Aufgabenstellung (WS 13/14)

Das Skript `tspVorlage` basiert auf der Toolbox GEATbx und definiert einen evolutionären Algorithmus zur Lösung von Travelling Salesman Problems (TSP). Im Folgenden sollen einige der im Skript 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 `\toolbox\geatbx\objfun\tsp` finden sich einige TSP-Beispiele. Diese gehören zu einer bekannten Gruppe von Benchmark-Testinstanzen. Verwenden Sie in den nachstehend beschriebenen Testläufen die Probleminstanz `bays29.tsp` [1]. Sie enthält die Entfernungstabelle von 29 Städten Bayerns. Das Ziel besteht darin, eine kürzeste Rundreise durch diese 29 Städte zu finden. (Deren Länge beträgt, wie man schon weiß, 2020 km.)

a) Beschreiben Sie das in `bays29.tsp` verwendete Datenformat.

b) Führen Sie eine größere Anzahl von Aufrufen des Skripts `tspVorlage.m`, angewandt auf `bays29.tsp`, durch. Stellen Sie die Ergebnisse dar und beurteilen Sie die Güte dieser Testläufe.

c) Beschreiben Sie die in `tspVorlage.m` explizit vorgegebenen Parameter und Optionen.

d) 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 dass die einzelnen Testergebnisse in für statistische Auswertungen geeigneter Form festgehalten werden.

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

e) Populationsgrößen, Unterpopulationen und Migration:

   Was versteht man unter diesen Begriffen?

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

   Experimentieren Sie dabei auch mit den *Migrationsparametern.*

f) Selektion:

   Ersetzen Sie die klassische *Roulette Wheel Selection* durch die Verfahren *Stochastic Universal Sampling* bzw. *Tournament Selectio*n. Beschreiben Sie die beiden letztgenannten Verfahren kurz. Welche der drei Selektionsmethoden schneidet in Ihren Testläufen am besten ab?

[1] Nicht zu verwechseln mit der Datei `bay**g**.tsp`!!

g) Rekombination:

Neben dem in der Vorlesung erwähnten *Partially matched Crossover* (`recpm`) gibt es für das TSP unter anderem auch das so genannte *Order Crossover (OX)* von L. Davis.

Leider gehört dieses Rekombinationsverfahren nicht zum Angebot der Toolbox. Implementieren Sie es daher selbst, nachdem Sie sich kundig gemacht haben, welcher Algorithmus dahinter steckt. Führen Sie wiederum einige vergleichende Testläufe durch.

h) Mutation:

Neben dem `mutswap` bietet die Toolbox auch die Mutationsverfahren `mutmove` und `mutinvertan`. Was steckt dahinter?

Führen Sie Testläufe, in denen alle drei Mutationsverfahren verwendet werden, durch und analysieren Sie deren Ergebnisse. Variieren Sie dabei auch die Mutationsraten.

i) Optimale Einstellungen:

Fassen Sie schließlich die Kombination von Operatoren und Parameterwerten, die Ihnen am besten erscheint, in einer m-Funktion zusammen.