Search-Based Software Engineering

Sommersemester 2024, Übungsblatt 3
Exploitative Multi-State Meta-Heuristics & Representation

Diskussion am: **05.06.2024**

SOFTWARE SYSTEME
Prof. Dr.-Ing. Norbert Siegmund
Johannes Dorn, M. Sc.
Max Weber, M. Sc.
Stefan Jahns, B. Sc.

Aufgabe 1: Wissen

a) Was unterscheidet Genetische Algorithmen von Evolutionären Algorithmen?

b) Welche Eigenschaft des Sample Sets garantiert Stochastic Universal Sampling (SUS)? Fertigen Sie eine Skizze an und beschreiben Sie den Algorithmus.

c) Wie unterscheidet sich der Steady-State Genetic Algorithm vom regulären Genetischen Algorithmus?

d) Was unterscheidet den Memetic Algorithm vom regulären Genetischen Algorithmus?

e) Was unterscheidet den Differential Evolution Algorithm vom regulären Genetischen Algorithmus?

f) Wie setzt sich die Update-Formel für die “Velocity” im Particle Swarm Algorithm zusammen?

g) Was beschreiben Genotyp und Phänotyp allgemein im Bereich Genetischer Algorithmen? Beschreiben Sie den Genotyp und Phänotyp für ein Beispiel aus der Natur.

h) Was bezeichnet die Hamming Cliff und warum ist sie problematisch? Erklären Sie anhand eines einfachen Beispiels.

i) Was ist ein Abstract Syntax Tree? Wie unterscheidet er sich von einem Parse Tree?

Aufgabe 2: Exploitative Multi-State Meta-Heuristics

Im Rahmen des Software Engineerings treten häufig Optimierungsprobleme auf, bei denen weder eine klare Zielfunktion vorliegt noch die Bewertung der Lösungsgüte frei von Rauschen ist. In der Vorlesung wurden verschiedene Algorithmen zu Exploitative Multi-State Meta-Heuristics vorgestellt. Implementieren Sie entweder einen Algorithmus mit Elitism, mit Steady-State, den Memetic Algorithm, Scatter Search, oder Differential Evolution.

Die Bewertung Ihrer Implementierung erfolgt über Black-Box-Funktionen, die im Moodle-Kurs bereitgestellt werden. Diese Funktionen simulieren die realen Unsicherheiten und Schwierigkeiten, die Sie in praktischen Anwendungen antreffen könnten. Ihr Ziel ist es, das Minimum dieser Funktionen zu ermitteln, wobei Sie ein selbst gewähltes Budget zur Ressourcennutzung festlegen.

Aufgabe 3: Introspection

Die Automatisierung im Software-Engineering, insbesondere das automatisierte Debugging und die Generierung von Testsets, ist entscheidend für die Entwicklung robuster und effizienter Softwarelösungen. Eine Voraussetzung für diese Aufgaben ist es, einen unbekannten Quelltext automatisch analysieren und ausführen zu können.

Aus diesem Grund sollen Sie in dieser Aufgabe eine Funktion zu entwickeln, die fähig ist, alle Methoden einer beliebigen Klasse automatisch auszuführen. Diese Implementierung soll nicht nur die Methoden und deren Parameter erkennen, sondern auch selbstständig aufrufen können. Lösen Sie dafür folgende Schritte:

a) Implementieren Sie eine Funktion, die einen Typ (Int/Float) erhält und eine valide Instanz des Typs zurückgibt. Hierbei sollen die Grenzwerte [1,17] für Int und [3.2, 4.5] eingehlten werden.

b) Nutzen Sie Ihre Implementierung aus a) um jede Methode der im beiliegenden Jupyter Notebook gegebenen Klasse aufzurufen. Dabei soll die Klasse dynamisch ausgetauscht werden können.

Diese Aufgabe dient als Vorarbeit für die Test-Set-Generierung im kommenden Aufgabenblatt.

Aufgabe 4: Representation (Teil 1)

Folgendes Szenario ist gegeben: Sie arbeiten in einem Unternehmen, das als Dienstleistung für andere Unternehmen Tests für deren Quelltext schreibt. Als Teil des Innovations-Teams sollen Sie durch den Einsatz eines evolutionären Algorythmus‘ für ein eine gegebene Klasse eine Test-Suite generieren. Ziel ist es, die Zweigabdeckung zu maximieren und die Ausführzeit zu minimieren. Ein zentraler Aspekt des Algorithmus wird die Repräsentation der Test Suite sein. Hierfür gibt es unter anderem folgende Rahmenbedingungen:

- Eine Test-Suite beinhaltet eine Abfolge von Testfällen.
- Der Konstruktor gilt als Methode.
- In einem Testfall muss die Klasse zuerst instanziiert werden.
- Ein Testfall ruft eine oder mehrere Methoden einer instanziierten Klasse auf.
- Es kann mehrere Testfälle geben, die dieselbe Methode aufrufen.
- Die Signatur der Methoden ist bekannt.
- Es ist eine Fabrik für jeden Datentyp der Methoden-Signaturen gegeben, die eine gültige, zufällige Instanz liefern. Ausgenommen ist der Konstruktor der gegebenen Klasse.

a) Beschreiben Sie zwei verschiedene Repräsentationen für eine Test-Suite. Schlagen Sie eine konkrete, möglicherweise zusammengesetzte, Datenstrukturen vor.

c) Beschreiben Sie Mutations-Operatoren für jede Repräsentation aus Aufgabe a).

d) Beschreiben Sie Crossover-Operatoren für jede Repräsentation aus Aufgabe a).

e) Schätzen Sie die Fitnesslandschaften anhand der Mutations-Operatoren für jede Repräsentation aus Aufgabe a) ab. Ergeben sich kleine Änderungen am Phänotypen

bzw. der Fitnessfunktion? Ist absehbar, dass mehrere Mutationen gemeinsam auftreten müssten um eine Verbesserung des Phänotypen bzw. der Fitnessfunktion zu bewirken?