Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik I
Effiziente Algorithmen und wissensbasierte Systeme

Bachelorarbeit

Visualisierung von gewichteten Graphen unter Platzbeschränkung

Maximilian Aulbach

Eingereicht am 21. Januar 2014

Betreuer:
Prof. Dr. Alexander Wolff
Dipl.-Inf. Martin Fink

Inhaltsverzeichnis

1 Einleitung

Ein soziales Netzwerk im Internet ist eine lose Verbindung von Menschen in einer Netzgemeinschaft. Dort können Nutzer persönliche Profile erstellen, Kontaktlisten anlegen und Nachrichten von anderen Mitgliedern empfangen und an diese versenden. Online-Netzwerke, wie beispielsweise Facebook[1] oder Google+[2], kann man als *Graphen* modellieren, wobei Mitglieder, wie dort angemeldete Menschen oder Unternehmen, als *Knoten* und Beziehungen, wie Freundschaften und Partnerschaften, zwischen Paaren dieser Mitglieder als *Kanten* repräsentiert werden können. Die Wichtigkeit eines Mitglieds lässt sich dabei durch das Gewicht seines Knotens innerhalb des Graphen abbilden, welches sich beispielsweise aus der Anzahl seiner Freundschaften berechnen lässt. Durch die Anwendung graphentheoretischer Verfahren existiert ein methodischer Ansatz, um die Strukturen sozialer Netzwerke studieren und dadurch verstehen zu können [BMBL09]. Das Interesse solche sozialen Strukturen zu studieren, um soziale Phänomene aus psychologischer oder auch wirtschaftlicher Sicht zu erklären, ist in den letzten zehn Jahren enorm gestiegen.

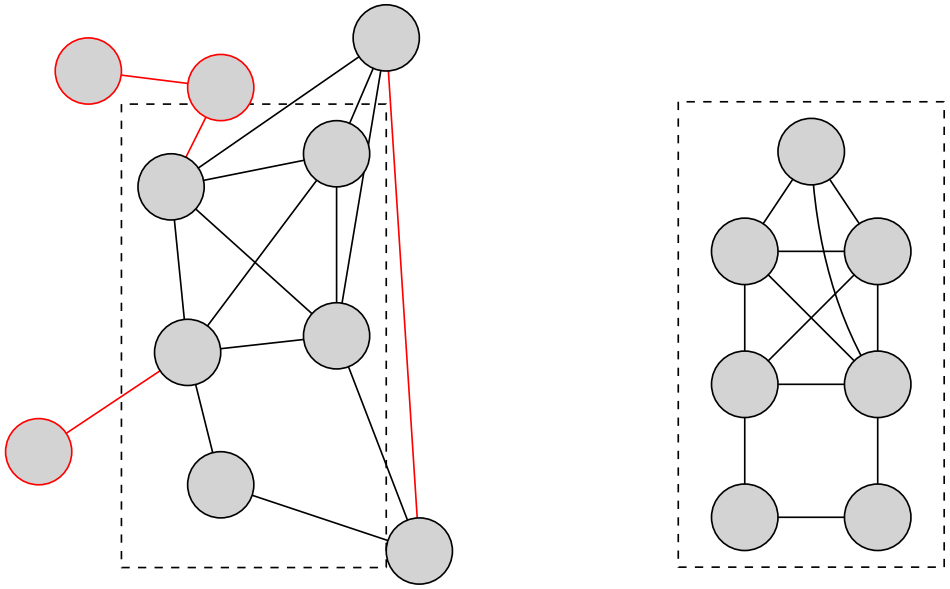
Es existieren schon gute Verfahren um solche Netzwerke als Graphen zu modellieren. Die Visualisierung solcher Graphen ist problematisch, da diese häufig sehr groß sind und nur ein beschränkter Raum, wie beispielsweise ein Blatt Papier oder ein Bildschirm, zur Verfügung steht. Da man die Mitglieder (Knoten) erkennen möchte, ist es oft unpraktikabel die Zeichnung nach unten zu skalieren, weil dann die Beschriftung nicht mehr lesbar ist. Wenn man dies aber nicht tut, kann man nicht alle Entitäten und Kantenbeziehungen abbilden.

In dieser Arbeit werden die Knoten des Graphen als geometrische Objekte dargestellt, deren Beschriftung innerhalb der Knoten steht. Es werden beispielhaft kreisförmige und rechteckige Knoten verwendet. Die Kanten werden generell als Strecken gezeichnet. Es soll nun ein Algorithmus entwickelt werden, der solche Graphen einliest, einen Teilgraphen in der gewünschten Flächenbegrenzung ausgibt und Überlappungen zwischen Knoten bzw. zwischen Kanten und Knoten vermeidet. Nachdem der Graph eingelesen wurde, wird eine rechteckige Flächenbegrenzung um die zufällig generierte Startzeichnung gelegt, die Schrittweise verkleinert wird, bis die gewünschten Werte für Breite und Höhe erreicht sind. Die Zeichnung wird dabei mehr und mehr zusammengedrückt, sodass zwischendurch Knoten oder Kanten aus dem Graph entfernt werden müssen, damit die Zeichnung überschneidungsfrei und damit leserlich bleibt. Dafür werden unter Anderem Knoten- und Kantengewichte betrachtet, die aus Anfangsgraph berechnet werden. Für die Anordnung der Knoten innerhalb der Flächenbegrenzung wird ein *kräftebasiertes* Verfahren angewendet. Nachdem die Zeichenflächenbegrenzung verändert oder ein

[1] http://www.facebook.com/
[2] http://www.plus.google.com/

Knoten bzw. eine Kante aus dem Graph entfernt wurde, werden solange Kräfte berechnet, die auf die Zeichnung einwirken, bis ein Zustand erreicht wird, an dem sich die Knoten nicht mehr großartig bewegen. Durch die Simulation von Kräften, werden die Knoten gleichmäßig im Zeichenbereich verteilt und man erhält eine schöne, weitestgehend symmetrische und gut lesbare Zeichnung des Graphen. Falls nach der Ausführung des kräftebasierten Algorithmus in der Endzeichnung Überschneidungen zwischen geradlinigen Kanten und Knoten existieren, werden diese Kanten als quadratische *Bézierkurve* um diese Knoten herum gezeichnet. Dies geschieht ebenfalls durch die Simulation von Kräften, die auf die Kontrollpunkte der Kanten wirken.

In Abbildung 1.1(a) wird beispielhaft eine Startzeichnung abgebildet, bei der das gestrichelte Rechteck den gewünschten Zeichenflächenbereich darstellt und die Knoten und Kanten, die während des Algorithmus aus den Graph entfernt werden, rot umrandet gezeichnet werden. Eine mögliche Endzeichnung ist in Abbildung 1.1(b) dargestellt, bei der die Knoten durch das kräftebasierte Verfahren relativ symmetrisch angeordnet wurden und eine der Kanten als Bézierkurve gezeichnet wurde, da diese sonst einen Knoten schneiden würde.



(a) Zufällig generierte Startzeichnung eines Graphen

(b) Ausgabe des Algorithmus mit geradlinigen und kurvigen Kanten

Abb. 1.1: Ausgangslage und Wunschendergebnis

Kräftebasierte Verfahren wurden in erster Linie für kleine Graphen entwickelt, in denen Knoten durch Punkte ohne Beschriftung repräsentiert werden, und funktionieren dort

auch dementsprechend gut. Damit man diese Verfahren auf größere Graphen mit nicht-punktartigen Knoten anwenden kann, muss man die vorhanden Verfahren dementsprechend erweitern. Es existieren bereits Ansätze, in denen kräftebasierte Verfahren auf sehr große Graphen angewendet werden, wie das Vorgehen von Bannister et al. [BEGT13], indem den Knoten Gewichte auf Grundlage ihrer Zentralität innerhalb des entsprechenden Graphen zugewiesen werden und dadurch Knoten mit größerem Gewicht in die Mitte des Graphen gezogen werden. Allerdings werden dort Knoten durch Punkte repräsentiert und es existiert keine Flächenbegrenzung, sodass weder Kanten noch Knoten entfernt werden müssten. Des Weiteren haben Dwyer et al. [DMW09] und He und Marriott [HM98] sich mit der Zeichnung von Graphen mit nicht-punktartigen Knoten in begrenzten Flächen beschäftigt, wobei kein Teilgraph erstellt wird, sondern die Zeichenfläche, sollte sie nicht ausreichen, erweitert wird. Dort wurden aber weder Teile des Graphen entfernt, noch wurde die Auslegung der Knoten im Zeichenbereich durch einen kräftebasierten Algorithmus realisiert. Es existieren aber schon Arbeiten, bei denen Kräfte auf Bézierkurven angewendet werden, wie beispielsweise die von Fink et al. [FHS$^+$12] oder Fink et al. [FHN$^+$13].

Das Neue in dieser Arbeit ist die Kombination von kräftebasierten Verfahren, Graphen mit nicht-punktartigen Knoten und der fest vorgegeben Größe des Zeichenbereichs. Damit diese Kombination zu einer schönen und übersichtlichen Zeichnung führt, werden neben (erweiterten) Kräften aus existierenden Verfahren neue Kräfte benutzt, die auf Knoten oder auf Kontrollpunkte der Bézierkurven wirken. In Kapitel 2 werden grundlegende Begrifflichkeiten der Graphentheorie und Arbeiten zu kräftebasierten Verfahren vorgestellt. Kapitel 3 beschreibt die Vorgehensweise des Algorithmus und die Kräfte, die verwendet wurden. In Kapitel 4 wird dann näher auf die Implementierung des Algorithmus und die Ausgaben, die dieser produziert, eingegangen. Anschließend werden in Kapitel 5 einige Ideen vorgestellt, wie man den Algorithmus noch verbessern könnte.

2 Grundlagen

Für den weiteren Verlauf der Arbeit ist es wichtig zu definieren, was Graphen überhaupt sind. Krumke und Noltemeier [KN09] bieten einen guten Einstieg in das Thema der Graphentheorie. Ein *(ungerichteter) Graph* G ist ein Paar (V, E), das aus einer endlichen Menge von *Knoten* V und einer endlichen Menge von *Kanten* E, bei der jede Kante $e = (u, v) \in E$ ein ungeordnetes Paar von Knoten $u, v \in V$ ist, besteht. V wird dabei als *Knotenmenge* und E als *Kantenmenge* bezeichnet. Eine Kante $e = (v, v)$ nennt man eine *Schleife*. Falls ein Graph eine Menge von Kanten $e_1 = e_2 = \ldots = e_k = (u, v)$ mit $k \geq 2$ enthält, dann nennt man $e_1, \ldots, e_k$ *Mehrfachkanten*. Graphen ohne Schleifen und ohne Mehrfachkanten nennt man *einfache* Graphen. Eine Kante e und einen Knoten v werden dann als *inzident* bezeichnet, wenn v einer der Knoten von e ist. Zwei Kanten e und f sind inzident, falls es einen Knoten v gibt, der zu beiden Kanten inzident ist. Falls e die inzidenten Knoten u und v hat, dann nennt man u und v *adjazent* bzw. *benachbart*. Der *Grad* $d(v)$ eines Knotens v ist die Anzahl seiner adjazenten Knoten. Ein *(ungerichteter) gewichteter Graph* G ist ein Quadrupel (V, E, w_v, w_e), das aus dem Graph (V, E) und zwei *Gewichtsfunktionen* $w_v \colon V \to \mathbb{R}$ und $w_e \colon E \to \mathbb{R}$ besteht. Das Gewicht eines Knotens v bzw. einer Kante e wird mit $w(v)$ bzw. $w(e)$ bezeichnet und *Knotengewicht* bzw. *Kantengewicht* genannt. Ein *benannter Graph* G ist ein Tripel (V, E, n), das zusätzlich eine Funktion $n \colon V \to \mathit{String}$ enthält, die jedem Knoten v einen Namen $n(v)$ zuordnet. Die in dieser Arbeit betrachteten Graphen sind einfach, ungerichtet, gewichtet und benannt.

Ein Graph $G = (V, E)$ wird allgemein durch eine zweidimensionale *Zeichnung* $Z(G)$ dargestellt. In einer Zeichnung Z eines Graphen G werden die Knoten als Punkte, Kreise, Ellipsen oder Polygone dargestellt. In dieser Arbeit werden Knoten ausschließlich durch Kreise oder Rechtecke visualisiert. Das kleinste achsenparallele Rechteck, das die graphische Darstellung eines Knotens v umgibt, nennt man den *Rahmen (Bounding Box)* des Knotens v. Kanten werden in dieser Arbeit entweder als Strecken oder als quadratische Bézierkurven gezeichnet.

Ein Graph wird als *planar* bezeichnet, falls er so in der Ebene gezeichnet werden kann, dass Kanten weder Knoten, noch andere Kanten außer in ihrem gemeinsamen Endpunkt schneiden. Der Schnitt zweier Kanten $i(e, f)$ in der Zeichnung, der nicht in einem gemeinsamen Endpunkt erfolgt, nennt man eine *Kantenkreuzung*. Die Qualität der Zeichnung eines Graphen wird häufig durch *(ästhetische) Kriterien* bestimmt, wie die Anzahl an Kantenkreuzungen, die Summe der Kantenlängen, die durchschnittliche Kantenlänge, die Anzahl an sich überlappenden Knoten oder die Erkennbarkeit vorhandener Symmetrien.

2.1 Kräftebasierte Verfahren

Kräftebasierte Verfahren zum Zeichnen von Graphen bestehen aus zwei Komponenten: Ein *Kräfte-Modell*, welches aus physikalischen Objekten (z.B. abstoßende Teilchen, metallische Federn) besteht, dass auf den gegebenen Graphen $G = (V, E)$ übertragen wird, und einem Algorithmus, der versucht die Knoten so in der Ebene zu platzieren, dass der gesamte Energiegehalt des physikalischen Systems ein (lokales) Minimum erreicht, indem man die wirkenden Kräfte für alle Knoten iterativ berechnet und anwendet. Es wird davon ausgegangen, dass ein minimaler Energiegehalt zu einer schönen und leicht interpretierbaren Zeichnung führt.

Kräftebasierte Verfahren werden aus zweierlei Gründen häufig für die Zeichnung von Graphen eingesetzt. Durch die Beziehung mit der physikalischen Realität sind sie vergleichsweise einfach zu verstehen und zu implementieren. Des Weiteren werden bei Graphen mittlerer Größe (ca. 50 Knoten) oft recht gute Ergebnisse erzielt.

Ein modelliertes Kriterium vieler kräftebasierter Verfahren sind Graphen mit geraden Kanten, bei denen die gewünschte Kantenlänge so gut wie möglich erreicht wird. Daraus ergibt sich, dass adjazente Knoten nahe beieinander gezeichnet werden, wenn die gewünschte Kantenlänge aller Knoten gleich ist. Des Weiteren sollen die Knoten gut in der Zeichenfläche verteilt werden und Knoten sollen sich in der Zeichnung nicht überlappen. Algorithmen, die diese beiden Kriterien kombinieren, erzeugen oft Graphen, die vorhandene Symmetrien gut darstellen können.

Das mechanische Modell von Eades [Ead84], bekannt unter dem Namen *Spring Embedder*, ist für Graphen von ca. 30 Knoten entwickelt worden und implementiert die zuvor genannten Kriterien. Der Einheitsvektor von einem Knoten u zu einem Knoten v wird im Folgenden mit $\overrightarrow{uv}$ bezeichnet, $d(u, v)$ ist der geometrische Abstand zwischen u und v. Das Modell benutzt abstoßende Kräfte

$$f_{rep}(u, v) = \frac{c_\varrho}{d(u, v)^2} \cdot \overrightarrow{uv}$$

zwischen jedem Paar von nicht adjazenten Knoten $u, v \in V$, wobei c_ϱ eine Konstante ist. Entgegengesetzte Federkräfte zwischen adjazenten Knoten $u, v \in V$, sollen diese ausreichend auseinanderziehen, aber dennoch nahe beieinanderhalten. Anstelle von realistischeren Kräften nach dem Hooke'schen Gesetz, werden (imaginäre) logarithmische Federn benutzt, deren Kraftwirkung auf weit entfernte Knoten schwächer ist. Diese üben Kräfte

$$f_{spring}(u, v) = c_\sigma \cdot \log \frac{d(u, v)}{l} \cdot \overrightarrow{vu}$$

so aus, dass die Richtung der Kraftwirkung davon anhängt, ob die aktuelle Distanz zwischen den beiden Knoten kleiner oder größer als die natürliche Länge l der Feder ist. Die Konstante c_σ kontrolliert hierbei die Federstärke. Knoten werden pro Iteration t bewegt, nachdem für alle Knoten v der resultierende Verschiebungsvektor $F_v(t)$, der die Summe aller auf v wirkenden Abstoßungs- und Federkräfte ist, berechnet und mit

der Konstanten δ multipliziert wurde. Die Konstante soll verhindern, dass Knoten aufgrund der synchronen Bewegung zu stark oszillieren. Nach der iterativen Berechnung und Anwendung der Kräfte erreicht das System einen stabilen Zustand, bei dem keine lokale Verbesserung mehr möglich ist. Der Spring Embedder erzielt, trotz seines simplen Aufbaus, in vielen Fällen schöne Ausgaben.

Der Algorithmus von Fruchterman und Reingold [FR91] versucht neben den früheren beiden Kriterien nun noch gleichmäßige Knotenverteilung zu erreichen und behandelt Knoten im Graph als atomare Partikel, die anziehende und abstoßende Kräfte aufeinander ausüben. Die Kräfte von Eades [Ead84] werden modifiziert, um eine schnellere Konvergenz zu erreichen. Abstoßende Kräfte

$$f_{rep}(u,v) = \frac{k^2}{d(u,v)} \cdot \overrightarrow{uv}$$

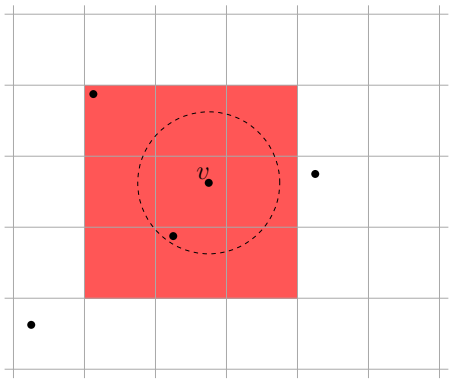
werden zwischen jedem Knotenpaar und zusätzlich anziehende Kräfte

$$f_{attr}(u,v) = \frac{d(u,v)^2}{k} \cdot \overrightarrow{vu}$$

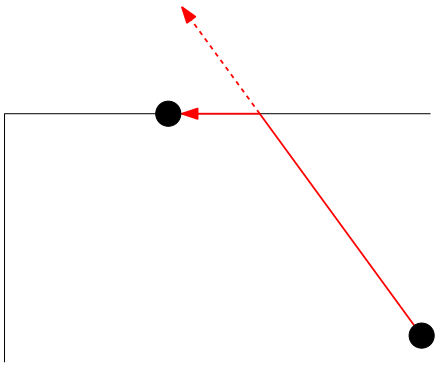
zwischen adjazenten Knoten berechnet, wobei k die optimale Kantenlänge darstellt. Die Kombination von Anziehung und Abstoßung zwischen adjazenten Knoten verhält sich wie eine Federkraft

$$f_{spring}(u,v) = f_{attr}(u,v) + f_{rep}(u,v)$$

ähnlich in der Funktion bei Eades [Ead84]. Um die Berechnung zu beschleunigen, wird die Abstoßung von weit entfernten Knoten nicht berücksichtigt, da diese vernachlässigbar wenig Einfluss auf den Verschiebungsvektor hat. Bei der verwendeten *Gitternetztechnik* werden nur Knoten, die innerhalb benachbarter Zellen liegen und deren Abstand unter einem festen Grenzwert liegt, berücksichtigt und in die Vektorsumme miteinbezogen (siehe Abb. 2.1(a)). Auch der Verschiebungsvektor wurde modifiziert. Anstelle eines konstanten Dämpfungsfaktors δ wird $F_v(t)$ von einer zeitabhängigen maximalen Verschiebungslänge $\delta(t)$ begrenzt, um starke Veränderungen, besonders in den späteren Stufen der Iteration, wenn die Verteilung nahe an einem stabilen Zustand ist, zu verhindern. Dieses Vorgehen nennt man *Simuliertes Abkühlen (Simulated Annealing)* und ist dem physikalischen Prozess des Abkühlens von geschmolzenem Material nachempfunden. Wenn geschmolzener Stahl zu schnell abgekühlt wird entstehen Bläschen, die den Stahl brüchig machen. Um dies zu verhindern muss der Stahl stetig und langsam abgekühlt werden. Dieser Prozess wird in der Metallurgie als stählen bezeichnet. Die zweite Modifikation soll sicherstellen, dass der Graph innerhalb einer vorgegebenen rechteckigen Fläche, z.B. einem Bildschirm oder einem Blatt Papier, gezeichnet wird. Falls die Verschiebung einen Knoten hinter eine Begrenzung positionieren würde, wird einfach die entsprechende Koordinate des Verschiebungsvektors begrenzt (siehe Abb. 2.1(b)). Ähnlich wie bei Eades [Ead84] ist dieses Verfahren für kleine Graphen mit weniger als 40 Knoten entwickelt worden und funktioniert dort auch dementsprechend gut.



(a) Vernachlässigung von schwachen abstoßenden Kräften



(b) Begrenzung des Vektors

Abb. 2.1: Modifikation des Spring Embedders von Fruchterman und Reingold

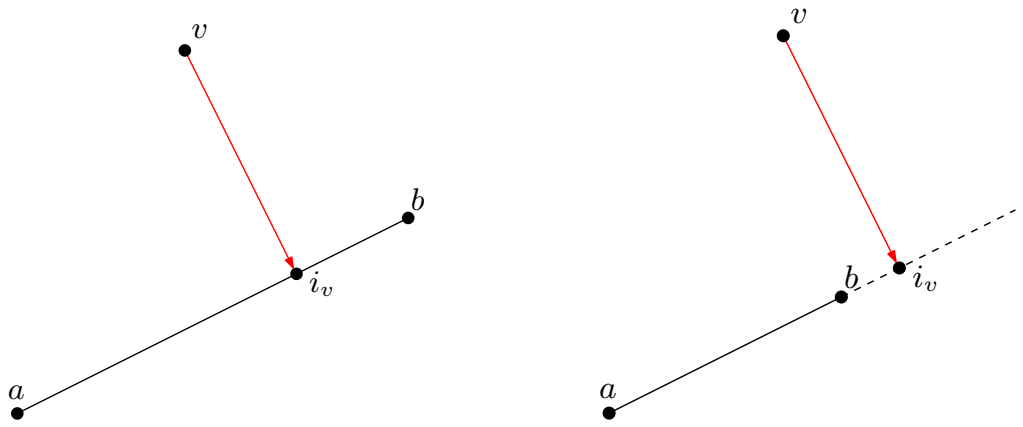
Der Algorithmus *PrEd* von Bertault [Ber00] nutzt neben den anziehenden und abstoßenden Kräften von Fruchterman und Reingold [FR91] auch noch eine weitere abstoßende Kraft zwischen Kanten und Knoten, die verhindern soll, dass aufgrund von Knotenverschiebungen neue Kantenkreuzungen entstehen. Kantenkreuzungen aus der Eingabe werden dabei beibehalten. Um diese Kraft

$$f_{edge}\left(v,(a,b)\right) = \left(\gamma - d\left(v,i_v\right)\right)^2 \cdot \overrightarrow{i_v v}$$

zwischen einem Knoten v und einer Kante (a,b) zu berechnen, wird ein neuer virtueller Knoten i_v erstellt, der die orthogonale Projektion (siehe Abb. 2.2) des Knotens v auf den Vektor, der durch die Knoten der Kante aufgespannt wird, darstellt. Die errechnete Kraft wird nur auf die Knoten v, a und b angewendet, falls der virtuelle Knoten i_v auf der Kante (a,b) liegt und der Abstand zwischen v und i_v kleiner als ein Parameter γ ist. Bertault benutzt für γ den Wert der vierfachen gewünschten Kantenlänge. Falls $v = a$ oder $v = b$ werden die Kräfte ignoriert.
Durch die Aufsummierung aller anziehenden und abstoßenden Kräfte, die auf einen Knoten v wirken, erhält man den entsprechenden Gesamtkraftvektor

$$F_v\left(t\right) = \sum_{(u,v)\in E} f_{attr}\left(u,v\right) + \sum_{u\in V} f_{rep}\left(u,v\right) + \sum_{(a,b)\in E} f_{edge}\left(v,(a,b)\right) - \sum_{\substack{u\in V, w\in V \\ (v,w)\in E}} f_{edge}\left(u,(v,w)\right).$$

Da die einzelnen Kräfte, die auf die Knoten des Graphen wirken, für jeden Knoten parallel berechnet und angewendet werden, kann nicht garantiert werden, dass ein Knoten nicht über eine Kante hinweg verschoben wird und so eine neue Kantenkreuzung entsteht. Um dieses Problem zu lösen wird jedem Knoten v eine *Zone* $Z(v)$, die angibt wohin der Knoten v verschoben werden darf, zugewiesen. Eine Zone besteht dabei aus acht Kreisbögen $Z_1(v), \ldots, Z_8(v)$. Demnach besitzt die Zone eines Knotens v acht Werte



(a) i_v liegt auf der Kante (a, b) (b) i_v liegt nicht auf der Kante (a, b)

Abb. 2.2: Bestimmung der orthogonalen Projektion i_v auf der Geraden durch a und b

$R_1(v), \ldots, R_8(v)$ für die entsprechenden Radien der Kreisbögen, die in jeder Iteration neue berechnet werden. Um die Zonen jedes Knotens des Graphen berechnen, werden zuerst die acht Werte jeder Zone auf unendlich gesetzt. Danach werden für jedes Paar von Knoten v und Kante (a, b) die Position des virtuellen Knotens i_v betrachtet:

- Falls i_v auf der Kante (a, b) liegt, ermittelt man welche Kreisbögen s von $Z(v)$ das Segment $[v, i_v]$ schneiden. Die Werte der Zonen werden wie folgt berechnet:

$$\begin{aligned} R_j(v) &= \min\left(R_j(v), d(v, i_v)/3\right) \qquad j = r(s-2), \ldots, r(s+2), \\ R_j(a) &= \min\left(R_j(a), d(v, i_v)/3\right) \qquad j = r(s+2), \ldots, r(s+6), \\ R_j(b) &= \min\left(R_j(b), d(v, i_v)/3\right) \qquad j = r(s+2), \ldots, r(s+6), \\ &\text{wobei} \quad r(j) = 1 + (j \mod 8). \end{aligned}$$

- Falls i_v nicht auf der Kante (a, b) liegt, werden die Werte der Zonen wie folgt berechnet:

$$\begin{aligned} R_j(v) &= \min\left(R_j(v), \min\left((v, a), (v, b)\right)/3\right) \qquad j = 1, \ldots, 8, \\ R_j(a) &= \min\left(R_j(a), d(v, a)/3\right) \qquad j = 1, \ldots, 8, \\ R_j(b) &= \min\left(R_j(b), d(v, b)/3\right) \qquad j = 1, \ldots, 8. \end{aligned}$$

Die Länge der Verschiebung eines Knotens v, durch die Gesamtkraft $F_v(t)$, wird begrenzt durch den Radius des Kreisbogens, der $F_v(t)$ enthält. In Abbildung 2.3 wird der Knoten v in die Richtung der Kraftwirkung von $F_v(t)$ verschoben und die Länge der Verschiebung durch $R_8(v)$ begrenzt. Kreisbögen mit einem unendlich großen Radius werden dabei gestrichelt dargestellt.
Durch dieses Vorgehen wird sichergestellt, dass neue Kantenüberschneidungen verhindert werden, während man nur einen Knoten und eine Kante betrachtet. Daraus resultiert,

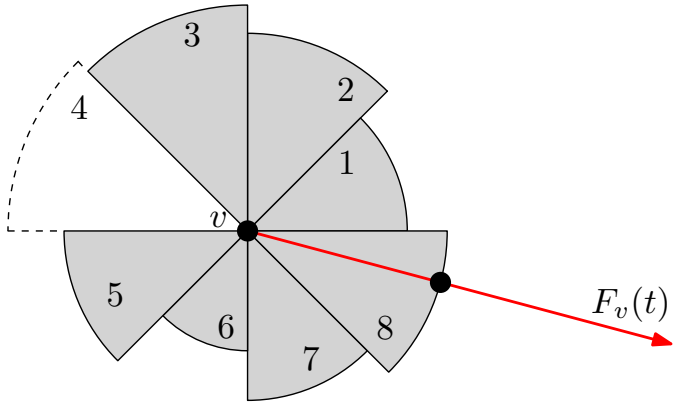


Abb. 2.3: Verschiebung des Knotens v entsprechend seiner Zone $Z(v)$

dass planare Graphen aus der Eingabe auch am Ende immer noch planar sind. Der Algorithmus erzielt schnell gute Ergebnisse für kleine Graphen. Eine Erweiterung *ImPrEd* von Simonetto et al. [SAAB11] reduziert die Berechnung signifikant und erzielt bessere Ergebnisse für größere Graphen durch verbesserte Aufteilung von Knoten und einer beschleunigten Konvergenz der Zeichnung.

Der von Lin und Yen [LY12] entwickelte Algorithmus benutzt für Anziehung und Abstoßung von Knoten die Kräfte von Eades [Ead84]. Um sehr kleine Winkel zwischen inzidenten Kanten und damit auch eine Überlappung der Kanten zu verhindern, wird eine neue abstoßende Kraft zwischen inzidenten Kanten eingeführt. Die abstoßende Kraft zwischen zwei inzidenten Kanten wird dabei nur aus den beiden Kantenlängen und dem Winkel zwischen diesen beiden Kanten berechnet. Die Stärke der Kraft soll dabei positiv mit den Abständen $d(v,a)$, $d(v,b)$ und negativ mit dem Winkel θ zwischen (v,a) und (v,b) korrelieren. Die Kraft ist definiert als

$$f_{edge}((v,a),(v,b)) = c_1 \cdot \left(\tan^{-1} \cdot \left(\frac{d(v,a)}{c_2}\right) + \tan^{-1} \cdot \left(\frac{d(v,b)}{c_2}\right)\right) + c_3 \cdot \cot\left(\frac{\theta}{2}\right),$$

wobei c_1, c_2 und c_3 Konstanten sind, um die Stärke und Auswirkung der Kraft zu kontrollieren. Da der Winkel zwischen $\overrightarrow{va}$ und $\overrightarrow{va} + \overrightarrow{vb}$ gleich $\theta/2$ ist (siehe Abb. 2.4), kann man $\theta/2$ durch

$$\frac{\theta}{2} = \cos^{-1}\left(\left(\overrightarrow{va} + \overrightarrow{vb}\right) \cdot \overrightarrow{va}\right),$$

berechnen. Die erzielten Ergebnisse verhindern 0°-Winkel zwischen inzidenten Kanten und produzieren Zeichnungen mit einem hohen Grad an Symmetrie und einer größeren durchschnittlichen Winkelauflösung als andere kräftebasierte Ansätze.

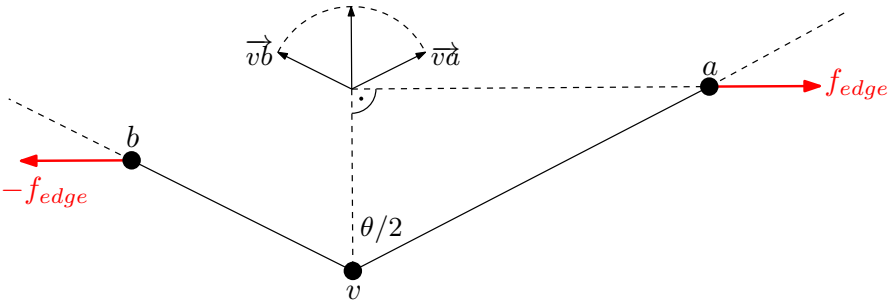


Abb. 2.4: Kraftausrichtung der abstoßenden Kraft zwischen (v, a) und (v, b) und dem eingeschlossenen Winkel θ

2.2 Energiebasierte Verfahren

Der Algorithmus von Kamada und Kawai [KK89] wählt einen anderen Weg als Eades [Ead84] und Fruchterman und Reingold [FR91] um eine gute Zeichnung zu erzielen. Anstelle Knoten anhand der wirkenden Kräften zu verschieben, wird versucht die interne Energie des physikalischen Systems direkt zu minimieren. Kamada und Kawai betrachten den gewünschten euklidischen Abstand zwischen zwei Knoten in der Zeichnung als den graphentheoretischen Abstand zwischen ihnen im entsprechenden Graphen. In diesem Modell existieren keine separaten anziehenden und abstoßenden Kräfte zwischen Knotenpaaren, stattdessen stoßen sich Knoten ab oder ziehen sich an, falls der geometrische Abstand zwischen einem Knotenpaar größer oder kleiner ist, als der entsprechende Abstand innerhalb des Graphen. Die Stärke der Feder zwischen den Knoten u und v wird durch

$$k_{u,v} = K/d_{u,v}^2,$$

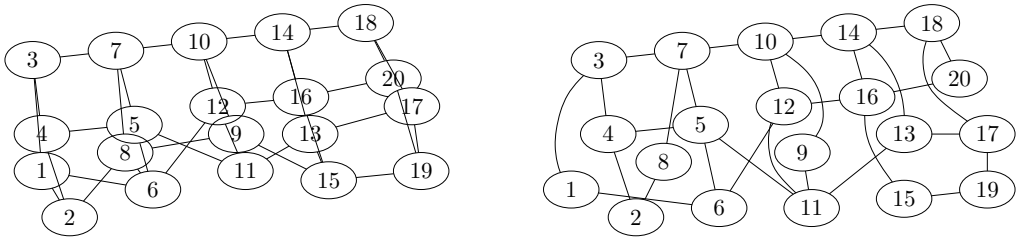
definiert, wobei K eine Konstante ist und $d_{u,v}$ die Länge des kürzesten Pfads zwischen den Knoten im Graph. Die übergeordnete Energiefunktion ist die Summe der potentiellen Energien aller $n \cdot (n-1)/2$ Federn,

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{i,j} \left((x_i - x_j)^2 + (y_i - y_j)^2 + l_{i,j}^2 - 2 l_{i,j} \sqrt{(x_i - y_i)^2 + (x_j - y_j)^2} \right)^2$$

wobei $l_{u,v} = l \cdot d_{u,v}$ die ideale Länge der Feder zwischen u und v und l die gewünschte Länge einer Kante in der Zeichnung ist. Der Algorithmus versucht durch Anwendung eines modifizierten Newton-Raphson-Verfahrens Werte für die Variablen zu finden, die die Energiefunktion $E(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$ minimieren. In einem lokalen Minimum sind alle Teilableitungen null. Dieser Zustand kann durch ein System abhängiger nichtlinearer Gleichungen ausgedrückt werden. Deshalb wird die stabile Position jedes Knotens nacheinander berechnet und angewendet. Der Algorithmus erzielt eine gute Knotenverteilung in der Zeichenfläche, bei der die graphentheoretischen Abstände miteinbezogen

werden. Allerdings ist der benötigte Rechenaufwand und Speicherplatz größer als bei anderen kräftebasierten Verfahren.

Gansner and North [GN98] haben sich mit der Kombination von kräftebasierten Verfahren und nicht-punktartigen Knoten beschäftigt. Der Algorithmus besteht aus zwei Teilen. Im ersten Teil wird der Algorithmus von Kamada und Kawai [KK89] auf die Mittelpunkte der Knoten aus dem Eingabegraph angewendet, um die initiale Auslegung zu erhalten in der dann wieder die Knoten vollständig dargestellt werden. Aufgrund der vielen möglichen Formen und Größen von Knoten können diese sich überdecken sich Knoten oder liegen auf Kanten (siehe Abb. 2.5(a)). Im zweiten Teil des Algorithmus wird nun die Zeichnung aufgeräumt, indem Überlappungen entfernt werden. Um die Überlappungen von Knoten zu beseitigen, werden iterativ Voronoi-Diagramme aus den Mittelpunkten der Knoten in der Zeichenfläche gezeichnet und die Knoten auf den Schwerpunkt ihrer jeweiligen Voronoi-Zelle bewegt. Dabei stellt der Schwerpunkt den Punkt dar, der am weitesten von jedem anderen Knoten entfernt ist und es wird sichergestellt, dass die relative Auslegung der Knoten in der Zeichenfläche beibehalten wird. Falls die Positionen aller Knoten konvergieren und noch Überlappungen vorhanden sind, muss die Zeichenfläche vergrößert werden. Nachdem die endgültige Position der Knoten gefunden wurde, bei der diese überschneidungsfrei gezeichnet werden können, müssen noch Überschneidungen zwischen Kanten und Knoten beseitigt werden. Die Kanten sollen dabei als Segmente von Linien oder Bézierkurven gezeichnet werden. Falls dies nicht möglich ist, ohne Knoten zu schneiden, wird das Problem in zwei Teilprobleme aufgeteilt und jeweils rekursiv gelöst [DGKN97]. Ein mögliches Ergebnis ist in Abbildung 2.5(b) dargestellt.



(a) Unübersichtliche Startzeichnung bei sich Knoten gegenseitig überlappen und Kanten durch Knoten verlaufen

(b) Ergebnis ohne Knotenüberlappungen und Überschneidungen zwischen Kanten und Knoten

Abb. 2.5: Startzeichnung und Ergebnis des Algorithmus von Gansner und North

Der Algorithmus erzeugt lesbare Zeichnungen durch eine Kombination aus einem physikalischen Modell und geometrischen Einschränkungen. Die originale Struktur wird in den Ausgaben gut dargestellt und es wird versucht wenig zusätzlichen Platz zu verbrauchen.

3 Kräfte-Algorithmus

Der hier vorgestellte Algorithmus wird entwickelt, um automatisch Graphen mit nicht-puntkartigen Knoten, deren Kanten und Knoten gewichtet sind, in einer vorgegeben Zeichenflächenbegrenzung zu zeichnen. Dabei werden, falls nötig, Knoten oder Kanten aus dem Graphen entfernt. Die Verteilung der Knoten des Graphen in der Zeichenfläche baut auf den bekannten kräftebasierten Verfahren auf. Durch die zufällige Anfangsverteilung der Knoten und das Zusammendrücken der Zeichnung, um die gewünschte Flächenbegrenzung zu erhalten, muss das Verfahren an einigen Stellen erweitert werden, damit man am Ende eine sinnvolle Ausgabe erhält.

Nachdem der Graph eingelesen und eine zufällige Startzeichnung erstellt wurde, wird in der Vorverarbeitung der zugrunde liegende Graph der Zeichnung verkleinert, indem gewichtsminimale Knoten und Kanten entfernt werden. Damit werden die nachfolgenden Berechnungen des Algorithmus beschleunigt. Während des Algorithmus soll die Zeichenfläche regelmäßig verkleinert werden bis die gewünschte Größe erreicht ist. Dies führt dazu, dass die Zeichnung zusammengedrückt wird und somit unübersichtlich wird. Deshalb werden zwischendurch, falls nötig, Knoten oder Kanten entfernt. Am Ende des Algorithmus sollen übriggebliebene Knoten-Kanten-Überschneidungen aus der Zeichnung aufgelöst werden, indem solche Kanten überschneidungsfrei als quadratische Bézierkurven gezeichnet werden.

In Algorithmus 1 wird das grobe Vorgehen in Pseudocode vorgestellt. In den folgenden Kapiteln werden die angesprochenen Erweiterungen und der Algorithmus genauer vorgestellt.

3.1 Vorverarbeitung

In der Vorverarbeitung wird geprüft ob ein Teil der Knoten und damit auch Kanten entfernt werden kann. Dieses Vorgehen ist sehr wichtig, da sonst die Laufzeit bei im Vergleich zum gewünschten Zeichenbereich sehr großen Graphen schwer zu kontrollieren ist. In Abschnitt 3.3 wird die Berechnung eines Gleichgewichtszustands vorgestellt. Dieser wird auch nach jeder Knoten- oder Kantenentfernung neu berechnet (siehe Abschnitt 3.5). Deshalb ist es unerlässlich in der Vorverarbeitung zu prüfen, ob Teile des Graphen gelöscht werden können, damit man die benötigte Laufzeit reduziert werden kann.

Im ersten Schritt wird die Knotenmenge V betrachtet. Da nicht-punktartige Knoten verwendet werden, kann man über die Größe der Bounding Box der Knoten ungefähr abschätzen, wie viele Knoten in den Zeichenbereich passen. Dies ist aber auch abhängig von der gewählten gewünschten Kantenlänge l_{unit}; je größer diese gewählt ist, desto weiter sind die Knoten im Schnitt in der Zeichenfläche voneinander entfernt. Da hier eine obere

Algorithmus 1: Automatisches Zeichnen von gewichteten Graphen in einer begrenzten Zeichenfläche

```
1  lese Graph G aus Datei ein und erstelle zufällig Startzeichnung Z
2  entferne Knoten und Kanten in Vorverarbeitung aus G
3  berechne Gleichgewichtszustand (F_r, F_a, F_g, F_e)
4  erzeuge den Rahmen A' um Z
5  berechne Gleichgewichtszustand (F_r, F_a, F_e, F_f)
6  while A' ≠ A do
7      passe A' um konstanten Wert an A an
8      drücke Knoten, die nicht vollständig in A' sind, in A' hinein
9      berechne Gleichgewichtszustand(F_r, F_a, F_e, F_f)
10     lösche falls nötig Knoten oder Kanten aus G
11 versuche Knoten-Kanten-Überschneidungen durch Bézierkurven aufzulösen
12 return Z
```

Abschätzung der Knotenanzahl berechnen möchte, wird jeweils die kleinste Höhe $h_{\min}$ und Breite $b_{\min}$ der Knotenrahmen verwendet. Des Weiteren wird als Abstand zwischen den Knoten nur ein Teil der gewünschten Kantenlänge verwendet. Nun wird berechnet, wie oft $b_{\min}$ unter Beachtung des Abstands in die Länge B des Rahmens hineinpasst. Analog wird dies für $h_{\min}$ und h durchgeführt. Daraus ergibt sich die maximal zulässige Knotenanzahl

$$n' = \frac{h}{l_{\text{unit}} \cdot f_{\text{pre}} + h_{\min}} \cdot \frac{b}{l_{\text{unit}} \cdot f_{\text{pre}} + b_{\min}}$$

als Produkt der beiden Berechnungen, wobei $l_{\text{unit}} \cdot f_{\text{pre}}$ eine untere Abschätzung des Abstands zwischen den Knoten darstellt. Nun werden die Knoten in V absteigend nach Gewicht sortiert und das Gewicht des Grenzknotens v' betrachtet. Der Grenzknoten ist der Knoten mit dem n'-größten Gewicht unter den Knoten in V. Abschließend werden alle Knoten deren Gewicht kleiner als das Gewicht von v' ist und damit auch deren inzidente Kanten aus dem Graphen entfernt.

3.2 Die Kräfte

Eine geeignete Auswahl und Definition der Kräfte ist der zentrale Punkt in einem kräftebasierten Algorithmus. Dieser Abschnitt beschreibt die verschiedenen Arten der Kräfte, die auf Knoten und Kanten, repräsentiert durch die beiden Endknoten, einwirken, und deren Auswirkung auf die endgültige Zeichnung.

3.2.1 Abstoßende Kraft zwischen Knoten

Die wichtigste Kraft, die nahezu jedes kräftebasierte Modell enthält, ist die abstoßende Kraft zwischen Paaren von Knoten. Diese sorgt dafür, dass Knoten sich gegenseitig weg-

drücken und in der Zeichenfläche gleichmäßig verteilt werden. Dadurch wird verhindert, dass Knoten zu nahe beieinander gezeichnet werden; es erhöht sich die Übersichtlichkeit und die Zeichnung ist dadurch angenehmer zu lesen. Je weiter zwei Knoten voneinander entfernt sind, desto schwächer ist die wirkende abstoßende Kraft. Um die Beeinflussung weit entfernter Knoten auszuschließen, wird ab einem gewissen Abstand keine Kraft mehr berechnet. Dies führt dazu, dass andere Kräfte stärkere Auswirkungen haben.

In Abbildung 3.1 sieht man wie sich die Kraft auf die drei Knotenpaare (v_1, v_2), (v_1, v_3) und (v_2, v_3) auswirkt. Die blauen Pfeile stellen dabei die Kraftvektoren zwischen den Knotenpaaren dar, die roten Pfeile zeigen die wirkende Vektorsumme des entsprechenden Knotens an. Es ist gut zu erkennen, dass sich v_1 und v_3 stärker voneinander abstoßen, als v_1 und v_2.

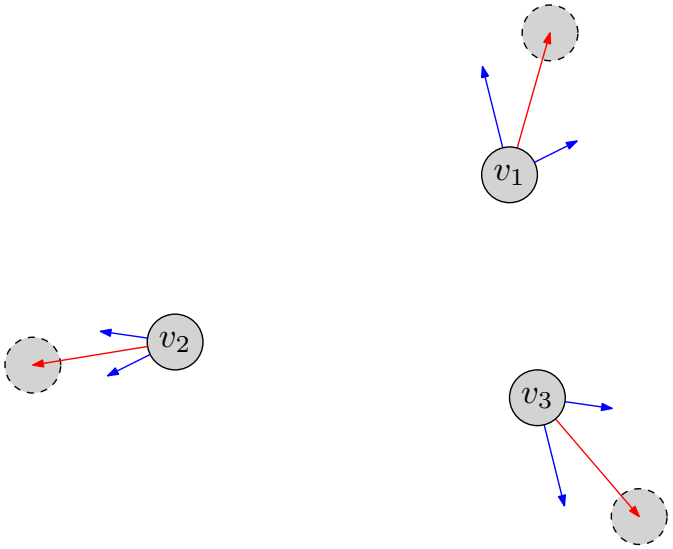


Abb. 3.1: Wirkung der abstoßenden Kräfte auf Knotenpaare

Sei (u, v) ein beliebiges Knotenpaar. Der geometrische Abstand zwischen u und v sei $d(u, v)$ und $\overrightarrow{uv}$ der dazugehörige Einheitsvektor, der von u nach v zeigt. Der gewünschte Abstand zwischen allen Knotenpaaren wird mit l_{unit} bezeichnet. Dies führt dazu, dass Knoten gleichmäßig in der verfügbaren Zeichenfläche verteilt werden. Die abstoßende Kraft, die von einem Knoten u auf einen Knoten v wirkt ist definiert als

$$F_r(u, v) = \begin{cases} \frac{{l_{\text{unit}}}^2}{d(u,v)} \cdot \overrightarrow{uv} \cdot f_r, & \text{wenn } d(u, v) < c_r \\ 0, & \text{sonst.} \end{cases},$$

wobei c_r eine Schranke für den Abstand darstellt, ab der keine Kraft mehr berechnet wird und f_r ein Faktor ist, um die Stärke der Kraft zu kontrollieren. Die abstoßende Kraft entspricht der Definition von Fruchterman und Reingold [FR91], erweitert um eine Schranke c_r.

3.2.2 Anziehende Kraft zwischen benachbarten Knoten

Eine Kante zwischen zwei Knoten stellt eine Art der Zusammengehörigkeit dieser beiden Knoten dar und deshalb sollen diese in einer Zeichnung nahe zusammen dargestellt werden. Aus diesem Grund wird eine anziehende Kraft zwischen adjazenten Knotenpaaren eingeführt, die der abstoßenden Kraft zwischen allen Knotenpaaren teilweise entgegenwirkt. Dies führt dazu, dass Knoten und ihre Nachbarknoten nahe beieinander gehalten werden und so der Verlauf von Kanten besser erkennbar ist, da diese weniger langgezogen werden. Des Weiteren nimmt der von der Zeichnung Z benötigte Platzbedarf ab.

Der Effekt der Kraft auf die drei Knotepaare (v_1, v_2), (v_1, v_3) und (v_2, v_3) wird in Abbildung 3.2 verdeutlicht. Je weiter zwei adjazente Knoten voneinander entfernt sind, desto stärker wirkt die Kraft. Die blauen Pfeile stellen die wirkende Kraft zwischen dem entsprechenden Knotenpaar dar. Die Vektorsumme, die auf Knoten wirken ist in rot dargestellt.

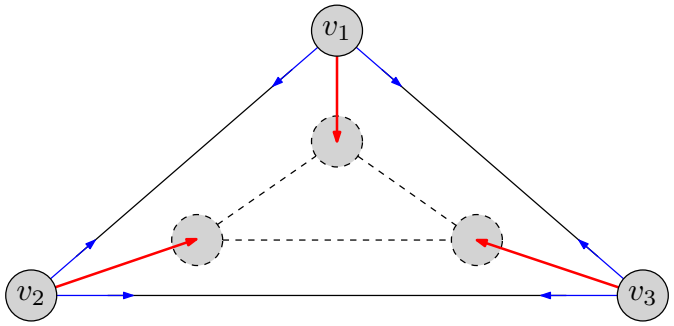


Abb. 3.2: Wirkung der anziehenden Kräfte auf Knotenpaare

Falls sich zwei adjazente Knoten berühren oder sogar überlappen sollten, wirkt diese anziehende Kraft nicht mehr, um schneller diesen Zustand zu beseitigen. Die abstoßende Kraft, die von einem Knoten u auf einen Knoten v wirkt ist dabei definiert als

$$F_a(u,v) = \begin{cases} \frac{d(u,v)^2}{l_{\text{unit}}} \cdot \overrightarrow{vu} \cdot f_a, & \text{wenn } u, v \text{ benachbart} \\ 0, & \text{sonst.} \end{cases},$$

wobei f_a die Stärke der Kraft kontrolliert. Die Kombination der abstoßenden und anziehenden Kräfte zwischen Knotenpaaren hat zur Folge, dass nicht benachbarte Knoten sich gegenseitig abstoßenden, bis der erforderliche Mindestabstand erreicht wird ab dem die abstoßende Kraft nicht mehr wirkt, und benachbarte Knoten sich abstoßen, falls sie zu nahe zusammen liegen, oder anziehen, falls sie zu weit auseinander liegen. Dieses Verhalten zwischen benachbarten Knoten kann man als Federkraft

$$F_s\left(u,v\right) = F_r\left(u,v\right) + F_a\left(u,v\right)$$

interpretieren, die sich im Gleichgewichtszustand befindet, sobald der Abstand zwischen dem Knotenpaar bzw. die Länge der Kante zwischen den beiden Knoten der gewünschten

Kantenlänge l_{unit} entspricht. Die Definition der anziehenden Kraft zwischen benachbarten Knotenpaaren und damit auch der resultierenden Federkraft enstpricht der Definition von Fruchterman und Reingold [FR91].

3.2.3 Anziehende Kraft zum Mittelpunkt der Zeichenfläche

Bei der Berechnung des ersten Gleichgewichtszustands (siehe Abschnitt 3.3) ist noch kein äußere Begrenzung A' vorhanden, die Knoten davon abhält sich ihren Kraftvektoren entsprechend beliebig auszubreiten. Dies führt dazu, dass die Größenunterschiede zwischen A und A' in Abschnitt 3.4 größer sind und dadurch entweder größere oder mehr Anpassungsschritte vorgenommen werden müssen und kann zu schlechteren Ergebnissen oder höheren Laufzeiten führen. Aus diesem Grund wird eine Anziehungskraft zwischen Knoten und dem Mittelpunkt der Zeichenfläche p_{center} eingeführt, die dazu führen soll, dass Knoten zentral angezogen werden und die flächenmäßige Ausbreitung der Knoten reduziert wird. Die Wirkung dieser Kraft ist in Abbildung 3.3 dargestellt. Je weiter eine Knoten von p_{center} entfernt ist, desto größer wirkt die Anziehung.

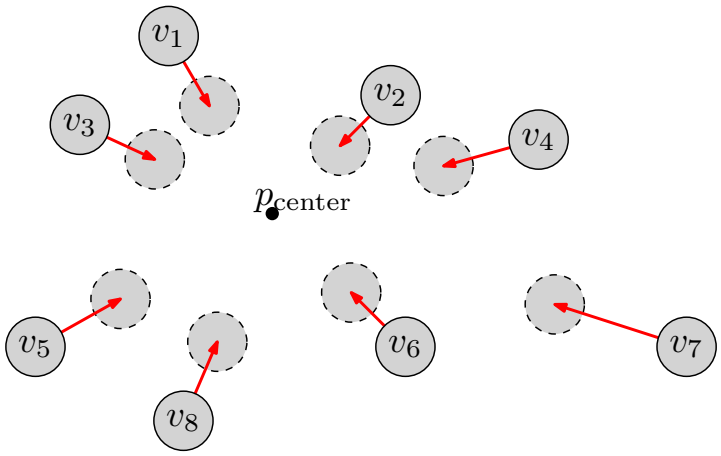


Abb. 3.3: Wirkung der zentralen Anziehungskraft auf Knoten

Die anziehende Kraft, die zwischen einem Knoten v und dem Zentrum der Zeichenfläche wirkt, ist gegeben durch

$$F_g(v) = d(v, p_{\text{center}}) \cdot \overrightarrow{vp_{\text{center}}} \cdot f_g,$$

wobei f_g ein konstanter Kontrollfaktor ist. Diese anziehende Kraft wird für jeden Knoten berechnet, unabhängig welche Entfernung dieser zum Mittelpunkt besitzt. Diese Kraft kann auch noch angewendet werden, nachdem A' erstellt wurde. Dies reduziert die Rechenzeit auf Kosten der Ästhetik der Zeichnung, da Knoten sich nicht optimal bewegen können und damit schneller ein Gleichgewichtszustand gefunden werden kann.

3.2.4 Abstoßende Kraft zwischen Kanten und Knoten

Eine Zeichnung, bei der Knoten auf Kanten liegen und diese Überdecken, ist in der Regel schwerer zu lesen, da unter Umständen nicht nachvollziehbar ist, ob eine Kante nur nahe des Mittelpunkts durch einen Knoten verläuft oder zwei Kanten darstellen soll. Deshalb wird versucht die Übersichtlichkeit der Zeichnung zu verbessern, indem eine Abstoßung zwischen Knoten und dessen nicht-inzidenten Kanten verursacht wird. Dabei wird sowohl der Knoten in eine Richtung abgestoßen, als auch die Endknoten der Kante in die entgegengesetzte Richtung (siehe Abb. 3.4). Ähnlich der abstoßenden Kraft zwischen Knotenpaaren aus Abschnitt 3.2.1, nimmt die Kraft ab, je weiter der Knoten und die orthogonale Projektion v_i auf die Kante voneinander entfernt sind. Ab einer gewissen Entfernung wird diese Kraft nicht mehr berechnet.

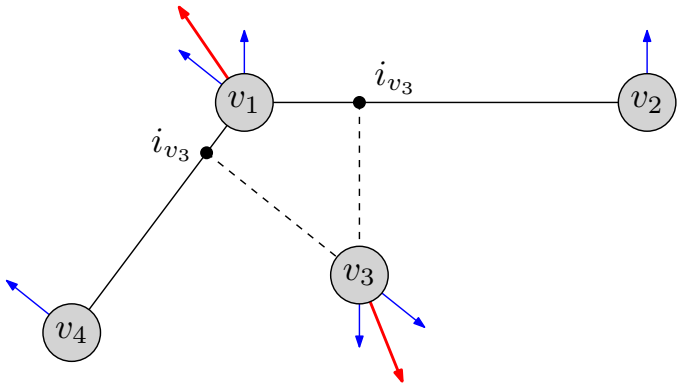


Abb. 3.4: Wirkung der abstoßenden Kraft auf Knoten und deren nicht-inzidenten Kanten

Die abstoßende Kraft zwischen einem Konten v und einer nicht-inzidenten Kante (a, b) ist definiert als

$$F_e\left(v,(a,b)\right) = \begin{cases} \left(l_{\text{unit}} - d\left(v,i_v\right)\right)^2 \cdot \overrightarrow{i_v v} \cdot f_e, & \text{wenn } i_v \text{ auf } (a,b) \text{ und } d\left(v,i_v\right) < c_e \\ 0, & \text{sonst.} \end{cases},$$

wobei f_e der entsprechende Kontrollfaktor ist und c_e eine Schranke darstellt, ab der die Kraft berechnet wird. Diese Kraftdefinition entspricht weitestgehend der von Bertault [Ber00], wobei die Schranke c_e im Gegensatz so gewählt wurde, dass die abstoßende Kraft erst dann einsetzt, falls die Kante nahe der Mitte des Knotens verläuft. In einem äußeren Bereich ist es der Kante erlaubt zu verlaufen, da dort die Lesbarkeit nicht zu stark beeinträchtigt wird und diese Überschneidung in einem späteren Schritt durch das Zeichen der Kante als Bézierkurve in Abschnitt 3.6 beseitigt werden kann. Dies ist vor allem für Knoten mit rechteckiger Form wichtig.

3.2.5 Abstoßende Kraft zwischen Knoten und Zeichenflächenbegrenzung

Damit der Graph in die vorgegebene Flächenbegrenzung gezeichnet werden kann, müssen in den meisten Fällen Teile des Graphen gelöscht werden. Um nun zu entscheiden, welcher ein Knoten gelöscht werden soll, damit der neue Teilgraph erstellt werden kann, wird in Abschnitt 3.5.1 der Druck P für Knoten berechnet. Da Knoten, die im äußeren Bereich der Zeichenfläche liegen nur eine eher einseitige Belastung aus dem inneren der Zeichenfläche erfahren, ist es wichtig dafür eine entgegengesetzte Kraft einzuführen, die diese Knoten vom Rand der Zeichenfläche wegdrückt (siehe Abb. 3.5). Dies sorgt dafür, dass der wirkende Druck auf Knoten vergleichbar ist, unabhängig ob der Knoten sich im inneren oder äußeren Bereich der Zeichenfläche befindet.

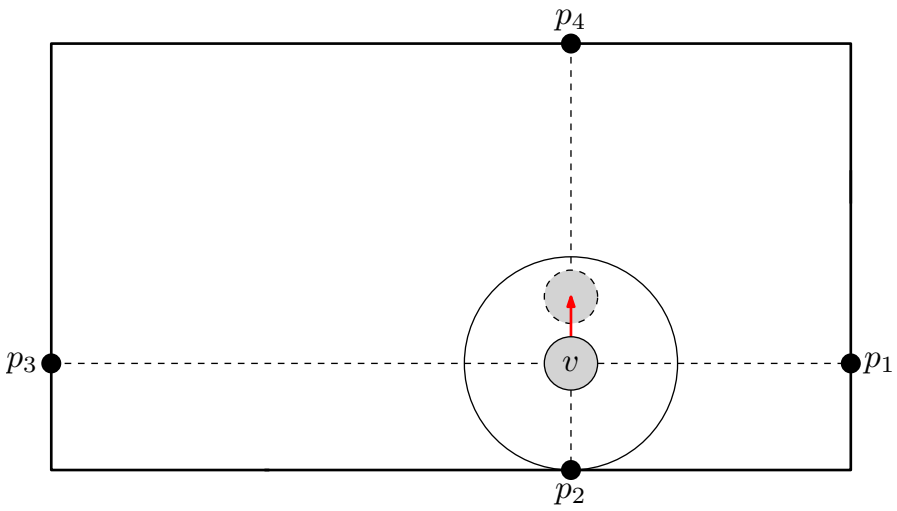


Abb. 3.5: Wirkung der abstoßenden Kraft zwischen Knoten und dem Rand der Zeichenfläche

Um die Kraftrichtung und nötige Intensität der Kraft zu ermitteln, wird der jeweils nächstgelegene Punkt auf der Zeichenflächenbegrenzung

$$p_{\text{frame}} = \arg \min_{p_i \in \{p_1, p_2, p_3, p_4\}} d(v, p_i)$$

berechnet. Die abstoßende Kraft entspricht in ihrer Berechnung und Anwendung der Abstoßungskraft zwischen Knotenpaaren aus Abschnitt 3.2.1 und ist definiert als

$$F_f(v) = \begin{cases} \frac{{l_{\text{unit}}}^2}{d(v, p_{\text{frame}})} \cdot \overrightarrow{p_{\text{frame}} v} \cdot f_f, & \text{wenn } d(v, p_{\text{frame}}) < c_f \\ 0, & \text{sonst.} \end{cases}.$$

Auch hier stellt c_f die obere Schranke für den Abstand dar, bis der die Kraft berechnet wird. Diese Kraft spielt nicht in die Verschiebung der Knoten mit ein, da dies dazu führen würde, dass Knoten nicht den Rand berühren und somit Platz im Zeichenbereich nicht genutzt werden kann. Einzig für die Berechnung des Drucks wird das Ergebnis dieser Kraft beachtet.

3.2.6 Laufzeitbewertung

Die asymptotischen Laufzeiten der in diesem Abschnitt vorgestellten Kräfte sind in Tabelle 3.1 dargestellt. Da die Berechnung der abstoßenden Kräfte zwischen Knotenpaaren mit $\mathcal{O}(|V|^2)$ und der abstoßenden Kräfte zwischen Knoten und Kanten mit $\mathcal{O}(|V| \cdot |E|)$ am aufwendigsten sind, nimmt die Berechnung aller Kräfte $\mathcal{O}(|V| \cdot (|V| + |E|))$ Zeit in Anspruch.

	F_r	F_a	F_g	F_e	F_f
Laufzeit	$\mathcal{O}(\|V\|^2)$	$\mathcal{O}(\|E\|)$	$\mathcal{O}(\|V\|)$	$\mathcal{O}(\|V\| \cdot \|E\|)$	$\mathcal{O}(\|V\|)$

Tab. 3.1: Asymptotische Laufzeit der Kräfte für eine Iteration

3.3 Berechnung des Gleichgewichtszustands

Der Gleichgewichtszustand in einem kräftebasierten Modell stellt den Zustand der Zeichnung dar, bei der der Energiegehalt des simulierten physikalischen Systems minimal ist, d.h. die Kräfte auf Knoten gleichen sich in soweit aus, dass die resultierenden Verschiebungsvektoren sehr klein sind und somit keine großartige Bewegung mehr stattfindet.

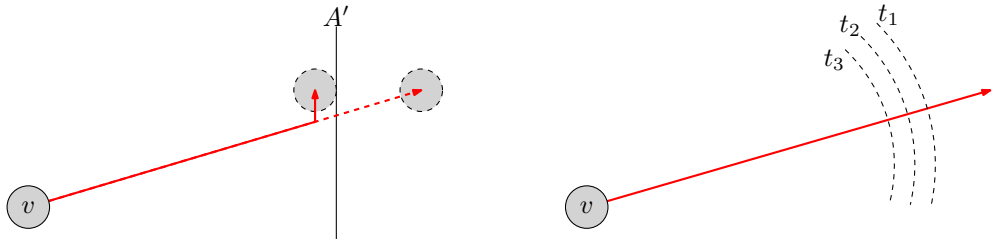
Dieser Zustand wird durch iterative Berechnung der Knotenkräfte aus Abschnitt 3.2 erreicht und ist in Algorithmus 2 dargestellt. Falls der größte Verschiebungsvektor eines Knotens kleiner als ein Wert $l_{\min}$ ist, wird die Berechnung abgebrochen. In den folgenden Abschnitten stellt t die jeweilige Iteration der Kräfteberechnung dar.

Algorithmus 2: Berechnung eines Gleichgewichtszustands

```
1  while größter Verschiebungsvektor F_v(t) < l_min do
2      for v ∈ V do
3          berechne Kräfte für v
4          if F_v(t) > l_max then
5              begrenze Länge von F_v(t)
6          verschiebe v entsprechend von F_v(t)
7          if v außerhalb von A' then
8              drücke v in A' hinein
9          l_max = l_max · f_cool
10 setze l_max auf Ausgangswert
11 berechne Stress für alle Knoten und Kanten aus G
```

In jeder Iteration werden zuerst alle Kräfte für alle Knoten berechnet. Nach der Verschiebung der Knoten muss noch getestet werden, ob diese aus dem Zeichenbereich hinausgeschoben wurden. Dies wird ähnlich dem Verfahren von Fruchterman und Reingold [FR91] durch Stutzen der x- oder y-Koordinate des Verschiebungsvektors umgesetzt (siehe Abb. 3.6(a)). Durch dieses Vorgehen kann es vorkommen, dass Knoten sehr nahe

beieinander liegen oder sich sogar überlappen. Dies führt dazu, dass bei der darauffolgenden Kräfteberechnung die Abstoßung sehr groß ist. Um dies zu unterbinden wird die maximale Länge des Verschiebungsvektors $l_{\max}$ vorgegeben (siehe Abb. 3.6(b)).



(a) Die x-Komponente des Verschiebungsvektors wird begrenzt, damit v innerhalb der Begrenzung F bleibt

(b) Der Verschiebungsvektor von v hat abhängig von der Iteration t eine maximal erlaubte Länge

Abb. 3.6: Bearbeitung und Verschiebung der Knoten nach den Kraftvektoren

Um eine schnellere Konvergenz zu erreichen wird die maximale Verschiebungslänge nach jeder Iteration um den konstanten Dämpfungsfaktor f_{cool} verkleinert. Dies hat den weiteren Vorteil, dass ein Oszillieren der Knoten, welches typisch für Spring Embedder ist, unterbunden wird. Unter dem Oszillieren der Knoten versteht man, dass ein Knoten sich nach zwei Iterationen wieder ungefähr an der Ausgangspostion befindet und so kein Zustand minimaler Energie gefunden werden kann.

Nachdem ein Gleichgewichtszustand gefunden wurde, wird die maximale Verschiebungslänge wieder auf den Ausgangswert zurückgesetzt und für alle Knoten und Kanten der Knoten- und Kantenstress 3.5 berechnet.

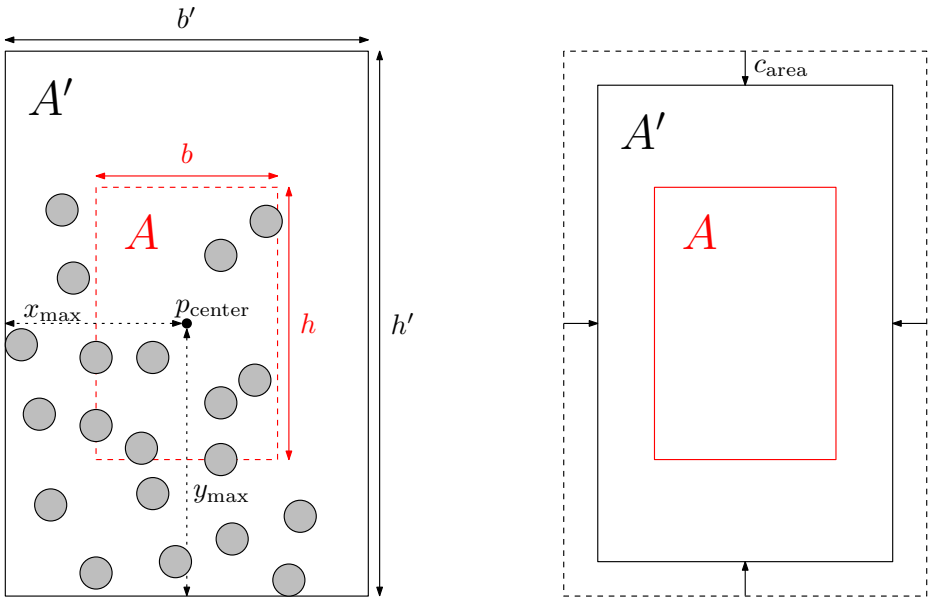
3.4 Begrenzung der Zeichenfläche

Das hier vorgestellte Verfahren versucht die Zeichenfläche zu begrenzen, indem der äußere Rahmen iterativ verkleinert und so die Zeichnung Zeichnung zusammendrückt wird. Aus der Startzeichnung Z wird sich aus dem Mittel der x- und y-Koordinaten der Knoten der Mittelpunkt p_{center} berechnet (siehe Abb. 3.7(a)). Im nächsten Schritt wird ausgehend von p_{center} die größte x- und y-Abweichung $x_{\max}$ und $y_{\max}$ berechnet. Ausgehend von diesen Werten wird der äußere Rahmen A', der durch

$$h' = y_{\max} \cdot 2 \qquad \text{und} \qquad b' = x_{\max} \cdot 2$$

gegeben ist, erstellt. Um die gewünschte Flächenbegrenzung zu erreichen, wird A' iterativ verkleinert, indem h' und b' um einen konstanten Wert c_{area} verkleinert werden (siehe Abb. 3.7(b)), bis b' gleich b und h' gleich h ist, d.h. A' entspricht A.

Je nachdem wie die Knoten in der Zeichenfläche vor der Verkleinerung des Rahmen verteilt waren und die Größe von c_{area} gewählt wurde, kann es passieren, dass Knoten



(a) Der äußere Rahmen A' wird auf Basis der Verteilung der Knoten erstellt

(b) Der Breiten- und Höhenunterschied von A' werden um den gleichen Wert angepasst

Abb. 3.7: Erstellung und Anpassung des äußeren Rahmens A'

nach dem Verkleinerungsschritt außerhalb von A' liegen oder A' schneiden. Damit nun für die folgende Berechnung eines Gleichgewichts keine Probleme mit der Berechnung Abstoßungskraft zwischen Knoten und A' entstehen, muss gewährleistet werden, dass nach der Anpassung alle Knoten innerhalb von A' liegen. Ähnlich wie in Abschnitt 3.3 werden auch hier die die x- und y-Werte der Knoten solange angepasst, bis der Knoten wieder vollständig innerhalb der Begrenzung liegen (siehe Abb. 3.8).

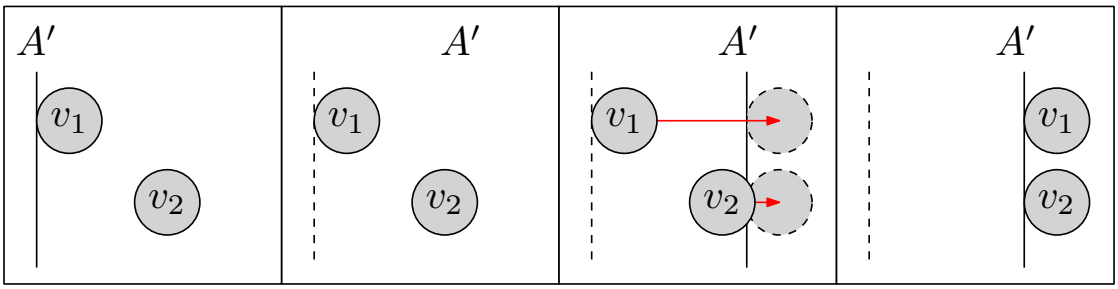


Abb. 3.8: Hineindrücken der Knoten in die neue Flächenbegrenzung

3.5 Bildung des Teilgraphen

Um den Graphen mit der gewünschten Kantenlänge l_{unit} in die vorgegebene Zeichenfläche zeichnen zu können ist es in den meisten Fällen unerlässlich bestimmte Teile des Graphen zu löschen und so einen *Teilgraph* zu erstellen.

Der Ablauf der Teilgraphbildung innerhalb des Zeichenalgorithmus wird im Algorithmus 3 veranschaulicht. Um herauszufinden, ob überhaupt Elemente des Graphen weggelassen werden müssen, betrachtet man den festgelegten minimalen Abstand zwischen nicht-adjazenten Knoten l_{nadj} und die festgelegte minimale Kantenlänge l_{adj}. Damit kann man abschätzen, ob die Zeichnung zu sehr zusammengedrückt ist und man Elemente löschen muss, um mehr Platz zu schaffen. Die Unterteilung der Abstände zwischen adjazenten und nicht-adjazenten Knoten ist wichtig, da bei der kräftebasierten Knotenverteilung nicht-adjazente Knoten weiter voneinander entfernt sein sollen als Knoten, die durch eine Kante miteinander verbunden sind.

Algorithmus 3: Bildung des Subgraphen

```
1 while minimale Kantenlänge < l_adj or minimale Abstand zwischen
    nicht-benachbarten Knoten < l_nadj do
2     if durchschnittliche Kantenlänge < f_decision · l_unit then
3         entferne Knoten mit größtem Stress
4     else
5         entferne Kante mit größtem Stress
6     berechne Gleichgewichtszustand (F_r, F_a, F_e, F_f)
```

Falls eines dieser Kriterien erfüllt ist, muss entschieden werden, ob ein Knoten oder eine Kante entfernt werden soll. Dies geschieht über die durchschnittliche Kantenlänge der Zeichnung. Falls diese im Vergleich zur gewünschten Kantenlänge, dargestellt als $f_{\text{decision}} \cdot l_{\text{unit}}$, als zu klein bewertet wird, kann man daraus folgen, dass zu viele Knoten in der Zeichenfläche sind und Kanten deshalb nicht länger gezeichnet werden können. Deshalb ist es in dieser Situation sinnvoll einen Knoten anstatt einer Kante auszuwählen. Andernfalls wird eine Kante entfernt.

Nachdem ein Knoten oder eine Kante entfernt wurde, berechnet man einen neuen Gleichgewichtszustand, um eine neue Zeichnung zu berechnen und so aktuelle Informationen zu erhalten, ob der Vorgang wiederholt werden muss.

Das Ziel ist es die wichtigsten Kanten und Knoten im Endgraphen zu repräsentieren. Dadurch muss nun konkret entschieden werden, welcher Knoten oder welche Kante entfernt wird. Das Gewicht der Kanten und Knoten beschreibt in die Wichtigkeit in der Regel ziemlich gut, aber um eine gute Zeichnung am Ende zu erhalten, müssen weitere Faktoren mit eingerechnet werden, um bestimmen zu können welche Kante oder welchen Knoten entfernt werden soll. Aus diesem Grund wird der *Knotenstress* und der *Kantenstress* eingeführt, durch den der zu löschende Knoten oder die zu löschende Kante bestimmt wird.

3.5.1 Stressberechnung für Knoten

Damit man Knoten in der Zeichenfläche mehr Bewegungsraum zur Verfügung stellen kann, ist es in manchen Fällen sinnvoll einen Knoten aus dem Graphen bzw. der Zeichnung zu entfernen. Aus diesem Grund wird im ersten Schritt derjenige Knoten gesucht, der am Ende des letzten stabilen Zustands die größte „Unruhe“ besitzt, d.h. den größten Drang hat sich zu bewegen, aber dies durch die auf ihn wirkenden Kräfte nicht tun kann. Der resultierende Gesamtkraftvektor $F_v(t)$ bildet dieses Kriterium aber nicht gut genug ab. Wenn beispielsweise $F_v(t)$ die Länge 0 besitzt, heißt das nicht automatisch, dass momentan keine Kräfte auf v wirken, denn es ist möglich, dass sich alle Kraftvektoren gegenseitig ausgleichen und so in der Summe einen Nullvektor erzeugen.

Deshalb wird der *Knotendruck* $P_v(t)$ eingeführt, der genau dieses Verhalten besser beschreiben soll, als $F_v(t)$. Um den Druck zu berechnen teilt man die einzelnen wirkenden Kraftvektoren, die auf v wirken, in acht Zonen ein und betrachtet fortan nur noch die Gesamtvektoren $\Upsilon_i(v)$ der entsprechenden Zonen (siehe Abb. 3.9, Links und Mitte). Die Vektoren jeder Zone sollen mit der gegenüberliegende Zone und deren beiden Nachbarzonen verglichen werden. Der Teildruck berechnet sich dann folgendermaßen:

$$Z_i(v) = \max\left(\min\left(\Upsilon_i(v), \Upsilon_{r(i+3)}(v)\right), \min\left(\Upsilon_i(v), \Upsilon_{r(i+4)}(v)\right), \min\left(\Upsilon_i(v), \Upsilon_{r(i+5)}(v)\right)\right),$$
$$\text{wobei } r(j) = j \mod 8.$$

Dabei wird der Kraftvektor jeder Zone mit den der drei gegenüberliegenden verglichen und am Ende das Maximum aller Vergleiche gewählt. Im Einzelvergleich ist es aber nötig das Minimum zu berechnen, da sonst Fälle in denen keine gegensätzlichen Kräfte existieren nicht erkannt werden. Durch die Berechnung des Teildrucks jeder Zone erhält man den Gesamtdruck als

$$P_v(t) = \max\left(Z_1(v), Z_2(v), \ldots, Z_8(v)\right).$$

Die Berechnung des Teildrucks zweier Zonen und des Gesamtdrucks eines Knotens v wird in Abbildung 3.9 dargestellt, wobei die betrachteten Zonen grau schraffiert sind.

Mit dem Knotendruck wurden nun die Knoten des Graphen hinsichtlich ihrer Position innerhalb der Zeichnung bewertet. Allerdings ist das Ziel möglichst unwichtige Knoten aus der Zeichnung zu entfernen. Die *Wichtigkeit* wird hierbei durch das Knotengewicht und den Knotengrad ausgedrückt. Deshalb wird im zweiten Schritt der Knotenstress

$$S_v(t) = \frac{P_v(t)}{w(v) \cdot (d(v) + 0.001)}$$

berechnet, der den Knotendruck und das Knotengewicht miteinbezieht, wobei $d(v)$ de Grad des Knotens ist. Der Knotenstress $S_v(t)$ soll somit eine aussagekräftige Bewertung darstellen, nach der Knoten aus der Zeichnung entfernt werden.

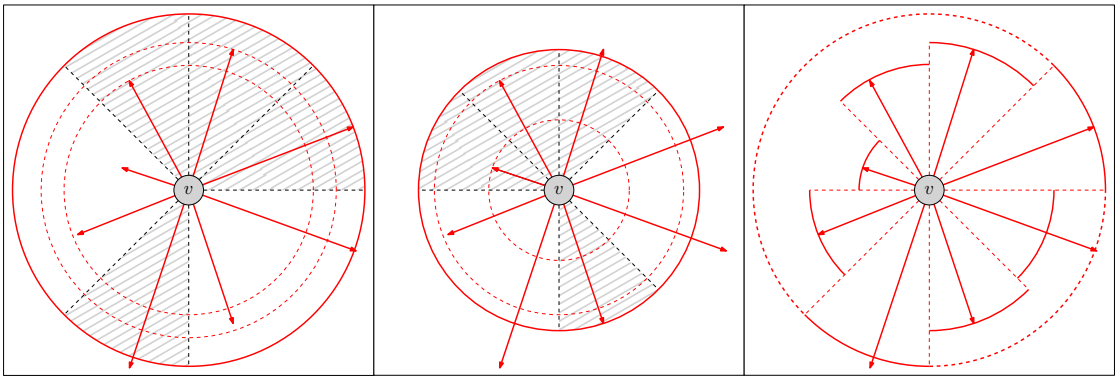


Abb. 3.9: Druckberechnung zweier Zonen des Knotens v (Links, Mitte) und der resultierende Gesamtdruck $P(v)$ (Rechts)

3.5.2 Stressberechnung für Kanten

Die Bewertung der Kanten findet in einem ähnlichen Verfahren statt. Auch hier soll sowohl die Wichtigkeit, also das Gewicht, als auch der Einfluss auf die Ästhetik der Zeichnung mit einfließen.

Innerhalb der Zeichnung werden diejenigen Kanten als störend dargestellt, die im Vergleich besonders viele Kantenkreuzungen besitzen. Denn dies bedeutet, dass diese Kanten in einem eher dicht vernetzten Abschnitt verlaufen und verhindern, dass sich ihre Start- und Endknoten optimal verteilen können.

Damit aber trotzdem Kanten mit geringerem Gewicht bevorzugt entfernt werden, wird das Gewicht der Kante mit der Summe der Gewichte der geschnittenen Kanten verglichen. Daraus resultierend lässt der Kantenstress berechnen als

$$S(e) = \frac{|E_i|}{w\,(e)} \cdot \sum_{f \in E_i} w\,(f),$$

wobei E_i die Menge der Kanten ist, die eine Kantenkreuzung mit der Kante e besitzen.

3.6 Verhindern von Überschneidungen durch Bézierkurven

Vor dem Nachbearbeitungsschritt werden Kanten in der Zeichnung als Strecken zwischen Knotenpaaren dargestellt. Durch das Platzproblem in der Zeichenfläche und das dadurch bedingte Zusammendrücken der Zeichnung, kann es vorkommen, dass Kanten durch Knoten verlaufen und diese schneiden. Mit einer andere Definiton der abstoßenden Kraft zwischen Knoten und Kanten aus Abschnitt 3.2.4 hätte man dieses Phänomen beseitigen können. Dies würde aber dazu führen, dass mehr Kanten aus der Zeichnung entfernt werden. Die Überschneidungen treten vor allem bei Zeichnungen auf bei denen Knoten als Rechtecke gezeichnet werden, deren Breite und Höhe sehr unterschiedlich zueinander sind (siehe Abb. 3.10). Aus diesem Grund sollen die Kanten, die sich mit

Knoten überschneiden, nicht mehr als Strecken, sondern als quadratische Bézierkurven dargestellt werden.

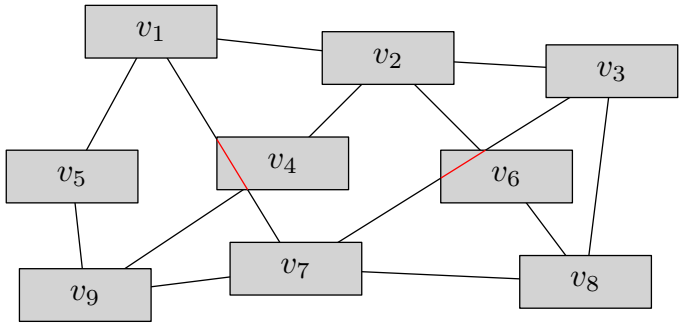


Abb. 3.10: Überschneidungen zwischen rechteckigen Knoten und Kanten

Eine *Bézierkurve* ist eine parametrisch modellierte Kurve, die durch zwei Endpunkte und beliebig viele Kontrollpunkte definiert ist. Zum Zeichnen der Kanten werden hier quadratische Bézierkurven verwendet, welche nur einen Kontrollpunkt besitzen. Man nennt sie quadratisch, da das Polynom, das die Kurve beschreibt ein Polynom zweiten Grades ist. Die Punkte, die auf einer quadratischen Bézierkurve liegen, sind festgelegt durch

$$B_2(t) = (P_0 - 2P_1 + P_2) \cdot t^2 + (-2P_0 + 2P_1) \cdot t + P_0 \text{ mit } t \in [0; 1],$$

wobei P_0 und P_2 die Endpunkte sind und P_1 der Kontrollpunkt ist. Eine geradlinige Kante in eine quadratische Bézierkurve umzuwandeln ist sehr einfach, da die Endpunkte genau den Knoten entsprechen, die die Kante miteinander verbinden soll. Für den Anfang reicht es, den Kontrollpunkt auf den Mittelpunkt der Kante zu legen. Damit wird durch die Bézierkurve immer noch eine Strecke dargestellt. Die Bézierkurven werden durch eine vorgegebene Anzahl c_{bezier} an Streckensegmenten genähert, was geometrische Berechnungen, wie Schnittpunkte und Abstände, vereinfachen soll. Um die endgültigen Positionen für die Kontrollpunkte der Kanten zu finden, wird ein ähnliches Verfahren wie in Abschnitt 3.3 angewendet und ist in Algorithmus 4 dargestellt.

Die endgültigen Positionen der Kontrollpunkte werden ermittelt, indem für jeden Kontrollpunkt ein Gleichgewichtszustand berechnet. Dabei werden iterativ die abstoßenden und anziehenden Kräfte zwischen Knoten und dem Kontrollpunkt (siehe Abschnitt 3.6.1 und 3.6.2) berechnet. Auch hier wird die Berechnung abgebrochen sobald der Kraftvektor $F_{P_1}(t)$ unter den Wert l_{min} fällt. Da in vielen Fällen die Kante durch einen Knoten verläuft, ist die Abstoßungskraft der ersten Iteration sehr groß. Aus diesem Grund muss auch die maximale Länge des Verschiebungsvektors einer Iteration um den Wert l_{max} begrenzt werden, der nach jeder Iteration um den Dämpfungsfaktor f_{cool} reduziert wird. Falls nach der Berechnung des Gleichgewichtszustands immer noch Knoten oder der Rand der Zeichenfläche geschnitten werden, wird die Kante aus der Zeichnung entfernt.

Algorithmus 4: Verhinden von Knoten-Kanten-Überschneidungen durch quadratische Bézierkurven

```
for e ∈ E do
    if e verläuft zu nahe an oder schneidet Knoten aus V then
        while größter Verschiebungsvektor F_v(t) < l_min do
            berechne Kräfte für P_1
            if F_{P_1}(t) > l_max then
                begrenze Länge von F_{P_1}(t)
            verschiebe P_1 entsprechend F_{P_1}(t)
            l_max = l_max · f_cool
        setze l_max auf Ausgangswert
    if e schneidet einen Knoten aus V or e schneidet den Rahmen A then
        entferne e aus Z
```

3.6.1 Abstoßende Kraft zwischen Kontrollpunkten und Knoten

Diese Kraft stößt Kanten von Knoten ab, indem der Kontrollpunkt P_1 der Kante verschoben wird. Diese Funktion verhindert Überschneidungen und sorgt dafür, dass Kanten nicht zu nahe an Knoten vorbei verlaufen und die Übersichtlichkeit der Endzeichnung erhöht wird.

Für jedes Paar von Knoten und Kante wird der Punkt auf dem Knotenrand und der Punkt auf der Bézierkurven-Repräsentation der Kante gesucht, für den der Abstand zwischen diesen beiden Punkten minimal ist. Die Kraft wird dabei für den berechneten Punkt auf der Kante berechnet, aber auf den Kontrollpunkt angewendet. In Abbildung 3.11 ist dieses Vorgehen dargestellt worden, wobei die berechneten Punkte auf den Knotenrändern als Quadrate, die auf den Kanten als Punkte dargestellt wurden. Der Kontrollpunkt wird um den Gesamtvektor aller drei Knoten verschoben.

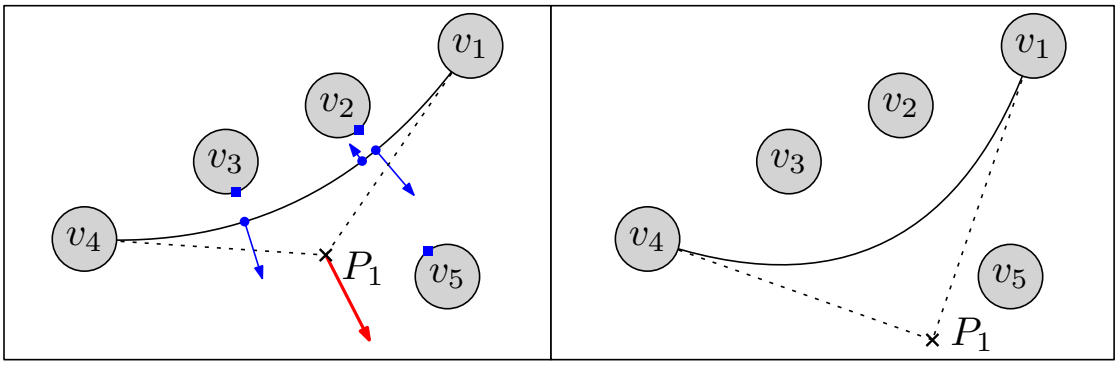


Abb. 3.11: Wirkung der abstoßenden Kraft zwischen Knoten und dem Kontrollpunkt einer Kante

Die abstoßende Kraft ist in ihrer Funktionsweise ähnlich der abstoßenden Kraft zwischen Knoten aus Abschnitt 3.2.1. Der Abstand und die Richtung werden aus den berechneten Punkten auf dem Knotenrand und der Bézierkurve ermittelt. Der gewünschte Abstand entspricht dem Kreis um v mit Radius $r = \sqrt{{b_v}^2 + {h_v}^2}$, wodurch sichergestellt werden soll, dass die Kante nach der Berechnung des Kräftegleichgewichts außerhalb von Knoten verläuft. Dadurch lässt sich die abstoßende Kraft berechnen als

$$F_{rc}\left(v,(a,b)\right) = \begin{cases} \frac{{b_v}^2+{h_v}^2}{d\left(p_v,p_{(a,b)}\right)} \cdot \overrightarrow{p_v p_{(a,b)}} \cdot f_{rc}, & \text{wenn } d\left(p_v,p_{(a,b)}\right) < c_{rc} \\ 0, & \text{sonst.} \end{cases},$$

wobei diese nur für Knoten berechnet wird, deren Abstand zur Kante kleiner als der Parameter c_{rc} ist. Die Punkte p_v und $p_{(a,b)}$ sind die beiden berechneten Bezugspunkte des Knotens und der Kante.

3.6.2 Anziehende Kraft zwischen Kontrollpunkten und Knoten

Die anziehende Kraft stellt das Pendant zur zuvor vorgestellten abstoßenden Kraft zwischen Kontrollpunkten und Knoten dar. Dabei soll verhindert werden, dass Kanten, bzw. ihre Kontrollpunkte, zu weit von ihrer ursprünglichen Position wegbewegt werden.

Auch hier werden die gleichen zwei Bezugspunkte auf dem Rand des Knotens und der Kante ermittelt und die Kraft auf den Kontrollpunkt angewendet (siehe Abb. 3.12).

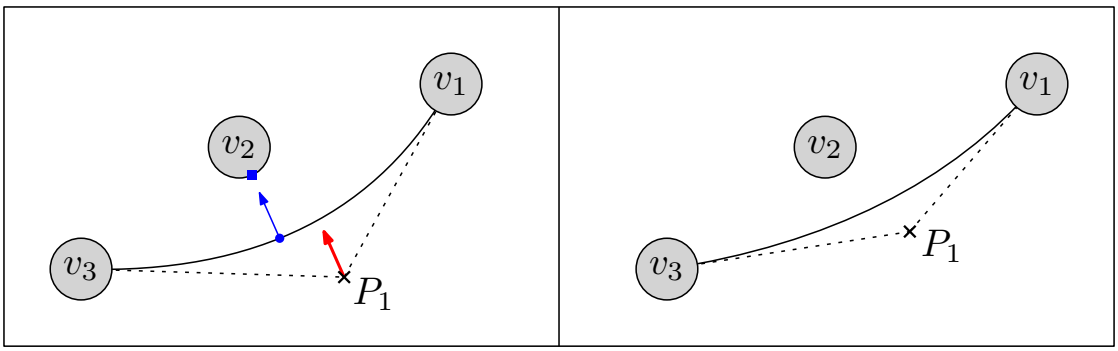


Abb. 3.12: Wirkung der anziehenden Kraft zwischen einem Knoten und dem Kontrollpunkt einer Kante

Diese Kraft stellt eine angepasste Version der Anziehungskraft zwischen benachbarten Knotenpaaren aus Abschnitt 3.2.2 dar. Die Kraft ist definiert als

$$F_{ac}\left(v,(a,b)\right) = \begin{cases} \frac{d\left(p_v,p_{(a,b)}\right)^2}{\sqrt{{b_v}^2+{h_v}^2}} \cdot \overrightarrow{p_{(a,b)} p_v} \cdot f_{ac}, & \text{wenn } v \in V_i \\ 0, & \text{sonst.} \end{cases},$$

wobei V_i die Menge der Knoten ist, die von der Kante (a,b) geschnitten wurden oder an denen die Kante zu nahe vorbei verlaufen ist, bevor versucht wurde dies in Abschnitt 3.6 zu ändern.

4 Implementierung und Auswertung

Der Algorithmus wurde in Java implementiert, wobei zur Speicherung des Graphen die Java-Graphen-Bibliothek JUNG[1] verwendet wurde. Die Speicherung und Verwaltung des Graphen ist dadurch ziemlich einfach und viele Grundfunktionen, wie das Finden von inzidenten Kanten oder adjazenten Knoten, sind schon vorhanden. Des Weiteren ist es bei dieser Bibliothek möglich eigene Knoten- und Kantenklassen zu definieren, die dann im Graph gespeichert und verwendet werden. Dies erleichtert die Arbeit ungemein und lässt zu, dass man alle benötigten Informationen, wie Position, Form und Kräfte in den Knoten oder die Position der Kontrollpunkte in den Kanten speichern kann.

Die Eingaben bestehen entweder aus puren XML- oder aus GraphML-Dateien[2]. Das Letztgenannte ist ein XML-basiertes Dateiformat, dass die Struktur von Graphen definiert und einfach um applikationsspezifische Daten erweiterbar ist.

Die Ausgaben des Algorithmus werden im Ipe-, DOT- und GraphML-Format gespeichert. Das Ipe-Format basiert ebenfalls auf XML und kann mit dem Zeichenprogramm Ipe[3] bearbeitet und in PDF-Dateien umgewandelt werden. Dateien im DOT-Format können durch GraphViz[4], einem Programm zur Visualisierung von Graphen, eingelesen und mit den dort angebotenen Algorithmen gezeichnet und ausgegeben werden.

Für die in den folgenden Abschnitten durchgeführten Tests wurden fünf verschiedene Graphen benutzt (siehe Tabelle 4.1). Die Graphen beinhalten Informationen über Mehrautorenschaft unter den eingereichten und akzeptierten wissenschaftlichen Veröffentlichungen bei dem *International Symposiom on Graph Drawing*[5] über den Zeitraum von 1994-2012. Für alle durchgeführten Test wurde als Größe der Zeichenfläche die eines Din-A4-Blattes in quer (29,7 cm x 21 cm) verwendet. Für die Tests wurde ein Rechner mit einem Intel Core i5-2500K Prozessor und 8 GB Arbeitsspeicher benutzt.

Graph	Knotenanzahl	Knotengewicht	Kantenanzahl	Kantengewicht
G_1	181	240	199	432
G_2	368	677	606	1496
G_3	545	1202	1034	2746
G_4	715	1754	1520	4144
G_5	950	2559	2334	6564

Tab. 4.1: Die verwendeten Testgraphen

[1] http://www.jung.sourceforge.net/
[2] http://www.graphml.graphdrawing.org/
[3] http://www.ipe7.sourceforge.net/
[4] http://www.graphviz.org/
[5] http://www.graphdrawing.org/

4.1 Aufbau des Programms

Die wichtigsten Klassen des Programms werden im Folgenden vorgestellt:

- **Vertex:** Ist eine abstrakte Klasse, die alle Knotenübergreifenden Informationen speichert und Berechnungen ausführt. Dazu zählen die Stressberechnung, Berechnung des Abstands und das Prüfen auf Überschneidung mit anderen Knoten. Dadurch ist es ziemlich einfach neue Knotenklassen zu implementieren und in das Programm einzubinden. Hier werden auch die Kräfte für Knoten gespeichert, bis sie am Ende einer Iteration angewandt werden.

- **CircleVertex** und **RectangleVertex:** Diese Klassen sind von Vertex abgeleitet und stellen die konkreten Knotenklassen dar, die die Form der Knoten vorgeben. Hier werden Abstände und eventuelle Schnittpunkte mit Kanten berechnet.

- **BezierEdge:** Speichert die Informationen der Kanten, wie beispielsweise das Gewicht, den Stress, die Position des Kontrollpunkts und das Polynom der quadratischen Bézierkurven. Des Weiteren ist sie für die Berechnung von Schnittpunkten mit anderen Kanten verantwortlich. Geradlinige Kanten werden auch als Bézierkurven dargestellt, indem der Kontrollpunkt auf dem Start- oder Endknoten der Kante liegt.

- **Frame:** Diese Klasse ist für die Repräsentation der Zeichenflächenbegrenzung zuständig. Hier wird der Mittelpunkt der Zeichnung ermittelt, die Anpassung des Rahmens vorgenommen, der nächstliegendste Punkt eines Knotens auf der Begrenzung berechnet und getestet ob die Bézierkurven-Repräsentation einer Kante außerhalb der Zeichenfläche verläuft.

- **GraphMLReader** und **XMLReader:** Lesen Eingaben im GraphML- und XML-Format ein. Hier kann auch festgelegt werden, welche Knotenform verwendet werden soll und ob die Abmessungen der Knoten aus der Datei ausgelesen oder selbst aus der Größe der Beschriftung berechnet werden sollen.

- **IPEWriter**, **GraphMLWriter** und **DOTWriter:** Enthalten jeweils Methoden um die aktuelle Zeichnung in eine IPE-Datei, GraphML-Datei oder eine DOT-Datei zu schreiben.

- **ForceDirectedAlgorithm:** Dies ist die Hauptklasse des Algorithmus. Hier ist das kräftebasierte Verfahren und die Definition der Kräfte implementiert. Des Weiteren findet hier der Vorverarbeitungsschritt, die Berechnung des Gleichgewichtszustands und die abschließende Auflösung von Überschneidungen zwischen Knoten und Kanten statt.

- **Main:** Die Klasse enthält die Main-Methode und startet das Programm. Hier können viele Werte des Algorithmus verändert werden, die die Qualität der Ausgabe und Laufzeit des Algorithmus beeinflussen.

4.2 Optimierung der Laufzeit durch Mehrkernnutzung

Ein Großteil der Laufzeit des Algorithmus wird durch die Kräfteberechnung bei der Ermittlung der Gleichgewichtszustände (siehe Abschnitt 3.3) in Anspruch genommen. Da innerhalb einer Iteration die Berechnung der Kräfte die auf die Knoten des Graphen wirken nicht voneinander abhängen, ist es sinnvoll diese Berechnungen zu parallelisieren.

Es wird für jeden der zur Verfügung stehenden Kerne ein *Thread* gestartet, der für einen Teil der Knotenmenge des Graphen, die auf die Knoten dieser Teilmenge wirkenden Kräfte berechnet. Dazu werden jedem Thread die entsprechende Teilknotenmenge und die Referenzen auf das Graph-Objekt, die gewünschte Kantenlänge l_{unit} und das Objekt, welches die Zeichenflächenbegrenzung repräsentiert, übergeben. Die Zeichenflächenbegrenzung ist nötig, um p_{center} aus Abschnitt 3.2.3 und p_{frame} aus Abschnitt 3.2.5 berechnen zu können. Da die einzelnen Kraftvektoren in den Knoten-Objekten gespeichert werden, können, nachdem alle Threads einer Iteration terminiert sind, die Gesamtvektoren $F_v(t)$ berechnet und die Knoten dementsprechend verschoben werden.

Um die Auswirkung dieser Mehrkernoptimierung zu verdeutlichen, wurde ein Test mit dem Testgraph G_5 durchgeführt. Es wurden die in Abschnitt 4.3 vorgestellten Standardeinstellungen und eine gewünschte Kantenlänge von 2 cm verwendet. Mit Parallelisierung der Kräfteberechnung terminierte der Algorithmus in 254,9 Sekunden. Die Version des Algorithmus, die zur Berechnung nur einen Kern verwendet, benötigte hingegen 448,0 Sekunden. Durch die Benutzung mehrerer Kerne konnte in diesem Test eine Zeitersparnis von 43,1% erreicht werden. In den folgenden Abschnitten wurden alle Ergebnisse mit der Mehrkernnutzung berechnet.

4.3 Standardeinstellungen

Die in diesem Abschnitt vorgestellten Standardeinstellungen, sind solche, die sich über einen längeren Testzeitraum mit den verwendeten Testgraphen als geeignet erwiesen haben. Für drei dieser Parameter werden Tests durchgeführt bei denen der jeweilige Parameter verändert wird und die entsprechenden Ergebnisse bezüglich Qualität und Laufzeit verglichen werden. Je höher das abgebildete Knoten- und Kantengewicht, desto qualitativ hochwertiger ist das Ergebnis. Als Ergebnisse wurden immer die Durchschnittswerte über fünf Durchläufe aller Testgraphen gewertet. Anzumerken ist, dass die hier vorgestellten Einstellungen für die verwendeten Testgraphen geeignet sind und nicht zwangsläufig ohne Anpassungen für andere Testdaten gute Ergebnisse erzielen. Die Größe der Zeichenfläche im Din-A4-Format und die gewünschte Kantenlänge von 2 cm wurden vorgegeben. Die einzelnen Parameter werden im Folgenden, geordnet nach den Einzelabschnitten innerhalb des Algorithmus, vorgestellt:

- In der Vorverarbeitung (Abschnitt 3.1) wurde der Faktor f_{pre} eingeführt, der bestimmt, wie man den durchschnittlichen Abstand zwischen Knoten einschätzt, damit man abschätzen kann, wie viele Knoten in der Flächenbegrenzung maximal Platz finden. Je nachdem wie f_{pre} gewählt wird, ist die Abschätzung schärfer und damit die Laufzeit des Algorithmus geringer. Allerdings kann eine schärfere

Abschätzung ein schlechteres Ergebnis zur Folge haben. Deshalb werden mehrere Werte für f_{pre} getestet. Aus den Ergebnissen in Tabelle 4.2 lässt sich erkennen, dass ab einem Wert von 0,65 die Laufzeit extrem ansteigt, wobei sich das Ergebnis aber nicht mehr verbessert, in den Tests sogar leicht verschlechtert. Deshalb wurde als Standardwert $f_{\text{pre}} = 0,7$ gewählt, da zwar die Laufzeit ein wenig ansteigt, aber dies mit den besseren Ergebnissen gerechtfertigt werden kann.

f_{pre}	Knotengewicht		Kantengewicht		Laufzeit (Sek.)	
	Gewicht	Änderung	Gewicht	Änderung	Laufzeit	Änderung
0,8	659,4	0	716,4	0	111,7	0
0,75	670,4	+ 1,7 %	756,4	+ 5,6 %	135,8	+ 21,6 %
0,7	675,4	+ 2,4 %	769,6	+ 7,4 %	163,2	+ 46,1 %
0,65	668,6	+ 1,4 %	768,0	+ 7,2 %	242,1	+ 116,7 %
0,6	667,8	+ 1,3 %	745,2	+ 4,0 %	708,7	+ 534,5 %

Tab. 4.2: Auswirkung des Abschätzungsfaktors f_{pre} auf die Laufzeit des Algorithmus und die Qualität des Ergebnisses

- Bei der Berechnung des Gleichgewichtszustands aus Abschnitt 3.3 werden die Knoten maximal um einen Vektor der Länge $l_{\max}$ verschoben. Dadurch sollen einzelne zu starke Bewegungen verhindert werden. Als Länge wurde $l_{\max} = 0,25 \cdot l_{\text{unit}}$ gewählt. Die Berechnung eines Gleichgewichtszustand wird abgebrochen, falls die Länge des größten Verschiebungsvektors unter den Knoten in einer Iteration kleiner als $l_{\min} = 0,02$ cm ist. Zwischen den Iterationen der Berechnung wird $l_{\max}$ schrittweise um den Dämpfungsfaktor f_{cool} verkleinert, um eine schnellere Konvergenz zu erreichen und ein Oszillieren der Knoten zu verhindern. Die Tabelle 4.3 zeigt eine durchgeführte Testreihe, die als optimalen Wert $f_{\text{cool}} = 0,9$ ermittelt hat, da dort Laufzeit und Gewicht in einem guten Verhältnis zueinander stehen. Die Abnahme des Knotengewichts bei größeren Werten für f_{cool} ist durch die bessere Verteilung und der dadurch niedrigeren durchschnittlichen Kantenlänge zu erklären, die dazu führen, dass in Abschnitt 3.5 bevorzugt Knoten entfernt werden.

f_{cool}	Knotengewicht		Kantengewicht		Laufzeit (Sek.)	
	Gewicht	Änderung	Gewicht	Änderung	Laufzeit	Änderung
0,8	671,2	0	679,0	0	88,2	0
0,85	659,3	– 1,8 %	693,8	+ 2,2 %	102,9	+ 16,7 %
0,9	660,1	– 1,7 %	740,6	+ 9,1 %	122,1	+ 38,5 %
0,95	663,7	– 1,1 %	779,2	+ 14,8 %	222,2	+ 151,9 %
0,96	651,8	– 2,9 %	768,8	+ 13,2 %	270,5	+ 206,7 %

Tab. 4.3: Auswirkung des Dämpfungsfaktors f_{cool} auf die Laufzeit des Algorithmus und die Qualität des Ergebnisses

- Die Zeichenfläche A' wird in Abschnitt 3.4 immer um den konstanten Wert c_{area} angepasst, bis die Fläche mit der von A übereinstimmt. Damit bei großen Unter-

schieden in den Maßen von A und A' nicht zu viele Anpassungen vorgenommen werden müssen, ist $c_{\text{area}} = \max\{\frac{h'-h}{25}, \frac{b'-b}{25}\}$. Dadurch werden unabhängig von den Größen von A und A' immer 25 Verkleinerungsschritte durchgeführt.

- Mehrere Parameter bestimmen, wie Teilgraphen in Abschnitt 3.5 erstellt werden. Es existieren fest vorgegebene Abstände d_{adj} zwischen adjazenten Knoten und d_{nadj} zwischen nicht-adjazenten Knoten. Als passende untere Schranken für diese Werte, bei denen die Lesbarkeit der Zeichnung gewährleistet ist und gleichzeitig kleinere Abstände zwischen Knoten erlaubt sind, haben sich $d_{\text{adj}} = 0,1 \cdot l_{\text{unit}}$ und $d_{\text{nadj}} = 0,15 \cdot l_{\text{unit}}$ ergeben. Als Wert wurde $f_{\text{decision}} = 0,9$ gewählt, da dort Knoten- und Kantengewicht in der Summe maximal sind und auch ein positiver Effekt auf die benötige Rechenzeit erzielt werden konnte.

$f_{\text{precision}}$	Knotengewicht		Kantengewicht		Laufzeit (Sek.)	
	Gewicht	Änderung	Gewicht	Änderung	Laufzeit	Änderung
0,7	687,7	0	719,0	0	144,4	0
0,8	664,1	– 3,4 %	741,2	+ 3,1 %	149,0	+ 3,2 %
0,9	673,5	– 2,1 %	742,6	+ 3,3 %	127,6	– 11,6 %
1,0	660,7	– 3,9 %	739,6	+ 2,9 %	139,1	– 3,7 %
1,1	628,5	– 8,6 %	705,0	– 1,9 %	133,7	– 7,4 %

Tab. 4.4: Auswirkung des Entscheidungsfaktors $f_{\text{precision}}$ auf die Laufzeit des Algorithmus und die Qualität des Ergebnisses

- Die Anzahl der Streckensegmente, durch die quadratische Bézierkurven in Abschnitt 3.6 genähert werden, hat direkten Einfluss darauf, wie genau und wie schnell Schnittpunkte mit Knoten oder der kürzeste Abstand zu Knoten berechnet werden können. Da dieser Teil des Algorithmus nur relativ wenig Zeit in Anspruch nimmt, wurde eine Segmentzahl $c_{\text{bezier}} = 25$ gewählt, was in den hier durchgeführten Tests mehr als ausreichend war.

Um ein noch besseres Ergebnis zu erreichen, wurde jeweils der Gleichgewichtszustand zweimal hintereinander berechnet. Im ersten Durchlauf wurde dabei auf die abstoßende Kraft zwischen Knoten und Kanten aus Abschnitt 3.2.4 verzichtet, wohingegen diese im zweiten Durchlauf wieder miteinbezogen wurde. Dadurch können Knoten sich leichter über Kanten hinweg bewegen. Dies führt zu insgesamt kürzeren Kanten, weniger Kantenkreuzungen und einer besseren Knotenverteilung. Der Unterschied ist in Tabelle 4.5 dargestellt. Durch einen vergleichsweise niedrigen Laufzeitanstieg wird ein sehr viel besseres Ergebnis hinsichtlich des resultierenden Kantengewicht erzielt. Der Anstieg des Kantengewichts lässt sich dadurch erklären, da bei höheren durchschnittlichen Kantenlängen bevorzugt Kanten entfernt werden (siehe Abschnitt 3.5).

	Knotengewicht		Kantengewicht		Laufzeit (Sek.)	
	Gewicht	Änderung	Gewicht	Änderung	Laufzeit	Änderung
1 x Berechnen	663,1	0	412,4	0	107,1	0
2 x Berechnen	666,7	+ 0,5 %	739,6	+ 79,3 %	124,9	+ 16,6 %

Tab. 4.5: Durch die doppelte Berechnung der Gleichgewichtszustände lassen sich die Ergebnisse des Algorithmus verbessern

4.4 Abwägen der Stärke der Kräfte

Die in den Abschnitten 3.2, 3.6.1 und 3.6.2 vorgestellten Kräfte, die auf Knoten oder Kontrollpunkte von Kanten wirken, wurden mit Gewichtungsfaktoren $f_r, f_a, f_g, f_e, f_f, f_{rc}, f_{ac}$ versehen. Bei einer falschen Gewichtung ergeben sich unschöne Zeichnungen und es kann passieren, dass sich Kräfte gegenseitig aufschaukeln (siehe Abb. 4.1). Die Anziehungskraft ist dort so stark, dass sich der Abstand zwischen den Knoten vergrößert und in der nächsten Iteration zu noch stärker wirkenden Kräfte führen würde.

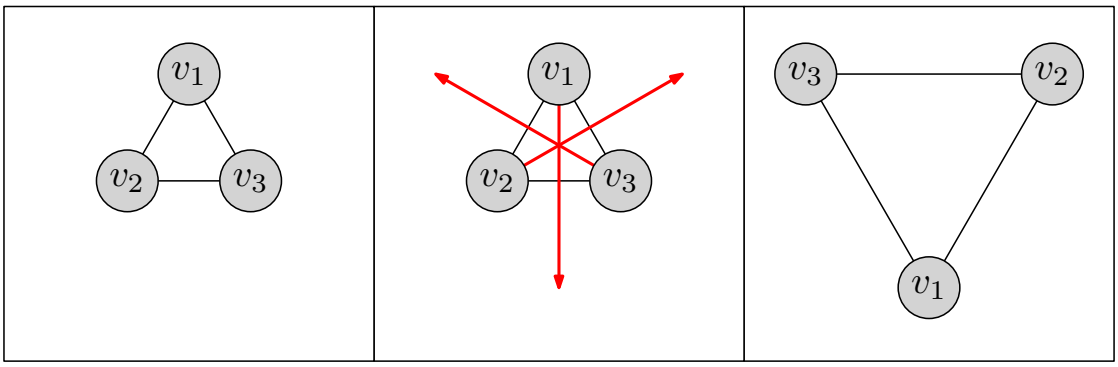


Abb. 4.1: Aufschaukeln der Kräfte, da diese zu stark wirken

In der Praxis haben sich diese Gewichtungsfaktoren als sinnvoll ergeben: $f_r = 0,01; f_a = 0,01; f_g = 0,005; f_e = 0,0075; f_f = 0,01; f_{rc} = 0,01; f_{ac} = 0,01$. Die Gewichtung der Kräfte sind ziemlich identisch mit der Ausnahme der abstoßenden Kraft zwischen Knoten und Kanten von Bertault [Ber00] und der selbst definierten Anziehungskraft zum Mittelpunkt der Zeichenfläche. Die Kraft von Bertault wirkt sehr viel stärker und musste somit auch schwächer gewichtet werden, damit sinnvolle Ausgaben erzielt werden konnten. Die Anziehungskraft zum Mittelpunkt wurde etwas schwächer als die restlichen Kräfte gewichtet, da durch diese die Wirkung der restlichen Kräfte nicht zu sehr abgeschwächt werden sollten. Die restlichen Kräfte sind identisch oder bauen auf den Kräften von Fruchterman und Reingold [FR91] auf. Dadurch lassen sich diese ähnlich gewichten und es gab dort wenige Probleme.

4.5 Beispielausgaben des Algorithmus

Die Abbildungen 4.2 und 4.3 zeigen die Ausgaben des Algorithmus für den Graph G_5 mit den Standardeinstellungen und einer Kantenlänge von 2 cm bzw. 3 cm. Um das Gewicht der Knoten einschätzen zu können, wurden diese eingefärbt. Das Farbspektrum verläuft von Rot (Schwer) über Gelb nach Grün (Leicht). Das Kantengewicht wurde durch die Dicke der Kanten in der Zeichnung verdeutlicht. Man sieht, dass in der Zeichenfläche die Knoten relativ platzsparend angeordnet und damit eine befriedigende Menge an Knoten abgebildet werden konnten. Der Vergleich der prozentual abgebildeten Anzahl und Gewichts von Kanten und Knoten zeigt, dass die wichtigsten Knoten und Kanten abgebildet werden konnten. In Abbildung 4.4 wurde der Zustand des Algorithmus dargestellt, bevor Kanten in Bézierkurven umgewandelt wurden. Man sieht, dass diese Umwandlung in allen Fällen problemlos funktioniert hat und die Lesbarkeit der Ausgabe verbessert wurde. In Abbildung 4.5 wurde der Graph, der am Ende des Algorithmus in die DOT-Datei gespeichert wurde, durch den kräftebasierten *fdp*-Algorithmus von GraphViz gezeichnet. Dort lässt sich zwar eine Flächenbegrenzung angeben, allerdings wird nur das Seitenverhältnis zur Berechnung beibehalten und nach der Auslegung das Ergebnis herunter skaliert. Deshalb können die Ergebnisse leider nicht mit denen des hier vorgestellten Algorithmus verglichen werden. Da der Algorithmus auch für kreisförmige Knoten entwickelt wurde soll auch davon eine Ausgabe vorgestellt werden. Da die Zeichnungen mit vollständigen Namen Kreise mit sehr großen Radien ergeben würden, wird deshalb nur der letzte durch Leerzeichen getrennte Namensteil verwendet. Das Ergebnis für einen Testlauf mit Kantenlänge 2 cm und Standardeinstellungen ist in Abbildung 4.6 zu sehen. Auch hier werden gute Werte erzielt, wenn die prozentuale abgebildete Anzahl und das prozentuale abgebildete Gewicht der Graphelemente miteinander verglichen wird.

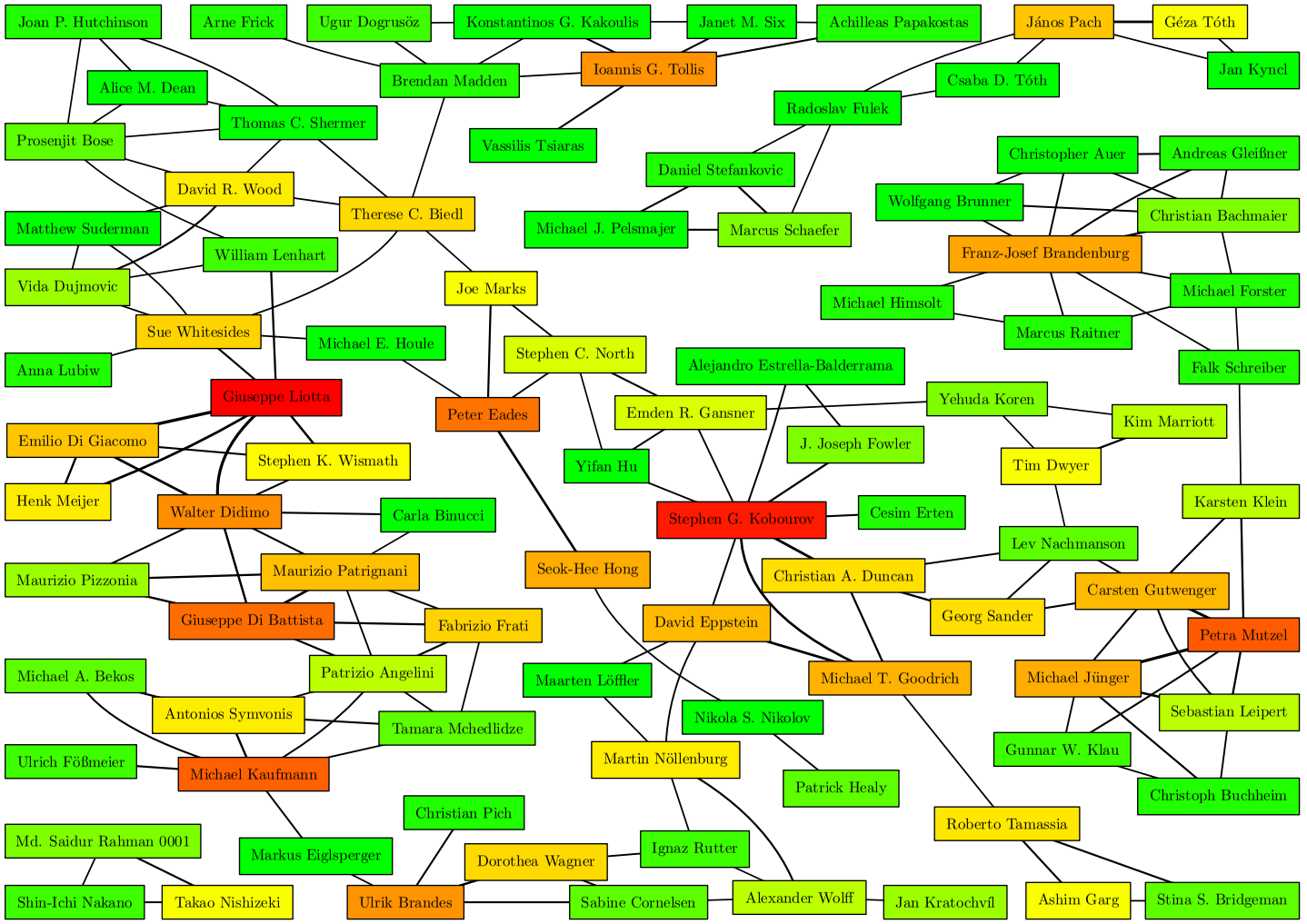


Abb. 4.2: Beispielzeichnung von G_5 mit $l_{\text{unit}} = 2{,}0$ cm; enthaltene Knoten: 10,1 %; enthaltenes Knotengewicht: 49,4 %; enthaltene Kanten: 6,9 %; enthaltenes Kantengewicht: 21,2 %

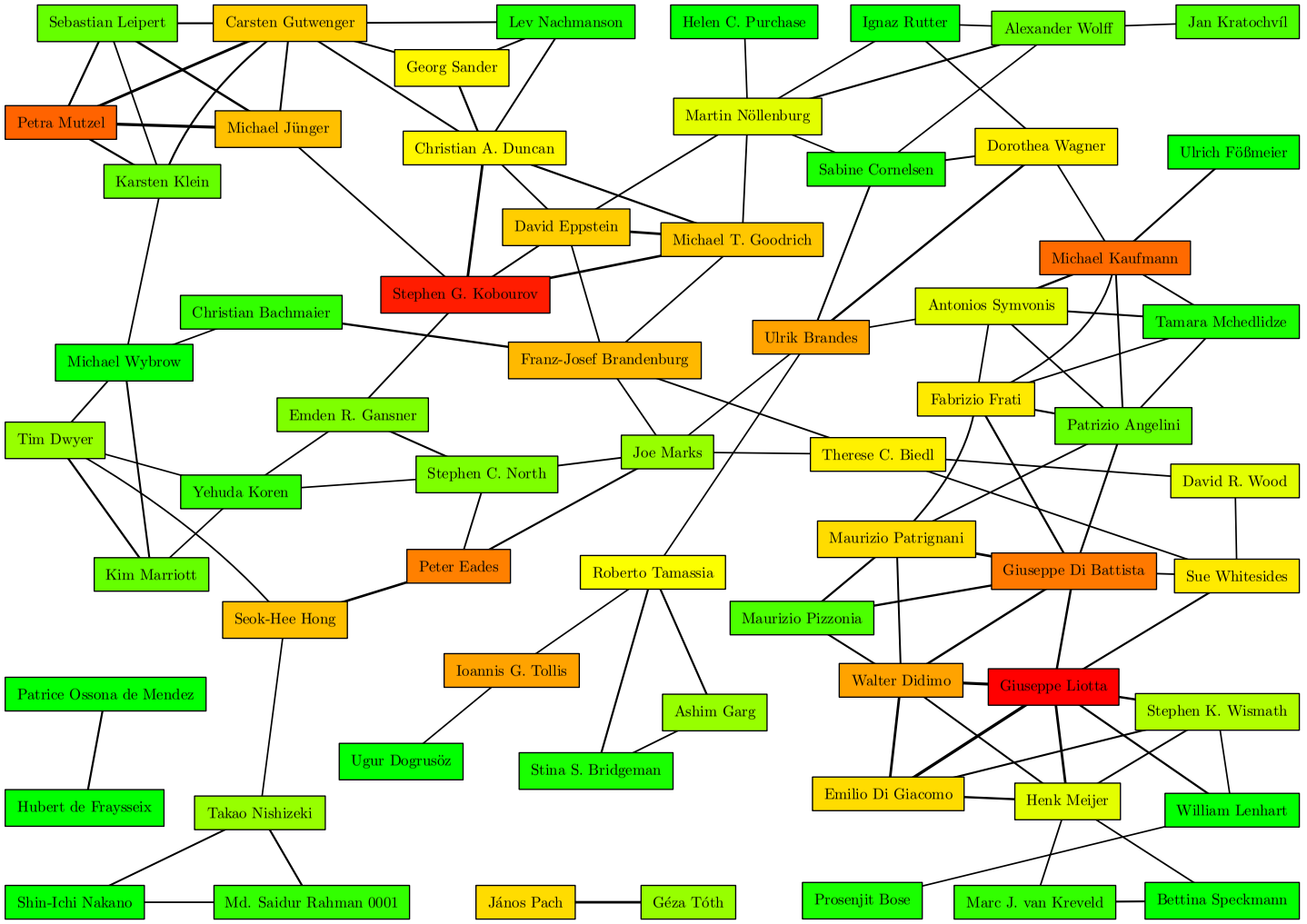


Abb. 4.3: Beispielzeichnung von G_5 mit $l_{\text{unit}} = 3{,}0$ cm; enthaltene Knoten: 6,6 %; enthaltenes Knotengewicht: 42,1 %; enthaltene Kanten: 4,9 %; enthaltenes Kantengewicht: 17,0 %

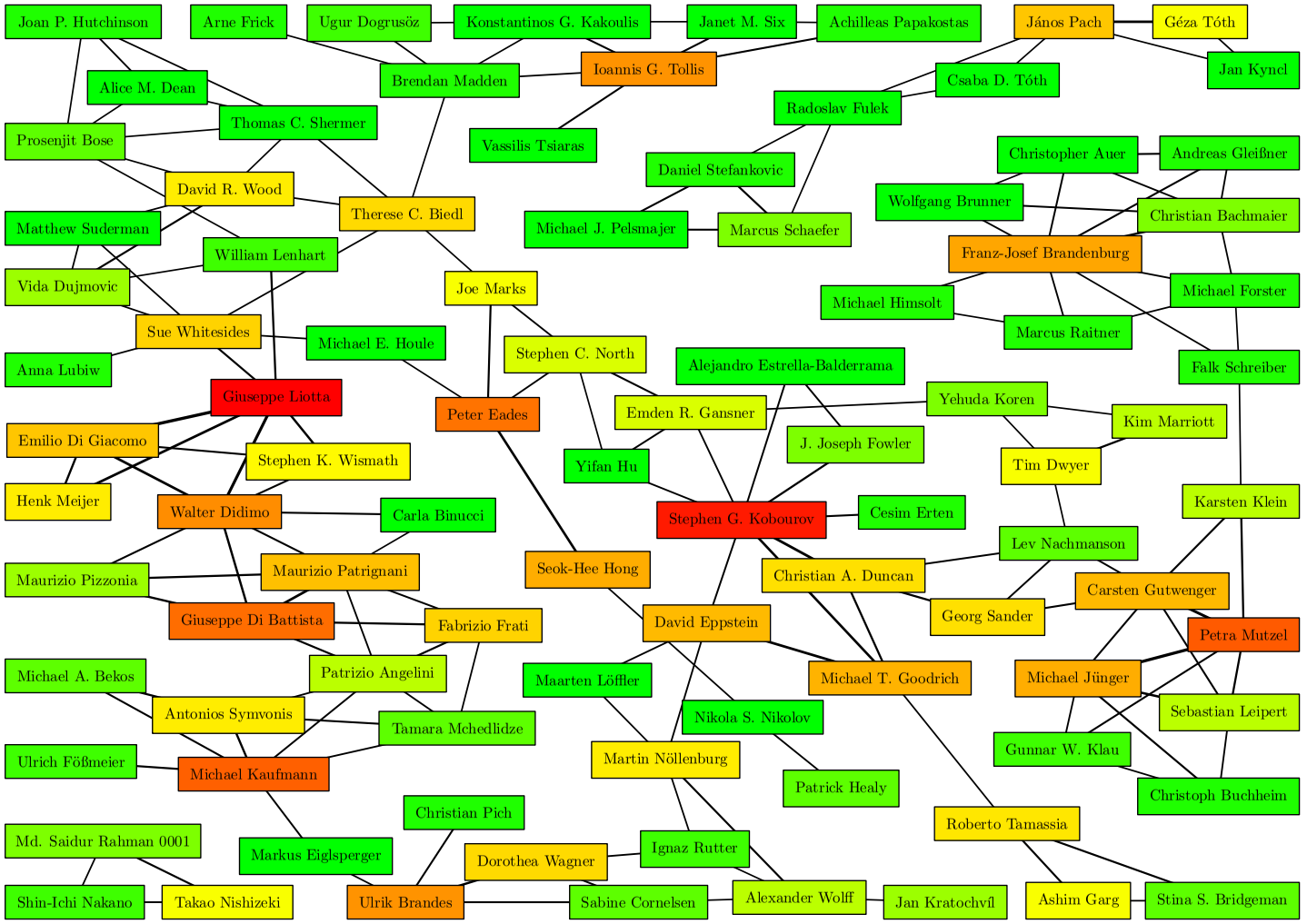


Abb. 4.4: Ausgabe des Durchlaufs aus Abbildung 4.2 vor der Umwandlung der Kanten in quadratische Bézierkurven

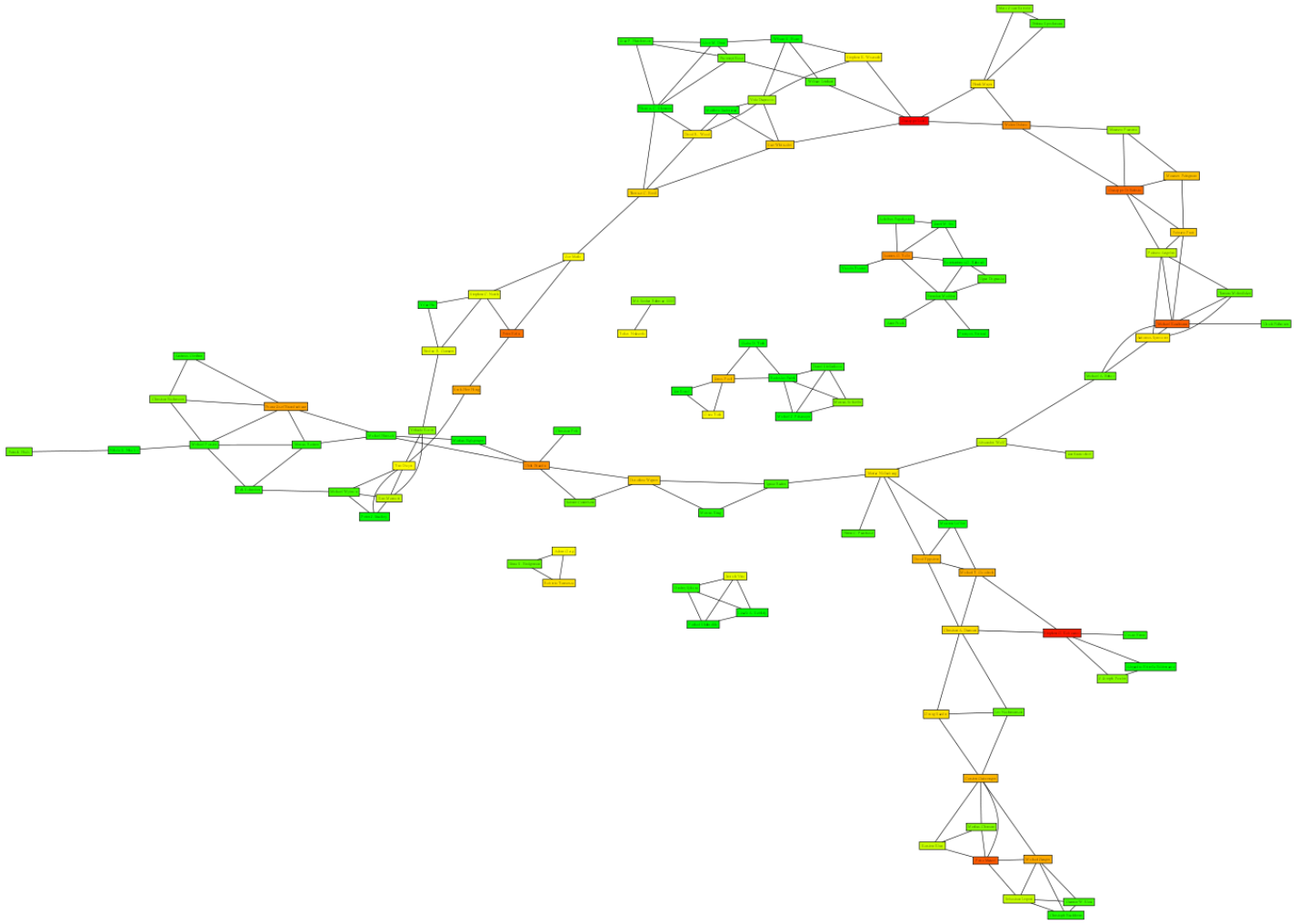


Abb. 4.5: Ausgabe von GraphViz mit dessen fdp-Algorithmus für den Endgraphen des Durchlaufs aus Abbildung 4.2; GraphViz hält sich nicht an die Größenbegrenzung, sondern nur an das Seitenverhältnis und skaliert das Ergebnis dann entsprechend

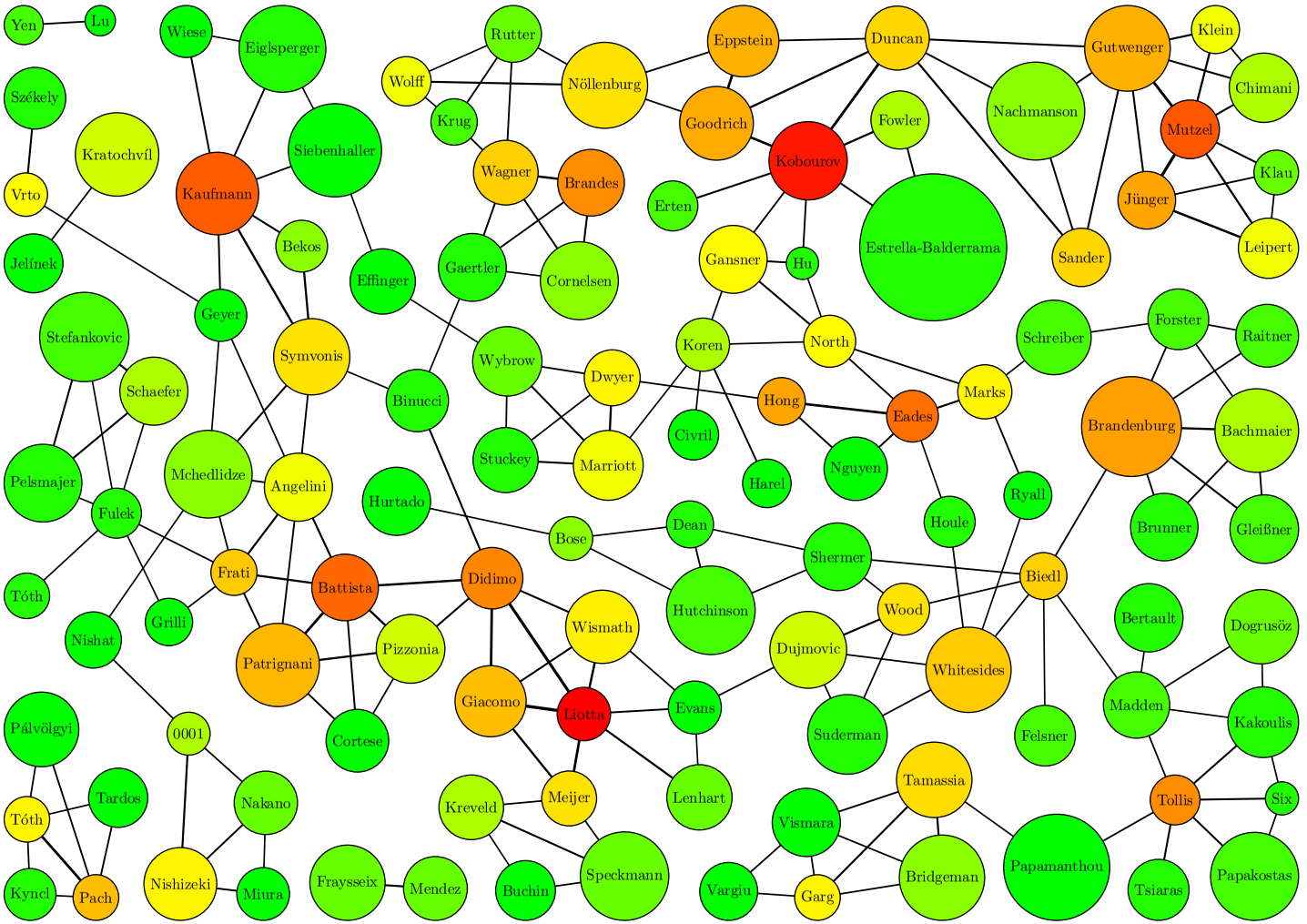
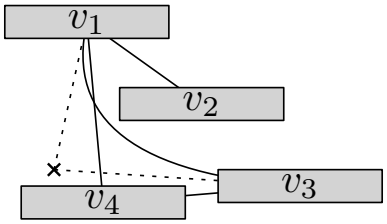


Abb. 4.6: Beispielzeichnung von G_5 (nur letzter Teil des Namens) mit $l_{\text{unit}} = 2,0$ cm; enthaltene Knoten: 12,8 %; enthaltenes Knotengewicht: 54,2 %; enthaltene Kanten: 8,6 %; enthaltenes Kantengewicht: 24,0 %

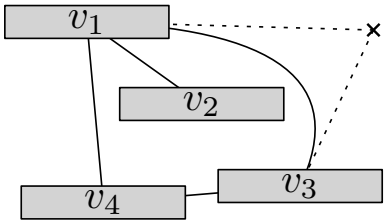
5 Fazit und Ausblick

Aufbauend auf bekannten kräftebasierten Verfahren wurde ein Algorithmus entwickelt, der aus einem gewichteten Eingabegraph, einer vorgegeben gewünschten Kantenlänge, einem vorgegeben Zeichenbereich und der gewünschten Knotenform eine Zeichnung eines Teilgraphen mit möglichst großem Knoten- und Kantengewicht erzeugt. Die Knoten werden dabei durch den kräftebasierten Algorithmus in der Zeichenfläche verteilt. Falls nach der Berechnung des Algorithmus Überschneidungen zwischen Knoten und Kanten übrig bleiben sollten, werden Kanten als quadratische Bézierkurven gezeichnet, um die Überschneidungen aufzulösen. Die Ausgaben im letzten Kapitel zeigen, dass der Algorithmus gute Ausgaben für die genutzten Testgraphen liefert. Die Berechnungszeit steigt aber für größere Zeichenflächen stark an. Auch die Zeichnung der Bézierkurven kann an manchen stellen noch verbessert werden. Die Bewertung der Ergebnisse war in vielen Fällen recht schwierig, da es kein vergleichbares Verfahren gibt mit dem man sich messen könnte.

Die Umwandlung der Kanten in quadratische Bézierkurven führt aber in manchen Fällen zu unschönen Ergebnissen, da im Vordergrund steht die Kanten überschneidungsfrei in Hinsicht auf Knoten in der Zeichnung zu platzieren. In dieser Arbeit wurden Kantenkreuzungen nicht beachtet. Allerdings können durch die Bézierkurven unschöne Kantenkreuzungen entstehen. Abbildung 5.1(a) zeigt ein häufiger wiederkehrendes Muster in den Ausgabezeichnungen. Dies ließe in den meisten Fällen eleganter lösen, indem die Kurve in solchen Fällen auf der anderen Seite am Knoten vorbei läuft, wie in Abbildung 5.1(b). Allerdings lässt sich dies durch Kräfte nicht so einfach implementieren, da die Kante vom Mittelpunkt des Knotens weggedrückt wird und so auf der ungünstigeren Seite vorbei verläuft.



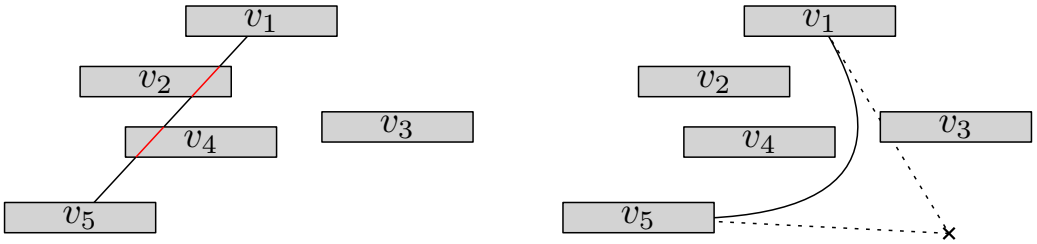
(a) Bisheriges Ergebnis des Algorithmus



(b) Schöneres Vorgehen für eine übersichtlichere Zeichnung

Abb. 5.1: Verhinderung von unschönen Kantenkreuzungen

Ein weiteres Problem bei dem jetzigen kräftebasierten Ansatz zur Verschiebung des Kontrollpunkts der Kante ist die Auflösung von Überschneidungen mit mehr als zwei Knoten. Da initial der Kontrollpunkt auf den Mittelpunkt der Kante platziert wird, kann die Kante nicht immer auf jeweils einer Seite an den Knoten vorbei geleitet werden. Die Überschneidung ist nur auflösbar, wenn die Kante bei allen Knoten jeweils auf der gleichen Seite am Mittelpunkt der Knoten verläuft, da dort die Kräfte sich nicht gegenseitig blockieren. Ein Problemfall wurde in Abbildung 5.2(a) geschildert. Anstatt diese Kanten zu löschen, weil man die Überschneidung nicht auflösen kann, wäre es schöner die Kräfte von nur einem der Schnittknoten anzuwenden, bis man es geschafft die Kante aus allen Knoten heraus zudrücken (siehe Abb. 5.2(b)). Wenn das geschafft ist, können wieder alle Kräfte miteinbezogen werden.



(a) Kante schneidet zwei Knoten vor der Nachbearbeitung

(b) Besseres Ergebnis durch vorübergehende Unterdrückung der Kräfte von einem der beiden geschnittenen Knoten

Abb. 5.2: Mögliche Vorgehensweise um Überschneidungen einer Kante mit zwei Knoten aufzulösen

Literaturverzeichnis

[BEGT13] Michael J. Bannister, David Eppstein, Michael T. Goodrich und Lowell Trott: Force-Directed Graph Drawing using Social Gravity and Scaling. In: Walter Didimo und Maurizio Patrignani (Herausgeber): *Proceedings of the 20th International Symposium on Graph Drawing (GD'12)*, Band 7704 der Reihe *Lecture Notes in Computer Science*, Seiten 414–425. Springer Berlin Heidelberg, 2013. http://dx.doi.org/10.1007/978-3-642-36763-2_37.

[Ber00] François Bertault: A Force-Directed Algorithm that preserves Edge Crossing Properties. *Information Processing Letters*, 74(1–2):7–13, 2000. http://dx.doi.org/10.1016/S0020-0190(00)00042-9.

[BMBL09] Stephen P. Borgatti, Ajay Mehra, Daniel J. Brass und Giuseppe Labianca: Network Analysis in the Social Sciences. *Science*, 323(5916):892–895, 2009. http://dx.doi.org/10.1126/science.1165821.

[DGKN97] David P. Dobkin, Emden R. Gansner, Eleftherios Koutsofios und Stephen C. North: Implementing a General-Purpose Edge Router. In: Giuseppe DiBattista (Herausgeber): *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Band 1353 der Reihe *Lecture Notes in Computer Science*, Seiten 262–271. Springer Berlin Heidelberg, 1997. http://dx.doi.org/10.1007/3-540-63938-1_68.

[DMW09] Tim Dwyer, Kim Marriott und Michael Wybrow: Topology Preserving Constrained Graph Layout. In: Ioannis G. Tollis und Maurizio Patrignani (Herausgeber): *Proceedings of the 16th International Symposium on Graph Drawing (GD'08)*, Band 5417 der Reihe *Lecture Notes in Computer Science*, Seiten 230–241. Springer Berlin Heidelberg, 2009. http://dx.doi.org/10.1007/978-3-642-00219-9_22.

[Ead84] Peter Eades: A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

[FHN+13] Martin Fink, Herman Haverkort, Martin Nöllenburg, Maxwell Roberts, Julian Schuhmann und Alexander Wolff: Drawing Metro Maps using Bézier Curves. In: Walter Didimo und Maurizio Patrignani (Herausgeber): *Proceedings of the 21th International Symposium on Graph Drawing (GD'13)*, Band 7704 der Reihe *Lecture Notes in Computer Science*, Seiten 463–474. Springer Berlin Heidelberg, 2013. http://dx.doi.org/10.1007/978-3-642-36763-2_41.

[FHS+12] Martin Fink, Jan Henrik Haunert, André Schulz, Joachim Spoerhase und Alexander Wolff: Algorithms for Labeling Focus Regions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2583–2592, 2012. http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.193.

[FR91] Thomas M. J. Fruchterman und Edward M. Reingold: Graph Drawing by Force-Directed Placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. http://dx.doi.org/10.1002/spe.4380211102.

[GN98] Emden R. Gansner und Stephen C. North: Improved Force-Directed Layouts. In: Sue H. Whitesides (Herausgeber): *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, Band 1547 der Reihe *Lecture Notes in Computer Science*, Seiten 364–373. Springer Berlin Heidelberg, 1998. http://dx.doi.org/10.1007/3-540-37623-2_28.

[HM98] Weiqing He und Kim Marriott: Constrained Graph Layout. *Constraints*, 3(4):289–314, 1998, ISSN 1383-7133. http://dx.doi.org/10.1023/A%3A1009771921595.

[KK89] Tomihisa Kamada und Satoru Kawai: An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1):7–15, 1989. http://dx.doi.org/10.1016/0020-0190(89)90102-6.

[KN09] Sven Oliver Krumke und Hartmut Noltemeier: *Graphentheoretische Konzepte und Algorithmen*. Leitfäden der Informatik. Vieweg+Teubner Verlag, 2009. http://books.google.de/books?id=TcYcRlIqZnoC.

[LY12] Chun-Cheng Lin und Hsu-Chun Yen: A New Force-Directed Graph Drawing Method based on Edge-Edge Repulsion. *Journal of Visual Languages and Computing*, 23(1):29–42, 2012. http://dx.doi.org/10.1016/j.jvlc.2011.12.001.

[SAAB11] Paolo Simonetto, Daniel Archambault, David Auber und Romain Bourqui: ImPrEd: An Improved Force-Directed Algorithm that prevents Nodes from Crossing Edges. *Computer Graphics Forum*, 30(3):1071–1080, 2011. http://dx.doi.org/10.1111/j.1467-8659.2011.01956.x.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen Hilfsmittel und Quellen als die angegebenen benutzt habe. Weiterhin versichere ich, die Arbeit weder bisher noch gleichzeitig einer anderen Prüfungsbehörde vorgelegt zu haben.

Würzburg, den _____, _____

(Maximilian Aulbach)