Hinweise zur Bewertung der Übungsaufgabe 3

Während Ihre Lösungen zu den ersten beiden Übungsaufgaben lediglich am Rechner vorzuführen sind, wird für die nachstehende Aufgabe 3 eine detaillierte schriftliche Ausarbeitung erwartet.

1. Manche der in Übungsaufgabe 3 gestellten Fragen beziehen sich auf Verfahren oder Modelle, die nicht in der Vorlesung besprochen wurden.
   - Es wird erwartet, dass Sie sich zu deren Beantwortung selbstständig in einschlägiger Literatur umschauen, jedenfalls dann, wenn Ihnen die in der GEATbx-Dokumentation gegebenen Erklärungen nicht ausreichend erscheinen.

2. Zur Aufgabenstellung gehören auch Untersuchungen zum Einfluss von Parametereinstellungen und unterschiedlichen Operatoren auf die Performanz des evolutionären Algorithmus. Um hierbei zu signifikanten Ergebnissen zu gelangen, müsste eigentlich eine große Anzahl von Testläufen durchgeführt werden.
   - Weil dies aber nur mit erheblichem Zeitaufwand möglich wäre, dürfen Sie diese Anzahl "in vernünftigen Grenzen" halten. Eine Hilfe für die Abschätzung dieser Grenzen bietet die folgende Angabe:
   - Die für die Bearbeitung der gesamten Aufgabe veranschlagte Zeit beträgt ca. 25 Stunden (über 7 Wochen verteilt).
   - Von den Teilnehmern, die die Aufgabe zu zweit bearbeiten, werden daher natürlich deutlich mehr Testergebnisse erwartet als von den "Einzelkämpfern".

3. Einige Hinweise zur schriftlichen Ausarbeitung:
   - Bitte geben Sie alle verwendeten Quellen an und zitieren Sie korrekt.
   - Testergebnisse sind am besten in Tabellen und/oder Diagrammen darstellbar. Da es sich um "Zufallsergebnisse" handelt, sind stets Mittelwert und Standardabweichung aus mehreren gleichartigen Testläufen anzugeben.
   - Begriffe und Verfahren werden oft am besten durch einfache Beispiele und graphische Darstellungen beschrieben.
   - Liefern Sie die Ausarbeitung bitte in gedruckter Form ab.
   - Fügen Sie unbedingt auf CD den Quellcode Ihres optimierten Algorithmus bei, darunter insbesondere auch die M-Funktion `reccycle`, deren Implementierung im Aufgabenteil e) verlangt wird.
   - Die Teilnehmer, die die Aufgabe zu zweit bearbeiten, dürfen eine gemeinsame Ausarbeitung erstellen. In dieser sollte jedoch kenntlich gemacht werden, wer für welche Ergebnisse verantwortlich zeichnet.
   - Definitiv spätester Abgabetermin ist der 16. Juli 2007.

Übungsaufgabe 3

Das Skript `demotsplib` der Toolbox GEATbx definiert einen evolutionären Algorithmus zur Lösung von Travelling Salesman Problemen (TSP). Im Folgenden sollen einige der darin verankerten Parameterwerte und Optionen modifiziert und die Auswirkungen solcher Änderungen auf die Güte des Algorithmus untersucht werden. Das Ziel besteht natürlich darin, eine Konstellation zu finden, unter der der Algorithmus möglichst gute Lösungen in möglichst kurzer Zeit ermittelt.

Unter `\objfun\tsp` finden sich einige TSP-Beispiele. Verwenden Sie in den nachstehend beschriebenen Testläufen zunächst die Probleminstanz `bays29.tsp`; diese Datei enthält die Entfernungstabelle von 29 Städten Bayerns. Unser Ziel besteht also darin, eine kürzeste Rundreise durch diese 29 Städte zu finden.

a) Führen Sie einige Erstaufrufe von `demotsplib.m`, angewandt auf `bays29.tsp`, durch. Versuchen Sie herauszufinden, wie gut oder schlecht die Ergebnisse dieser Testläufe sind.

   Welche Parameter und Optionen sind voreingestellt?

b) Automatisieren Sie Ihre weiteren Testläufe:

   In einigen der nachfolgenden Aufgabenteile sollen einzelne Parameter ceteris paribus in von Ihnen festzulegenden Schrittweiten verändert werden und jeweils anschließend einige Testläufe erfolgen. Sorgen Sie dafür, dass dies weitgehend programmgesteuert geschieht und auch die einzelnen Testergebnisse in geeigneter Form festgehalten werden.

   Beschreiben Sie die Gesamtarchitektur Ihres Testsystems, d. h. die von Ihnen hierfür verwendeten Dateien und Scripts.

c) Populationsgrößen, Unterpopulationen und Migration:

   Führen Sie einige Testläufe mit unterschiedlichen Populationsgrößen durch. Geben Sie zunächst nur 1 Population vor; variieren Sie dann aber auch die Anzahl der *Unterpopulationen*.

   Machen Sie sich dabei mit dem Begriff *Migration* vertraut: Beschreiben Sie das zu Grunde liegende Modell und die durch den Parameter `Migration.Topology` zu steuernden Varianten. Welche Rolle spielen die Parameter `Migration.Rate` und `Migration.Interval`?

d) Selektion:

   Ersetzen Sie das Default-Verfahren `selsus` (Welches ist das?) durch die klassische *Roulette Wheel Selection*. Hat dies sichtbare Auswirkungen auf die Leistung Ihres Algorithmus? Oder ist vielleicht die *Tournament Selection* (`seltour`) geeigneter?

e) Rekombination:

   Ersetzen Sie den in `demotsplib` verwendeten Rekombinationsoperator `recgp` (*generalized position crossover*) durch `recpm` und vergleichen Sie die Resultate in einigen Testläufen. Welches Verfahren steckt hinter `recpm`?

   Den *Cycle Crossover* (`reccycle`) gibt es leider nicht im Angebot der Toolbox. Implementieren Sie diesen daher selbst und führen Sie wiederum einige vergleichende Testläufe durch.

f) Mutation:

   Beschreiben Sie mit Hilfe einiger Beispiele die Verfahren `mutswap`, `mutmove` und `mutinvert`. Warum ist es wohl sinnvoll, alle drei Mutationsoperatoren einzusetzen?

   Versuchen Sie, ein paar Testläufe mit unterschiedlichen Mutationsraten durchzuführen.

g) Optimale Einstellungen:

   Geben Sie die Kombination von Operatoren und Parameterwerten an, die Ihnen am besten erscheint. Führen Sie mit Ihrem so konfigurierten Algorithmus auch ein paar Testläufe mit den TSP-Beispielen `bayg29.tsp` und `berlin52.tsp` durch und geben Sie die Resultate an.