Graph Layout Problems Parameterized by Vertex Cover

Ziel ist es gemeinsam das Papier Graph Layout Problems Parameterized by Vertex Cover zu lesen.

Betrachtete Probleme

Es werden vier Probleme betrachtet, bei denen einen Permutation für die Knoten eines Graphen gefunden werden muss. Dabei wird eines der folgenden vier Maße minimiert.

- **Imbalance:** Die Inbalance eines einzelnen Knotens ist die absolute Differenz zwischen der Anzahl linker und der Anzahl rechter Nachbarn. Die Inbalance der Permutation ist die Summer dieser Inbalances über alle Knoten.
- **Cutwidth:** Eine Permutation definiert $n - 1$ Schnitte: für jedes $1 \leq i < n$ trennt man die ersten i Knoten von den restlichen Knoten. Die Cutwidth der Permutation ist das Maximum dieser Schnitte.
- **Bandwidth:** Länge der längsten Kante.
- **Distortion:** Für eine einzelne Kante uv ist die Distortion der kürzeste Weg den man erhalten kann, wenn man im Graphen von u zu v läuft und dabei alle Knoten zwischen u und v in der von der Permutation gegebenen Reihenfolge besucht. Die Distortion der Permutation ist dann das Maximum über alle Kanten.

Allgemeiner Lösungsansatz

Die Probleme werden auf das Lösen eines LPs reduziert, wobei die Anzahl Variablen nur von der Größe k des minimalen Vertex Covers abhängt.

Dazu macht man zunächst folgende Kernbeobachtungen:

- Es gibt nur $k!$ viele Ordnungen für die Knoten des Vertex Covers. Man kann diese also alle durchgehen und muss das LP nur für eine feste Ordnung aufstellen.
- Die restlichen Knoten lassen sich durch ihre Nachbarschaft charakterisieren. Da sie nur Nachbarn im Vertex Cover haben gibt es auch nur 2^k verschiedene Nachbarschaften. Wir erhalten also 2^k Typen von Knoten und zwei Knoten desselben Typs sind nicht unterscheidbar.
- Statt für jeden einzelnen Knoten eine Position zu suchen, bestimmt man stattdessen, wie viele Knoten welchen Typs wo liegen. Damit das funktioniert muss man die Anzahl von möglichen "wo"s beschränken. Wenn es beispielsweise nur relevant ist, welche Knoten zwischen zwei Knoten des Vertex Covers liegen, ihre Reihenfolge aber egal ist, dann gibt es nur $k + 1$ viele "wo"s.

Imbalance: Ordnung ist egal

Kernbeobachtung: Betrachte die Knoten zwischen zwei aufeinanderfolgenden Knoten aus dem Vertex Cover. Ändert man die Reihenfolge dieser Knoten, so ändert sich die Inbalance nicht, denn die Knoten sind untereinander nicht verbunden und die relative Lage zu allen Knoten aus dem Vertex Cover bleibt erhalten.

Damit muss man nur noch bestimmen, wie viele Knoten welchen Typs zwischen welchen VC-Knoten liegen. Das sind die Hauptvariablen in dem LP. Basierend darauf kann man die Inbalance jedes Knotens ausrechnen, wobei man für die VC-Knoten dafür noch extra variablen braucht.

Da man für die Inbalance eigentlich Beträge betrachten muss, erhält man so erstmal keine linearen Gleichungen. Allerdings kann man das Vorzeichen der Differenzen “raten” (sprich: alle Kombinationen durchprobieren), da es nicht zu viele gibt.

Beachte: Die Inbalances, die für die nicht-VC-Knoten berechnet werden beinhalten ebenfalls Beträge. Warum ist das nicht schlimm? Weil diese Inbalances bei der Aufstellung des LPs bekannt und damit konstant sind. Daher ist auch das Produkt in der Summe der Minimierungsfunktion unproblematisch.

Beachte: Ich glaube, dass die Formulierung der Inbalance für VC-Knoten nicht ganz sauber ist. Da werden die Kanten zu anderen VC-Knoten und zu nicht-VC-Knoten getrennt betrachtet und jeweils der Betrag genommen. Eigentlich darf man nur einmal am Ende den Betrag nehmen. Das kann man aber leicht fixen. Das heißt man rät wie gehabt, in welche Richtung die Inbalance jedes VC-Knotens geht und fordert dann, dass die Inbalance ≥ 0.

Cutwidth: Halte Ordnung fest

Kernbeobachtung: Für die Knoten zwischen zwei VC-Knoten gibt es einen kanonische Ordnung, die nicht schlechter ist als jede andere Ordnung. Wir können also diese Ordnung annehmen. Damit muss man wie oben nur noch bestimmen, wie viele Knoten eines Typs zwischen welchen VC-Knoten liegen.

Diese Ordnung kann wie folgt bestimmt werden. Betrachte die Knoten zwischen zwei aufeinanderfolgenden VC-Knoten. Für jeden Knotentyp kann man dann schauen, wie viele Kanten nach vorne und wie viele nach hinten er hat. Je mehr nach vorne und je weniger nach hinten, desto weiter vorne in der Ordnung sollte er stehen.

Um das zu formalisieren wird der *Rang* definieren als die Anzahl Kanten nach vorne minus die nach hinten. Dieser Rang eines Knotens v gibt einem dann die Differenz zwischen den Schnittgrößen direkt vor v und direkt nach v.

Die Knotentypen zwischen zwei VC-Knoten werden dann immer nach Rang sortiert. Außerdem kann man die Schnittgröße an der Stelle i so durch die Summe der Ränge der vorherigen Knoten berechnen.

Das Maximum über alle diese Schnittgrößen bildet man dann mit einer Hilfsvariable.

Bandwidth: Einfache Struktur

Kernbeobachtung: Die Knoten zwischen zwei VC-Knoten lassen sich in ζ Zonen unterteilen, wobei jede Zone nur Knoten desselben Typs beinhaltet. Man kann dann zeigen, dass die Anzahl nötiger Zonen nur von k abhängt (Beweis nicht im Papier). Damit kann dann alle möglichen Kombinationen von Zonen-Anzahlen durchgehen (hängt nur von k ab).

Alles weitere ist sehr ähnlich zu den Ansätzen zuvor.

Distortion: Mehr oder weniger einfache Struktur

Kernbeobachtung: Hier benutzt man die gleiche Kernbeobachtung wie für Bandwidth, wobei die Anzahl nötiger Zonen deutlich höher ist (aber weiterhin nur von k abhängt).

Zusätzlich gibt es noch ein bisschen Bonusinfo zur Distortion.