Leistungsanalyse d. Graphenanlyseprogramms ‚Musical Spoon‘

1. Einleitung

Zu den großen Problemen der computergestützten Suche nach Graph- bzw. Subgraphisomorphismen gehört die oft exponentielle Wachstumsrate der Komplexität in Relation zur Größe, d. h. insbesondere der Knotenzahl der betrachteten Graphen. Tatsächlich gehört das Problem der Subgraphisomorphie zur Klasse der NP-vollständigen Problemen. Die Programmbibliothek ‚Musical Spoon‘ implementiert hauptsächlich zwei Algorithmen zur vergleichenden Analyse von Graphen. Der erste ist eine Variante des bekannten *Bron-Kerbosch-Algorithmus* zur Detektion maximaler Cliquen und über das modulare Produkt indirekt zum Auffinden von Graph-Subgraph-Isomorphismen einzusetzen. Weiterhin wurde der von Cordella et al. vorgestellte laufzeitoptimierte Algorithmus zur Detektion vollständiger Isomorphismen zwischen Graphen implementiert. Zur besseren Vergleichbarkeit in Bezug auf Rechenleistung und Systemlast wurden in diesem Bericht die Ergebnisse einiger vorgenommener Tests mit randomisierten Graphen zusammengefasst.

2. Laufzeitanalyse

Unter Verwendung des in der Programmbibliothek enthaltenen *random_graph* Moduls wurden für jeden Testzyklus Sets von 10 Graphen mit jeweils fixer Knotenzahl generiert und mit dem zu untersuchenden Algorithmus auf Isomorphie überprüft. Dabei wurde bei jedem Zyklus die Anzahl der Knoten jedes Graphen um 1 erhöht. Es wurde in jedem Zyklus jeder Graph mit allen anderen gematcht, womit sich pro Zyklus 45 Läufe des betrachteten Algorithmus ergaben. Die aufgeführten Laufzeiten berücksichtigen dabei jeweils die komplette Laufzeit eines Zyklus.

Das verwendete Testsystem wies folgende Spezifikationen auf:

Betriebssystem: Windows 10 (64-bit)
CPU: Intel Core i7-8700, 3.20 Ghz
RAM: 16 Gigabyte, DDR4-3200 Mhz

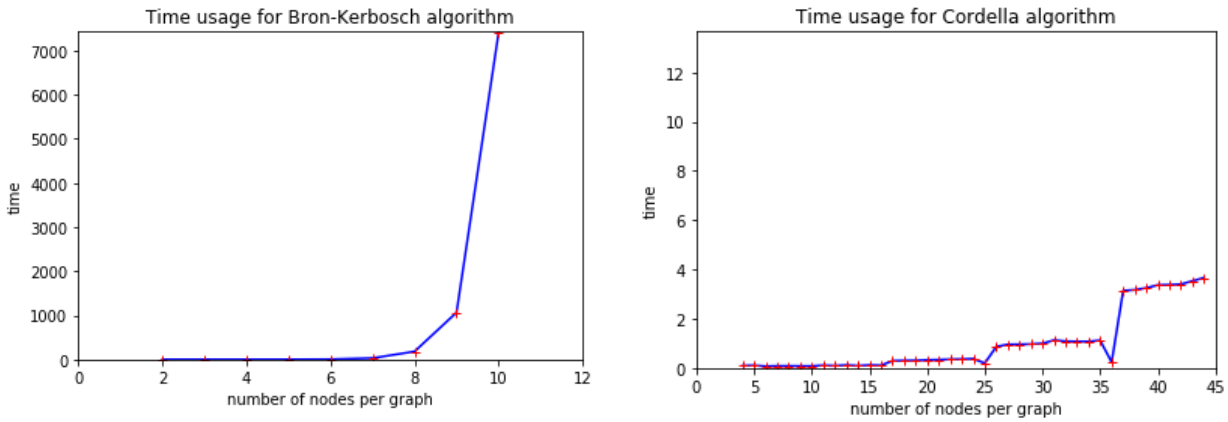

Abbildung 1: Gegenüberstellung der Laufzeit der Implementierung des Bron-Kerbosch Algorithmus mit der des laufzeitoptimierten Cordella-Algorithmus. Die Datenpunkte beziehen sich jeweils auf die summierte Laufzeit eines vollständigen Matching-Zyklus mit der angegebenen Anzahl an Knoten. Die Zeit wurde in Sekunden (s) angegeben.

Der Verlauf der Zeit zeigt einen distinkt exponentiellen Verlauf für den Bron-Kerbosch-Algorithmus, was den Erwartungen entspricht. Ab 12 Knoten betrug die Laufzeit für einen Zyklus länger als 24 Stunden, weshalb der Testlauf an dieser Stelle unterbrochen wurde. Da ein Zyklus sich aus 45 individuellen Programmläufen zusammensetzte, betrug die Laufzeit für einen Programmlauf für diese Eingabe etwa 32 Minuten. Mit dem hier verwendeten System kann eine Anwendung des Bron-Kerbosch-Algorithmus für die Detektion von Graph-Subgraph Isomorphie in einzelnen Graphenpaaren bis etwa 14 Knoten als umsetzbar gesehen werden.

Die Laufzeit des Cordella-Algorithmus zeigt einen näherungsweise linearen Verlauf und erlaubt die Untersuchung von Graph-Graph-Isomorphismen für wesentlich größere Graphen. Zu Beachten ist hierbei, dass die Verbesserung in der Laufzeit wesentlich durch die Fähigkeit des Algorithmus bedingt wird, frühzeitig matching-Runden zu beenden, in denen ein Isomorphismus bereits ausgeschlossen wurde, sowie durch die gezielte Einschränkung des Suchraums.
Da der Cordella-Algorithmus allerdings lediglich zwei Graphen auf vollständige Isomorph prüft, ist er nicht direkt zur Identifikation von Subgraphen anwendbar. Aufgrund der erheblich besseren Laufzeit könnte es allerdings in gewissen Situationen strategischer sein, für einen gegebenen Graphen alle induzierten Subgraphen einer bestimmten Größe zu bestimmen, und diese dann auf Isomorphie mit dem Zielgraphen zu testen. Ob die Komplexität dieses Problems kleiner oder größer als die der Graph-Subgraph-Isomorphie ist, hängt von den analysierten Graphen ab.

3. Speicheranalyse und CPU-Nutzung

Bei den obigen Läufen wurde auch der von der als Umgebung verwendeten Python IDE angeforderte Arbeitsspeicher beobachtet. Da sich die Filterung nach einzelnen Prozessen und deren Speicherverbrauch als äußerst unzuverlässig erwies, wurden im Folgenden nur die Änderungen der Gesamtspeichernutzung der Applikation berücksichtigt. Je nach Genauigkeit der von Windows 10 geführten Speicherdokumentation können diese Werte von der tatsächlichen Arbeitsspeicherlast abweichen. Insbesondere ist hervorzuheben, dass bei keinem der Algorithmen nennenswerte Fluktuationen der Arbeitsspeichernutzung zu beobachten waren. Dies kann einerseits auf die im Vergleich zu in der Forschung üblichen Projekten doch sehr klein gehaltenen Testgraphen zurüchzuführen sein, oder es liegt ein technisches Problem bei der Erfassung der Speichernutzung vor.
Die CPU-Nutzung wurde ebenfalls betrachtet. Bei Ausführung eines der Algorithmen bewegt sich die Auslastung durch die Python-IDE (Ruhewert: <0.1%) im Intervall zwischen 12.6% und 14.2% der möglichen Gesamtleistung. Dieser Wert konnte nicht mit der Anzahl der Knoten in Relation gebracht werden und ist hier deshalb nicht weiter aufgeführt. Da in der Programmbibliothek kein Multithreading implementiert wurde, handelt es sich hierbei um die Maximalrechenleistung, die von einem verwendeten CPU-Kern zur Ausführung dieser Algorithmen erbracht werden kann.

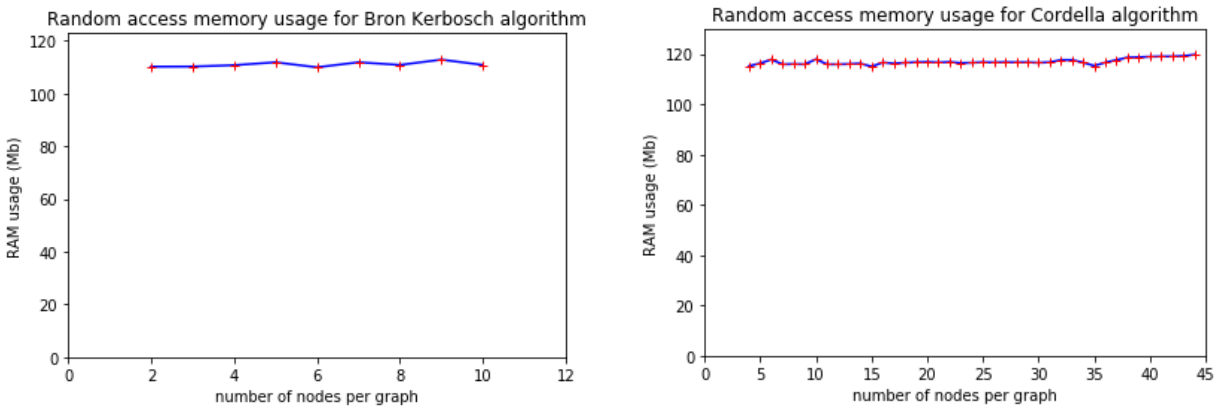

Abbildung 2: Gegenüberstellung des beobachteten Arbeitsspeichergebrauchs für den Bron-Kerbosch-Algorithmus und den Cordella-Algorithmus. Die Datenpunkte beziehen sich jeweils auf die summierte Laufzeit eines vollständigen Matching-Zyklus mit der angegebenen Anzahl an Knoten.