Dokumentation zum genetischen Algorithmus

Dokumentation zum genetischen Algorithmus 1
- **Import von Bibliotheken (Zeile 1-2)** 2
- **Funktion generate_individual (Zeile 4-7)** 2
- **Funktion mutate_individual (Zeile 9-12)** 2
- **Funktion crossover (Zeile 14-19)** 2
- **Funktion fitness (Zeile 21-44)** 3
- **Funktion genetic_algorithm (Zeile 46-73)** 3
- **Hauptprogramm (Zeile 75-84)** 3
- **Erklärung der Ausgaben** 4
- **Beispielaufbau** 4

Das Skript implementiert einen genetischen Algorithmus, um ein Optimierungsproblem zu lösen. Der Algorithmus wird verwendet, um die beste Zuweisung von Aufgaben an Roboter zu finden und um die Gesamtzeit für die Verarbeitung aller Aufgaben zu minimieren.

Import von Bibliotheken (Zeile 1-2)

Das Skript beginnt mit dem Import von zwei Bibliotheken: random und numpy als np. Die random-Bibliothek wird verwendet, um zufällige Zahlen zu generieren, während die numpy-Bibliothek für numerische Operationen verwendet wird.

Funktion generate_individual (Zeile 4-7)

Die Funktion generate_individual generiert ein zufälliges Individuum, indem sie zwei zufällige Permutationen von person1 und person2 (Abb. 2) kombiniert. Ein Individuum besteht aus zwei Teilen: der ersten Hälfte, die die Aufgaben repräsentiert, und der zweiten Hälfte, die die Zuweisung der Aufgaben an Roboter repräsentiert (Abb. 1).

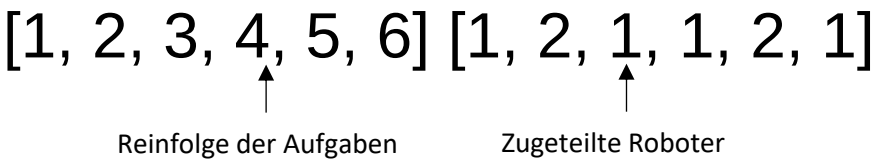

Abb.1

Funktion mutate_individual (Zeile 9-12)

Die Funktion mutate_individual mutiert ein Individuum, indem sie zwei zufällige Elemente innerhalb jeder Hälfte des Individuums austauscht. Dieser Vorgang wird verwendet, um die Vielfalt der Population zu erhöhen.

Funktion crossover (Zeile 14-19)

Die Funktion crossover führt einen Crossover-Vorgang zwischen zwei Eltern-Individuen durch. Der Crossover-Punkt wird zufällig ausgewählt, und die beiden Eltern-Individuen werden an diesem Punkt geteilt. Die beiden resultierenden Offspring-Individuen werden dann kombiniert.

Funktion fitness (Zeile 21-44)

Die Funktion fitness berechnet die Fitness (Abb. 2) eines Individuums. Die Fitness wird als die Gesamtzeit für die Verarbeitung aller Aufgaben berechnet. Die Funktion überprüft auch, ob alle Sicherheitsabstände größer als der minimale Sicherheitsabstand sind. Wenn ein Individuum ungültig ist (z. B., wenn es Duplikate enthält), wird die Fitness auf einen hohen Wert gesetzt.

Funktion genetic_algorithm (Zeile 46-73)

Die Funktion genetic_algorithm implementiert den genetischen Algorithmus. Die Funktion erstellt eine Population von Individuen und führt dann eine bestimmte Anzahl von Generationen durch. In jeder Generation werden die Fitness-Werte (Abb. 2) aller Individuen berechnet, und die beiden fittesten Individuen werden ausgewählt. Die Population wird dann durch den Crossover-Vorgang (Z. 62) und Mutationen aktualisiert.

Hauptprogramm (Zeile 75-84)

Das Hauptprogramm definiert die Eingabeparameter person1, person2, safety_distance und process_duration (Abb. 2). Es ruft dann die Funktion genetic_algorithm (Z. 81) auf und gibt das fitteste Individuum und seine Fitness (Abb. 2) aus.

Erklärung der Variablen

Variable	Aufgabe
person1 und person2	Diese Variablen repräsentieren die Aufgaben und die Zuweisung der Aufgaben an Roboter.
safety_distance	Dies ist der minimale Sicherheitsabstand, der zwischen den Aufgaben eingehalten werden muss.
process_duration	Dies ist die Verarbeitungszeit für jede Aufgabe.
population_size	Dies ist die Größe der Population, die im genetischen Algorithmus verwendet wird.
generations	Dies ist die Anzahl der Generationen, die im genetischen Algorithmus durchlaufen werden.
fitness	Berechnet die Dauer der Reinfolge durch die Wegdauer und die Bearbeitungsdauer

Abb.2

Erklärung der Ausgaben

Die Aufgabe des Algorithmus ist es die optimale reinfolge 2er Roboter zu finden die aufgaben auf einer schiene erfüllen sollen, ohne zu kollidieren. Dabei wird zur Bestimmung der optimalen reinfolge auf den niedrigsten fitness wert (Abb. 2) geachtet

Beispielaufbau

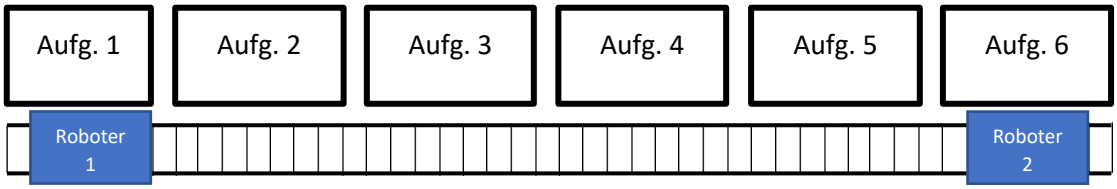

Abb.3