Die Hausaufgaben müssen einzeln bearbeitet und abgegeben werden. Geben Sie die schriftlich zu bearbeitenden Aufgaben als pdf-Dateien ab.

Abgabefrist ist der 14.01.2024 – 23:59 Uhr

Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studierenden selbst sowie für die Betreuer und Korrektoren sichtbar.

Für die Hausaufgabe benötigen Sie **kein** neues Repository. Es wird das Repository benutzt, das für Übungsblatt 2 angelegt wurde.

Damit das Repository im Laufe der Vorlesung übersichtlich bleibt und wir sinnvoll korrigieren können, achtet bitte auf das Folgende:

- Der finale Commit für ein Übungsblatt muss mit einem Tag oder einer klaren Commit-Message versehen sein, der/die diesen Commit eindeutig als die finale Abgabe erkennen lässt.
- Erstellt **pro Übungsblatt** einen Ordner, in dem die Files liegen, die kein Code sind (Textfiles, Plots, ...).

Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet!

Aufgabe 1 – Symbolic regression durch einen evolutionären Algorithmus auf Graphen (40P)

Auf diesem Blatt beschäftigen wir uns mit *symbolic regression* per evolutionärem Algorithmus auf Graphen. Wir verwenden dafür die Python-Variante der Bibliothek igraph. Ziel ist es, durch einen evolutionären Algorithmus (basierend auf Graphen als Repräsentation) eine Funktion zu finden, die die Funktion

$$f(x, y) = 6 \sin(x) \cos(y)$$

möglichst exakt approximiert. Gehen Sie in den folgenden Schritten vor.

1. Implementieren Sie die in der Vorlesung besprochene Kodierung von Funktionen als Graphen (z. B. Folie 132). Dazu gehört das Folgende bzw. sind die folgenden Dinge zu beachten.
   - Ihre Graphen sollen eine feste Größe von 15 Knoten haben. Von diesen Knoten sind zwei als Input- und einer als Outputknoten gekennzeichnet. Die übrigen 12 Knoten sind über dem Alphabet

     $$L = \{\text{add}, \text{sub}, \text{mul}, \text{div}, \exp, \log, \text{sqrt}, \sin, \cos, \tan, \text{const}\}$$

     gelabelt. Ein Knoten, der mit const (für Konstante) gelabelt ist, soll zusätzlich einen Float als Wert der Konstanten speichern. Die Kanten, die einen Knoten verlassen, sollen eine Reihenfolge haben. Zusätzlich sollen die Nebenbedingungen erfüllt sein:

(i) jeden Knoten verlassen exakt so viele Kanten, wie der Stelligkeit der repräsentierten Funktion entspricht; (ii) der Graph ist kreisfrei. Label (und Werte für Konstanten) können Sie in igraph zum Beispiel als Attribute eines Knoten speichern.

- Implementieren Sie eine Dekodierfunktion, die einen solchen Graphen als Eingabe nimmt und in diejenige Funktion übersetzt, die durch den Graphen repräsentiert wird (also die Funktion, die sich aus dem *expression tree* ergibt, der am Output-Knoten startet).
- Implementieren Sie einen Zufallsgenerator für das Erzeugen einer Startpopulation. Sie dürfen als Ausgangspunkt einen der Zufallsgeneratoren verwenden, die igraph zur Verfügung stellt. Zum Beispiel die Funktion Graph.Degree_Sequence erlaubt das Erzeugen eines zufälligen Graphen mit vorgegebenem Grad für die Knoten.

  Beachten Sie aber, dass Sie Labels setzen und die Nebenbedingungen sicherstellen müssen. Zum Beispiel die Methode is_dag überprüft, ob ein Graph kreisfrei ist, und die Methode feedback_arc_set berechnet eine Menge von Kanten, deren Entfernen einen Graph kreisfrei macht.

2. Implementieren Sie die Auswertung der Fitness eines Graphen. Ziel ist die Minimierung der quadratischen Abweichung von der Funktion $f(x,y)$; um dennoch *fitness proportional selection* anwenden zu können, gehen wir wie folgt vor: Wir verwandeln das Minimierungs- in ein Maximierungsproblem, tun dies aber abhängig von der aktuellen Population.

   - Zerlegen Sie das Gebiet $[0,2\pi] \times [0,2\pi] \subseteq \mathbb{R}^2$ gleichmäßig in ein Gitter mit 10 000 Stützstellen. Werten Sie an jeder dieser Stellen (z_i^x, z_i^y), $0 \leq i \leq 9999$, die Funktion f aus und speichern Sie die Ergebnisse als einen Vektor valuesf.
   - Kodiert ein Graph G die Funktion f_G, so berechnen Sie den Abstand zu f als
   $$d_f = \sum_i (\mathsf{valuesf}[i] - f_G(z_i^x, z_i^y))^2,$$
   wobei wieder $0 \leq i \leq 9999$. Sie können d_f auch als Attribut des Graphen G speichern, um den Wert nicht erneut berechnen zu müssen.
   - Während des evolutionären Algorithmus, gegeben die aktuelle Population P, bestimmen Sie
   $$c_P = \max\{d_{f_G} \mid G \in P\}.$$
   Die Fitness fit_P eines Graphen G (in Abhängigkeit von P) soll dann definiert sein als
   $$fit_P(G) = c_P - d_{f_G}$$
   (insbesondere kann sich die Fitness eines Graphen, abhängig von der Population, in jeder Iteration ändern).

3. Implementieren Sie *mindestens zwei* Mutationsoperatoren für Ihre Graphen (wir verzichten auf Crossover). Die Operatoren dürfen die Knoten- und Kantenmutation aus der Vorlesung sein (Folien 185 und 186), Sie dürfen aber auch eigene entwerfen. Achten Sie darauf, dass (i) die Mutationsoperatoren in jedem Fall Graphen wieder in Graphen überführen müssen, die die Nebenbedingungen einhalten, und (ii) Sie unterstützen müssen, dass sich Werte von Konstanten ändern.

   Sie dürfen die Methoden, die igraph zum Manipulieren von Graphen zur Verfügung stellt, nutzen.

4. Wenden Sie nun Ihren SGA an, um die Funktion f evolutionär zu approximieren. Setzen Sie die Crossoverrate auf 0. Die Permutationsrate entscheidet wieder, ob ein Graph überhaupt permutiert wird oder nicht. Soll ein Graph permutiert werden, entscheiden Sie zunächst zufällig gleichverteilt, welcher der zur Verfügung stehenden Mutationsoperatoren angewendet wird. Gehen Sie wie folgt vor:

   a) Überlegen Sie sich **drei** verschiedene Einstellungen für die Parameter des SGA (Mutationswahrscheinlichkeit, Populationsgröße und Anzahl der Iterationen) und lassen Sie Ihren SGA pro Wahl **zehnmal** laufen, um die Funktion f zu approximieren. Die Anzahl der Iterationen soll jeweils mindestens 1000 betragen.

   b) Berechnen Sie für die drei Set-ups für jede Iteration a) den Durchschnitt (Median) über die 10 Durchläufe über die durchschnittliche Fitness (Median) und b) den Durchschnitt (Median) über *den Abstand* d_f der besten Lösung. Plotten Sie jeweils die Entwicklung der durchschnittlichen Fitness und des Abstands gegen die Anzahl der Iterationen.

   c) Bewerten Sie Ihr durchgeführtes Experiment, indem Sie (zunächst ohne Wertung) Ihre Beobachtungen beschreiben und dann Schlussfolgerungen ziehen, welche Vor- und Nachteile welche Einstellungen hatten.

Frohe Weihnachten und einen guten Rutsch ins neue Jahr!