sich im Paket `optimizers` und gibt Ihnen bereits das Grundgerüst des auf Seite 192 des Vorlesungsmanuskriptes beschriebenen Algorithmus vor. Dabei greift die `ES`-Klasse auf Implementierungen der $Interfaces$ aus den beiden Paketen `pop` (für die zu optimierende Population und die darin enthaltenen Individuen) sowie `operators` zu. In `operators` finden Sie die Schnittstellen `Selection`, `Crossover` und `Mutation`. Weitere Erläuterungen finden Sie in den JavaDoc-Kommentaren im Quellcode.

(a) Schreiben Sie eine Klasse, die das $Interface$ `Individual` implementiert und legen Sie diese im Paket `pop` ab. (2 Punkte)

(b) Implementieren Sie die Schnittstelle `Crossover` und legen Sie Ihre erstellte Klasse im Paket `operators` ab. Der Crossover-Operator soll dabei ein intermediäres Crossover gemäß Seite 205 im Manuskript durchführen. (2 Punkte)

(c) Erstellen Sie im Paket `operators` eine Klasse, die das $Interface$ `Mutation` implementiert. Der Mutationsmechanismus entspreche der Variante 1 auf Seite 197 des Vorlesunsmanuskriptes. Verwenden Sie für τ und τ' sinnvolle Werte. (4 Punkte)

(d) Legen Sie weiterhin eine Implementierung der `Selection`-Schnittstelle im selben Paket ab, die jeweils die μ besten Individuen aus einer Population der Größe λ auswählt und in einer (neuen) Population zurückliefert. (2 Punkte)

(e) Implementieren Sie in der Klasse `ES` die mit dem Schlagwort `TODO` gekennzeichneten Methoden. Führen Sie dazu, falls nötig, auch Feld-Variablen ein. Beachten Sie, dass die `ES`-Klasse sowohl eine (μ, λ)- als auch eine $(\mu + \lambda)$-Strategie (Elitismus) realisieren können soll. (4 Punkte)

Aufgabe 26 Optimierung mit Evolutionsstrategien - (6 Punkte)

Nutzen Sie sowohl Ihre Implementierung aus der vorangehenden Aufgabe als auch die im `lib`-Verzeichnis zur Verfügung gestellten Problemklassen `FourierApproximation` und `PolynomApproximation`, um die Optimierungsleistung der Evolutionsstrategien zu untersuchen. Alternativ können Sie gern Ihre

[1] `http://www.cogsys.cs.uni-tuebingen.de/lehre/ss11/ga_es_ueb.html`

eigene Implementierung dieser Probleme aus dem vorherigen Übungsblatt verwenden. Ziehen Sie für Ihre Untersuchungen den Datensatz `SampleFunction1.txt` heran. Evaluieren Sie folgenden Einstellungen:

(a) Elitismus ohne Crossover (2 Punkte)

(b) Crossover ohne Elitismus (2 Punkte)

(c) Crossover mit Elitismus (2 Punkte)

Erläutern kurz Sie Ihre dabei gemachten Beobachtungen und geben Sie für jede der beiden Problemklassen ein Bildschirmfoto des besten Optimierungsergebnisses an.

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, denn dies kostet meist deutlich mehr Zeit als geplant. Wer es dennoch für unerlässlich hält, möge ausschließlich das AWT und Swing verwenden.