Anleitung

Dies ist die Anleitung für den Programmierteil der Masterarbeit „Evaluation verschiedener Lösungen des Graphisomorphieproblems“. Die Programme sind in Prolog geschrieben.

Initialisierung

Um das Programm zu initialisieren, muss die Datei „isomorph_main.pl“ kompiliert werden. Diese Datei lässt sich im Ordner „Programme“ finden. Der Prologbefehl ist:

?- consult('Verzeichnis//isomorph_main.pl').

Erstellung eines isomorphen Graphen

Um einen isomorphen Graphen zu erstellen, benötigen wir einen Graphen im DIMACS-Format. Der Aufruf sieht so aus:

?- open_dim2('Verzeichnis//dateiname',Graph), create_isomorph(Graph,Graph_iso), write_dim("Verzeichnis2//dateiname2',Graph_iso).

Wichtig ist, dass open_dim2 benutzt wird.

Möchte man für einen gerichteten Graphen einen isomorphen Graphen erstellen, dann benutzen wir den Aufruf:

?- open_dim2('Verzeichnis//dateiname',Graph), create_isomorph_di(Graph,Graph_iso), write_dim("Verzeichnis2//dateiname2',Graph_iso).

Erstellung vieler isomorphen Graphen

Der Aufruf ist:

?- create_isomorph_tests('Verzeichnis','Verzeichnis2).

Wichtig ist, dass in Verzeichnis nur Dateien sind, welche Graphen im DIMACS-Format sind. Nach dem Aufruf sind im Verzeichnis2 die isomorphen Graphen. Die Namen der isomorphen Graphen sind dieselben, wie die der ursprünglichen Graphen.

Kanonische Form

Die kanonische Form wird so gebildet:

?- open_dim('Verzeichnis//dateiname',Graph), kan_e(Graph,Kan_Form).

Wollen wir einen Isomorphismus von zwei isomorphen Graphen finden, dann machen wir das so:

?- open_dim('Verzeichnis//dateiname',Graph), kan_e(Graph,Kan_Form), open_dim('Verzeichnis2//dateiname2',Graph2), kan_e(Graph2,Kan_Form2), connect_colors(Kan_Form,Kan_Form2,Isomorphismus).

Es gibt kan_e, kan_k, kan_g, kan_r3 und kan_n zur Auswahl. Zu beachten ist, dass kan_r3 nur auf kubischen Graphen und kan_n nur auf quadratischen Gitternetzen funktioniert.

Direkte Vergleich

Um zwei Graphen direkt miteinander zu vergleichen benutzen wir folgenden Aufruf:

?- open_dim('Verzeichnis//dateiname',Graph), open_dim('Verzeichnis2//dateiname2',Graph2), isomorph(Graph,Graph2,Isomorphismus).

Wobei hier isomorph/3 nur iso_e/3 aufruft.

Wir haben wieder mehrere Verfahren zur Auswahl: iso_e, iso_k, iso_g, iso_v, iso_u und iso_r3.

Auch hier gilt, dass iso_r3 nur auf kubischen Graphen funktioniert.

Überprüfung der Korrektheit

Falls wir wirklich sichergehen wollen, dass ein Isomorphismus auch wirklich gefunden wurde, dann können wir das überprüfen mit:

?- isomorphism_test(Graph,Graph2,Isomorphismus).

isomorphism_test/3 transformiert Graph zu Graph2 mit den Isomorphismus. Ein wichtiger Hinweis noch, isomorphism_test/3 ist nicht optimiert und kann bei größeren Graphen (mehrere Hundertausend Kanten) sehr lange dauern.

Gerichtete Graphen

Wollen wir einen Graphen als gerichteten Graphen haben, dann müssen wir den Graphen mit open_dim_di('Verzeichnis//dateiname',Graph) einlesen.

Testen von vielen Knoten

Um für eine größere Menge an Graphen die Isomorphie zu finden gibt es den Befehl:

?- testing_iso_e(Verzeichnis1,Verzeichnis2,'Verzeichnis//Tabelle.csv').

Im Verzeichnis1 sollten sich die gleichnamigen Graphen befinden wie in Verzeichnis2. In der Tabelle befinden sich die Zeiten der Isomorphiefindung. Außerdem sind die Anzahl der Knoten und Kanten angegeben.

Es wird jeder gefundene Isomorphismus getestet mit isomorphism_test/3. Das kann den gesamten Prozess unnötig verzögern. Notfalls muss man diesen manuell entfernen.

Beispiel eines Aufrufs

Um ein konkretes Beispiel zu geben:

?-
open_dim('Data//Isomorph//random_10//ran10_1000_a.bliss',X),open_dim('Data//Isomorph2//rando

m_10//ran10_1000_a.bliss',Y),kan_k(X,XF), kan_k(Y,YF),connect_colors(XF,YF,R),
isomorph_function_test(X,Y,R).