Kreuzungsminimierung in Nachbarschaftsgraphen

Bachelorarbeit
von

Patricia Bachmann

An der Fakultät für Informatik und Mathematik
Lehrstuhl für Informatik mit Schwerpunkt Theoretische Informatik

Erstgutachter: Prof. Dr. I. Rutter

Bearbeitungszeit: 19. Dezember 2019 – 16. März 2020

Eigenständigkeitserklärung

Hiermit bestätige ich, Patricia Bachmann, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die wörtlich und sinngemäß übernommenen Passagen aus anderen Werken kenntlich gemacht habe. Die Arbeit ist weder von mir noch von einer anderen Person an der Universität Passau oder an einer anderen Hochschule zur Erlangung eines akademischen Grades bereits eingereicht worden.

Passau, 16. März 2020

Zusammenfassung

Sei $G = (V, E)$ ein Graph mit einer Zeichnung Γ. Wir wählen einen Knoten $v \in V$ als Zentrum. Die Nachbarschaft dieses Zentrums wollen wir optimiert darstellen. Hierzu soll die Anzahl der Kreuzungen des Nachbarschaftsgraphen $G(v) = G[N_G(v)]$ reduziert werden und eine geradlinige Nachbarschaftszeichnung Γ^* von $G(v)$ berechnet werden. Die Nachbarschaftszeichnung Γ^* soll dabei ähnlich zur Ausgangszeichnung Γ sein, weshalb wir die Nachbarn von v nur um einen positiven Faktor entlang einer Verschiebungsgerade durch v bewegen. Zur Minimierung der Kreuzungen dient ein Algorithmus, welcher in linearer Laufzeit eine kreuzungsfreie Nachbarschaftszeichnung Γ^* liefert, sofern G keine alternierenden 3-Kreise enthält. Für Nachbarschaftsgraphen, die alternierenden 3-Kreisen enthalten, wird ein weiterer Algorithmus beschrieben, um diese 3-Kreise aufzulösen, damit dann der erste Algorithmus angewandt werden kann. Das Auflösen der 3-Kreise läuft in $O(n^2 \log n)$ Zeit. Des Weiteren wurde eine JavaScript Anwendung implementiert, welche für einen gewählten Knoten eines Graphen seine Nachbarschaft wie beschrieben optimiert darstellt. Anhand einer Evaluation durch praktische Beispiele wurde gezeigt, dass mit steigender Dichte einer Nachbarschaft sowohl die Anzahl von alternierenden 3-Kreisen steigt als auch die Anzahl von Kanten, die gelöscht werden müssten, um diese 3-Kreise aufzulösen. Der Prozentsatz der minimierten Kreuzungen sinkt dabei. Im Durchschnitt kann die Anzahl der Kreuzungen in Nachbarschaftszeichnungen etwa um die Hälfte reduziert werden, während man durchschnittlich 20% der Kanten löschen müsste, um alle alternierenden 3-Kreise aufzulösen.

Inhaltsverzeichnis

1. Einleitung

Graphen sind Strukturen, die zur Visualisierung von Informationsmodellen genutzt werden, welche als eine Menge von Objekten und eine Menge von Beziehungen oder Verbindungen zwischen diesen Objekten beschrieben werden können. Das Zeichnen von Graphen, also das Konstruieren einer geometrischen Darstellung eines Graphen, ist ein wichtiger Teil auf dem Gebiet der Informationsvisualisierung. Es findet Anwendung in vielen unterschiedlichen Bereichen, beispielsweise der Biochemie [8], der Darstellung von Karten [6] und Stromnetzwerken [3], oder der Netzwerkvisualisierung [11, 7].

Somit ist ein naheliegendes und interessantes Problem das Zeichnen von Graphen, sodass sie für Menschen gut lesbar und leicht nachzuvollziehen sind. Es wurden bereits mehrere Algorithmen für verschieden Arten von Graphen untersucht, die bestimmte geometrische Einschränkungen oder Vorgaben beim Zeichnen eines Graphen einhalten [2, 9]. Dabei wurde unter anderem versucht, die Anzahl der Kreuzungen in geradlinigen Zeichnung zu minimieren.

Zeichnungen von Graphen sind grundsätzlich dazu gedacht, die globale Struktur des Graphen zu visualisieren. Allerdings kann diese Darstellung für die Nachbarschaft eines einzelnen Knoten suboptimal sein, ein Beispiel dafür wird in Abbildung 1.1 dargestellt. Hier entstehen durch die Zeichnung des Graphen in der ausgewählten Nachbarschaft relativ viele Kreuzungen. In Abbildung 1.2 wird eine mögliche Nachbarschaftszeichnung dargestellt, die noch ähnlich zur ursprünglichen Zeichnung ist, aber deutlich weniger Kreuzungen enthält.

In dieser Arbeit wird der Versuch, die geradlinige Zeichnung eines Graphen für Menschen lesbarer zu machen, reduziert auf die verbesserte Darstellung der Nachbarschaft eines Knoten v. In unserem Fall bedeutet das, dass die Anzahl der Kreuzungen in der Nachbarschaftszeichnung minimiert werden sollen. Gleichzeitig soll jedoch die Ähnlichkeit zur ursprünglichen Zeichnung bewahrt werden, weshalb die Knoten der Nachbarschaft nur um einen positiven Faktor von v verschoben werden sollen. Die Winkel zwischen den von v ausgehenden Kanten sowie die Reihenfolge der Knoten um v bleiben unverändert.

Der folgende Abschnitt beschäftigt sich mit den grundlegenden Eigenschaften von Graphen sowie der Beschreibung der Problemstellung. Zudem werden erste Begriffe definiert, die für die späteren Beweisführungen benötigt werden.

In Kapitel 3 wird die Kreuzungsminimierung in Nachbarschaftsgraphen behandelt. Zunächst wird gezeigt, dass für bestimmte Nachbarschaftszeichnungen immer eine ähnliche kreuzungsfreie Nachbarschaftszeichnung existiert. Hierfür wird ein Algorithmus beschrieben,

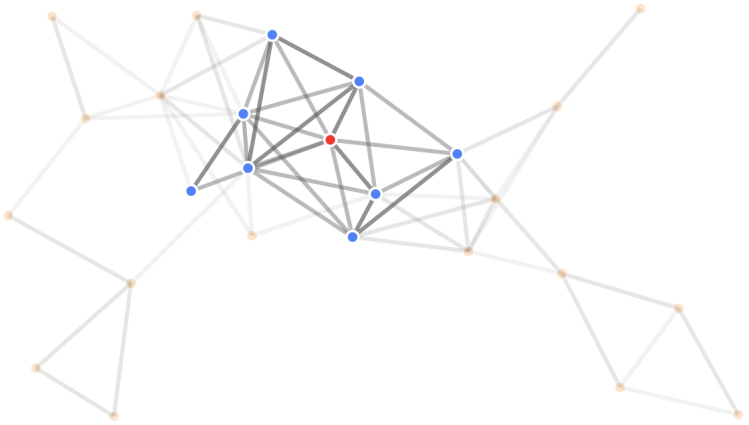


Abbildung 1.1.: Eine Zeichnung eines Graphen mit einer hervorgehobenen Nachbarschaft

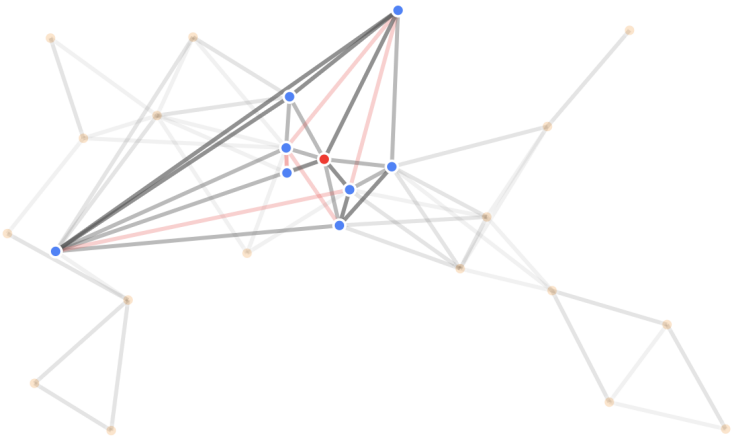


Abbildung 1.2.: Die transformierte hervorgehobene Nachbarschaft. Alle Kreuzungen liegen hier auf roten Kanten.

der diese in linearer Zeit berechnet. Dieser Ansatz wird anschließend auf allgemeine Nachbarschaftszeichnungen durch einen weiteren Algorithmus erweitert. Dieser Algorithmus entfernt schrittweise eine gewisse Anzahl an Kanten, abhängig von ihrem Gewicht. Das Gewicht einer Kante e entspricht dabei der Anzahl von Kanten, die von e gekreuzt werden. Wenn alle Kanten ein Gewicht von 0 erreicht haben, kann der vorherige Algorithmus auf den entstandenen Teilgraphen angewandt werden kann. Die gelöschten Kanten werden dann wieder in die neue Zeichnung eingefügt. Für das Berechnen der zu Löschenden Kanten wird $O(n^2 \log n)$ Zeit benötigt.

Anschließend wird in Abschnitt 4 das Programm beschrieben, welches im Rahmen der Arbeit entwickelt wurde. In diesem Programm können Knoten in einem Graphen angewählt werden, um deren Nachbarschaft optimiert darzustellen. Dazu kennzeichnet das Programm zunächst alle Kanten, die im Zuge des zweiten Algorithmus' entfernt werden würden. Anschließend sollen die Nachbarknoten so verschoben werden, dass sich die übrigen Kanten nicht kreuzen. Die Anwendung wird anhand der Kreuzungsminimierung und der Anzahl von zu Löschenden Kanten im Verhältnis zur Dichte des Graphen ausgewertet und das Ergebnis wird diskutiert.

Die Arbeit wird in Abschnitt 5 abschließend zusammengefasst und es wird sowohl ein Ausblick gegeben als auch eine Übersicht über offene Fragen.

2. Vorbereitung

Sei Ω eine Grundmenge. Ein geordnetes Paar (V, E) mit $V \subseteq \Omega$ und $E \subseteq \binom{V}{2}$ heißt Graph. Dabei ist $\binom{V}{2}$ die Menge der zweielementigen Teilmengen von V. Die Menge V wird als Knotenmenge und E als Kantenmenge bezeichnet. Ist G ein Graph, so setzen wir $V = V(G)$ und $E = E(G)$.

Sei $G = (V, E)$ ein Graph und $v, u \in V$ und $e \in E$. Wir sagen v ist *inzident* mit e, wenn $v \in e$ gilt. Zwei Kanten sind inzident zueinander, wenn sie zu einem gemeinsamen Knoten inzident sind. Die Knoten u und v heißen *benachbart*, *verbunden* oder *adjazent* in G, wenn gilt $\{u, v\} \in E$. Sind zwei Knoten benachbart, so werden sie auch Nachbarn genannt.

Seien G und H Graphen. H heißt *induzierter Untergraph* von G (kurz $H = N_G$), wenn H durch fortgesetztes Löschen von Knoten aus G entsteht, d.h. wenn gilt $V(H) \subseteq V(G)$ und $E(H) = E(G) \cap \binom{V(H)}{2}$

2.1. Problemstellung

Definition 2.1. *Sei $G = (V, E)$ ein Graph mit einer geradlinigen Zeichnung Γ und $v \in V$. Wir nennen $G(v) = G[N_G(v)]$ den* Nachbarschaftsgraph *von v und v das* Zentrum *der Nachbarschaft. Die Ordnung um v ist in $G(v)$ fest. Eine geradlinige Zeichnung Γ^* von $G(v)$ nennen wir* Nachbarschaftszeichnung*, wenn für Γ^* gilt:*

(1) $\Gamma(v) = \Gamma^*(v)$

(2) $\Gamma(u) - \Gamma(v) = c \cdot (\Gamma^*(u) - \Gamma^*(v))$ *für alle $u \in N_G(v) \setminus \{v\}$ wobei $c > 0$*

Die Winkel der von v ausgehenden Kanten bleiben also unverändert. Lediglich der Abstand der Nachbarn zu v darf sich um einen positiven Faktor c unterscheiden.

Definition 2.2. *Sei $G(v)$ ein Nachbarschaftsgraph mit einer geradlinigen Zeichnung Γ. Sei $u \in V$. Die Halbgerade $[vu$ nennen wir die* Verschiebungsgerade *von u.*

Um die Ähnlichkeit zu einer geradlinigen Zeichnung Γ zu bewahren, bilden wir die Nachbarn von v nur auf Punkte ab, die auf der jeweiligen Verschiebungsgerade liegen. Dieses Abbilden eines Knotens u auf einen Punkt u' auf seiner Verschiebungsgerade nennen wir auch das *Verschieben* von u um einen Faktor c, wobei $c = \frac{|vu'|}{|vu|}$. In Abbildung 2.1 wird ein Beispiel dargestellt, in dem zwei Knoten u_1 und u_2 jeweils um einen Faktor $c_1, c_2 > 0$ verschoben werden.

Gesucht ist nun eine Nachbarschaftszeichnung Γ^* mit einer minimalen Anzahl an Kreuzungen.

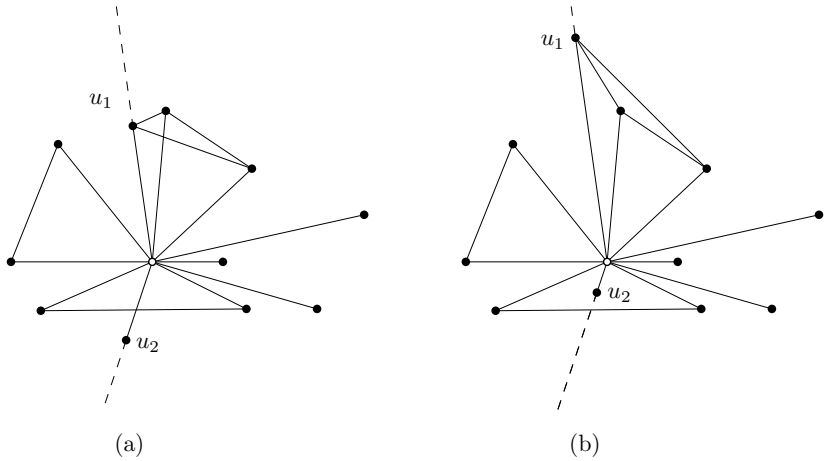


Abbildung 2.1.: Zwei Nachbarschaftszeichnungen eines Nachbarschaftsgraphen $G(v)$. Das Zentrum v ist weiß hervorgehoben. Die Verschiebungsgeraden für u_1 und u_2 sind als gestrichelte Linie dargestellt. (a) Eine Nachbarschaftszeichnung mit Kreuzungen (b) Die Knoten u_1 und u_2 wurden um einen positiven Faktor verschoben. Die Zeichnung wird dadurch kreuzungsfrei.

2.2. Alternierende 3-Kreise

Sei $G = (V, E)$ ein vollständiger Graph mit $|V| = 3$. Einen derartigen Graphen nennen wir im Folgenden einen *3-Kreis*.

Definition 2.3. *Sei $G(v)$ ein Nachbarschaftsgraph mit Zentrum v und $u_1, u_2, w_1, w_2 \in V$. Die Knoten u_1vu_2 und w_1vw_2 sollen dabei jeweils einen 3-Kreis bilden. Wir nehmen zudem an, dass die von v ausgehenden Kanten der 3-Kreise an v alternieren, d.h., dass die Kanten in der Reihenfolge $\{v, u_1\}, \{v, w_1\}, \{v, u_2\}, \{v, w_2\}$ an v geordnet sind. Eine solche Anordnung von 3-Kreisen in $G(v)$ nennen wir* alternierende 3-Kreise.

Für ein Beispiel für alternierende 3-Kreise, siehe Abb. 2.2.

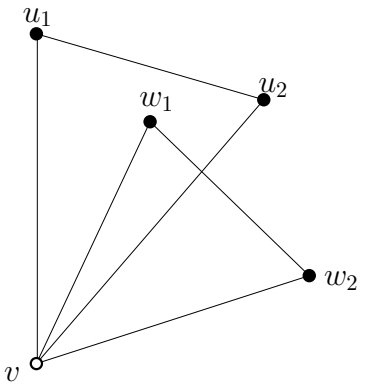


Abbildung 2.2.: Beispiel für alternierende 3-Kreise

Definition 2.4. *Sei $G(v)$ ein Nachbarschaftsgraph mit $u, w \in V(G(v))$. Wenn $\{v, u\}$ und $\{v, w\}$ in der Ordnung um v direkt aufeinander folgen, dann nennen wir u und w* an v benachbart.

3. Kreuzungsminimierung in Nachbarschaftsgraphen

3.1. Alternierende 3-Kreise in Nachbarschaftsgraphen

Lemma 3.1. *Sei $G(v)$ ein Nachbarschaftsgraph. Wenn $G(v)$ alternierende 3-Kreise enthält, dann existiert keine kreuzungsfreie Nachbarschaftszeichnung Γ von $G(v)$.*

Beweis. Sei $G(v)$ ein Nachbarschaftsgraph, der alternierende 3-Kreise enthält. Angenommen, es existiert eine kreuzungsfreie Zeichnung Γ. Dann besteht diese zunächst aus einem 3-Kreis u_1vu_2 mit $u_1, u_2 \in V$. Die Knoten v, w_1 und w_2 mit $w_1w_2 \in V$ sollen nun ebenfals einen 3-Kreis w_1vw_2 bilden, dessen von v ausgehenden Kanten mit denen des 3-Kreises u_1vu_2 an v alternieren. Durch diese Alternierung befindet sich genau eine der Kanten von w_1vw_2 zwischen den Schenkeln vu_1 und vu_2. Ohne Beschränkung der Allgemeinheit sei das die Kante $\{v, w_1\}$ (s. Abb. 3.1). Damit diese keine Kreuzung erzeugt, muss w_1 innerhalb des Dreiecks u_1vu_2 liegen, da sie sonst die Kante $\{u_1, u_2\}$ schneidet. Nun muss w_2 durch die Alternierung außerhalb von u_1vu_2 liegen. Dadurch aber kreuzt $\{w_1, w_2\}$ mindestens eine der Kanten von u_1vu_2. Dies ist ein Widerspruch zur Annahme, dass Γ kreuzungsfrei ist. Somit existiert keine kreuzungsfreie Zeichnung für $G(v)$. □

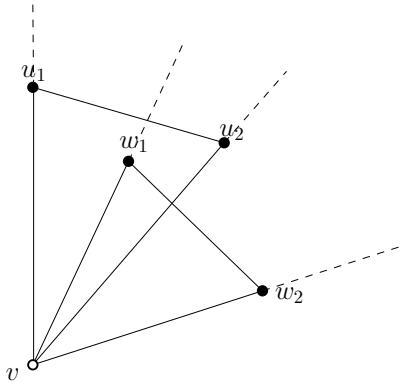


Abbildung 3.1.: Mögliche Verschiebungsgeraden für eine Zeichnung mit alternierenden 3-Kreisen

Lemma 3.2. *Sei $G(v) = (V, E)$ ein beliebiger Nachbarschaftsgraph ohne alternierende 3-Kreise. Angenommen, wir fügen eine Kante $\{u, w\}$ ein und erhalten einen neuen Graphen $G'(v)$. Wenn u und w an v benachbart sind, dann enthält $G'(v)$ keine alternierenden 3-Kreise.*

Beweis. Seien $u, w \in V$ an v benachbart. Angenommen, $G'(v)$ enthält alternierende 3-Kreise. Dann existiert ein 3-Kreis $u'vw'$, dessen von v ausgehenden Kanten mit denen des 3-Kreises uvw an v alternieren. Das bedeutet, dass sich genau eine der Kanten von $u'vw'$ zwischen $\{v, u\}$ und $\{v, w\}$ befindet. Dies ist ein Widerspruch zur Annahme, dass u und w an v benachbart sind. Somit enthält $G'(v)$ keine alternierenden 3-Kreise. □

Lemma 3.3. *Sei $G(v)$ ein Nachbarschaftsgraph. Wenn $G(v)$ keine alternierenden 3-Kreise enthält, dann ist $G(v)$ planar.*

Beweis. Sei $G(v)$ ein Nachbarschaftsgraph, der keine alternierenden 3-Kreise enthält. Zwischen allen an v benachbarten Knoten, die noch nicht über eine Kante verbunden sind, fügen wir Kanten ein und erhalten einen neuen Nachbarschaftsgraphen $G'(v)$. Nach Lemma 3.2 enthält $G'(v)$ keine alternierenden 3-Kreise. Die von v ausgehenden Kanten zusammen mit den Kanten zwischen den an v benachbarten Knoten bilden das *Rad* $W_n = H$ [10] (vgl. Abbildung 3.2 (b)).

Sei nun Γ eine Zeichnung von $G'(v)$. Die Kanten zwischen nicht an v benachbarten Knoten lassen wir in Γ außerhalb von H verlaufen (vgl. Abb. 3.2 (b)). Angenommen, Γ ist nicht kreuzungsfrei. Dann existieren zwei Kanten $\{u_1, u_2\}, \{w_1, w_2\}$, die sich kreuzen. Wir können annehmen, dass H planar ist und kreuzungsfrei gezeichnet werden kann. Alle Kanten, die nicht Teil von H sind, liegen nach Konstruktion außerhalb von H und haben v nicht als einen ihrer Endpunkte. Dann existieren zwei 3-Kreise u_1vu_2 und w_1vw_2. Da $\{u_1, u_2\}$ die Kante $\{w_1, w_2\}$ kreuzt, befindet sich ohne Beschränkung der Allgemeinheit u_1 innerhalb von w_1vw_2 und u_2 außerhalb. Also sind die Kanten in der Reihenfolge $\{v.w_1\}, \{v, u_1\}, \{v, w_2\}, \{v, u_2\}$ an v geordnet, womit wir alternierende 3-Kreise erhalten. Dies ist ein Widerspruch zur Annahme, dass $G'(v)$ keine alternierenden 3-Kreise enthält.

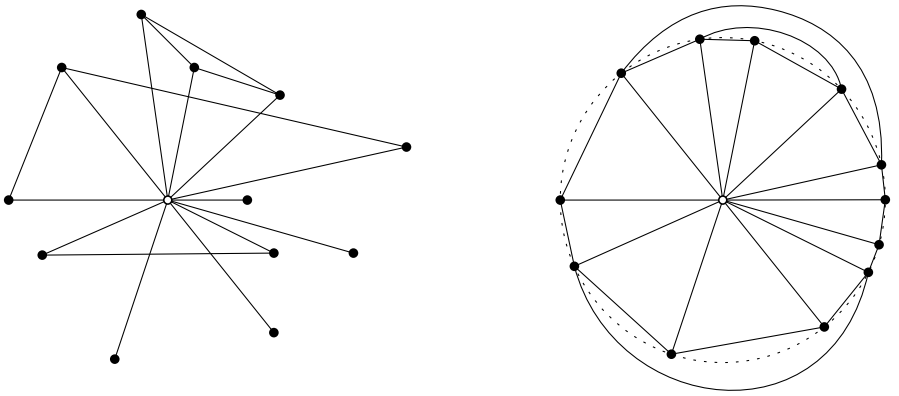
Somit ist Γ kreuzungsfrei. Da W_n und damit insbesondere $G'(v)$ 3-zusammenhängend sind (nach [10]) und dadurch, dass die Ordnung um v fest ist, können wir annehmen, dass Γ eine feste, planare Einbettung ist, womit $G'(v)$ und insbesondere $G(v)$ planar sind. □

Definition 3.4. *Sei $G(v)$ ein Nachbarschaftsgraph ohne alternierende 3-Kreise. Sei $G'(v)$ der Graph, den wir durch das Einfügen von Kanten zwischen an v benachbarten Knoten erhalten. Wir nennen $G'(v)$ den* erweiterten Nachbarschaftsgraphen *von $G(v)$. Eine Einbettung Γ' von $G'(v)$, wie sie in Lemma 3.3 konstruiert wurde, nennen wir eine* Nachbarschaftseinbettung von $G'(v)$ *(vgl. Abb. 3.2 (b)).*

Lemma 3.3 zeigt, dass für jeden Nachbarschaftsgraphen $G(v)$ ohne alternierende 3-Kreise ein planarer erweiterter Nachbarschaftsgraph $G'(v)$ sowie eine planare Nachbarschaftseinbettung Γ' von $G'(v)$ existieren.

Lemma 3.5. *Sei $G(v)$ ein Nachbarschaftsgraph ohne alternierende 3-Kreise. Dann existiert eine geradlinige, kreuzungsfreie Zeichnung Γ' von $G(v)$.*

Beweis. Sei $G(v)$ ein beliebiger Nachbarschaftsgraph, der keine alternierenden 3-Kreise enthält. Nach Lemma 3.3 wissen wir, dass sowohl für $G(v)$ der erweiterte Nachbarschaftsgraph $G'(v)$ existiert, als auch die planare Einbettung Γ' von $G'(v)$. Wir wählen Γ' mit folgenden Eigenschaften:



(a) (b)

Abbildung 3.2.: (a) Zeichnung eines Nachbarschaftsgraphen $G(v)$ ohne alternierende 3-Kreise. Das Zentrum der Nachbarschaft v ist weiß hervorgehoben. (b) Zeichnung von $G'(v)$, Die von v ausgehenden Kanten zusammen mit den Kanten zwischen den an v benachbarten Knoten bilden das Rad H. Die Kanten ziwschen nicht an v benachbarten Knoten verlaufen außerhalb von H.

(1) Alle von v ausgehenden Kanten haben die gleiche Länge.

(2) Alle Kanten $\{u, w\}$ mit $u \neq v$, $w \neq v$ liegen auf der Seite des Graphen, auf der der Winkel zwischen $\{v, u\}$ und $\{v, w\}$ kleiner als 180° ist.

Seien $G'_0(v), G'_1(v), ..., G'_n(v)$ Teilgraphen von $G'(v)$, wobei n die Anzahl der Kanten zwischen nicht an v benachbarten Knoten ist. Sei e_i mit $0 \leq i \leq n$ eine beliebige aber feste Kante der äußeren Facette in $G'_i(v)$ zwischen zwei nicht an v benachbarten Knoten w_i, w'_i. Dann ist $G'_{i-1}(v) = (V, E_{i-1})$, wobei $V = V(G'(v))$ und $E_{i-1} = (E(G'_i(v)) \setminus e_i)$. Es gilt $G'_n(v) = G'(v)$ und $G'_0(v) = H$.

Sei Γ_i die Nachbarschaftseinbettung von $G'(v)$ mit Einschränkung auf $G'_i(v)$.

Wir zeigen induktiv, dass für $G'_i(v)$ eine kreuzungsfreie, geradlinige Zeichnung Γ' existiert.

Induktionsanfang: $i = 0$

Betrachte $G_0(v) = H$. Da H ein Rad ist, besitzt $G_0(v)$ eine kreuzungsfreie, geradlinige Zeichnung Γ'_0 von $G'_0(v)$.

Induktionsannahme: Der Graph $G'_i(v)$ mit Einbettung Γ_i besitzt eine geradlinige, kreuzungsfreie Zeichnung Γ'_i.

Induktionsschritt: $i \to i + 1$

Betrachte $G'_{i+1}(v)$. Nach Induktionsannahme wissen wir, dass $G_i(v)$ eine geradlinige, kreuzungsfreie Zeichnung Γ'_i besitzt. Seien $v_1, v_2, ...u_k$ die Knoten innerhalb der Facette $w_{i+1}vw'_{i+1}$, wobei $v_1 = w_{i+1}$ und $v_k = w'_{i+1}$. Nach Konstruktion von $G'_i(v)$ bilden diese Knoten die Dreiecke $\Delta_{l,l'} = v_l v v_{l'}$ für $1 \leq l < l' \leq k$. Ein Knoten v_l kann dabei einen Eckpunkt von höchstens $k - 1$ Dreiecken bilden.

Sei die Kante $e'_{i+1} = \{w_{i+1}, w'_{i+1}\}$ nun geradlinig in Γ'_{i+1}. Dann bildet diese das Dreieck $w_i v w'_i$. Seien nun $\Delta'_{l,l'}$ Dreiecke mit den Eckpunkten $v'_l v v'_{l'}$ und $1 \leq l < l' \leq k$, wobei v'_l bzw. $v'_{l'}$ sowohl auf der Verschiebungsgerade von v_l bzw. $v_{l'}$ als auch innerhalb des Dreiecks $w_i v w'_i$ liegt. Wir bilden jedes Dreieck $\Delta_{l,l'}$ affin auf das Dreieck $\Delta'_{l,l'}$ ab, wodurch es vollständig und kreuzungsfrei innerhalb des Dreiecks $w_i v w'_i$ liegt (vgl. Abbildung 3.3).

Wir bewahren hierbei die Ähnlichkeit zu Γ, da wir die Nachbarn von v auf Punkte abbilden, die auf den entsprechenden Verschiebungsgeraden liegen. Auch erhalten wir eine geradlinige, kreuzungsfreie Zeichnung Γ'_{i+1}, da alle Knoten $v_2, ..., v_{k-1}$ innerhalb des Dreiecks $w_i v w'_i$ liegen und durch die affinen Abbildungen die Kreuzungsfreiheit aus Γ'_i erhalten bleibt.

Dieses schrittweise Zeichnen führen wir bis $G_n(v)$ durch, wodurch wir eine geradlinige und kreuzungsfreie Zeichnung $\Gamma'_n = \Gamma'$ erhalten.

□

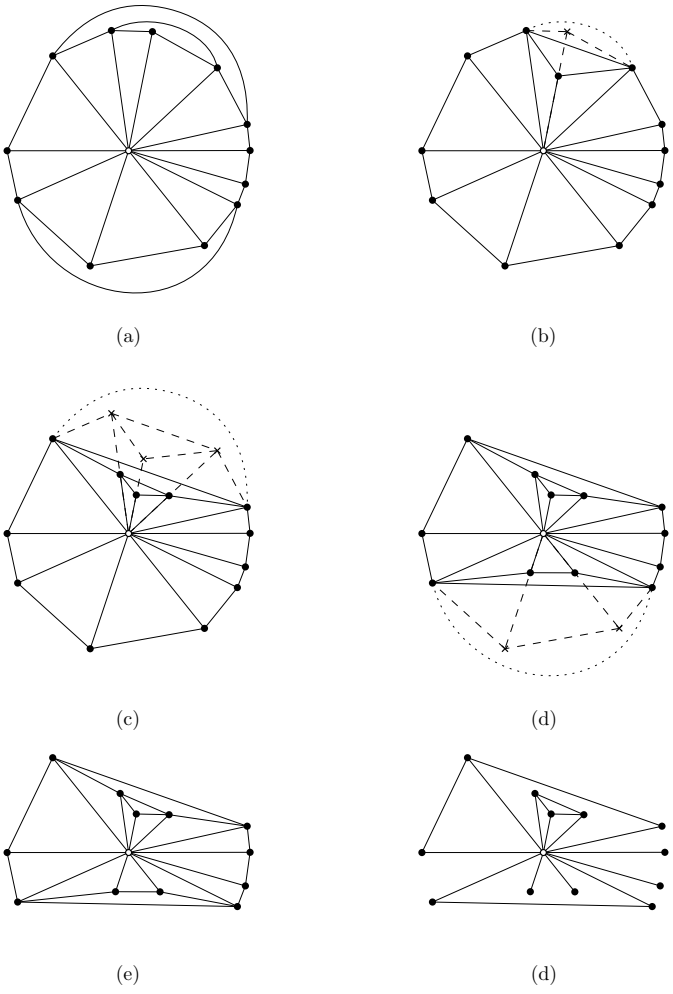


Abbildung 3.3.: (a) Nachbarschaftseinbettung Γ eines Nachbarschaftsgraphen $G(v)$ (b) Zeichnung Γ_1 für $G_1(v)$ (c) Zeichnung Γ_2 für $G_2(v)$ (d) Zeichnung Γ_3 für $G_3(v)$ (e) Zeichnung $\Gamma_4 = \Gamma'$ für $G_4(v) = G'(v)$ (f) Eine kreuzungsfreie Nachbarschaftszeichnung Γ^* von $G(v)$

Satz 3.6. *Sei $G = (V, E)$ ein Graph mit einer zugehörigen Zeichnung Γ und $v \in V$. Sei $G(v)$ ein Nachbarschaftsgraph. Wenn $G(v)$ keine alternierenden 3-Kreise enthält, dann existiert eine kreuzungsfreie Nachbarschaftszeichnung Γ^* von $G(v)$.*

Beweis. Sei $G(v)$ ein Nachbarschaftsgraph ohne alternierende 3-Kreise. Nach Lemma 3.3 wissen wir, dass sowohl für $G(v)$ der erweiterte Nachbarschaftsgraph $G'(v)$ existiert, als auch die planare Einbettung Γ' von $G'(v)$. Wir wählen Γ' mit den folgenden Eigenschaften:

(1) $\Gamma(v) = \Gamma'(v)$

(2) $\Gamma(u) - \Gamma(v) = c \cdot (\Gamma'(u) - \Gamma'(v))$ für alle $u \in N_G(v) \setminus \{v\}$ wobei $c > 0$

Falls wir also die Nachbarn von v verschieben, dann nur auf der entsprechenden Verschiebungsgerade um einen positiven Faktor c, um die Ähnlichkeit zu Γ zu bewahren. Dies liefert uns mit Lemma 3.5 per Induktion die geradlinige, kreuzungsfreie Zeichnung Γ^*. Da wir die Nachbarn von v von Γ' ausschließlich auf Punkte in Γ^* abbilden, die auf den entsprechenden Verschiebungsgeraden liegen, wird die Ähnlichkeit von Γ^* zu Γ weiter bewahrt.

Wir entfernen nun alle Kanten $E' = E(G'(v))\backslash(E(G(v))\cap E(G'(v)))$, also die Kanten, die wir in Lemma 3.3 zwischen den an v benachbarten Knoten hinzugefügt haben um die erweiterte Nachbarschaft zu erhalten, aus Γ^* (vgl. Abbildung 3.3 (d)). Mit den Eigenschaften (1),(2) und der Geradlinigkeit von Γ^* erhalten wir nach Definition 2.1 eine Nachbarschaftszeichnung Γ^* von $G(v)$.

□

3.2. Der Algorithmus

Wir haben gezeigt, dass für einen Nachbarschaftsgraph $G(v)$ stets eine kreuzungsfreie Nachbarschaftszeichnung Γ^* existiert, wenn dieser keine alternierenden 3-kreise enthält. Diese Nachbarschaftszeichnung Γ^* wollen wir nun anhand eines Algorithmus möglichst effizient konstruieren.

Lemma 3.7. *Sei $G(v)$ ein Nachbarschaftsgraph mit einer kreuzungsfreien Zeichnung Γ. Sei $u \in V(G(v))$ ein Knoten der äußeren Facette. Wenn wir u um einen Faktor $c \geq 1$ verschieben, wodurch wir eine neue Zeichnung Γ' erhalten, dann ist Γ' kreuzungsfrei.*

Beweis. Sei $G(v)$ ein Nachbarschaftsgraph mit einer kreuzungsfreien Zeichnung Γ. Sei $u_1 \in V(G(v))$. Angenommen, wir verschieben u_1 um einen Faktor $c \geq 1$ und erhalten eine Zeichnung Γ', die nicht kreuzungsfrei ist. Dann existiert eine von u_1 ausgehende Kante $\{u_1, u_2\}$, die in Γ' eine Kante $\{w_1, w_2\}$ kreuzt.

Fall 1: $u_2 = v$
Dann liegt u_1 in Γ innerhalb des 3-Kreises $\{w_1 v w_2\}$, was ein Widerspruch zur Annahme ist, dass u_1 ein Knoten der äußeren Facette ist (s.Abb 3.4 (a)).

Fall 2: $\{w_1, w_2\}$ hat v als einen Endpunkt
Dann liegt ohne Beschränkung der Allgemeinheit w_2 sowohl in Γ' als auch in Γ außerhalb des Dreiecks $u_1 v u_2$. Diese Kreuzung hätte dann allerdings bereits in Γ existiert, was ein Widerspruch zur Annahme ist, dass Γ kreuzungsfrei ist (s.Abb 3.4 (b)).

Fall 3: weder $\{u_1, u_2\}$ noch $\{w_1, w_2\}$ haben v als Endpunkt
Dann liegt $\{u_1, u_2\}$ in Γ innerhalb des Dreiecks $w_1 v w_2$. Da Γ kreuzungsfrei ist, bedeutet das insbesondere, dass das Dreieck $u_1 v u_2$ innerhalb des Dreiecks $w_1 v w_2$ liegt (s.Abb 3.4 (c)). Das ist ein Widerspruch zur Annahme, dass u_1 ein Knoten der äußeren Facette ist.

Somit kann es keine Kante geben, die $\{u_1, u_2\}$ in Γ' kreuzt. Also ist Γ' kreuzungsfrei. □

Im Folgenden wird ein Algorithmus beschrieben, der für einen Nachbarschaftsgraphen $G(v)$ ohne alternierende 3-Kreise eine kreuzungsfreie Nachbarschaftszeichnung Γ^* von $G(v)$ konstruiert.

Sei $G = (V, E)$ ein Graph mit einer geradlinigen Zeichnung Γ und $v \in V$. Sei $G(v)$ ein Nachbarschaftsgraph ohne alternierende 3-Kreise. Wir übernehmen schrittweise die Kanten und Knoten der einzelnen Facetten aus Γ in eine neue Zeichnung Γ^*, sodass die Kriterien aus Definition 2.1 erfüllt sind. Die Reihenfolge der zu übernehmenden Facetten erhalten wir durch eine Breitensuche über den Dualgraphen $G^*(v)$ der Nachbarschaftseinbettung Γ' von $G(v)$.

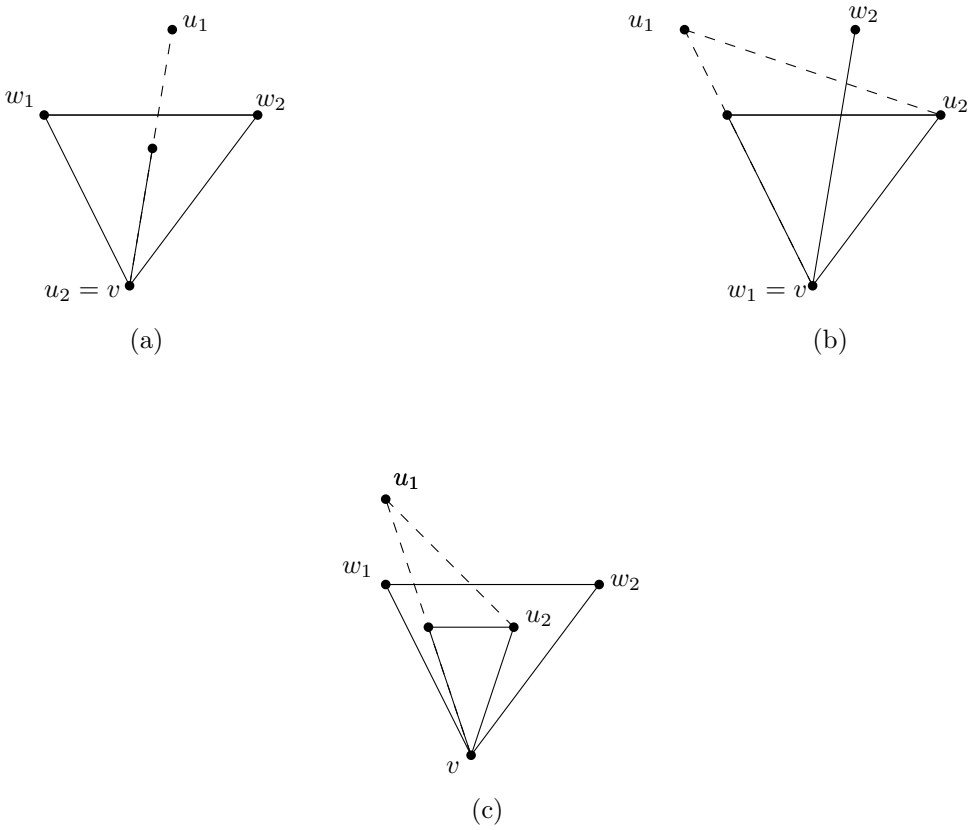


Abbildung 3.4.: Kreuzungsmöglichkeiten beim Verschieben eines Knotens der äußeren Facette um einen Faktor $c > 1$

Sei $F = \{f_1, f_2, ..., f_k\}$ die Menge von Facetten aus $G'(v)$, deren zugehörigen Knoten und Kanten noch nicht vollständig in Γ^* eingezeichnet wurden. Dabei gibt i mit $1 \leq i \leq k$ die (umgekehrte) Reihenfolge an, in welcher die Facetten aus F und deren zugehörigen Kanten und Knoten in die Zeichnung Γ^* übertragen werden.

Für jede Facette $f_i \in F$ führen wir folgende Schritte durch:

Die Knoten und Kanten von f_i werden kreuzungsfrei in Γ^* eingezeichnet. Falls die Kanten von f_i geradlinig eingezeichnet werden können, ohne dabei Kanten der anderen Facetten zu kreuzen, gehen wir weiter zu f_{i-1}. Andernfalls suchen wir nun die Kante vu innerhalb der Facette f_i bzw. des 3-Kreise u_1vu_2, für die der Knoten u am weitesten von v entfernt ist. Dann verschieben wir u_1, u_2 jeweils um einen Faktor c_j mit $j \in \{1, 2\}$, den wir wie folgt bestimmen:

Sei k_u ein Kreis mit Radius $r = |vu|$ und v als Mittelpunkt und sei t die Tangente an k_u in u. Dann schneiden die Verschiebungsgeraden von u_1 und u_2 die Tangente t in den Punkten u'_1 und u'_2. Dann ist $c_j = \frac{|vu'_j|}{|vu_j|} + c$ mit $c > 0$. Der Wert c stellt sicher, dass die Kante $e'_i = \{u'_1, u'_2\}$ nicht durch den Knoten u verläuft, sondern an ihm vorbei (vgl. Abb 3.5). Da $|vu'_i| \geq |vu_i|$ und $c > 0$ folgt $c_i \geq 1$. Zusammen mit der Eigenschaft, dass u_1 und u_2 jeweils Knoten der äußeren Facette sind, folgt mit Lemma 3.7, dass u_1 und u_2 verschoben werden können und Γ^* dabei kreuzungsfrei bleibt.

Haben wir die Facette f_1 erreicht und damit alle Facetten mit den dazugehörigen Knoten und Kanten aus $G(v)$ nach Γ^* übertragen, werden die Kanten, die in $G'(v)$ ergänzt wurden, entfernt. So erhalten wir die kreuzungsfreie Nachbarschaftszeichnung Γ^*. Die Laufzeit dieses Algorithmus wird in folgendem Lemma analysiert:

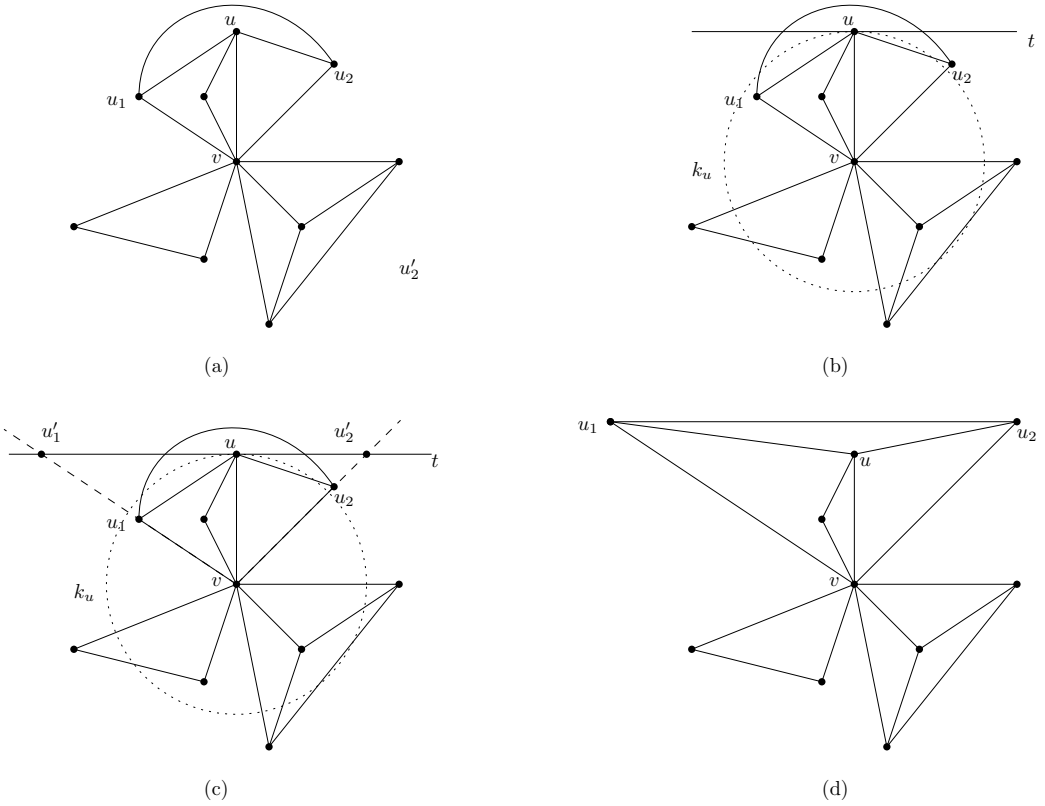


Abbildung 3.5.: Bestimmung des Faktors c_j zur Verschiebung der Endpunkte der eingefügten Kante $\{u_1, u_2\}$

Lemma 3.8. *Sei $G = (V, E)$ ein Graph und sei $G(v)$ ein Nachbarschaftsgraph ohne alternierende 3-Kreise. Dann berechnet der beschriebene Algorithmus in $O(n)$ mit $n = |V(G(v))|$ Zeit eine kreuzungsfreie Nachbarschaftszeichnung Γ^* von $G(v)$.*

Beweis. Sei $G = (V, E)$ ein Graph mit einer geradlinigen Zeichnung Γ und $v \in V$. Sei $G(v)$ ein Nachbarschaftsgraph ohne alternierende 3-Kreise mit $n = |V(G(v))|$ und $m = |E(G(v))|$. Nach Lemma 3.3 ist $G(v)$ planar und es existiert die Nachbarschaftseinbettung $G'(v)$, welche sich in $O(n)$ berechnen lässt. Sei $G^*(v)$ der Dualgraph von $G'(v)$. Dieser lässt sich ebenfalls in linearer Zeit berechnen. Um die Reihenfolge der zu zeichnenden Kanten und Knoten in linearer Zeit zu erhalten, führen wir eine Breitensuche über $G^*(v)$ durch. Der Startknoten sei $w_0 \in V(G^*(v))$, welcher die äußere Facette von $G'(v)$ repräsentiert. Alle weiteren Knoten w_i des Dualgraphen werden in der Reihenfolge, in der sie von der Breitensuche gefunden werden, durchnummeriert, wodurch wir insbesondere eine Nummerierung der Facetten von $G'(v)$ in $O(n + m)$ Zeit erhalten [1]. Die Facetten und ihre zugehörigen Knoten und Kanten werden schrittweise beginnend bei f_k in absteigender Reihenfolge in $O(1)$ Zeit in die Zeichnung Γ^* übernommen:

Sei f_i die zu übertragende Facette. Falls f_i geradlinig eingezeichnet werden kann, ohne dabei Kanten anderer Facetten zu kreuzen gehen wir zur f_{i-1} und übertragen f_i in $O(1)$ Zeit, da jede Facette in $G'(v)$ per Konstruktion höchstens 3 Knoten und höchstens 3 Kanten enthält. Andernfalls bestimmen wir die Kante vu innerhalb der Facette f_i für die der Knoten u am weitesten von v entfernt ist, wie folgt:

Bestehe f_i aus den Kanten vu_1, vu_l und u_1u_l. Nach Konstruktion von $G'(v)$ existiert ein Pfad $u_1, u_2, ..., u_l$ über die Kanten zwischen den an v benachbarten Knoten $u_1, u_2, ..., u_l$. Die Knoten dieses Pfades liegen per Konstruktion vor dem Einfügen der Kante u_1u_l auf dem Rand. Für jeden Knoten auf diesem Pfad speichern wir seine Entfernung zu v, beispielsweise

in einem Array, und bestimmen anhand davon den Knoten mit maximalem Abstand zu v in $O(l)$ Zeit. Da nach Einfügen der Kante u_1u_l die Knoten $u_2, ..., u_{l-1}$ nicht mehr auf dem Rand liegen, sondern innerhalb der Facette f_i, werden sie in späteren Schritten nicht mehr besucht. Wir besuchen also im Laufe des Algorithmus, um einen Knoten u mit maximaler Entfernung zu v für eine Facette zu finden, jeden der n Knoten höchstens ein Mal, weshalb wir weiterhin lineare Laufzeit erreichen.

Nun verschieben wir die Knoten u_1, u_2 jeweils in $O(1)$ Zeit um einen Faktor c_j mit $j \in \{1, 2\}$, den wir anhand des Abstands von u zu v berechnen. Somit lassen sich alle Facetten in konstanter Zeit in Γ^* einzeichnen. Nachdem alle k Facetten aus F in $O(k)$ Zeit in Γ^* eingezeichnet wurden, werden die Kanten, die eingefügt wurden, um die Nachbarschaftseinbettung $G'(v)$ zu erhalten, in $O(m)$ Zeit aus Γ^* entfernt. Die kreuzungsfreie Nachbarschaftszeichnung Γ^* von $G(v)$ kann somit in $O(n + m) \in O(n)$ Zeit berechnet werden. □

Satz 3.9. *Sei $G = (V, E)$ ein Graph und sei $G(v)$ ein Nachbarschaftsgraph mit $n = |V(G(v))|$. Dann lässt in $O(n)$ Zeit testen, ob eine kreuzungsfreie Nachbarschaftszeichnung Γ^* existiert. Falls ja, kann diese in $O(n)$ Zeit berechnet werden.*

Beweis. Sei $G = (V, E)$ ein Graph und sei $G(v)$ ein Nachbarschaftsgraph mit $n = |V(G(v))|$, $m = |E(G(v))|$ und Nachbarschaftszeichnung Γ. Nach Lemma 3.1 kann Γ nur kreuzungsfrei sein, wenn $G(v)$ keine alternierenden 3-Kreise enthält. Daraus folgt, dass, wenn $G(v)$ mit seiner festen Ordnung um v planar ist, er keine alternierenden 3-Kreise besitzt. Der Test auf Planarität für $G(v)$ erfolgt in linearer Zeit [5]. Falls $G(v)$ nicht planar ist, existiert folglich auch keine kreuzungsfreie Nachbarschaftszeichnung. Andernfalls enthält $G(v)$ keine alternierenden 3-Kreise und damit existiert nach Lemma 3.8 ein Algorithmus, der in $O(n + m)$ Zeit eine kreuzungsfreie Nachbarschaftszeichnung Γ^* von $G(v)$ berechnet. Somit lässt sich in $O(n)$ Zeit testen, ob für $G(v)$ eine kreuzungsfreie Nachbarschaftszeichnung Γ^* existiert und falls ja, kann diese in $O(n + m) \in O(n)$ Zeit berechnet werden. □

3.3. Nachbarschaftsgraphen mit alternierenden 3-Kreisen

In vorangegangenen Abschnitten haben wir gezeigt, dass Nachbarschaftsgraphen in linearer Zeit kreuzungsfrei dargestellt werden können, sofern sie keine alternierenden 3-Kreise enthalten. Wir wollen nun ein Verfahren betrachten, welches sich auf Graphen im Allgemeinen anwenden lässt. Es soll eine Menge von Kanten E_{del} bestimmt werden, welche gelöscht werden muss, um alle alternierenden 3-Kreise eines Nachbarschaftsgraphen aufzulösen. Auf den Teilgraphen ohne die Kanten aus E_{del} kann dann der Algorithmus 3.2 angewandt werden, um in linearer Zeit eine kreuzungsfreie, geradlinige Zeichnung zu berechnen. Anschließend werden die entfernten Kanten wieder in diese Zeichnung eingefügt.

Sei $G = (V, E)$ ein Graph und $G(v)$ ein Nachbarschaftsgraph mit $v \in V$ und Nachbarschaftszeichnung Γ. Sei $\{u_1, u_2\} \in E$ mit $u_1, u_2 \neq v$ eine Kante, die zwei Nachbarn von v als ihre Endpunkte hat. Eine solche Kante bezeichnen wir als *äußere Kante*.

Sei $G'(v)$ der erweiterte Nachbarschaftsgraph von $G(v)$ mit Nachbarschaftseinbettung Γ'. Wir nehmen an, dass $G(v)$ und damit insbesondere $G'(v)$ alternierende 3-Kreise enthält. Nach Lemma 3.3 entstehen Kreuzungen in Nachbarschaftseinbettungen nur durch äußere Kanten von alternierenden 3-Kreisen.

Sei $E_{\text{outer}} \subseteq E$ die Menge der äußeren Kanten in $G'(v)$ und sei $d : E_{\text{outer}} \to \mathbb{R}$ die Kantengewichtsfunktion, welche einer äußeren Kanten $e \in E_{\text{outer}}$ ein Gewicht $d(e)$ zuordnet. Das Gewicht entspricht der Anzahl an Kanten die e kreuzen.

Sei $e_i \in E_{\text{outer}}$ eine Kante mit dem höchsten Gewicht $d(e_i) > 0$ mit $1 \leq i \leq |E_{\text{outer}}|$. Die Menge E_i sei definiert als $E_i = E_{i-1} \setminus \{e_i\}$. Wir betrachten anschließend den daraus vorgehenden Teilgraph $G'_i(v) = (V, E_i)$. Es gilt $G'_0(v) = G'(v)$ und $E_0 = E$. Wenn das Gewicht aller Kanten 0 beträgt, dann enthält $G'_i(v)$ keine alternierenden 3-Kreise mehr. Wir berechnen dann mithilfe des Algorithmus 3.2 die kreuzungsfreie, geradlinige Zeichnung Γ^* in $O(n)$ und fügen anschließend die gelöschten Kanten wieder geradlinig in die Zeichnung Γ^* ein. Andernfalls bestimmen wir die Kante $e_{i+1} \in E_i$ mit dem höchsten Gewicht $d(e_{i+1}) > 0$ sowie $G'_{i+1}(v)$ und E_{i+1} und wiederholen den Vorgang.

Im Folgenden soll die Laufzeit dieses Verfahrens bestimmt werden.

Lemma 3.10. *Sei $G = (V, E)$ ein Graph und $G(v)$ ein Nachbarschaftsgraph mit $v \in V$. Wenn $G(v)$ alternierende 3-Kreise enthält, dann existiert ein Algorithmus, der in $O(m^2_{\text{outer}} \log m_{\text{outer}})$ Zeit die Menge von äußeren Kanten E_{del} berechnet, die aus $G(v)$ entfernt werden müssten, um alle alternierenden 3-Kreise in $G(v)$ aufzulösen, wobei m_{outer} die Anzahl der äußeren Kanten in $G(v)$ ist.*

Beweis. Sei $G = (V, E)$ ein Graph und $G(v)$ ein Nachbarschaftsgraph mit $v \in V$. Sei $G'(v)$ der erweiterte Nachbarschaftsgraph. Sei E_{outer} die Menge der äußeren Kanten in $G'(v)$. Wir nehmen an, dass $G(v)$ und damit $G'(v)$ alternierende 3-Kreise enthalten. Sei $d : E_{\text{outer}} \to \mathbb{R}$ die Kantengewichtsfunktion, welche einer äußeren Kanten $e \in E_{\text{outer}}$ ein Gewicht $d(e)$ zuordnet. Das Gewicht entspricht der Anzahl an Kanten, die e kreuzen. Das Gewicht einer Kante e berechnen wir wie folgt:

Seien $u_1, u_2, ..., u_{m_{\text{outer}}}$ die Nachbarn von v, geordnet nach ihrer Reihenfolge um v. Wir speichern alle äußeren Kanten $e = (u_l, u_p) \in E_{\text{outer}}$ mit $1 \leq l < p \leq m_{\text{outer}}$ in einem Rot-Schwarz-Baum ab und sortieren sie dabei nach dem Endpunkt u_p mit dem höheren Index p. Ebenso speichern wir sie zusammen mit ihrem Gewicht in einem Array ab. Anschließend suchen wir nach allen äußeren Kanten $e' = (u_{l'}, u_{p'}) \in E_{\text{outer}}$ mit $1 \leq l' < p' \leq m_{\text{outer}}$, für die gilt $l < l'$ und $p < p'$ oder $l > l'$ und $p > p'$. Mit anderen Worten, wir suchen nach Kanten, die einen Endpunkt innerhalb des 3-Kreises $u_l v u_p$ haben und einen außerhalb, da dies genau alle äußeren Kanten von 3-Kreisen sind, die mit $u_l v u_p$ alternieren. Für jede solche Kante erhöht sich das Gewicht $d(e)$ um 1. Jede Suchoperation im Rot-Schwarz-Baum kann in $O(\log m_{\text{outer}})$ Zeit durchgeführt werden [4]. Wir benötigen somit für jede der m_{outer} äußeren Kanten $O(m_{\text{outer}} \cdot \log m_{\text{outer}})$ Zeit, um ihr Gewicht zu berechnen und damit $O(m^2_{\text{outer}} \log m_{\text{outer}})$ Zeit, um das Gewicht aller Kanten zu bestimmen.

Nun wird in $O(m_{\text{outer}})$ Zeit eine Kante e_i mit dem höchsten Gewicht $d(e_i)$ bestimmt und sowohl zur Kantenmenge E_{del} hinzugefügt als auch in $O(\log m_{\text{outer}})$ Zeit aus dem Rot-Schwarz-Baum entfernt [4], wodurch wir die Kantenmente E_i erhalten. Wir reduzieren die Gewichte aller Kanten in E_i, die vom Löschen betroffen waren, in $O(d(e_i) \cdot m_{\text{outer}})$ Zeit um jeweils 1. Wenn es keine Kante e_{i+1} gibt mit $d(e_{i+1}) > 0$, was in $O(m_{\text{outer}})$ Zeit geprüft werden kann, dann enthält $G(v)$ keine alternierenden 3-Kreise mehr. Andernfalls wiederholen wir den Vorgang für E_{i+1}.

Wir erhalten somit eine Laufzeit von insgesamt $O(m^2_{\text{outer}} \log m_{\text{outer}})$ für das Löschen von $k = |E_{\text{del}}|$ äußeren Kanten, sodass $G(v)$ keine alternierenden 3-Kreise mehr enthält. □

Satz 3.11. *Sei $G = (V, E)$ ein Graph und sei $G(v)$ ein Nachbarschaftsgraph mit $n = |V(G(v))|$. Sei $G'(v)$ die erweiterte Nachbarschaftseinbettung von $G(v)$ und sei m_{outer} die Anzahl der äußeren Kanten in $G'(v)$. Wenn $G(v)$ alternierende 3-Kreise enthält, dann lässt sich in $O(m^2_{\text{outer}} \log m_{\text{outer}})$ Zeit eine Menge von Kanten E_{del} berechnen, sodass alle Kreuzungen der Nachbarschaftszeichnung Γ^* auf diesen Kanten liegen. Entfernt man die Kanten E_{del} aus Γ^*, so wird die Zeichnung kreuzungsfrei.*

Beweis. Sei $G = (V, E)$ ein Graph und sei $G(v)$ ein Nachbarschaftsgraph mit $n = |V(G(v))|$. Sei $G'(v)$ die erweiterte Nachbarschaftseinbettung von $G(v)$ und sei m_{outer} die Anzahl der äußeren Kanten in $G'(v)$. Wir nehmen an, $G(v)$ enthält alternierende 3-Kreise. Nach Lemma 3.10 können in $O(m_{\text{outer}}^2 \log m_{\text{outer}})$ Zeit k äußere Kanten aus $E(G(v))$ entfernt werden, sodass $G'(v)$ und damit insbesondere $G(v)$ keine alternierenden 3-Kreise mehr enthält. Die Menge an gelöschten Kanten sei E_{del}. Der Algorithmus 3.2 lässt sich dann auf den Teilgraphen $G_{\text{del}}(v) = (V, E \setminus E_{\text{del}})$ anwenden, da alle alternierenden 3-Kreise aufgelöst wurden, und eine kreuzungsfreie Nachbarschaftszeichnung Γ^* für diesen Teilgraphen kann somit in $O(n)$ Zeit berechnet werden. Anschließend fügen wir die Kanten aus E_{del} geradlinig in die Zeichnung Γ^* ein. Damit erhalten wir in $O(m_{\text{outer}}^2 \log m_{\text{outer}})$ eine Nachbarschaftszeichnung Γ^*, in der alle Kreuzungen auf Kanten aus E_{del} liegen. □

4. Experimentelle Evaluation

Im Rahmen der Arbeit wurde eine Java-Script Anwendung programmiert, um die beschriebenen Algorithmen graphisch zu veranschaulichen. Im Folgenden wird ein Überblick über die entwickelte Anwendung gegeben, sowie eine Evaluation bezüglich der Kreuzungsminimierung und grafischen Darstellung von Nachbarschaften.

4.1. Implementierung

Das Programm liest zunächst einen Graphen in XML Format ein und zeichnet diesen. Hierzu wurde die JavaScript Bibliothek D3.js[1] verwendet. Diese ermöglicht es unter anderem, die Id und Koordinaten der Knoten sowie die Attribute `source` und `target` für alle Kanten aus der XML Datei auzulesen und anschließend für alle Knoten und Kanten entsprechende Kreise und Linien in Form von HTML Elementen zu generieren. Jedem Knoten wird dabei die Funktion mitgegeben, dass bei Doppelklick auf ihn seine Nachbarschaft transformiert werden soll. Die neue Nachbarschaftszeichnung wird dabei wie folgt berechnet.

Zunächst werden die Knoten und Kanten der Nachbarschaft ermittelt sowie die Rotation um das gewählte Zentrum v. Alle Knoten und Kanten, die nicht Teil der Nachbarschaft sind, werden ausgeblendet, um die Nachbarschaft hervorzuheben. Anschließend werden die Gewichte der Kanten berechnet. Hierbei findet die Funktion `calcWeight()` für eine Kante e alle Kanten, die mit ihr alternieren. Die Anzahl dieser Kanten entspricht dem Gewicht $d(e)$. Alle äußeren Kanten der Nachbarschaft werden mit ihrem Gewicht im Array `weightedEdges` abgespreichert. Anschließend wird die Funktion `removeAndUpdate()` solange ausgeführt, bis jede Kante ein Gewicht von 0 hat. Die Funktion `removeAndUpdate()` sucht die letzte Kante mit dem höchsten Gewicht, entfert sie aus `weightedEdges` und aktualisiert anschließend das Gewicht der restlichen Kanten. Das grafische Linienelement der gelöschten Kante wird dabei rot eingefärbt, um sie als gelöscht zu markieren. Der Vorgang entspricht dem Algorithmus 3.3 mit der Abänderung, dass die vom Algorithmus betroffenen Kanten nicht gelöscht, sondern rot eingefärbt werden.

Zuletzt berechnet die Anwendung die neuen Koordinaten für alle Knoten, die Endpunkte von Kanten sind, die weder ausgeblendet noch rot markiert sind, um die verbleibenden Kanten der Nachbarschaft kreuzungsfrei zu zeichnen. Dafür wird zunächst die Reihenfolge der Kanten bestimmt, für die die Koordinaten der Endpunkte neu berechnet werden sollen. Für jede Kante $e = (u, w)$ in `weightedEdges` - also jeder äußeren Kante, die noch nicht

[1] `https://d3js.org/`

gelöscht wurde - berechnet die Funktion `getOrderNumber()` die Anzahl an Knoten, die in der Reihenfolge um v zwischen u und w auf der Seite der Zeichnung liegen, auf der der Winkel zwischen uv und wv kleiner als 180° ist. Die Kante e wird dann zusammen mit ihrer `orderNumber` im Array `orderedEdges` abgespeichert. Es wird nun nach der letzten Kante $e = (u, w)$ mit der kleinsten `orderNumber` in `orderedEdges` gesucht und für ihre Endpunkte werden neue Koordinaten berechnet. Hierfür suchen wir nach dem Knoten p, der am weitesten außerhalb des Dreiecks uvw liegt. Diese Entfernug wird genutzt, um mit Hilfe des Strahlensatzes die neuen Kantenlängen für wv und uv zu berechnen bzw. ein Dreieck, das den Knoten p enthält. Anschließend wird die Kante e aus dem Array `orderedEdges` entfernt und die nächste Kante mit der kleinsten `orderNumber` wird gesucht. Wir wiederholen dies, bis `orderedEdges` keine Kanten mehr enthält. Durch die `orderNumber` wird sichergestellt, dass wir keine Knoten mehr verschieben, die innerhalb eines Dreiecks liegen, das bereits neu berechnet wurde.

Per Doppelklick auf das gewählte Zentrum wird die Zeichnung wieder in die ursprüngliche Zeichnung zurück transformiert und die restlichen Knoten und Kanten werden wieder eingeblendet.

4.2. Testdaten

Um die Anwendung zu bewerten, wurden aus 10 Graphen jeweils 10 Nachbarschaften untersucht. Vier dieser Graphen wurden dabei generiert, indem die Kanten- und Knotenanzahl vorgegeben wurde. Dann wurden erst alle Knoten erzeugt und anschließend alle möglichen Kanten mit der gleichen Wahrscheinlichkeit p, sodass die vorgegebene Anzahl von Kanten nicht überschritten wurde. Die Anzahl von Knoten lag jeweils bei 35, 40, 40 und 30 und die Kantenanzahl jeweils bei 135, 160, 160 und 120. Für die restlichen Graphen wurden Beispieldatensätze aus dem Internet verwendet[2]. Untersucht wurden die Nachbarschaften auf ihre Dichte, die Anzahl von alternierenden 3-Kreisen, die Anzahl von Kanten, die gelöscht werden müssten, um alle alternierenden 3-Kreise aufzulösen, sowie die Kreuzungsminimierung in Prozent. Die Dichte $d = \frac{m}{n}$ der Nachbarschaftsgraphen mit m als Anzahl der Kanten und n als Anzahl der Knoten der Nachbarschaft reicht bei den Beispielen von 1.125 bis 6.25. Eine Dichteverteilung der Nachbarschaftsgraphen wird in Abbildung 4.1 dargestellt.

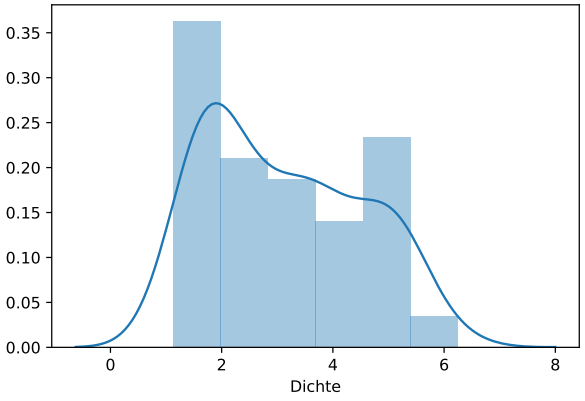


Abbildung 4.1.: Dichteverteilung der 100 getesteten Nachbarschaftsgraphen

[2] `http://visone.info/wiki/index.php/Knecht_Classroom_(data)`
`http://visone.info/wiki/index.php/Introductory_workshop_(data)`
`https://observablehq.com/@d3/force-directed-graph`

4.3. Evaluation

Im Folgenden wird die Anwendung sowohl hinsichtlich der Kreuzungsminimierung als auch der grafischen Darstellung von Nachbarschaften ausgewertet.

4.3.1. Kreuzungsminimierung

Es wurde zunächst untersucht, inwiefern sich die Anzahl der alternierenden 3-Kreise, die Kreuzungsminimierung und die Anzahl der gelöschten Kanten verhält, je höher die Dichte eines Nachbarschaftsgraphen ist. Die Auswertung über die 100 Nachbarschaftsgraphen wird in Abbildung 4.2 dargestellt. Darin lässt sich beobachten, dass mit wachsender Dichte die Anzahl von alternierenden 3-Kreisen sowie die Anzahl an zu löschenden Kanten steigt, während der Prozentsatz der reduzierten Kreuzungen sinkt. Mit steigender Komplexität des Graphen wird es also scheinbar schwieriger, Kreuzungen zu reduzieren und wenig Kanten entfernen zu müssen, um alle alternierenden 3-Kreise aufzulösen.

Für drei der generierten Graphen wurden exemplarisch die Zahl der Kreuzungen vor und nach der Transformation untersucht, sowie die Anzahl der Kanten der Nachbarschaft und die Anzahl der gelöschten Kanten. Die Graphen und deren Auswertung werden im Anhang A in den Abbildungen A.1 bis A.6 dargestellt. Die Zentren der ausgewerteten Nachbarschaft sind in den entsprechenden Zeichnungen beschriftet.

Insgesamt wurde für 100 Nachbarschaften eine durchschnittliche Kreuzungsminimierung von $49,4\%$ erreicht sowie ein Durchschnittswert von $20,8\%$ für Kanten, die für eine kreuzungsfreie Zeichnung gelöscht werden müssten.

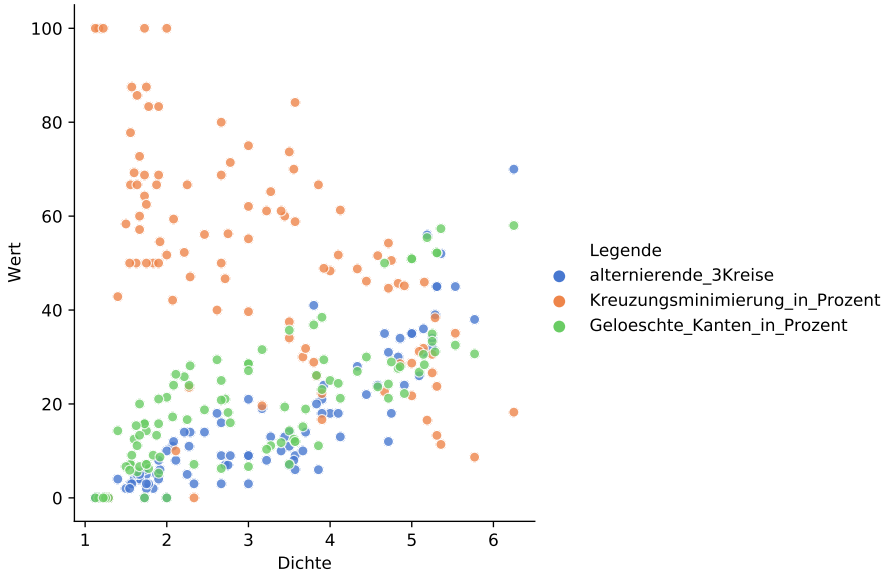


Abbildung 4.2.: Gesamtauswertung über 100 Nachbarschaften

4.3.2. Grafische Darstellung

Grundsätzlich lässt sich sagen, dass die grafische Darstellung der optimierten Nachbarschaften für Graphen mit geringer Dichte und wenigen alternierenden 3-Kreisen gut funktioniert (s. Abbildung 4.3 und 4.4). Jedoch wird die Darstellung für einige Fälle problematisch, beispielsweise müssen die Eckpunkte eines Dreiecks uvw immer weiter verschoben werden, je größer der Winkel am Zentrum v ist. Dadurch werden Knoten oft über den Bildschirmrand hinaus verschoben, damit die in uvw enthaltenen Dreiecke nicht die Kanten von uvw schneiden (s. Abbildung 4.5 und 4.6).

Zusätzlich kommt es durch Anwendung des Strahlensatzes oft zu Transformationen, bei denen Dreiecke größer werden, als nötig. Ein möglicher Ansatz, dieses Verhalten zu beheben, wäre es, erst einen Eckpunkt eines Dreiecks zu verschieben und anschließend, falls noch Kreuzungen für dieses Dreieck bestehen, den zweiten Eckpunkt zu verschieben. Man könnte das Ergebnis dieses Ansatzes auch mit dem Ergebnis des Strahlensatzes vergleichen und sich dann beispielsweise für die Zeichnung mit geringerer Fläche entscheiden.

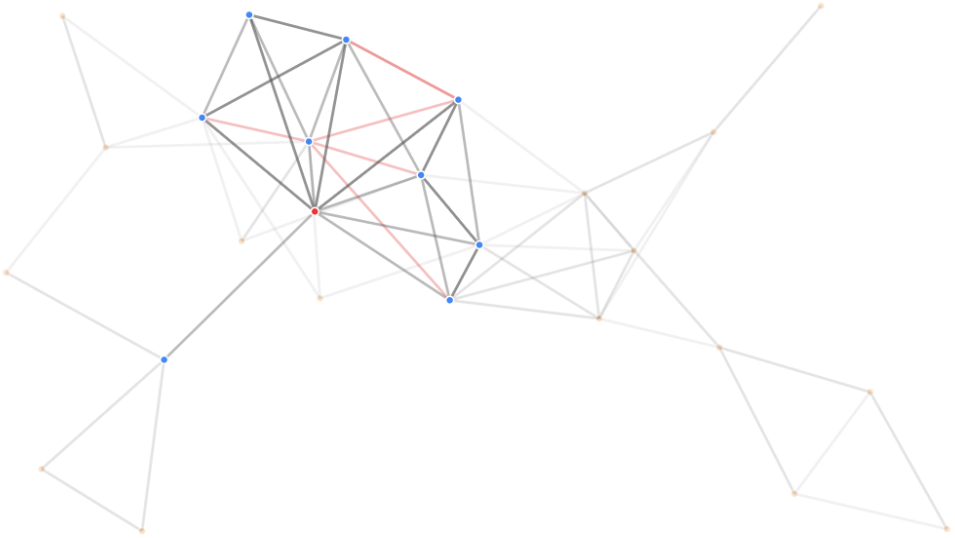


Abbildung 4.3.: Nachbarschaft mit einer Dichte von $3,5$ und 11 alternierenden 3-Kreisen vor der Transformation

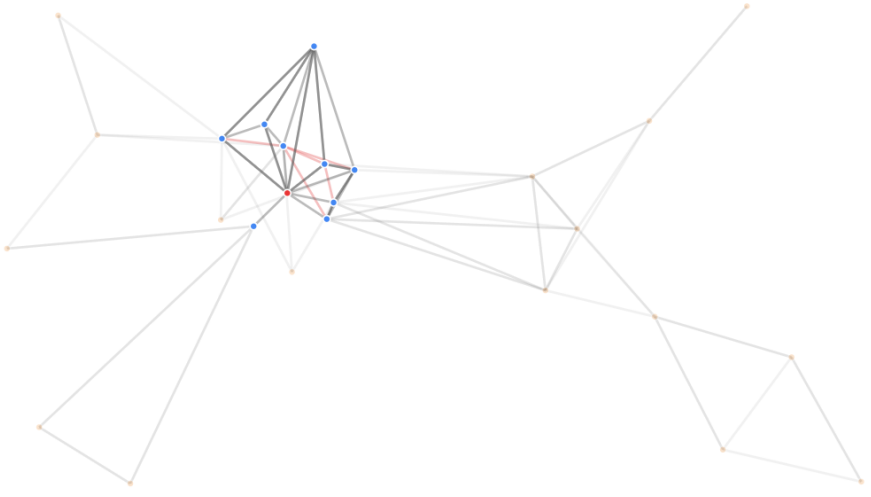


Abbildung 4.4.: Nachbarschaft aus Abbildung 4.3 nach der Transformation

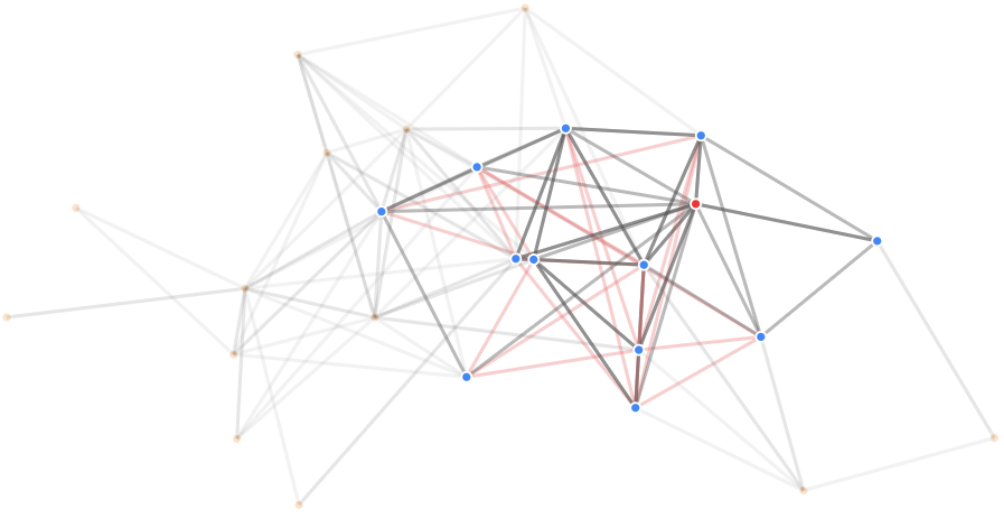


Abbildung 4.5.: Nachbarschaft eines Graphen vor der Transformation

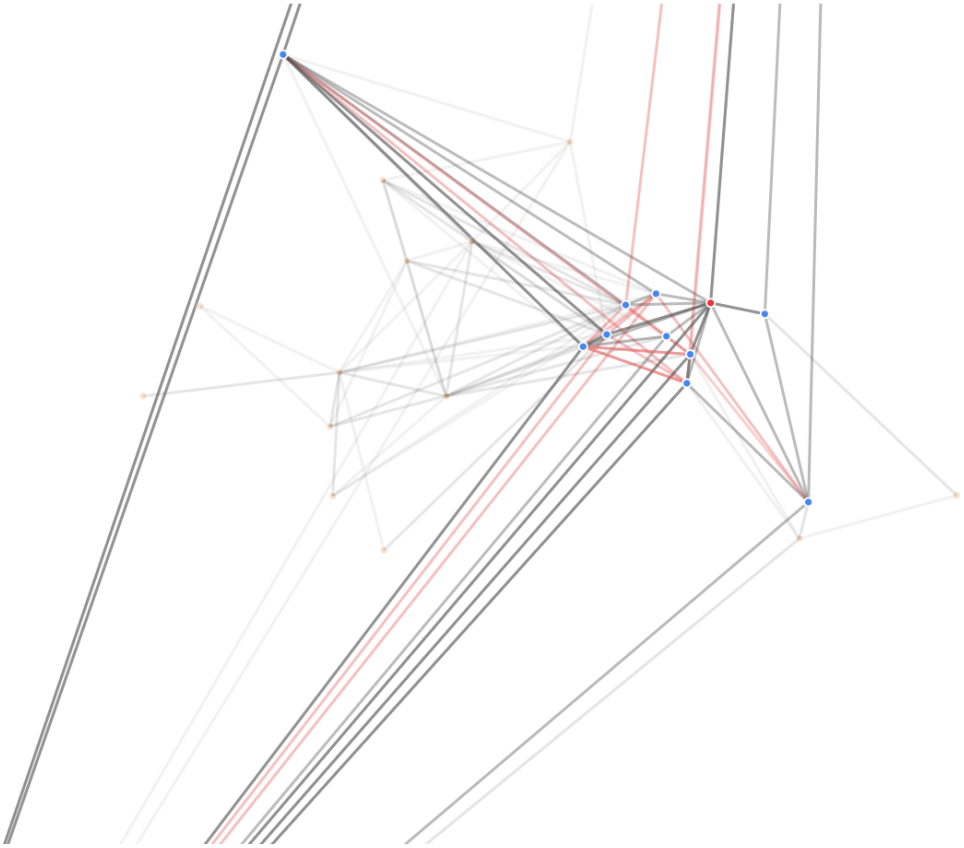


Abbildung 4.6.: Nachbarschaft des Graphen aus Abbildung 4.5 nach der Transformation. Durch große Winkel am Zentrum (rot hervorgehobener Knoten) müssen einige Knoten sehr weit oder sogar über den Bildschirmrand verschoben werden, damit die Dreiecke groß genug werden und keine Kreuzungen mehr verursachen.

5. Zusammenfassung und Ausblick

Wir haben gezeigt, dass sich eine Nachbarschaft stets kreuzungsfrei zeichnen lässt, wenn sie keine alternierenden 3-Kreise enthält. Diese kreuzungsfreie Zeichnung lässt sich in linearer Zeit berechnen. Ebenso wurde ein Verfahren für Graphen im Allgemeinen beschrieben, dass in $O(n^2 \log n)$ Zeit k Kanten berechnet, welche gelöscht werden müssten, um alle alternierenden 3-Kreise aus dem Nachbarschaftsgraphen aufzulösen.

Die entwickelte Anwendung ergänzt den theoretischen Teil der Arbeit und ermöglicht es im Durchschnitt die Kreuzungen in Nachbarschaftszeichnungen auf etwa die Hälfte zu reduzieren. Sie zeigt außerdem, dass man durchschnittlich etwa 20% der Kanten löschen müsste, um eine Nachbarschaftsgraphen zu erhalten, der keine alternierenden 3-Kreise enthält.

Durch die Auswertung anhand von Beispielen kann beobachtet werden, dass mit des steigenden Dichte eines Nachbarschaftsgraphen die Anzahl an alternierenden 3-Kreisen steigt und damit auch die Anzahl an zu löschenden Kanten. Die prozentuale Minimierung von Kreuzungen sinkt dabei.

In Zukunft könnte untersucht werden, ob sich die bisherige Laufzeit des Algorithmus zum Löschen von Kanten von $O(n^2 \log n)$ optimieren lässt. Insbesondere könnte die minimale Anzahl von zu löschenden Kanten mit Hilfe von Schnittgraphen und VertexCover bestimmt werden. Ebenso könnte man versuchen, Schranken für die Anzahl der Kreuzungen in Nachbarschaftszeichnungen zu ermitteln, insbesondere im Zusammengang mit der Anzahl von alternierenden 3-Kreisen. Interessant wäre auch zu untersuchen, inwiefern sich die Anzahl an Kreuzungen reduzieren lässt, wenn man das Verändern der Winkel zwischen den vom Zentrum ausgehenden Kanten erlauben würde.

Literaturverzeichnis

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[2] E. Di Giacomo, W. Didimo, and G. Liotta. Radial Drawings of Graphs: Geometric Constraints and Trade-Offs. *Journal of Discrete Algorithms*, 6(1):109–124, March 2008.

[3] A. Grinvald, A. Manker, and M. Segal. Visualization of the spread of electrical activity in rat hippocampal slices by voltage-sensitive optical probes. *The Journal of Physiology*, 333(1):269–291, 1982.

[4] L. J. Guibas and R. Sedgewick. A Dichromatic Framework for Balanced Trees. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 8–22. IEEE Computer Society, 1978.

[5] J. E. Hopcroft and R. E. Tarjan. Efficient Planarity Testing. *Journal of the ACM*, 21(4):549–568, October 1974.

[6] Y. Hu, E. R. Ganser, and S. Kobourov. Visualizing Graphs and Clusters as Maps. *IEEE Computer Graphics and Applications*, 30(6), 2010.

[7] C. Muelder, K.-L. Ma, and T. Bartoletti. Interactive visualization for network and port scan detection. In A. Valdes and D. Zamboni, editors, *Recent Advances in Intrusion Detection*, pages 265–283, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] I. Surovtsova, N. Simus, T. Lorenz, A. König, S. Sahle, and U. Kummer. Accessible methods for the dynamic time-scale decomposition of biochemical systems. *Bioinformatics*, 25(21):2816–2823, 07 2009.

[9] R. Tamassia. Constraints in Graph Drawing Algorithms. *Constraints*, 3:87–120, 1998.

[10] W. T. Tutte. A Theory of 3-Connected Graphs. *Indagationes Mathematicae*, 23:441–455, 1961.

[11] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

Anhang

A. Zusatzmaterial für Kapitel 4

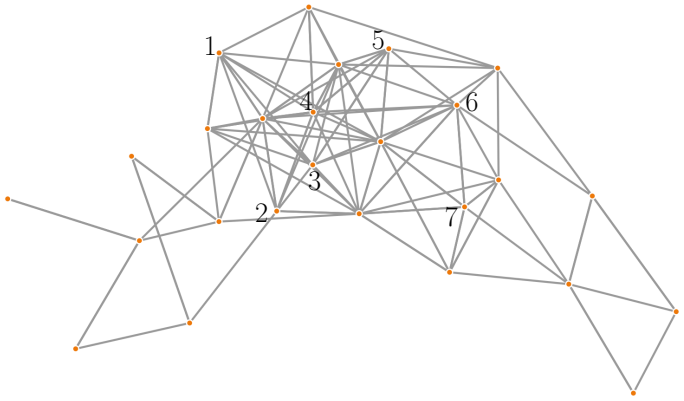


Abbildung A.1.: Graph 1

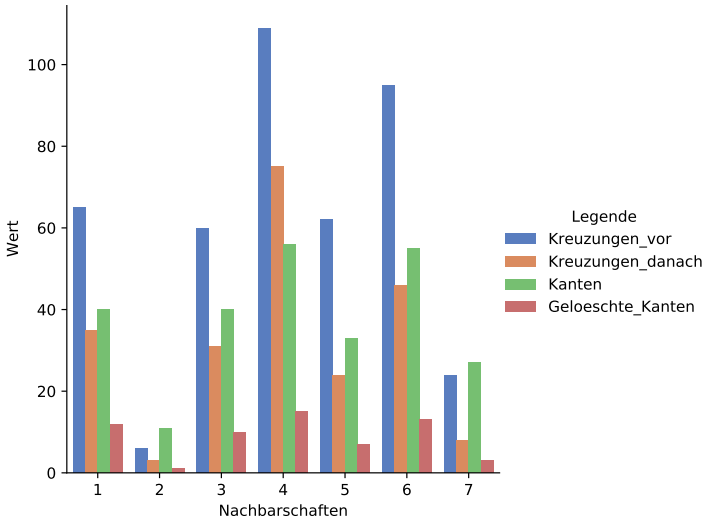


Abbildung A.2.: Auswertung Graph 1

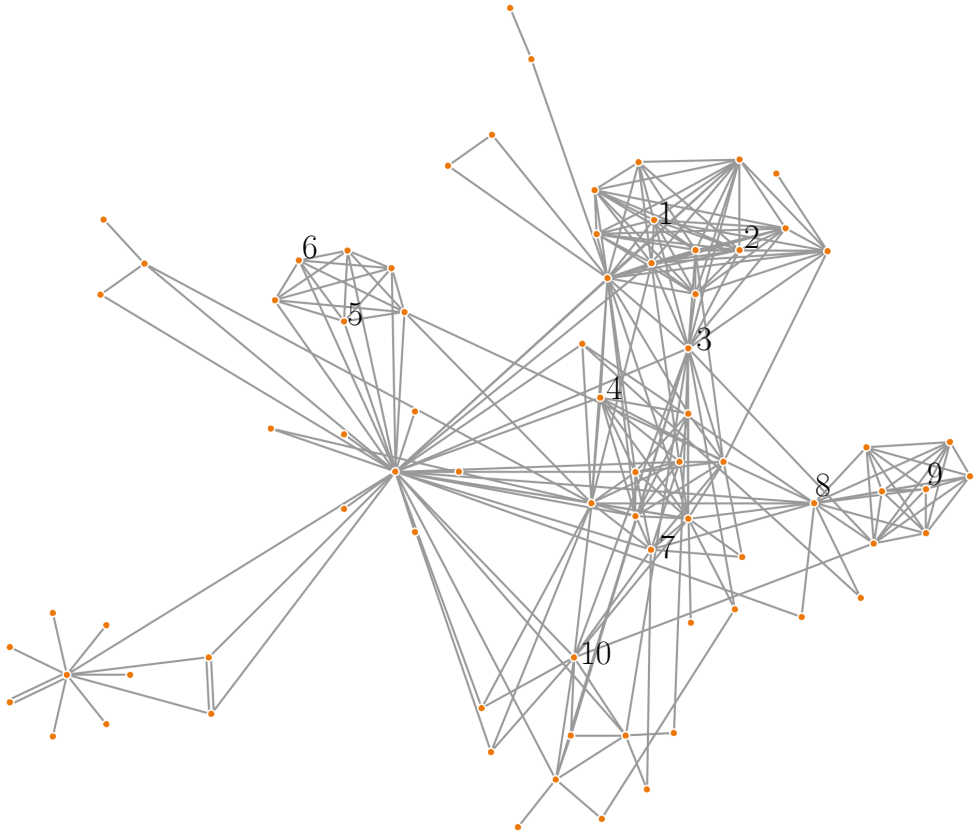


Abbildung A.3.: Graph 2

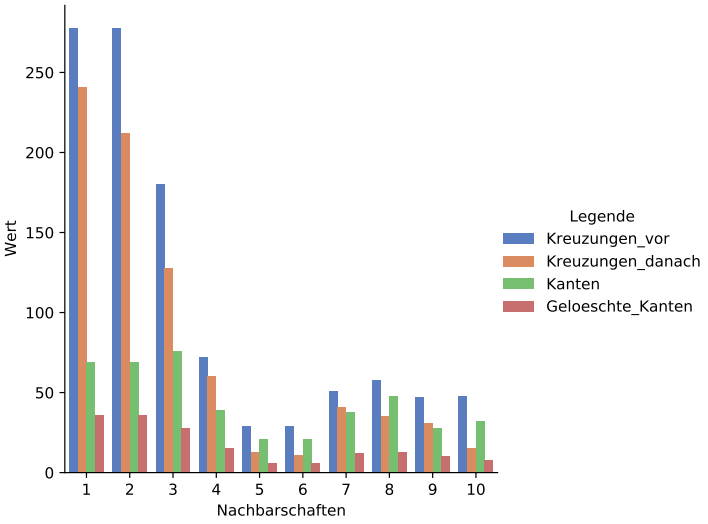


Abbildung A.4.: Auswertung Graph 2

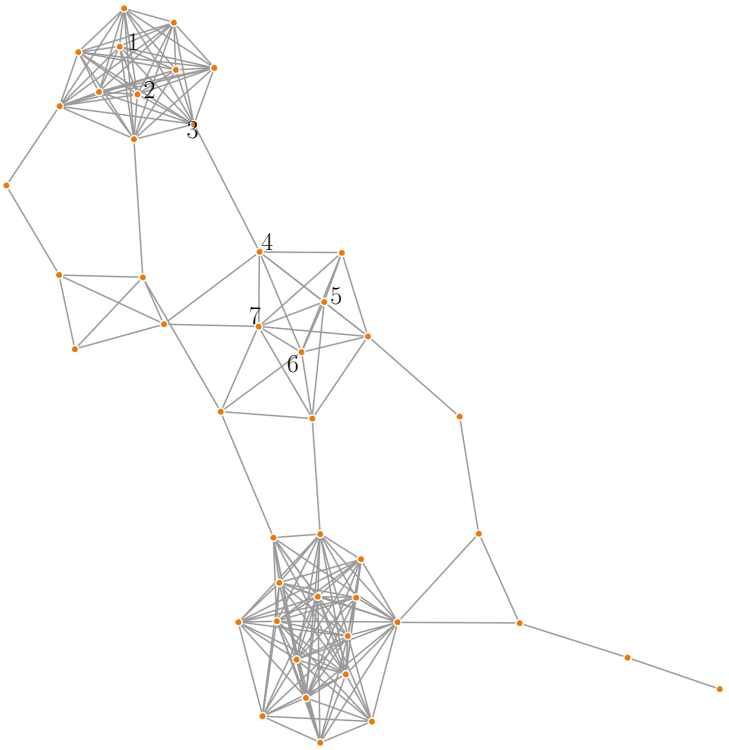


Abbildung A.5.: Graph 3

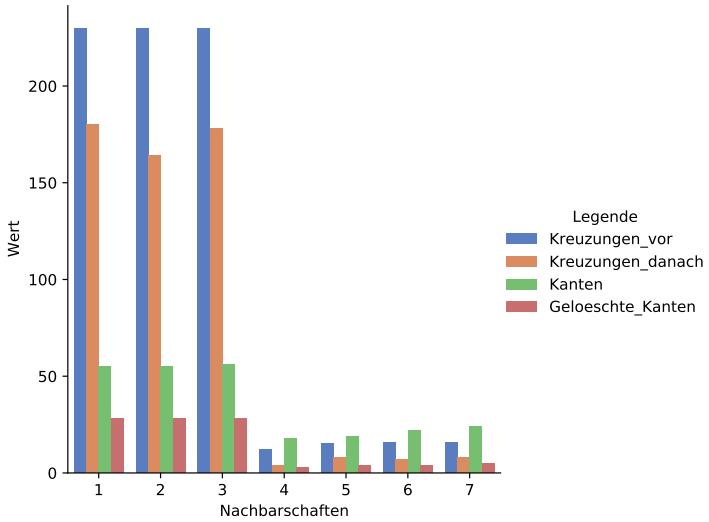


Abbildung A.6.: Auswertung Graph 3