EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

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

Evolutionäre Algorithmen

Übungsblatt 9, SS 2011

Abgabe: 28.06.11

Aufgabe 23 Die (1+1)-Evolutionsstrategie - (6 Punkte)

Implementieren Sie den (1+1)-ES-Algorithmus auf Seite 171 des Manuskriptes. Laden Sie sich dazu das neue *Framework* von der Übungswebseite[1] herunter. Das Framework beinhaltet eine JAR-Datei, die eine einfache grafische Ausgabe der Optimierungsergebnisse verwirklicht. Außerdem finden Sie darin das Verzeichnis `assignments` mit den Unterverzeichnissen `optimizers` und `problems`. Die Klasse `ES` im `optimizers`-Verzeichnis bietet ein Gerüst für Ihre zu implementierende Klasse. Ausführen können Sie die Implementierung durch die Klasse `Application` im `assignment`-Paket. Dabei wird der `ES`-Klasse eine `Problem`-Instanz übergeben. Alle Optimierungsprobleme befinden sich im Paket `problems` und implementieren das *Interface* `Problem`. Damit die grafische Oberfläche über verbesserte Optimierungsergebnisse informiert wird, benachrichtigt `ES` seine Instanzen der `SolutionChangedListener` wann immer eine bessere Lösung gefunden wird. Die Klasse `Application` verwendet diesen Mechanisumus und fügt der `ES`-Klasse genau einen solchen *Listener* hinzu. Damit die Optimierung nebenläufig zur grafischen Oberfläche funktionieren kann, implementiert `ES` das `Runnable`-*Interface* und wird von `Application` in einem eigenen `Thread` gestartet. Deshalb sollte die Hauptroutine des ES-Algorithmusses in der geerbten `run`-Methode stattfinden. Die grafische Darstellung zeigt nicht, wie bisher üblich, die Entwicklung der Fitness, sondern den aktuell besten Phänotypen der Population.

Als Problemstellung soll Ihre ES-Klasse vorgegebene Funktionswerte approximieren. Um Ihre (1+1)-ES-Klasse erfolgreich testen zu können, befindet sich im `problems`-Paket bereits die Klasse `LinearApproximation`, die das `Problem`-*Interface* implementiert. Als Fehlermaß verwendet diese Klasse den quadratischen Abstand zwischen Approximation und vorgegebenen Funktionswerten. Funtionswerte der Zielfunktionen finden Sie im `resources`-Verzeichnis in den Dateien `SampleFunction1.txt` und `SampleFunction2.txt`. Das Parsen dieser Dateien wird bereits durch die Klasse `SampleValues` übernommen, die im `framework.jar` enthalten ist.

Aufgabe 24 Polynom- und Fourierapproximation - (14 Punkte)

Wie Sie beim Testen Ihrer ES-Implementation aus der vorherigen Aufgabe erkennen können, bietet eine lineare Approximation keine zufriedenstellende Annäherung an die vorgegebenen Funktionspunkte. Aus diesem Grund sollen nun zwei weitere Approximationstechniken implementiert werden. Für jede Approximationsmethode legen Sie dazu eine eigene Klasse an, die jeweils die `Problem`-Schnittstelle implementiert und in das `problems`-Paket eingefügt wird.

(a) Implementieren Sie zunächst eine Polynomapproximation gemäß folgender Formel:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \,,$$

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

wobei $a_n, \ldots, a_0$ die reelwertigen Elemente des Individuums darstellen, d. h. die Funktionsparameter. (5 Punkte)

(b) Implementieren Sie als zweite Approximation eine Fourier-Reihe:

$$f(x) = \frac{a_0}{2} \sum_{i=1}^{n} a_i \cos(ix) + b_i \sin(ix) \,.$$

Hinweis: Die Zahl der Parameter muss ungerade und echt größer null sein. (7 Punkte)

(c) Testen und vergleichen Sie die Approximationsmethoden bezüglich ihrer Konvergenzgeschwindigkeit und der Qualität derer gefunden Lösung. (2 Punkte)

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.