Computational Intelligence

M. Sc. Andreas Buschermöhle, M. Sc. Jan H. Schoenke

Übungsblatt 2 - Evolutionäre Algorithmen II: Evolutionäre Strategien

Abgabe bis Mittwoch, 21.05.2014, 12:00 Uhr

In diesem Übungsblatt werden Algorithmen in Matlab geschrieben und zur Auswertung ein- und zweidimensionale Plots angefertigt. Erstellen Sie im Dateiordner Ihrer Gruppe in Stud.IP einen Ordner mit dem Namen `Blatt 2` und laden Sie zu zweit sowohl die m-Files, als auch die fertigen Matlab-Figures mit sinnvoller Achsenbeschriftung rechtzeitig in diesen Ordner hoch. Geben Sie in Vips je nach Aufgabenstellung die textuelle Interpretationen ihrer Ergebnisse sowie ggf. die Namen der Dateien an, die sich auf die jeweilige Aufgabe beziehen.

Ein Framework für Evolutionäre Strategien in Matlab lässt sich analog zur Umsetzung für Genetische Algorithmen in Übungsblatt 1 aufbauen. In diesem Fall beschränken wir uns auf ein Hauptprogramm, das nach folgendem Schema aufgebaut ist. Die einzelnen hier angedeuteten Teilaspekte sollen jeweils für die Aufgaben ausgefüllt werden.

```
% Parametersetup
mu = ...;
lambda =  ...;
mutationRate = ...;
...

% Speicher für Verlaufsgrößen

%Erzeugen der Startpopulation
population = ...
fitness = ...;

for i=1:100
    % Neuen Pool erstellen

    % Neuen Pool bewerten
    fitness =

    % Neuen Pool mutiern

    % Neuen Pool nach Fitness sortieren

    % Bestes Individuum speichern

    % Nächste Generation erzeugen
end

%Auswertung
```

Die Population besteht wie im ersten Übungsblatt aus einer zweidimensionalen Matrix, die in einer Dimension die Individuen und in der zweiten Dimension die Gene enthält. In diesem Fall sind die Gene jedoch reelle Zahlen und nicht binäre Werte. Zunächst soll eine zufällige

Startpopulation erzeugt werden, wobei die Anzahl der Individuen, die Anzahl der Gene und der Wertebereich der Zufallszahlen aus $[minVal, maxVal]$ über Parameter einstellbar sein sollen.
Zur Evaluation der Fitness können in diesem Fall die reellen Zahlen der Gene direkt an die zu maximierende Funktion übergeben werden. Diese Funktion wird wiederum in einem separaten m-File abgelegt. In einer Schleife wird dann eine Bewertung jedes Individuums durchgeführt, indem die Funktion an der entsprechenden Stelle ausgewertet wird. Anschließend kann die Population absteigend nach der Fitness sortiert werden. So können im nächsten Schritt für das (λ, μ) Überlebenskriterium die besten μ Individuen für die Erzeugung der nächsten Generation genutzt werden.
Zur Generierung der λ Nachkommen sollen zufällig zwei Eltern aus den besten μ gewählt werden und zwischen deren Genomen global linear Interpoliert werden, wobei die Gewichtung der linearen Interpolation zufällig gewählt wird (s.Vl.: Arithmetische Rekombination mit gleichem α für alle Gene). Das so gewonnene Genom wird dann mutiert, indem auf jedes Gen eine normalverteilte Zufallszahl addiert wird. Die Zufallszahl soll nach einer skalierten Normalverteilung gezogen werden. Die Matlab-Funktion `randn` liefert Werte anhand der Standardnormalverteilung, die mit einem Faktor `mutationRate`, der die Stärke der Mutation angibt, skaliert werden können. Nach jeder Mutation wird geprüft ob diese Mutation die Fitness verbessert hat. Ist dies nicht der Fall wird die Mutation zurückgenommen, sodass sich durch die Mutation die Fitness nicht verschlechtern kann.
Somit steht die nächste Generation zur Verfügung und die nächste Iteration kann durchlaufen werden. Dabei soll zunächst als Abbruchkriterium wieder eine feste Anzahl an Generationen vorgegeben werden.

Aufgabe 2.1: Maximumssuche mit ES (10 + 10 + 15 + 10 = 45 P)

In dieser Aufgabe soll die Evolutionäre Strategie auf das Beispiel aus Aufgabe 1.2

$$f(x) = e^{-(0{,}003x-3)^2} + 1{,}5e^{-\frac{1}{\left(1+e^{-\left(\frac{x-3500}{682}\right)^2}\right)^9}} + 1{,}15e^{-\frac{(x-2000)^2}{1000}} + \frac{\sin\left(\frac{x}{20}\right)+\cos\left(\frac{x}{21}\right)}{20} - 0{,}45\ .$$

angewendet werden, wobei nun die reelle Zahl direkt die Eingabe darstellt und somit der Wertebereich nicht mehr diskret ist. Die Population soll im Wertebereich auf das Intervall $[0, 4095]$ eingeschränkt werden.

a) Füllen Sie den oben vorgegebenen Rahmen aus, sodass der Algorithmus der Evolutionären Strategie auf das Beispiel angewendet werden kann. Wählen Sie hierzu zunächst $\lambda = 15$ und $\mu = 10$. Der Abbruch soll nach einer festen Generationenzahl von 100 Generationen durchgeführt werden. Die Mutationsrate sollte mit `mutationRate` $= 5$ zunächst fest gewählt werden. Tragen Sie wie im Übungsblatt 1 die Fitness des besten Individuums, des schlechtesten Individuums und den Durchschnitt der Population in einem Plot über die Zeit auf.

   Hinweis: Bei der Bearbeitung von Aufgabenteil a) empfiehlt es sich den Code so anzulegen, dass Aufgabenteil c) mit nur einer Parameteranpassung von τ bearbeitet werden kann.

b) Variieren Sie auch hier wie in Übungsblatt 1 die Populationsgröße μ zwischen 2 und 16 Individuen und wählen Sie jeweils $\lambda = \mu+5$. Variieren Sie zudem die Mutationsrate zwischen 1 und 50. Tragen Sie die resultierende Fitness nach 100 Generationen in einem dreidimensionalen Plot auf (Fitness über Populationsgröße und Mutationsrate). Um den Einfluss

der zufälligen Mutationen zu untersuchen, simulieren Sie auch hier 50 Läufe für jede Parameterkonstellation und erstellen wiederum drei Grafiken mit der besten, schlechtesten und durchschnittlichen Fitness der Läufe.

c) Fügen Sie nun als weiteres Gen die Mutationsrate zum Genom hinzu. So kann die Population neben der Optimierung des gesuchten Wertes auch die Mutationsrate anpassen und sich so an das gestellte Problem adaptieren. Verwenden Sie zur Adaption der Mutationsrate eine log-normale Verteilung (siehe Vorlesungsskript, $\tau = 1$) und mutieren Sie erst danach die übrigen Gene mit der angepassten Mutationsrate. Erstellen Sie auch hier einen Plot der Fitness der Population über die Zeit. Wie verändert sich der Verlauf der Fitness im Vergleich zu Aufgabenteil a)?

d) Lassen Sie die Funktion f durch ein Ensemble $E = \{e_1, \ldots, e_n\}$ von Optimierungsverfahren, die auf Gradientenabstieg basieren, maximieren. Verwenden Sie dazu die Funktion `fminsearch` aus Matlab. Die Größe des Ensemble $|E| = n$ soll der Anzahl der Individuen entsprechen, die in Teil a) bzw. c) in jeder Generation erzeugt werden, $|E| = n = \lambda = 15$. Schränken Sie die maximale Anzahl der Funktionsauswertungen des Ensembles so ein, dass sie der Anzahl der Funktionsauswertungen der ES-Maximierung entspricht. Wählen Sie die Startpositionen der einzelnen Ensemble-Mitglieder e_i einmal zufällig und einmal auf einem gleichmäßigen Gitter. Machen Sie bei der zufälligen Initialisierung mehrere Durchläufe und wählen Sie zwei oder drei repräsentative Ergebnisse. Markieren Sie für die Darstellung Ihrer Ergebnisse jeweils die Startposition der Optimierung und die Endposition der Optimierung auf dem Graphen der Funktion f und verbinden Sie die beiden Punkte mit einer Linie. Ist die Obergrenze der Funktionsauswertungen das relevante Abbruchkriterium für `fminsearch`? Wie viele Funktionsauswertungen fallen bei der linearen Verteilung der Startpositionen insgesamt an?

Aufgabe 2.2: Abbruchkriterien (5 + 7 + 8 = 20 P)

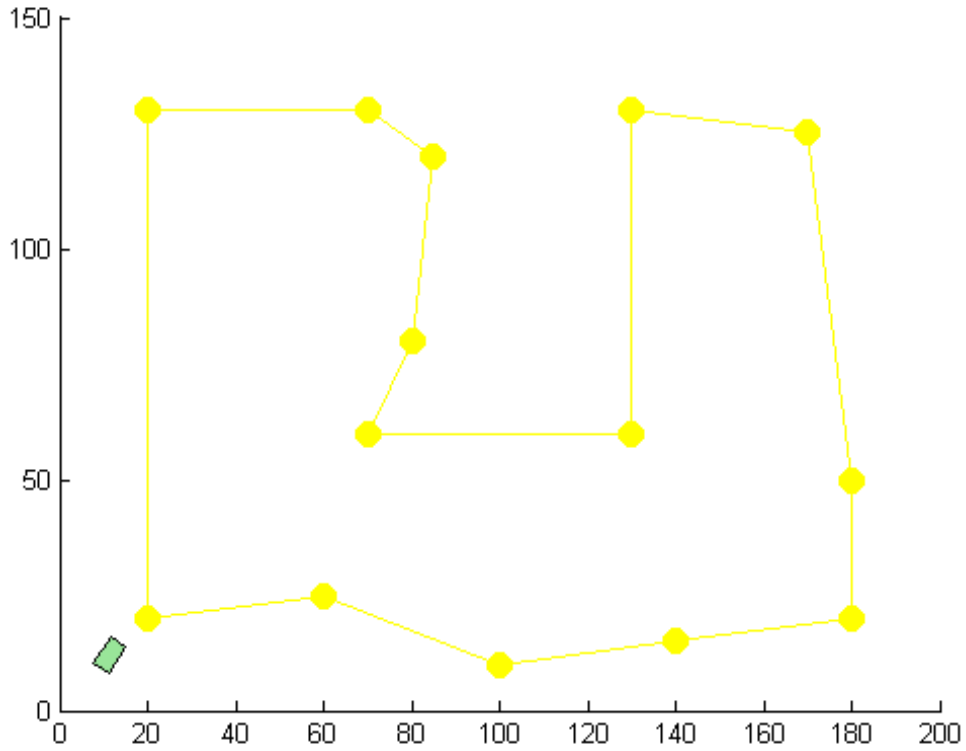
Bislang wurde zum Abbruch der Evolution eine feste obere Grenze für die Zahl der Generationen vorgegeben. In dieser Aufgabe sollen andere Abbruchkriterien getestet werden.

a) Implementieren Sie für das Beispiel aus Aufgabe 2.1 c) zunächst alternativ zur maximalen Generationenzahl eine Abbruchbedingung, bei der die Evolution beendet wird, sobald eine maximale Fitness überschritten wird. Es sollte dann ein ausreichend gutes Ergebnis erzielt worden sein. Wie sollte dieser Maximalwert sinnvoll gewählt werden? Zeigen Sie mögliche Vor- oder Nachteile anhand von ein oder zwei typischen Fitnessverläufen.

b) Implementieren Sie eine weitere Abbruchbedingung, bei der die Evolution beendet wird, sobald die Änderung der Fitness des besten Individuums eine Grenze unterschreitet. In diesem Fall sollte die Entwicklung der Individuen konvergiert sein. Wie sollte die untere Schranke sinnvoll gewählt werden? Zeigen Sie mögliche Vor- oder Nachteile anhand von ein oder zwei typischen Fitnessverläufen.

c) Implementieren Sie eine weitere Abbruchbedingung, bei der die Evolution beendet wird, sobald die Änderung der Fitness des besten Individuums länger als 50 Generationen eine Grenze unterschreitet. Zählen Sie hierzu die Anzahl Generationen sobald die Grenze unterschritten wird und setzen Sie den Zähler zurück sobald die Grenze wieder überschritten wird. Wie sollte die untere Schranke sinnvoll gewählt werden? Zeigen Sie mögliche Vor- oder Nachteile anhand von ein oder zwei typischen Fitnessverläufen.

Aufgabe 2.3: **Autorennen** **(35 P)**



In dieser Aufgabe sollen Sie frei eine Evolutionäre Strategie anwenden, um die Steuerung für ein simuliertes Fahrzeug zu optimieren. Das Fahrzeug muss einen Parcours von Zielpunkten abfahren. Hierzu hat es eine fest vorgegebene Zeit und jeder erreichte Zielpunkt erhöht die Fitness. Für diese Aufgabe ist eine Funktionen vorgegeben (siehe Stud.IP), die dazu dient die Autofahrt zu simulieren.

```
function fit = simCar(driver, n, show)
```

Als Übergabeparameter erwartet die Funktion in der Variable `driver` das Genom des Fahrers als (1×12)-Matrix und in der Variable `n` die Anzahl der zu simulierenden Zeitschritte. Über die boolsche Variable `show` kann eingestellt werden, ob der simulierte Lauf visualisiert werden soll. Der Rückgabewert `fit` enthält als Fitness die Anzahl der vom Fahrer während der Simulation erreichten Zielpunkte des Parcours. Wird in `driver` eine Matrix der Größe $(p \times 12)$ übergeben werden alle p Fahrer gleichzeitig simuliert und entsprechend ein Vektor von p Fitnesswerten zurückgegeben.
Sie können die Simulation direkt als Fitnessfunktion verwenden indem Sie die Funktion mit nur einem Übergabeargument aufrufen `simCar(driver)`. In diesem Fall wird automatisch die korrekte Simulationsdauer von `n = 100` Schritten eingestellt und die Visualisierung ausgeschaltet (`show = false`).
Ihre Aufgabe ist es nun, durch eine frei wählbare Evolutionäre Strategie eine bestmögliche Fahrzeugsteuerung zu entwickeln. Beschreiben Sie hierzu, welche Varianten Sie für die Populationsgröße, Crossover, Mutation, etc. gewählt haben und begründen Sie Ihre Wahl. Schicken Sie zudem das Genom der besten gefundenen Fahrzeugsteuerung ein. Nach wie vielen Generationen haben Sie dieses erreicht?