Arbeitsanleitung: Tutorial „Natur-inspirierte Schwarmintelligenz“

Zielgruppe:
Studierende Advanced Software Engineering, DHBW Mosbach

Teamgröße:
2 Studierende (ausnahmsweise ein Team mit 3 Studierenden)

Abgabe: 12.05.2025, 23:59 Uhr, Moodle (ZIP)

01 | Verteilung der Algorithmen

Die Studierenden organisieren die eigenständige Zuteilung der Algorithmen innerhalb der jeweiligen Kurse. Ziel ist, dass alle genannten Algorithmen von mindestens einem Team bearbeitet werden.

02 | Algorithmenliste

Wählen Sie einen Algorithmus aus der folgenden Liste (je Team ein Algorithmus):

- African Buffalo Optimization
- Aquila Optimizer
- Bat Algorithm
- Beluga Whale Optimization
- Cheetah Optimizer
- Cockroach Swarm Optimization
- Coyote Optimization Algorithm
- Cuckoo Search
- Elephant Herding Optimization
- Grasshopper Optimization Algorithm
- Green Anaconda Optimization
- Grey Wolf Optimizer
- Jaguar Algorithm
- Lion Optimization Algorithm
- Lion Pride Optimization
- Locust Swarm Optimization
- Manta Ray Foraging Optimization
- Monkey Search
- Reptile Search Algorithm
- Savannah Bengal Tiger Optimization

- Seagull Optimization Algorithm
- Scorpion Hunting Strategy
- Snow Leopard Optimization
- Social Spider Algorithm
- Termite Colony Optimization
- Tuna Swarm Optimization
- Whale Optimization Algorithm
- White Shark Optimizer

---

03 | Steckbrief-Erstellung (Markdown-Datei)

Erstellen Sie eine Datei `<Algorithmusname>.md` mit folgenden Abschnitten:

Biologisches Vorbild und natürliches Verhalten

- Beschreibung der Spezies bzw. Gruppe
- Wesentliche Verhaltensweisen und Adaptionsstrategien

Algorithmische Mechanik

- Grundprinzip (Exploration vs. Exploitation)
- Initialisierung (Population, Parameter)
- Update-Regeln (Positions- und Parameteranpassung)

Pseudocode

- Kernstruktur: Schleifen, Abbruchkriterium, Individual- und Sozialkomponenten

Soziale Interaktion

- Kommunikationsmodell unter Individuen
- Einflussmechanismen (z. B. Leader–Follower, Schwarmbildung)

Mathematische Formulierung

- Zentrale Gleichungen (Positionsupdate, Gewichtungsfaktoren)
- Parameterübersicht (z. B. Populationsgröße, Iterationen, Lernraten)

Hinweis:
Nutzen Sie ein beliebiges KI-Tool zur Generierung und formulieren Sie präzise Prompts. Überprüfen Sie alle Inhalte sorgfältig und abschließend.

---

04 | Implementierung: Java 21+ Konsolen-Applikation

01 | Architekturrichtlinien & SOLID

- **Interface `Optimizer`:**

```
public interface IOptimizer {
    void initialize();
    void iterate();
    boolean termination();
}
```

- **Konkrete Implementierung:** `<Algorithm>Optimizer implements Optimizer`
- **Objective Function:**

```
public interface IObjectiveFunction {
    double evaluate(double[] x);
}
```

  - `AckleyFunction implements ObjectiveFunction`
- **Utilities:**
  - Logger (mittels `java.util.logging` oder SLF4J)
  - CLI-Parser zur Übergabe von Parametern (Populationsgröße, Dimension, Iterationen)

03 | Ablauf und Protokollierung

1. **Initialisierung**
   - Einlesen aller Parameter aus einer JSON-Konfigurationsdatei
   - Instanziierung von `AckleyFunction` und `<Algorithm>Optimizer`
2. **Optimierungsschleife**
   - In `iterate()`: Ausführung individueller und sozialer Update-Schritte
   - Nach jeder Iteration: Logeintrag mit Iterationsnummer, Global-Best-Position, Fitnesswert und Zeitstempel
3. **Abschluss**

   Bei Erreichen des Abbruchkriteriums Ausgabe der besten Lösung in der Konsole und im Logfile

Log-Details: Iteration, globale Bestposition, aktueller Fitnesswert, Auswahlentscheidungen, Parameteränderungen

---

05 | Qualitätssicherung und Review

- **Funktionale Tests:**
  - Verifikation der Ackley-Funktion anhand bekannter Minima
  - Wiederholbarkeit mittels festem Zufallsseed

- **KI-gestütztes Review:**

  Wenden Sie ein zweites KI-Tool an, prüfen Sie Steckbrief und Code auf Vollständigkeit, fachliche Korrektheit, Pseudocode-Kohärenz und SOLID-Konformität

---

06 | Abgabe und Dokumentation

ZIP-Struktur:

```
/tutorial_<Kurs>_<Team>.zip
├─ readme.txt       # Teammitglieder, Matrikelnummern, eingesetzte KI-Tools (Erstellung & Review)
├─ <Algorithmus>.md # Steckbrief
└─ src/             # Java-Quellcode
   ├─ optimizer/
   ├─ function/
   └─ util/
```

readme.txt enthält Team, Matrikelnummern sowie Angaben zu den eingesetzten KI-Tools

---

Prof. Dr. Carsten Müller | Advanced Software Engineering