Anwendung heuristischer Algorithmen zur Lösung des Orientierungsproblems

Autor: Mahmoud Machaka (mahmoud.machaka@studium.fernuni-hagen.de)
Version: 1.0

Im vorliegenden Verzeichnis sind folgende Ordner und Dateien enthalten:

Name	Beschreibung
Balsamiq Wireframes	Enthält die Balsamiq Wireframes Arbeitsmappe und die entsprechenden Skizzen der GUI im PNG-Format.
Benchmarkings	Enthält die verwendeten Benchmark-Graphen und die Ergebnisse durchgeführter Benchmarking-Tests
Dokumentation	Enthält die JavaDoc-Dokumentation für die API der Anwendung. Öffnen Sie die index.html-Datei, um die Dokumentation anzuzeigen.
orienteering-problem	Quellcode der Anwendungen als Maven-Projekt
OPBenchmarking (.jar | .exe)	Applikation zur Durchführung von Benchmarking-Tests.
OPSolver (.jar | .exe)	Hauptapplikation zur Anwendung heuristischer Algorithmen.
Readme (.md | .pdf)	Readme-Datei, die Hinweise und eine Bedienungsanleitung für die Anwendung enthält.

Einleitung

Die vorliegende Anwendung wurde im Rahmen einer Masterarbeit an der FernUniversität in Hagen entwickelt und ermöglicht die Anwendung zweier heuristischer Algorithmen zur Lösung des Orientierungsproblems. Im Folgenden wird die Benutzung der Anwendung erläutert.

Voraussetzungen und Hinweise

- **Java SE 8**: Bitte verwenden Sie für beide JAR-Anwendungen Java SE 8 (Java Archive: Download). Die Verwendung anderer Java Versionen kann zu Darstellungs- und Anwendungsproblemen führen. Unter Windows können Sie die enthaltenen EXE-Dateien verwenden, welche bereits mit einer geeigneten Java Version gebündelt wurden.
- **Schreibrechte** In beiden Anwendungen müssen zur Speicherung der Ergebnisse entsprechende Speicherorte gewählt werden. Wenn Sie hier schreibgeschützte Bereiche angeben, müssen Sie sicherstellen, dass Sie die Anwendung mit entsprechend erweiterten Rechten ausführen. Andernfalls können die Ergebnisse nach der Berechnung nicht gespeichert werden.

Bedienungsanleitung

Mit dem Start der Anwendung erscheint ein Info-Fenster, in welchem das Orientierungsproblem und der Zweck der Anwendung kurz erläutert werden. Mit Betätigung des Weiter-Buttons gelangen Sie auf die Hauptseite der Anwendung, die Sie in vier Schritten zur Anwendung eines heuristischen Algorithmus führt.

Schritt 1

Im ersten Schritt müssen Sie einen gültigen Graphen laden. Durch Betätigung des „*Lade Graphen*"-Buttons öffnen Sie ein weiteres Fenster, in welchem Ihnen vier Optionen zur Wahl eines Graphen angeboten werden, die in den nächsten Abschnitten erläutert werden. Wenn ein gültiger Graph geladen wurde, schließt sich dieses Fenster wieder und Sie gelangen zum Hauptfenster zurück. Unter Schritt 1 wird nun mit grüner Schrift bestätigt, dass ein Graph geladen wurde. Außerdem wird die Größe bzw. Knotenanzahl des Graphen angezeigt.

Option 1
Hier können Sie eine entsprechend formatierte Textdatei laden, die einen Graphen beschreibt. Die Textdatei muss in jeder Zeile einen Knoten beinhalten, der sich aus seiner x- und y-Koordinate sowie seinem Profit zusammensetzt. Die Werte müssen jeweils mit einem Tabstopp getrennt werden. Als Dezimaltrennzeichen ist ein Punkt zu verwenden. Während die Koordinaten auch negativ sein dürfen, muss der Profit stets größer oder gleich 0 sein. Ein Graph muss mindestens zwei Knoten enthalten, wobei die ersten beiden Knoten seine Start- und Endknoten darstellen und einen Profit von 0 besitzen müssen. Im Folgenden ist beispielhaft ein Graph mit drei Knoten dargestellt:

```
-15.1   18.78   0
42.7    -17.1   0
11.35   22.36   15.7
```

Option 2
Mit dieser Option können Sie durch Klicken auf einer vereinfachten Deutschlandkarte Knoten hinzufügen, die dann sowohl auf der Karte eingezeichnet als auch in der nebenstehenden Tabelle aufgelistet werden. Die ersten beiden Knoten stellen auch hier den Start- und Endknoten des Graphen dar. In der Tabelle wird diesen Knoten daher ein Profit von 0 vergeben, der sich auch nicht editieren lässt. Jeder weitere Klick fügt Knoten an die entsprechende Stelle hinzu und vergibt diesen einen zufällig gewählten Profitwert (1-100), der sich manuell in der Tabelle anpassen lässt. Damit der Graph nicht an Übersichtlichkeit verliert, können neue Knoten nicht in unmittelbarer Nähe zu anderen Knoten gesetzt werden. Die Bereiche um einzelne Knoten sind entsprechend gesperrt. Wenn Sie mit Ihrer Auswahl zufrieden sind, können Sie mit Betätigung des „*Speichern*"-Buttons diesen Graphen laden. Andernfalls können Sie mit dem „*Reset*"-Button einen neuen Graphen zeichnen oder mit dem „*Abbrechen*"-Button eine andere Option zum Laden eines Graphen wählen.

Option 3
Hier können Sie kleinere vorgefertigte Graphen laden, die in der Literatur gerne referenziert und für Benchmarkings verwendet werden.
(siehe auch: https://www.mech.kuleuven.be/en/cib/op)

Option 4
Diese Option ermöglicht die Erzeugung eines Graphen mit beliebiger Größe. Der Profit eines Knotens wird zufällig in dem von Ihnen gewählten Profitbereich vergeben. Die Knoten werden gruppenweise hinzugefügt, um praktische Anwendungsfälle besser nachzustellen und eine Cluster-Bildung ermöglichen zu können. Die Gruppen können sich in ihrer Größe jeweils unterscheiden. Wenn Sie beispielsweise einen Graphen mit 500 Knoten erzeugen, können die Gruppen eine Größe von 10 bis 35 Knoten annehmen. Die Knoten jeder Gruppe werden um einen zufällig gewähltes Gruppenzentrum randomisiert verteilt.

Schritt 2

Nachdem Sie einen Graphen erfolgreich geladen haben, können Sie im nächsten Schritt den Speicherort für die Ergebnis-Datei auswählen. Das Ergebnis wird in der Datei „*result.txt*“ gespeichert. Um zu vermeiden, dass ältere Dateien überschrieben werden, werden neue Dateinamen mit dem Format „*result_[x].txt*“ gewählt, wobei *[x]* eine fortlaufende Zahl darstellt.

Schritt 3

Im dritten Schritt können Sie einen der beiden Algorithmen wählen. Abhängig von Ihrer Wahl erscheinen unterschiedliche Parameter-Felder, mit welchen Sie den gewählten Algorithmus konfigurieren können. Wenn Sie mit der Maus über die Eingabefelder fahren, wird Ihnen eine kurze Beschreibung des Parameters und sein Wertebereich angezeigt. Im Abschnitt Erläuterung der Parameter werden die einzelnen Parameter detailliert erläutert.

Hinweis: Die Kostenobergrenze Tmax wird nach dem Laden eines Graphen oder dem Betätigen des „Parameter zurücksetzen“-Buttons auf den Wert gesetzt, der wenigstens die Verbindung zwischen Start- und Endknoten in einer Tour erlaubt. Wird der Wert manuell kleiner gewählt, wird nach der Berechnung eine leere Tour zurückgegeben.

Schritt 4

Schließlich können Sie im vierten Schritt die Berechnung starten. Es wird geprüft, ob ein Graph geladen und ein Speicherort ausgewählt und ob zulässige Parameterwerte eingegeben wurde. Sollte diese Prüfung fehlschlagen, werden Ihnen die entsprechenden Probleme in einer Fehlermeldung angezeigt.

Ergebnis

Nach Abschluss der Berechnung wird Ihnen eine Zusammenfassung der Ergebnisse präsentiert. Sollten Sie in bei der Wahl des Graphen die Option 2 (Deutschlandkarte) ausgewählt haben, wird Ihnen zusätzlich die berechnete Tour auch auf der Karte dargestellt. Die Zusammenfassung der Ergebnisse samt der berechneten Lösung (Knoten des Graphen und Kanten der Tour) finden Sie in der Ergebnis-Datei im ausgewählten Speicherort.

Erläuterung der Parameter

In diesem Abschnitt werden die einzelnen Parameter der Algorithmen erläutert und ihre Wertebereiche definiert.

Parameter	Wertebereich	Beschreibung
Tmax	$\{t \in R \mid t \geq 0\}$	Ein gemeinsamer Parameter, der vom Orientierungsproblem selbst stammt, ist die Kostenobergrenze Tmax. Sie beschreibt, welche Kosten eine zulässige Tour nicht überschreiten darf und unter welcher der Profit maximiert werden soll. In der Anwendung wird Tmax nach dem Laden eines Graphen oder dem Betätigen des „Parameter zurücksetzen“-Buttons auf den Wert gesetzt, der wenigstens die Verbindung zwischen Start- und Endknoten in einer Tour erlaubt. Für eine sinnvolle Lösung sollte der Wert entsprechend erhöht werden, um das Hinzufügen weiterer Knoten zu erlauben.
Max. Laufzeit	$\{z \in \mathbb{N} \mid z \geq 0\}$	Ermöglicht die Beschränkung der maximalen Laufzeit eines Algorithmus in Minuten. Dieser Wert ist als Richtwert zu verstehen, der verhältnismäßig leicht überschritten werden kann. Dies ist daher notwendig, weil zur Terminierung eines Algorithmus noch finalisierende Methoden ausgeführt werden, um die Gültigkeit der Lösungen gewährleisten zu können. Wird der Wert auf '0' gesetzt, so wird die Laufzeit nicht beschränkt.

Parameter des Adaptive Large Neighbourhood Search (ALNS) Algorithmus

Parameter	Wertebereich	Beschreibung
Iterationen	$\{i \in \mathbb{N} \mid i \geq 0\}$	Beim ALNS-Algorithmus wird zunächst eine initiale Lösung erzeugt. Anschließend werden aus dieser Lösung iterativ Kanten mithilfe von Zerstörmethoden entfernt und mit Reparaturmethoden neue Kanten hinzugefügt. Ist die entstandene Lösung besser als die Vorangegangene, so wird sie als Grundlage für die nächste Iteration genommen. Je höher der Wert also gewählt wird, desto öfter werden Zerstör- und Reparaturmethoden angewendet und desto größer ist auch die Wahrscheinlichkeit, dass eine bessere Lösung gefunden wird. Dabei erhöht sich aber auch die Laufzeit, weshalb hier eine Abwägung zu treffen ist.
α	$\{\alpha \in \mathbb{R} \mid 0 < \alpha < 1\}$	Dieser Faktor beschreibt, wie aggressiv eine Zerstörmethode sein darf bzw. wie viele Knoten sie entfernen darf. Ist der Faktor hoch gewählt, werden in einer Iteration entsprechend mehr Knoten entfernt. Liegt der Wert nahe bei 1 bedeutet dies, dass fast alle Knoten der Tour (ausgenommen Start- und Endknoten) entfernt werden. Das führt dazu, dass in jeder Iteration im Prinzip eine komplett neue Lösung gebaut wird. Wird der Wert hingegen sehr niedrig gewählt, so führt dies dazu, dass sich die Lösung iterativ kaum verändern kann.
h	$\{h \in \mathbb{R} \mid 0 \leq h \leq 1\}$	Im ALNS Algorithmus wird in jeder Iteration die Wahl einer Zerstör- und Reparaturmethode getroffen. In der vorliegenden Anwendung werden die drei Zerstörmethoden RandomRemove, RandomSequenceRemove und ClusterRemove sowie die vier Reparaturmethoden RandomRepair, PrizeRepair, GreedyRepair und ClusterRepair implementiert. In jeder Iteration wird eine Zerstör- und Reparaturmethode angewendet und gemäß ihrem erzielten Ergebnis bewertet. Der Faktor h beschreibt, wie stark eine Bewertung auf die Gesamtbewertung der Methode eingehen darf. Je höher der Faktor, desto stärker der Einfluss auf die Gesamtbewertung. Mit einer steigenden Gesamtbewertung steigt auch die Wahrscheinlichkeit in einer Iteration per gewichtetem Zufall wieder gewählt zu werden.

Parameter des Evolutionären Algorithmus (EA)

Parameter	Wertebereich	Beschreibung
npop	$\{npop \in \mathbb{N} \mid npop \geq 1\}$	Beim evolutionären Algorithmus wird ein npop großer Pool bzw. Population an Lösungen gebildet und iterativ verändert. Am Ende des Algorithmus wird die beste Lösung der Population als Ergebnis zurückgegeben.
p	$\{p \in \mathbb{R} \mid 0 \leq p \leq 1\}$	Bei der Erzeugung jeder initialen Lösung werden Knoten mit der Wahrscheinlichkeit p zur Lösung hinzugefügt. Wird p auf 1 gesetzt bedeutet dies, dass alle initialen Lösungen alle Knoten des Graphen enthalten.
d2d	$\{d2d \in \mathbb{N} \mid d2d \geq 2\}$	Der evolutionäre Algorithmus führt iterativ einen von in zwei Blöcke aus. In einem Block werden genetische Operationen (SelectParents, Crossover, Mutate) auf die Population angewendet, um neue Lösungen zu finden. Im zweiten Block werden die gefundenen Lösungen optimiert und zu gültigen Lösungen gemacht, die die Kostenobergrenze nicht überschreiten. Der Wert d2d gibt an, wie häufig der zweite Block ausgeführt wird. Wird der Wert auf 2 gesetzt, so führt dies dazu, dass die Blöcke sich iterativ abwechseln. Je höher d2d gewählt wird, desto seltener wird der zweite Block ausgeführt.
ncand	$\{ncand \in \mathbb{N} \mid ncand \geq 1, ncand \leq npop\}$	Wird im Algorithmus der Block mit genetischen Operationen ausgeführt, so werden zunächst zwei Lösungen (Eltern) aus einer zufällig gewählten Untermenge der Population ausgewählt, die anschließend miteinander gekreuzt werden. Der Wert ncand beschreibt die Größe dieser Kandidatenliste und darf nicht größer sein als die Population selbst.