Symmetrisches Zeichnen von Graphen in drei Dimensionen

Sven Schaust

28. Mai 2002

Ausblick: Es soll das Problem der symmetrischen Darstellung von Graphen in drei Dimensionen (3D) betrachtet werden. Beginnend mit einer allgemeinen Beschreibung des Problems, folgt die Vorstellung eines Algorithmus für zwei-zusammenhängende planare Graphen, wobei hier besonders auf die Datenstruktur (SPQR-Baum) eingegangen wird.

1 Grundlegendes

- Rotation: rotieren um eine Achse
- Spiegelung(Reflexion): spiegeln an einer Ebene
- Inversion: spiegeln an einem Punkt
- Rotations Spiegelung: Komposition einer Rotation und einer Spiegelung

Es gibt drei Arten von endl. Rotations-Gruppen:

- zyklische Gruppe (C_n): $|C_n| = n$
- Dieder Gruppe (D_n): $|D_n| = 2n$
- platonische Körper (Würfel, Oktaeder,Dodekaeder, ...)

1.1 geforderte Eigenschaften des Graphen

- keine zwei Knoten befinden sich am selben Punkt
- keine zwei Kanten überlappen sich
- kein Knoten liegt auf einer Kante, zu der er nicht inzident ist

1.2 Symmetriegruppe

Die Symmetrie einer 3D Darstellung D eines Graphen G wird durch eine Permutation seiner Knotenmenge gegeben. Diese Permutation ist ein Automorphismus des Graphen G. Die symmetrische Darstellung D gibt also einen Automorphismus des Graphen wieder.

Lemma Die Symmetriegruppe einer 3D Darstellung D eines Graphen G erzeugt eine Untergruppe der Automorphismengruppe von G. Die Untergruppe P ist eine 3D Symmetriegruppe $\Leftrightarrow$ $\exists$ eine 3D Darstellung von G, welche jedes Element aus P als Symmetrie darstellt.

Theorem 1 [6] Das Problem, einen allgemeinen Graphen G symmetrisch im dreidimensionalen darzustellen, gehört in die Menge der NP-schweren Probleme.
$3DSGP$:

- *Eingabe:* Graph G
- *Ausgabe:* 3D Symmetriegruppe P von G mit maximaler Größe

Daraus wurde ein Unterproblem abgeleitet, welches NP-vollständig ist. Es handelt sich um die Frage, ob es zu einem Graphen G eine Symmetriegruppe gibt, welche mindestens die Größe drei hat.

2 planare Graphen

Schränkt man sich auf planare Graphen ein, so kann man $3DSGP$ in polynomieller Zeit lösen. Dazu muss für einen Graphen G zusätzlich gelten:

- zwei nichtadjazente Kanten schneiden sich nicht
- keine zwei ungerichteten Kreise bilden eine nicht-triviale Verbindung

2.1 Unterteilung von $3DSGP$

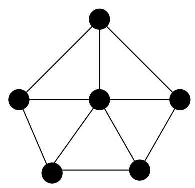
Das Problem wird in drei Fälle von Zusammenhang unterteilt:

- G ist dreifach-zusammenhängend
- G ist zweifach-zusammenhängend
- G ist zusammenhängend

3 Fall 1: Drei-Zusammenhang

Dies ist der einfachste Fall. Hierbei wird der Automorphismus genutzt um eine symmetrische 3D-Darstellung zu finden. Dies basiert auf einem Theorem von P.Mani.

Theorem 2. [5] Jeder drei-zusammenhängende planare Graph G kann als 1-Gerüst(*skeleton*) eines konvexen Polytopes P im $\mathcal{R}^3$ erzeugt werden, so dass alle Automorphismen von G durch Isometrie von P erzeugt werden.



Mittels dieses Theorems findet man eine 3D Symmetriegruppe in der Automorphismengruppe von G.

4 Fall 2: Zwei-Zusammenhang

Ein zweifach-zusammenhängender Graph wird in drei-Zusammenhangskomponenten zerlegt. Diese Komponenten werden in einem SPQR-Baum verwaltet. Der Baum wird, von den Blättern ausgehend, Ebene für Ebene durchlaufen. Dabei werden für jeden Knoten v eine Reihe von Symmetrieeigenschaften (*Marken*) berechnet und der zugehörigen virtuellen Kante im Elternknoten zugewiesen. Danach wird v gelöscht. Dies wird solange durchgeführt, bis man an der Wurzel angelangt ist.

Definition Das Zentrum eines Graphen G ist ein Knoten v (oder eine Kante e) von G, welche den minimalen Maximalabstand zu allen Knoten besitzt. Das Zentrum eines Graphen ist eindeutig.

Das Zentrum des Graphen G wird als Wurzel des SPQR-Baumes gewählt. Damit ist nun auch der SPQR-Baum eindeutig. Mit Hilfe der *Marken* kann eine maximal symmetrische 3D Gruppe im Zentrum berechnet werden.

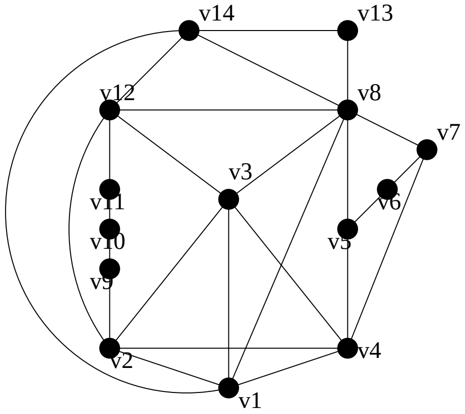
4.1 Datenstruktur für zweifach-zusammenhängende Graphen

Um zweifach-zusammenhängende Graphen G symmetrisch zeichnen zu können, benötigt man eine Datenstruktur, welche es erlaubt drei-zusammenhängende Komponenten von G zu speichern. Dies geschieht mittels des SPQR-Baumes.

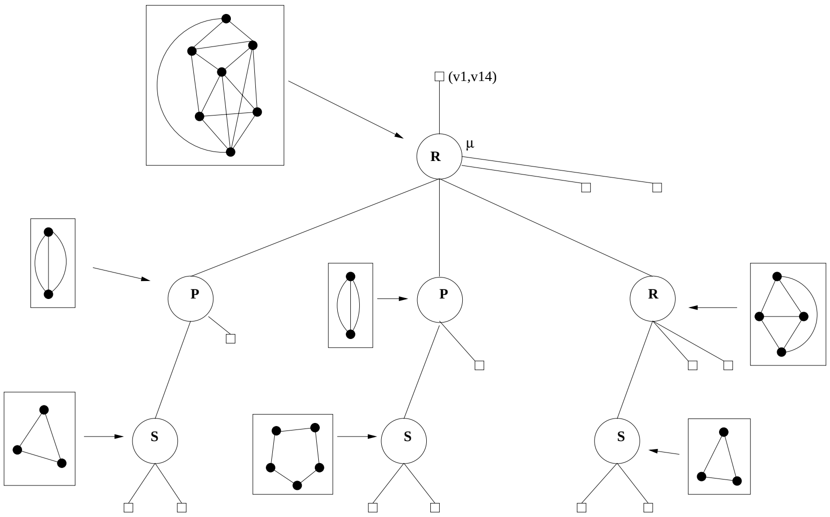
Zur Erinnerung: Ein Graph G ist k-zusammenhängend, wenn es k knotendisjunkte Wege zwischen je zwei Knoten aus G gibt. Eine 1-trennende Menge wird *Schnittknoten* genannt, eine 2-trennende Menge *Separationspaar*.

Zerlegung von G: Wenn G dreifach-zusammenhängend ist, ist G die einzige Komponente. Andernfalls, sei (u, v) ein Separationspaar in G. Partitioniere die Kanten von G in zwei disjunkte Mengen E_1 and $E_2(|E_1|, |E_2| \geq 2)$, so dass die dadurch erzeugten Untergraphen nur die Knoten u und v gemeinsam haben. Diese Zerlegung wird rekursiv auf $G_1' = G_1 + (u, v)$ und $G_2' = G_2 + (u, v)$ fortgesetzt, bis keine Zerlegung mehr möglich ist. Die dabei entstandenen Graphen sind nun entweder drei-zusammenhängend, eine Menge von dreifach Kanten(*triple bond*) oder ein Kreis der Länge 3 (Dreieck). Die beiden letzten Arten sind nicht dreifach-zusammenhängend. Um eindeutige dreifach-zusammenhängende Komponenten zu erhalten müssen diese "Trennungskomponenten" teilweise zusammengefasst(*mergen*) werden. Das Ergebnis ist ein Baum der nur noch eindeutige dreifach-zusammenhängende Komponenten enthält.

Beispiel Graph G:



SPQR-Baum:



Sei (v_1, v_{14}) *Referenzkante* in G. Dann gibt es folgende maximale Separationspaare:

- Kanten (v_1, v_4),(v_1, v_2),(v_2, v_4),(v_1, v_3),(v_4, v_3), (v_2, v_3),(v_1, v_8),(v_3, v_8),(v_3, v_{12}), (v_{12}, v_8),(v_{12}, v_{14}). Jede Kante ist Teil eines Q-Knotens, welcher Kind von μ ist.
- $\{v_4, v_8\}$, bilden die *Pole* des R-Kindes von μ.
- $\{v_8, v_{14}\}$, bilden die *Pole* eines P-Kindes von μ.
- $\{v_2, v_{12}\}$, bilden die *Pole* eines weiteren P-Kindes.

Das Gerüst von μ entsteht aus den maximalen Separationsmengen, wobei für eine Menge $\{s_i, t_i\}$ der Untergraph G_i durch eine Kante $e_i = (s_i, t_i)$ ersetzt wird. Diese Kante nennt man *virtuelle Kante*.

Definition Sei μ ein Knoten von $\mathcal{T}$:

- wenn μ ein R-Knoten ist, dann ist sein Gerüst (*skeleton*(μ)) ein drei-zusammenhängender Graph mit mind. 4 Knoten.
- wenn μ ein S-Knoten ist $\Rightarrow$ skeleton(μ) ist ein Kreis mit mind. 3 Knoten.
- wenn μ ein P-Knoten ist $\Rightarrow$ skeleton(μ) besteht aus mind. zwei Knoten, welche durch mind. 3 Kanten verbunden sind.
- wenn μ ein Q-Knoten ist $\Rightarrow$ skeleton(μ) besteht aus zwei Knoten, welche durch zwei Kanten verbunden sind.

4.2 Algorithmus im Detail

$3DBiconnected\_Planar$

1. Construct the SPQR-tree T_1 of G, and root T_1 at its center.
2. *Reduction:* For each level i of T_1 (from lowest to root level)
   (a) For each leaf node on level i, compute labels
   (b) For each leaf node on level i, label the corresponding virtual edge of the parent node with the labels
   (c) Remove the leaf nodes on level i.
3. Compute a maximum size three dimensional symmetry group at the labeled center.

4.3 Reduktionsschritt

Theorem 3. Das Zentrum eines SPQR-Baumes wird durch eine 3 dimensional symmetrische Gruppe eines zweifach-zusammenhängenden planaren Graphen fixiert.

Die Reduktion schreitet von den äußersten Knoten Richtung Zentrum, wobei sie auf jeder Ebene die Blätter löscht. Der Reduktionsschritt verringert nicht die Größe der Symmetriegruppe.
Aber: die Symmetriegruppe darf auch nicht größer werden! Um dies zu gewährleisten werden *Marken* verwendet. Die Informationen eines Blattes werden genutzt um sicherzustellen, dass jede 3D Symmetrie des reduzierten Graphen die Symmetrie des Original-Graphs erhält.

4.4 Beschriften der virtuellen Kanten (Marken) von R-Knoten

Ziel ist es die virtuellen Kanten derart zu beschriften, so dass man eine maximale 3D symmetrische Gruppe des Originalgraphs erhält, indem man die symmetrische Gruppe des *markierten* Zentrums berechnet (geschieht durch Überprüfung ob eine *Marke* die Symmetrie erhält oder nicht).

Defintion Sei v ein innerer Knoten von T_1. Die virtuelle Kante e von *skeleton(v)* heisst virtuelle Eltern(Kind) Kante, wenn es eine zu e korrespondierende virtuelle Kante in u gibt, welcher ein Eltern(Kind) Knoten von v ist.

Definition Das *Eltern Separationspaar* $s = (s_1, s_2)$ von v ist definiert als die beiden Endpunkte der virtuellen Kante e.

Definition Beim Berechnen der *Marken* von v wird die virtuelle Eltern-Kante e von *skeleton*(v) gelöscht. Der daraus resultierende Graph wird als *skeleton*$^-$*(*v*)* bezeichnet.

4.4.1 Marken

Seien $v_1, V_2, v_3, \ldots, v_k$ Knoten von T_1, die in einem Reduktionsschritt gelöscht werden. Es gibt korrespondierende Kanten $e_1, e_2, \ldots, e_k$ in der darüberliegenden Ebene. Für jede Kante e_i wird nun folgendes bestimmt:

- Isomorphiecode: [2] ein Paar $Iso(e_i)$ von Integerzahlen: Jede isomorphe Komponente besitzt die gleichen Werte.
- Rotationscode: eine Liste L_i, welche die Größe der möglichen Rotationsgruppen von $skeleton^-(v_i)$ angibt, die das Eltern Separationspaar fixieren.
- Reflexionscode:
  - $Ref_{swap}(e_i)$: ein Boolscher Wert der angibt, ob $skeleton^-(v_i)$ eine Reflexionssymmetrie besitzt, welche das Eltern-Separationspaar vertauscht.
  - $Ref_{fix}(e_i)$: ein Boolscher Wert der angibt, ob $skeleton^-(v_i)$ eine Reflexionssymmetrie besitzt, welche das Eltern-Separationspaar fixiert.
- Inversionscode: ein Boolscher Wert $Inv(e_i)$ der angibt ob $skeleton^-(v_i)$ eine Inversion besitzt, welche das Eltern-Separationspaar vertauscht.

Berechnen des Rotationscodes: Es wird eine Liste L_e generiert, wobei $\forall x \in L_e : x = (a, b)$ mit $a, b \in \mathcal{N}$ und $a =$ Größe der Rotationsgruppe und $b =$ Angabe ob Rotationsymmetrie eine feste Kante besitzt.
Die Rotationssymmetrie sollte den Isomorphiecode der virtuellen Kind Kante e beachten. Sollte e fixiert sein, so muss der Durchschnitt der Rotationsgruppen berechnet werden.

1. Compute the list L_e of rotation groups of $skeleton^-(v)$ which fixes the parent separation pair and respects the isomorphism codes of child virtual edges.
2. For each element p of L_e, if there is a child virtual edge e_j in $skeleton^-(v)$ which fixes by p, then compute the intersection of the rotation groups of L_e and L_{e_j}

Dieser Algorithmus nutzt die Vorgehensweise für drei-zusammenhängende Graphen, sofern $skeleton^-(v)$ drei-zusammenhängend ist. Andernfalls wird der Algorithmus $3DBiconnected\_Planar$ rekursiv aufgerufen. Der zweite Schritt kann in linearer Zeit ausgeführt werden (Bit-Array).

Berechnen des Reflexionscodes: Die Berechnung kann in zwei Fälle aufgeteilt werden

1. Reflexion vertauscht das Eltern Separationspaar($Ref_{swap}(e)$):
   $Compute\_Reflection\_Code\_Swap$

(a) Test $skeleton^{-}(v)$ whether it has a reflectional symmetry α which swaps the parent separation pair and respects the isomorphism codes of child virtual edges.

(b) If α exists, then

i. For each child virtual edge e_j that is fixed by α, check followings:

A. if α fixes the endpoints of e_j, then $Ref_{fix}(e_j) = true$

B. if α swaps the endpoints of e_j, then $Ref_{swap}(e_j) = true$

ii. If one of thes properties fails,
then $Ref_{swap}(e) = false$; else $Ref_{swap}(e) = true$.

else $Ref_{swap}(e) = false$

2. Reflexion fixiert das Eltern Separationspaar:
Der Algorithmus ist ähnlich und wurde nicht weiter ausgeführt ...

Berechnen des Inversionscodes: Dieser Algorithmus liefert Informationen darüber ob eine Inversion das Separationspaar vertauscht.

1. Test $skeleton^{-}(v)$ whether it has an inversion β which swaps the parent separation pair and respects the isomorhism codes of child virtual edges.
2. If β exists, then

   If for each child virtual edge e_j that is fixed by β, $Inv(e_j) = true$,
   then $Inv(e) = true$; else $Inv(e) = false$.

   else $Inv(e) = false$.

Die genannten Algorithmen werden in ähnlicher Weise für P- und S-Knoten verwendet.

4.4.2 Finden einer 3D Symmetrie im Zentrum

Das Zentrum eines SPQR-Baumes kann ein Knoten c, als auch eine Kante e sein. Ist das Zentrum ein Knoten, so kann man drei Fälle unterscheiden:

1. c ist ein R-Knoten: Nutze den Algorithmus für dreifach-zusammenhängende Graphen. Dazu konstruieren wir eine 3D Darstellung von $skeleton(c)$ und ersetzen jede virtuelle Kind Kante e_j durch die Darstellung von $skeleton^{-}(v_j)$. Dies wird rekursiv weitergeführt. Die Darstellung von $skeleton^{-}(v_j)$ wird auf eine Ebene gelegt um die Symmetrie zu maximieren.(siehe dazu 4.4.4)

2. c ist ein P-Knoten: [2] es wird ein Algorithmus verwendet, welcher sich an den Algorithmus zur Bestimmung von "parallel composition in series of parallel digraphs" anlehnt.

3. c ist ein S-Knoten: Algorithmus ähnelt dem Algorithmus zur Bestimmung der *Marken* eines Kreises.

Andernfalls ist das Zentrum eine Kante und es gibt folgende Fälle:

- Parallele Komposition (parallel composition)
- Reduktions Komposition (reduction composition)
- Einschließende Komposition (enclosing composition)

parallel composition: [2] Konstruiere eine Darstellung mit zwei *beschrifteten* Kanten ähnlich der Parallel Komposition in "series parallel digraphs"

reduction composition: Konstruiere eine Darstellung durch Berechnen der *label* eines Knotens u (und Löschen), sowie Beschriften der zugehörigen virtuellen Kante am Knoten v ((u, v) ist Kante). Berechne daraufhin die 3D Symmetriegruppe von v.

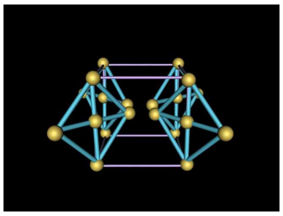
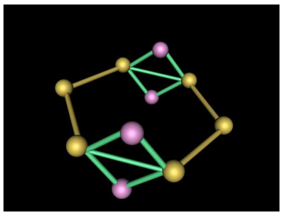
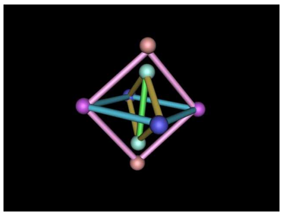
enclosing composition: Konstruieren von Sonderfällen

4.4.3 Laufzeitanalyse

Abschliessend soll noch die Laufzeit des Algorithmus erwähnt werden. Dazu folgendes Theorem:

Theorem 4. $3DBiconnected\_Planar$ benötigt $O(n^2)$ Zeit.

4.4.4 Darstellung von 3D Symmetrien

Literatur

1. Seok-Hee Hong, "Drawing Graphs Symmetrically in Three Dimensions" (extended abstract) [Graph Drawing 2001, Lecture Notes in Computer-Science (LNCS) 2265]

2. Seok-Hee Hong and Peter Eades, "An Algorithm for Finding Three Dimensional Symmetry in Series Parallel Digraphs" [ISAAC 2000, LNCS 1969]

3. G.Di Battista and R. Tamassia, "On-Line Maintenance of Triconnected Components with SPQR-Trees" [Algorithmica (1996) 15]

4. J.Hopcroft and R.E. Tarjan, "Dividing a Graph into Triconnected Components" [SIAM, Journal on Computing 2 von 1973]

5. P.Mani, " Automorphismen von Polyedrischen Graphen" [Math. Annalen (192) von 1971]

6. A. Lubiw, "Some NP-Complete Problems similar to Graph Isomorphism" [SIAM, Journal on Computing 10 von 1981]