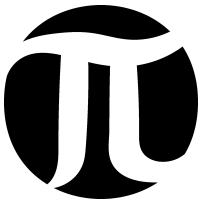
UNIVERSITÄT PASSAU
Fakultät für Mathematik und Informatik
Lehrstuhl für Theoretische Informatik



Diplomarbeit

Zeichnen ungerichteter Graphen mit gegebenen Knotengrößen durch ein Springembedder-Verfahren

Michael Forster

Betreuer
Prof. Dr. Franz J. Brandenburg

27. September 1999

Zusammenfassung

Graphen sind ein vor allem in der Informatik weitverbreitetes Mittel, um verschiedenartigste Daten und Strukturen intuitiv darzustellen. Im „Graph Drawing“ geht es darum, Graphen, die ohne Koordinaten gegeben sind, nach gewissen Kriterien möglichst schön zu zeichnen. Die sogenannten „Springembedder-Verfahren“ arbeiten mit kräftebasierten physikalischen Simulationen und sind dazu geeignet, beliebige Graphen übersichtlich darzustellen. In der Praxis hat man mit Springembedder-Verfahren gute Erfahrungen gemacht, jedoch gibt es nur wenige Ansätze, die unterschiedliche Knotengrößen berücksichtigen. Im Rahmen dieser Arbeit wird ausgehend von vorhandenen Arbeiten ein Springembedder-Verfahren entwickelt und im Graphlet[1]-System implementiert. Dieses soll flexibel und effizient sein und insbesondere die Größe von Knoten berücksichtigen.

[1] siehe `http://www.fmi.uni-passau.de/Graphlet/`

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Graph Drawing

„Ein Bild sagt mehr als tausend Worte.“ Dieses vielzitierte chinesische Sprichwort bringt die Intention dieser Arbeit auf den Punkt. Der Mensch kann komplexe Strukturen wesentlich besser erfassen und überblicken, wenn sie ihm nicht in textueller Form, sondern als Bild präsentiert werden. Dies gilt jedoch nur dann, wenn das Bild übersichtlich gezeichnet ist. Hier stellt sich auch schon das Problem: Was genau macht eine übersichtliche Darstellung aus, und vor allem: Wie kann man aus gegebenen Daten eine solche erstellen? Im konkreten Fall ist dies davon abhängig, für welchen Verwendungszweck die Bilder bestimmt sind und von welcher Art die gegebenen Daten sind. Das Forschungsgebiet des „Graph Drawing“ beschäftigt sich mit dem Zeichnen von Daten, die sich als Graph mit assoziierten Attributen modellieren lassen. Beispiele dafür sind leicht zu finden, da praktisch jede Information, die irgendwelche Abhängigkeiten darstellt, als Graph betrachtet werden kann. Typische Beispiele sind etwa Ablauf- oder Syntaxdiagramme

Wir beschäftigen uns also mit der Frage, wie man zu einem gegebenen Graphen eine schöne Zeichnung findet. Dazu gibt es für bestimmte Graphklassen Verfahren, die bei geringer Komplexität ein nach bestimmten Kriterien optimales Layout bestimmen. Der von REINGOLD und TILFORD in [RT81] vorgestellte Algorithmus ist ein Beispiel dafür. Dieser zeichnet Bäume in linearer Zeit so, daß innere Baumknoten zentriert über ihren Söhnen liegen und daß gleiche oder symmetrische Unterbäume auch gleich oder symmetrisch gezeichnet werden.

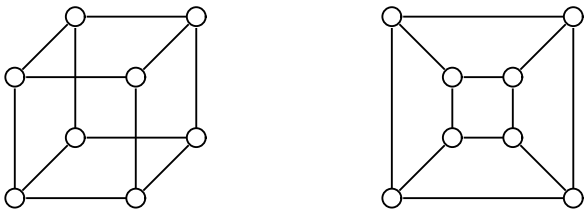


Abbildung 1.1: Verschiedene Layouts für denselben Graphen

Solch positive Ergebnisse hat man meistens leider nicht. Wenn man versucht, das Layout eines beliebigen Graphen zu optimieren, werden die Probleme fast immer ziemlich schnell $\mathbb{NP}$-hart. So wird z. B. das Problem der schön gezeichneten Bäume schon $\mathbb{NP}$-vollständig, wenn man zusätzlich zu den bereits genannten Kriterien verlangt, daß die Zeichnung minimale Breite haben soll (siehe [SR82]). Deshalb gilt es – wie immer in solchen Fällen – gute Heuristiken zu finden. Eine solche Heuristik für allgemeine Graphen stellen die sogenannten „Springembedder-Verfahren“ dar, die in dieser Arbeit behandelt werden.

1.2 Ästhetik

Was ist eigentlich eine „schöne Zeichnung“? Diese Frage kann sicherlich nicht endgültig beantwortet werden. Ästhetik ist ein grundsätzlich subjektiver Begriff. Jeder Mensch hat eine eigene Vorstellung davon. Zwei verschiedene Personen werden oft ein und dasselbe Layout eines Graphen verschieden schön finden. Wenn man etwa die Zeichnungen in Abbildung 1.1 vergleicht, so kann man nicht absolut sagen, welche der beiden schöner oder übersichtlicher ist. Jede davon hat seine Berechtigung.

Ziel bei der Bestimmung von Ästhetikkriterien kann es also nicht sein, absolut zu definieren, was Schönheit ist, sondern allgemeine Kriterien zu finden, die von möglichst vielen als schön empfunden werden. Das Finden von einigermaßen objektiven Ästhetikkriterien kann durchaus als Herausforderung gesehen werden. Die im folgenden genannten Kriterien sind also keineswegs als vollständige Liste zu sehen, sondern vielmehr als ein Anfang.

Knotenverteilung Die Knoten des Graphen sollen möglichst gleichmäßig über die ganze Zeichenfläche verteilt sein. Ungleichmäßige Knotenverteilungen erscheinen unausgewogen und damit unordentlich. Gleichzeitig sollen die Abstände zwischen zwei Knoten groß genug sein, damit man sie deutlich voneinander separieren kann. Knoten, die durch eine Kante verbunden sind, sollen nahe zusammenliegen, damit ihre Zusammengehörigkeit herausgestellt wird.

Kantenführung Die Führung der Kanten ergibt sich direkt aus der Zeichnung der Knoten, wenn man von Kantenknicken absieht, die von Springembedder-Verfahren meist nicht unterstützt werden. Deshalb ist die Positionierung der Knoten auch wichtig für ein gutes Bild der Kanten. Diese sollen sich möglichst wenig mit anderen Kanten schneiden, um die Verwirrung des Betrachters gering zu halten. Weiterhin wichtig ist eine weitgehend uniforme Länge der Kanten. Das macht einen aufgeräumten Eindruck und steigert dadurch die Übersichtlichkeit.

Symmetrien An sich ästhetische Strukturen, die bereits im Graphen enthalten sind, sollen vom Verfahren auch als solche dargestellt werden. Dazu gehören vor allem Symmetrien und räumliche Strukturen.

Flächenbedarf Um den Überblick nicht zu verlieren, ist es wichtig, daß die Zeichnung eines Graphen nicht zu viel Fläche benötigt. Idealerweise soll die gesamte Struktur mit einem Blick zu erfassen sein. Dies ist nur möglich, wenn die Zeichnung nicht zu groß ist.

Ein großes Problem dieser Kriterien ist, daß sie sich teilweise gegenseitig widersprechen. So brauchen Zeichnungen ohne Kantenschnitte oft mehr Platz als andere. Man wird manchmal das eine Kriterium wichtiger bewerten, manchmal das andere. Diese Entscheidung ist aber nur schlecht durch einen Algorithmus nachzubilden. Die Folgerung daraus ist, daß ein Layoutverfahren möglichst flexibel sein muß, um auf spezielle Wünsche des Anwenders eingehen zu können.

Kapitel 2

Grundlagen

Bevor wir uns näher mit den Details von Springembedder-Verfahren beschäftigen, ist es notwendig, einige grundlegende Begriffe zu klären, die zum Verständnis erforderlich sind.

2.1 Graphen

In den meisten Fällen definiert man Graphen als Zweitupel $G = (V, E)$ aus einer Menge von Knoten V und einer Menge von Kanten E, wobei die Kanten bei gerichteten Graphen als Zweitupel von Knoten und bei ungerichteten Graphen als zweielementige Mengen von Knoten definiert werden. Diese Definitionen haben den Nachteil, daß keine Mehrfachkanten möglich sind. Außerdem werden für manche Aspekte von Layoutverfahren Attribute für die Knoten und Kanten benötigt. Um dies alles zu berücksichtigen, wird ein Graph wie folgt definiert:

Definition 2.1 *Ein Graph über einem Beschriftungsalphabet* Σ *ist ein Dreitupel* $G = (V, E, A)$ *aus einer endlichen Knotenmenge* V*, einer endlichen Kantenmenge* $E \subseteq V \times V \times \mathbb{N}$ *und einer Attributfunktion* $A\colon V \cup E \to \Sigma^*$*. Sofern nicht anders angegeben, bezeichnet* $n = |V|$ *die Anzahl der Knoten und* $m = |E|$ *die Anzahl der Kanten.*

Zu dieser Definition sind ein paar Anmerkungen notwendig:

- Obwohl Springembedder-Verfahren hauptsächlich für ungerichtete Graphen gedacht sind, unterstützt diese Definition gerichtete Graphen. Dies wird an

einigen Stellen benötigt, etwa wenn von Richtungsvorgaben für Kanten die Rede ist. In den meisten Fällen wird die Richtung der Kanten ignoriert.

- Mehrere Kanten zwischen denselben Knoten sind möglich, sie werden einfach durchnumeriert. Der Einfachheit halber wird aber für Kanten weiterhin die Schreibweise $e = (u, v)$ anstatt von $e = (u, v, 1)$ verwendet, soweit dies nicht zu Mehrdeutigkeiten führt.

- Die genaue Modellierung der Attributfunktion A ist nebensächlich. Wichtig ist, daß ihr alle Informationen entnommen werden können, die zum Layout des Graphen benötigt werden, etwa die Größen der Knoten. Andere wichtige Attribute können Wunschwerte für die Länge oder Richtung von Kanten sein.

Es folgen noch einige Definitionen für in der Graphentheorie übliche Begriffe:

Definition 2.2 *Eine Kante $(u, v, i) \in E$ heißt inzident zu den beiden Knoten u und v. Zwei Knoten $u, v \in V$ heißen adjazent, wenn es eine Kante $e \in E$ gibt, die zu beiden inzident ist. Der Grad eines Knoten ist die Anzahl seiner inzidenten Kanten.*

Definition 2.3 *Ein Graph $G = (V, E, A)$ heißt Untergraph eines Graphen $G' = (V', E', A')$, falls folgende Bedingungen gelten:*

$$\begin{aligned} &V \subseteq V', \\ &E = E' \cap (V \times V \times \mathbb{N}) \quad \textit{und} \\ &A = A'|_{V \cup E}. \end{aligned}$$

Definition 2.4 *Ein (ungerichteter) Weg zwischen zwei Knoten $u, v \in V$ in einem Graphen $G = (V, E, A)$ ist eine Folge $(v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1})$ mit $v_1, \ldots, v_{n+1} \in V$, $e_1, \ldots, e_n \in E$ und $e_i = (x_i, x_{i+1}, j)$ oder $e_i = (x_{i+1}, x_i, j)$ für $i = 1, \ldots, n$. Dabei heißt n die Länge des Weges.*

Definition 2.5 *Der graphentheoretische Abstand zweier Knoten $u, v \in V$ in einem Graphen $G = (V, E, A)$ ist die Länge des kürzesten Weges zwischen u und v in G.*

2.2 Layout

Die Aufgabe von Layoutverfahren ist es, zu gegebenen attributierten Graphen eine Zeichnung zu erstellen. Diese wird hier *Layout* genannt.

Definition 2.6 *Ein Layout eines Graphen* $G = (V, E, A)$ *ist eine Funktion* $L_G \colon V \to \mathbb{R}^2$*, die jedem Knoten* $v \in V$ *eine Position zuweist.*

Die Beschränkung auf zweidimensionale Zeichnungen ist willkürlich. Praktisch alle Konzepte von Springembedder-Verfahren lassen sich problemlos ins Drei- oder Mehrdimensionale übertragen. Die Beschränkung auf zwei Dimensionen dient primär der Anschaulichkeit. Außerdem haben Zeichnungen in der Ebene die meisten Anwendungen.

Als eine zusätzliche Erweiterung könnte man nicht geradlinig verlaufende Kanten – etwa Polylinien ($E \to (\mathbb{R}^2)^*$) – erlauben. Da aber keiner der vorgestellten Algorithmen den Kantenverlauf berücksichtigt, soll hier darauf verzichtet werden. Der Verlauf der Kanten ergibt sich als Verbindungslinie der Knotenmittelpunkte.

2.3 Notationen

Ein in der Informatik weitverbreitetes Mittel, um die Komplexität von Funktionen zu vergleichen, ist die $\mathcal{O}$-Notation:

$$\mathcal{O}(f) = \{\, g \mid \exists c > 0 \colon \exists k \colon \forall n > k \colon g(n) \leq c \cdot f(n) \,\}.$$

Wenn eine Funktion $g \in \mathcal{O}(f)$ ist, dann bedeutet dies, daß g asymptotisch nicht stärker wächst als f.

Weiterhin soll mit $d(L_G, u, v)$ der euklidische Abstand zweier Knoten u und v in einem Layout L_G des Graphen bezeichnet werden.

$$d(L_G, u, v) = |L_G(u) - L_G(v)|$$

Da meistens eine Verwechslungsgefahr ausgeschlossen ist, wird anstatt $d(L_G, u, v)$ hauptsächlich $d(u, v)$ verwendet.

Kapitel 3

Existierende Verfahren

Dieses Kapitel soll einen groben Überblick über die historische Entwicklung der verschiedenen Varianten von Springembedder-Verfahren geben. Auf die genauen Details, in denen sich die Verfahren unterscheiden, wird dann in Kapitel 5 näher eingegangen.

3.1 EADES

Die grundlegende Idee aller Springembedder-Verfahren stammt von EADES. Er modelliert in [Ead84] die Knoten des Graphen als Stahlringe, die sich gegenseitig abstoßen, und die Kanten als Federn (*engl.* spring), die adjazente Knoten zueinanderziehen. Dies ist der Ursprung der Bezeichnung „Springembedder“.

In diesem physikalischen System versucht EADES, ein Energieminimum zu finden. Um dies exakt zu berechnen, wäre es notwendig, ein System von Differentialgleichungen zu lösen, das sich aus den Kräfteformeln ergibt. EADES geht aber einen anderen Weg. Sein Verfahren nähert sich einem Energieminimum durch eine physikalische Simulation. Diese beginnt mit zufälligen Startkoordinaten für die Knoten. In mehreren Schritten wird nun für jeden Knoten berechnet, welche Kräfte auf ihn in der aktuellen Situation wirken. Abhängig davon wird der Knoten ein Stück in Richtung der Gesamtkraft bewegt. Dies wird für eine feste Anzahl von Iterationen wiederholt.

EADES weicht in mehreren Punkten von den physikalischen Gesetzen ab, die in der Realität für ein solches System gelten würden. So gilt etwa in der realen

Welt für Federn das HOOK'sche Gesetz. Daraus ergeben sich Kräfte, die linear in der Federlänge wachsen. Nach EADES hätten dann aber weit voneinander entfernt liegende Knoten einen zu starken Einfluß aufeinander. Deshalb verwendet er in seiner Simulation einen logarithmischen Zusammenhang zwischen dem Abstand der Knoten und den zwischen ihnen wirkenden Kräften.

3.2 KAMADA/KAWAI

KAMADA und KAWAI verfolgen in [KK89] einen ähnlichen Ansatz wie EADES, sie bringen jedoch einige neue Ideen. In ihrem Ansatz gibt es nicht nur Federn zwischen adjazenten Knoten, sondern zwischen jedem Paar von Knoten – allerdings mit verschiedenen Längen und Federhärten.

Die Länge $l_{u,v}$ der Feder zwischen zwei Knoten $u, v \in V$ ergibt sich dabei aus dem graphentheoretischen Abstand der beiden Knoten, multipliziert mit der gewünschten Kantenlänge. Alternativ kann man auch gewichtete Kanten verwenden und als Federlänge die Länge des kürzesten Weges von u nach v setzen. Die Härte $k_{u,v}$ der Federn nimmt mit wachsendem graphentheoretischen Abstand der Knoten quadratisch ab. Die genauen Kräfteformeln richten sich bei diesem Verfahren direkt nach dem HOOK'schen Gesetz.

Der zweite Unterschied zu dem Verfahren von EADES ist die Methode, wie das Layout iterativ verbessert wird. Während bei EADES die Knoten einfach in Richtung der auf sie wirkenden Kraft bewegt werden, betrachten KAMADA und KAWAI die Gesamtenergie des Graphen

$$E_G = \sum_{\substack{u,v \in V \\ u \neq v}} k_{u,v} \cdot (d(u,v) - l_{u,v})^2$$

und wenden darauf eine modifizierte mehrdimensionale NEWTON-RAPHSON-Approximation an. Das Ergebnis davon ist, daß das Layout nachweislich zu einem (lokalen) Energieminimum konvergiert.

Die Ergebnisse dieses Verfahrens sind sehr ansprechend und auch die Laufzeit des Verfahrens ist akzeptabel.

3.3 DAVIDSON/HAREL

DAVIDSON und HAREL beschäftigen sich in [DH91] mit Layoutverfahren, die auf Simulated Annealing basieren.

Simulated Annealing ist ein allgemeines algorithmisches Prinzip, das aus einem gegebenen Suchraum von Konfigurationen ein bezüglich einer Kostenfunktion minimales Element sucht. Dies geschieht, indem, ausgehend von einer Startkonfiguration, kleine zufällige Änderungen vorgenommen werden. Wird dabei die Kostenfunktion günstiger, so wird der neue Zustand übernommen. Wenn nicht, dann wird es dem Zufall überlassen, ob man in dem alten Zustand bleibt oder in den neuen übergeht. Die Wahrscheinlichkeit, Verschlechterungen zu akzeptieren – auch Temperatur genannt – wird im Laufe des Verfahrens kontinuierlich gesenkt. Um wirklich gute Ergebnisse zu erzielen, ist es notwendig, die Temperatur sehr langsam zu senken. Deshalb haben Simulated-Annealing-Verfahren im allgemeinen eine sehr lange Laufzeit.

Übertragen auf das Zeichnen von Graphen besteht der Suchraum aus verschiedenen Layouts des Graphen, die kleinen Änderungen entsprechen Verschiebungen von Knoten. Als Kostenfunktionen verwendet man Indikatoren für die gewünschten Merkmale des Layouts.

Simulated Annealing ist strenggenommen kein Springembedder-Verfahren. Es weist jedoch viele Parallelen auf, so daß manche Ideen auf Springembedder-Verfahren übertragen werden können. Es lassen sich qualitativ sehr gute Ergebnisse erzielen, allerdings ist die Laufzeit inakzeptabel.

3.4 FRUCHTERMAN/REINGOLD

Ein weiteres Verfahren, das einige interessante Aspekte liefert, ist der von FRUCHTERMAN und REINGOLD in [FR91] vorgestellte Algorithmus. In diesem werden erstmals die Ideen aus dem Simulated Annealing in einem Springembedder-Verfahren verwendet. So werden Temperaturen verwendet, um das Konvergenzverhalten zu verbessern. Der große Unterschied zum Simulated Annealing ist, daß keine zufälligen Bewegungen von Knoten betrachtet werden, sondern zielgerich-

tete Bewegungen in Richtung der auf sie wirkenden Kraft. Die Temperatur regelt, wie weit sich Knoten bewegen dürfen. Je niedriger die Temperatur, desto geringer werden die Bewegungen, die den Knoten erlaubt werden.

Weitere Neuerungen sind erste Überlegungen, wie man die Zeichnung des Graphen auf eine vorgegebene Zeichenfläche begrenzen kann, und die Verwendung von im Vergleich zu EADES schnelleren Kräfteformeln.

Eine interessante Abänderung des Verfahrens ist die sogenannte Gittervariante (*engl.* grid variant). Hier werden die abstoßenden Kräfte nur für Knotenpaare berechnet, die voneinander einen bestimmten Maximalabstand haben. Dies hat zwei Effekte. Erstens lassen sich durch Anwendung von Hashing die Kräfte schneller berechnen und zweitens erlaubt diese Änderung das Zeichnen von Graphen, die nicht zusammenhängend sind.

3.5 FRICK/LUDWIG/MEHLDAU (GEM)

GEM, ein Verfahren das von FRICK, LUDWIG und MEHLDAU in [F+94] vorgestellt wurde, bringt eine Menge neuer Ideen. Ziel bei der Entwicklung von GEM war es, aufbauend auf den existierenden Ideen, ein Verfahren zu entwickeln, das bei schneller Berechnung und guter Qualität möglichst flexibel ist.

Die interessanteste Neuerung ist die Verwendung von unterschiedlichen Temperaturen für verschiedene Knoten. Diese führen zu einer deutlichen Beschleunigung des Verfahrens, weil dadurch einige ungünstige Effekte im Simulationsverlauf, wie Schwingungen und Rotationen, erkannt werden können.

Hinzu kommen noch viele Detailverbesserungen, auf die später genauer eingegangen wird.

Kapitel 4

Allgemeine Bewertung

In diesem Kapitel werden die allgemeinen Vor- und Nachteile von Springembedder-Verfahren dargelegt und gegenübergestellt, damit klar wird, worauf beim Entwurf besonders Wert gelegt werden muß.

4.1 Vorteile

4.1.1 Qualität des Layouts

Die wichtigste Eigenschaft eines Layoutverfahrens ist naturgemäß die Qualität seines Resultats. Hier können sich Springembedder-Verfahren im Vergleich zu anderen Algorithmen durchaus sehen lassen.

Eine ihrer großen Stärken ist die Darstellung von im Graphen vorhandenen Symmetrien oder räumlichen Strukturen. Obwohl keines der Verfahren explizit darauf abzielt, diese Strukturen herauszuarbeiten, werden sie meist sehr gut wiedergegeben, wie z. B. die Graphen in den Abbildungen 4.1 und 4.2 auf der nächsten Seite illustrieren. Dies liegt daran, daß meistens symmetrische Kräfteformeln verwendet werden, die sich dann im Layout wiederspiegeln.

In den Abbildungen ist noch eine weitere positive Eigenschaft erkennbar. In beiden Fällen ist die Länge der Kanten ziemlich uniform, und auch die Knoten sind relativ gleichmäßig über die Zeichenfläche verteilt. Dies ergibt sich unmittelbar aus den Kräfteformeln, die meist bewußt so gewählt werden, daß eine Kante in ihrem lokalen Energieminimum liegt, wenn sie „optimale“ Länge hat. Die Abstoßung von unverbundenen Knoten führt zu einer guten Verteilung der Knoten.

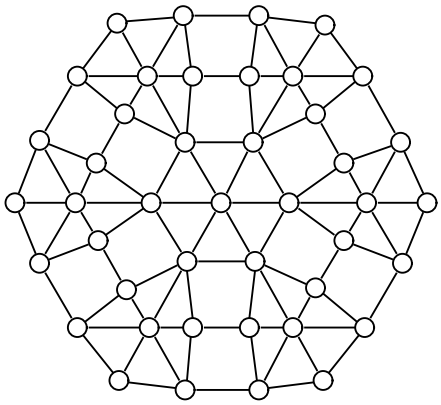


Abbildung 4.1: Darstellung von Symmetrien

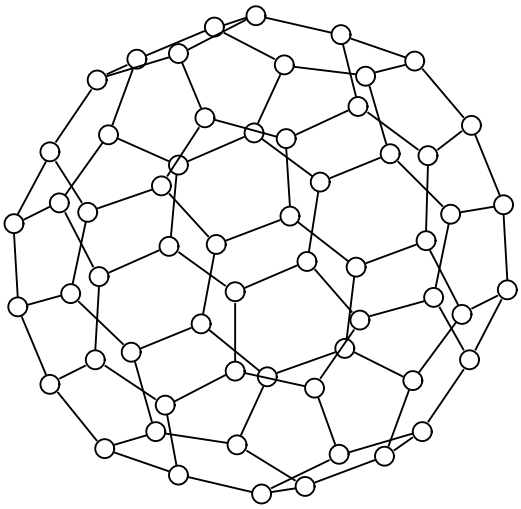


Abbildung 4.2: Darstellung von räumlichen Strukturen

4.1.2 Geschwindigkeit

Fast genauso wichtig wie ein gutes Layout ist die Geschwindigkeit des Verfahrens. Speziell bei Layoutverfahren erwartet der Benutzer interaktive Geschwindigkeit, also maximal einige Sekunden Antwortzeit für mittelgroße Graphen. Dies liegt an der Intuitivität der Problemstellung. Der Benutzer rechnet mit einem bestimmten Ergebnis und erwartet, daß der Computer genauso intuitiv „denkt“, wie er selbst. Die Schwierigkeit des Problems wird leicht unterschätzt.

Im allgemeinen haben Springembedder-Verfahren bei kleinen und mittleren Graphen eine gute bis akzeptable Laufzeit, auch wenn über ihre theoretische Komplexität meist keine Aussagen getroffen werden können. Allerdings ist darauf zu achten, keine zu komplexen Kräfteformeln zu verwenden. Die Komplexität der Kräfteberechnung pro Iteration sollte möglichst $\mathcal{O}(n^2)$ nicht überschreiten. Leider läßt sich für bestimmte Anwendungen $\mathcal{O}(n^3)$ nicht vermeiden.

4.1.3 Konzeptionelle Einfachheit

Ein wichtiger Vorteil von Springembedder-Verfahren ist ihre konzeptionelle Einfachheit. Um ihr Prinzip zu verstehen, bedarf es keiner Fachleute und ihre Arbeitsweise ist auch für Laien leicht nachvollziehbar. Entsprechend einfach sind die Algorithmen, die das Grundprinzip umsetzen. Eine erste naive Implementierung ist in kurzer Zeit machbar.

4.1.4 Flexibilität

Springembedder-Verfahren sind sehr flexibel und sehr gut auf individuelle Problemstellungen anpaßbar. Viele spezielle Anforderungen an das Layout können durch Änderungen der Kräfte und anderer Parameter des Verfahrens erfüllt werden. Eine Folge dieser Flexibilität ist, daß Springembedder-Verfahren für beliebige Graphen verwendet werden können. In diesem Gebiet gibt es zu Springembedder-Verfahren kaum Alternativen. Die meisten anderen Layoutverfahren konzentrieren sich auf bestimmte Graphklassen, wie etwa Bäume, gerichtete azyklische Graphen oder planare Graphen.

4.2 Nachteile

4.2.1 Nachweisbarkeit von Ergebnissen

Aufgrund ihrer Natur als Heuristiken ist es bei Springembedder-Verfahren in den meisten Fällen nicht möglich, theoretische Nachweise über die Qualität des Ergebnisses zu erbringen. Einzig und allein bei dem Verfahren von KAMADA und KAWAI (siehe Abschnitt 3.2) ist ein Nachweis möglich, daß das System zu einem lokalen Energieminimum konvergiert – ein in der Praxis eher nutzloses Ergebnis. Selbst wenn sich ein Graph in einem lokalen Energieminimum befindet, so bedeutet dies nicht unbedingt ein qualitativ gutes Layout. Ein lokal gesehen günstiger Energiezustand kann global gesehen immer noch sehr hoch sein. Viel wichtiger, als nachweislich ein lokales Minimum zu erreichen, ist es, dem globalen Minimum möglichst nahe zu kommen.

4.2.2 Erhalt bestimmter Eigenschaften

Springembedder-Verfahren garantieren kaum Eigenschaften des erzeugten Layouts. Bei der Anwendung auf z. B. planare Graphen ist das erzeugte Layout in vielen Fällen überschneidungsfrei – eine Garantie dafür gibt es jedoch nicht. Es ist für bestimmte Anwendungen möglich, bestimmte Eigenschaften des Layouts zu erzwingen, indem man die Simulation in einem zulässigen Zustand startet und dann bei jeder Änderung verhindert, daß die gewünschte Eigenschaft zerstört wird. Dies hat allerdings oft gravierende Auswirkungen auf die Laufzeit des Verfahrens, da diese Tests unter Umständen nicht effizient durchzuführen sind. Außerdem kann das gesamte Layout leiden, wenn durch Einschränkungen in der Simulation zusätzliche Energiebarrieren aufgebaut werden, die den Weg zum globalen Optimum versperren.

4.3 Herausforderungen

Die „klassischen“ Springembedder-Verfahren haben eine Menge an Problemen, die es in einem modernen Verfahren zu behandeln und zu lösen gilt.

4.3.1 Wahl der Parameter

In Abschnitt 4.1.4 wurde bereits die gute Anpaßbarkeit von Springembedder-Verfahren auf konkrete Problemfälle dargelegt. Diese große Flexibilität erschwert es jedoch um so mehr, ein allgemeines Verfahren zu entwickeln, das für möglichst viele Fälle gute Ergebnisse liefert. Durch eine bestimmte Wahl von Parametern entsteht für bestimmte Graphen ein sehr gutes Layout. Dieselben Kräfteformeln sind aber dann meistens für andere Graphen schlecht geeignet. So passiert es mit ungünstig gewählten Kräfteformeln sehr leicht, daß die Graphen bei schwachem Zusammenhang auseinanderdriften, sich dagegen bei starkem Zusammenhang sehr stark zusammenziehen und alle Knoten sich überlagern. Dies führt zu ästhetisch schlechten Layouts.

4.3.2 Konvergenz

Bedingt durch die physikalische Simulation bleiben die Graphen oft in energiereichen lokalen Energieminima „hängen“. Im Ergebnis des Layoutverfahrens erkennt man dies daran, daß der Graph in sich verworren ist. Für die Implementierung eines qualitativ hochwertigen Verfahrens ist es notwendig, lokale Minima zu meiden und durchaus auch temporäre Verschlechterungen in Kauf zu nehmen, wenn dadurch erreicht wird, daß der Graph endgültig ein besseres Layout erhält. Dies ist jedoch nicht immer mit der zielgerichteten Natur der Kräfte zu vereinbaren. In vielen Fällen ist es nicht möglich, das globale Optimum zu finden.

Ein anderes Problem, das weniger die Qualität des Ergebnisses, sondern mehr die Laufzeit des Verfahrens beeinträchtigt, sind Schwingungen, Rotationen und ähnliche Effekte, die durch die Diskretisierung der Kräfte und der Bewegungen der Knoten entstehen. Diese an sich nutzlosen Bewegungen sollten erkannt und beseitigt werden, um die Geschwindigkeit der Simulation zu erhöhen.

Kapitel 5

Algorithmische Details

Nachdem wir nun einen ungefähren Überblick über Springembedder-Verfahren gewonnen haben und in gewissem Maße auch deren Stärken und Schwächen kennen, soll nun Schritt für Schritt ein Algorithmus entwickelt werden, der allen Anforderungen an ein universelles Verfahren gerecht wird. Die wichtigsten Kriterien sind dabei die Laufzeit und die Qualität des Ergebnisses. Hinzu kommen Flexibilität und konzeptionelle Einfachheit.

Das Problem gliedert sich im wesentlichen in zwei Teile. Einerseits muß ein effizientes Simulationsverfahren entwickelt werden, das ausgehend von idealerweise beliebigen Kräfteformeln ein möglichst gutes Layout des Graphen berechnet. Andererseits müssen dann Kräfteformeln gefunden werden, die den Anforderungen an das Layout möglichst gut entsprechen und die trotzdem einfach und effizient zu berechnen sind.

5.1 Der Simulationsalgorithmus

Der zu entwickelnde Simulationsalgorithmus soll aus gegebenen Kräften und einer Startkonfiguration des Graphen ein möglichst tiefes Energieniveau finden. Dazu soll er folgende Eigenschaften aufweisen:

- Die wichtigste Eigenschaft ist die *Qualität* des Layouts. Die für Springembedder-Verfahren verwendeten Kräfteformeln tendieren dazu, im Energieverlauf viele energiereiche lokale Minima zu haben, die einem suboptimalen Layout des Graphen entsprechen. In den meisten Fällen ist es kaum möglich, das

globale Energieminimum in angemessener Zeit zu finden. Trotzdem gilt es, die energiereichen lokalen Energieminima so gut wie möglich zu meiden und dem globalen Energieminimum möglichst nahe zu kommen.

- Fast genauso wichtig ist die *Effizienz* des Verfahrens. Da man nicht erwarten kann, ein optimales Layout zu erhalten, ist man auch nicht bereit, große Laufzeiten zu investieren. Dafür gibt es bereits Verfahren, die bei wesentlich längerer Laufzeit ein besseres Ergebnis liefern, wie etwa Simulated Annealing (siehe [DH91]).

- Eine nicht zu vernachlässigende Forderung ist die nach *Flexibilität*. Es ist nicht immer vorauszusehen, welche Anforderungen Benutzer an das Layout eines Graphen stellen. Deshalb soll es mit vertretbarem Aufwand möglich sein, das bestehende Verfahren an neue Anforderungen anzupassen. Zu diesem Zweck soll der Simulationsalgorithmus mit nahezu beliebigen Kräften arbeiten können.

Anhand dieser Eigenschaften sollen nun die verschiedenen Aspekte der einzelnen bestehenden Verfahren miteinander verglichen und deren besten Lösungen mit neuen Ideen zu einem neuen Verfahren verbunden werden.

5.1.1 Das Grundverfahren

Alle vorgestellten Verfahren verfolgen im wesentlichen dasselbe Grundprinzip. Ausgehend von geeigneten Startkoordinaten wird in mehreren Iterationen für die Knoten die jeweils auf sie wirkende Gesamtkraft berechnet und anschließend ein oder mehrere Knoten in Richtung dieser Kraft bewegt. Dies wird solange wiederholt, bis ein bestimmtes Terminierungskriterium erfüllt ist. Die Unterschiede zwischen den Grundverfahren liegen einerseits in der Reihenfolge, in der die Knoten abgearbeitet werden, andererseits darin, wann die Kräfte neu berechnet werden. Das Ergebnis davon ist ein unterschiedlich gutes Konvergenzverhalten.

Die Grundverfahren sind mit folgenden Funktionen parametrisierbar, die vorerst als gegeben vorausgesetzt werden und auf deren Wahl anschließend genauer eingegangen wird:

Algorithmus 5.1 Die Verfahren von EADES und FRUCHTERMAN/REINGOLD

Eingabe: Graph $G = (V, E, A)$
Ausgabe: Layout L_G

```
L_G := init(G)
repeat
  for all u ∈ V do
    F_u := force(u, L_G)
  end for
  for all u ∈ V do
    move(u, L_G, F_u)
  end for
until finished()
```

- Die Funktion „init“ liefert zu einem Graphen G ein initiales Layout L_G, also Anfangskoordinaten für alle Knoten zum Beginn der Simulation.

- Die Funktion „force“ berechnet zu einem Knoten u die auf ihn wirkende Gesamtkraft F_u. Dabei ist F_u ein Vektor und bestimmt sowohl Betrag als auch Richtung der Kraft.

- Die Funktion „move“ bewegt in einem Layout L_G einen Knoten u unter Berücksichtigung der auf ihn wirkenden Kraft F_u und anderer Zusatzinformationen – wie etwa dem momentanen Gesamtzustand der Simulation – an eine neue Position.

- Die Funktion „finished“ liefert einen bool’schen Wert zurück, der festlegt, ob die Simulation beendet werden soll.

Die Verfahren von EADES und von FRUCHTERMAN und REINGOLD verwenden zueinander ähnliche Simulationsverfahren, die in Algorithmus 5.1 dargestellt sind. Sie berechnen pro Iteration zuerst die Kräfte für alle Knoten und bewegen die Knoten anschließend.

Die Positionsänderung etwa des ersten Knotens wird für die Bewegung weiterer Knoten also nicht berücksichtigt. Dies führt zu Ungenauigkeiten bei der Bewegung. Viele Knoten werden anhand einer Kraft bewegt, die nicht mehr aktuell ist. Die Folge ist ein mäßiges Konvergenzverhalten, es werden mehr Iterationen be-

Algorithmus 5.2 Das Verfahren von KAMADA und KAWAI

Eingabe: Graph $G = (V, E, A)$, Genauigkeit ε
Ausgabe: Layout L_G

```
L_G := init(G)
repeat
  for all u ∈ V do
    F_u := force(u, L_G)
  end for
  Wähle v ∈ V mit |F_v| = max{ |F_u| | u ∈ V }
  repeat
    move(v, L_G, F_v)
    F_v := force(v, L_G)
  until |F_v| < ε
until finished()
```

nötigt, als bei den anderen Verfahren. Andererseits werden die Kräfte nicht so oft neu berechnet, was sich positiv auf die Geschwindigkeit einer Iteration auswirkt.

KAMADA und KAWAI sprechen in ihrer Arbeit nicht von Kräften. Sie bewegen den aktuellen Knoten immer in Richtung des Gefälles der Energiefunktion. Physikalisch/mathematisch gesehen ist dies jedoch dasselbe. In ihrem Verfahren, das in Algorithmus 5.2 skizziert ist, werden pro Iteration zuerst die Kräfte für alle Knoten berechnet. Anschließend wird der Knoten gewählt, dessen Kraft vom Betrag her am größten ist. Dieser Knoten wird mehrmals hintereinander bewegt, wobei jedesmal die auf ihn wirkenden Kräfte neu berechnet werden. Dies wird solange wiederholt, bis die Kraft unter eine bestimmte Schwelle gefallen ist. Das Erreichen der Schwelle wird durch die verwendeten Kräfteformeln und durch die Berechnung der Bewegung garantiert. Das Simulationsverfahren von KAMADA und KAWAI ist schneller als die beiden anderen genannten, da es ein wesentlich besseres Konvergenzverhalten aufweist und somit mit weniger Iterationen auskommt.

Auch im GEM-Verfahren, das in Algorithmus 5.3 auf der nächsten Seite dargestellt ist, wird in jeder Iteration nur ein Knoten bewegt, allerdings nur einmal. Dieser Knoten wird zufällig gewählt, wobei aber garantiert wird, daß erst alle Knoten abgearbeitet werden, bevor ein Knoten ein zweites Mal bewegt wird. Bezüglich des Konvergenzverhaltens ist das GEM-Verfahren mit dem von KAMADA und KAWAI vergleichbar, dabei kommt es mit etwas weniger Kraftberechnungen

Algorithmus 5.3 Das GEM-Verfahren

Eingabe: Graph $G = (V, E, A)$
Ausgabe: Layout L_G

```
L_G := init(G)
repeat
  for all u ∈ V in zufälliger Reihenfolge do
    F_u := force(u, L_G)
    move(u, L_G, F_u)
  end for
until finished()
```

aus. Der große Vorteil ist seine Einfachheit. Dadurch wird die Integration neuer Funktionalität wesentlich vereinfacht.

5.1.2 Berechnung des Anfangszustands

Um die Simulation beginnen zu können, ist es notwendig, die Koordinaten der Knoten vor Beginn auf geeignete Werte zu initialisieren. Diese Koordinaten werden schon von Anfang an benötigt, einerseits um die wirkenden Kräfte zu berechnen, andererseits um eine Basis für die Bewegung der Knoten zu haben. Die Berechnung der Startkonfiguration entspricht der in Abschnitt 5.1.1 verwendeten Funktion „init“.

Bei bestimmten Anwendungen sind die Startkoordinaten bereits vorgegeben, etwa wenn ein Graph bereits gezeichnet wurde, aber durch kleine Änderungen ein neues Layout notwendig wird. Dabei soll sich das Gesamtbild des Graphen möglichst wenig ändern, die sogenannte „mental map“ soll erhalten bleiben. In diesen Fällen werden die Positionen der Knoten mit den gegebenen Koordinaten initialisiert.

In anderen Anwendungen sind keine Koordinaten vorhanden, etwa, wenn der zu zeichnende Graph vorher noch keine visuelle Repräsentation hatte. In diesem Fall müssen über eine geeignete Strategie Startwerte festgelegt werden. Was ist nun die ideale Ausgangsposition für die Knoten? Welchen Einfluß hat die initiale Plazierung überhaupt auf die Laufzeit des Verfahrens und die Qualität des Ergebnisses?

Die „klassischen“ Verfahren beschäftigen sich nicht näher mit verschiedenen Varianten für die Startkoordinaten. EADES und FRUCHTERMAN/REINGOLD verwenden zufällige Koordinaten. KAMADA und KAWAI plazieren die Knoten in beliebiger Reihenfolge auf die Ecken eines regulären n-Ecks. Sie behaupten, daß der Anfangszustand wenig Auswirkung auf das entstehende Layout hat. FRICK widerspricht ihnen hier – und auch sich selbst. Er behauptet in [F+94] in Übereinstimmung mit KAMADA und KAWAI, daß die initiale Plazierung wenig Einfluß auf die Qualität des Ergebnisses hat, in [Fri97] jedoch weist er darauf hin, daß sie großen Einfluß haben kann. Unabhängig davon ist er sich einig, daß die Laufzeit des Verfahrens stark davon abhängig ist. Dies ist leicht nachvollziehbar: Wenn ein Graph schon bei Beginn des Simulationsverfahrens nahe an dem globalen Energieminimum liegt – also per Definition schon einigermaßen schön gezeichnet ist, dann sind die wirkenden Kräfte von vornherein schwächer und das Verfahren konvergiert schneller.

Ob und wieviel die Startkoordinaten dagegen Auswirkungen auf die Qualität des Endergebnisses haben, hängt im wesentlichen davon ab, wie gut das Verfahren energiereiche lokale Minima vermeiden kann. Betrachtet man z. B. den vollständigen Binärbaum mit der Tiefe 5, dann stellt man fest, daß bei einer ungünstigen Ausgangsposition z. B. das Verfahren von KAMADA und KAWAI oft dazu tendiert, den Graphen mit Überschneidungen zu zeichnen, wie in Abbildung 5.1 auf der nächsten Seite. Verfahren, die jedoch darauf optimiert sind, ungünstige lokale Energieminima zu meiden, wie etwa das GEM-Verfahren oder das im Rahmen dieser Arbeit entwickelte, liefern wesentlich seltener verworrene Bilder, und dies ziemlich unabhängig von der Ausgangsposition. Man erhält fast immer ein Ergebnis wie in Abbildung 5.2 auf der nächsten Seite. Insgesamt ist es jedenfalls wünschenswert, mit möglichst guten Koordinaten zu beginnen.

FRICK nennt in [Fri97] einige neue Strategien für die Berechnung der Startkoordinaten, so daß sich insgesamt etwa folgende Möglichkeiten ergeben:

Vorgegebene Koordinaten Die Koordinaten der Knoten werden nicht initialisiert, sondern die vom Benutzer vorgegebenen Koordinaten werden übernommen.

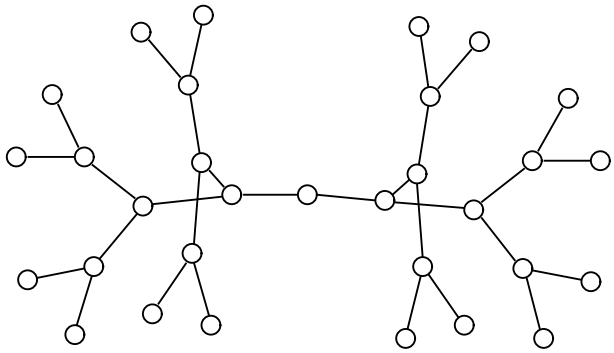


Abbildung 5.1: Vollständiger Binärbaum in hohem lokalem Energieminimum

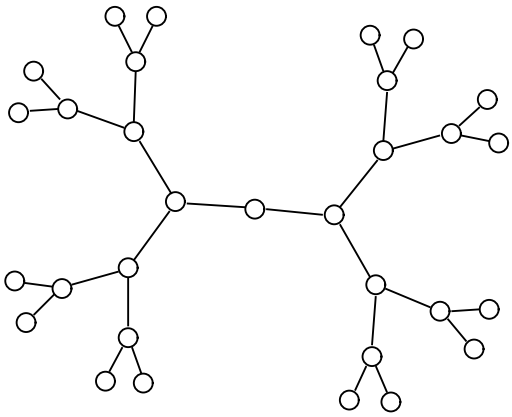


Abbildung 5.2: Vollständiger Binärbaum in globalem Energieminimum

Zufallsverteilung EADES und FRUCHTERMAN/REINGOLD initialisieren die Koordinaten der Knoten mit gleichverteilten Zufallszahlen. FRICK nennt zusätzlich die Möglichkeit, andere Verteilungen, wie etwa eine Normalverteilung, zu verwenden.

Reguläres n-Eck Die Knoten werden auf den Ecken eines regulären n-Ecks plaziert, wie im Verfahren von KAMADA und KAWAI. Dort wird nicht näher auf die Reihenfolge eingegangen. FRICK schlägt vor, mit geeigneten Strategien eine Reihenfolge zu bestimmen, die möglichst wenig Kantenschnitte erzeugt. Dazu könnten einfache Verfahren, wie Tiefensuche, Breitensuche oder ähnliches, verwendet werden.

Topologische Plazierung FRICK verwendet schließlich eine weitere, vielversprechende Methode, die ursprünglich von WATANABE aus [Wat88] stammt. Die Knoten werden ausgehend vom Zentrum des Graphen betrachtet, wobei Knoten bevorzugt werden, die nur noch wenige unplazierte Nachbarn haben. Jeder Knoten wird ausgehend vom Schwerpunkt seiner bereits plazierten Nachbarn durch mehrmalige Anwendung der bekannten Kräfte energetisch günstig plaziert.

Die besten Ergebnisse ergeben sich unter Verwendung der topologischen Plazierung. Allerdings ist die Laufzeit dieser Initialisierung nicht zu vernachlässigen. Für den Fall, daß man diese Zeit nicht investieren will, bieten sich Zufallskoordinaten an.

5.1.3 Verwendung globaler oder lokaler Temperaturen

Um die Geschwindigkeit des Verfahrens und die Qualität des Layouts zu verbessern, schlagen FRUCHTERMAN und REINGOLD erstmals vor, zu begrenzen, wie weit sich ein Knoten bewegen darf. Dies formulieren sie mit dem Begriff der „Temperatur“ des Graphen, der aus dem Simulated Annealing übernommen wurde. Je größer die Temperatur ist, desto weiter dürfen sich Knoten bewegen. Am Anfang der Simulation beginnt man mit einer relativ hohen Temperatur. Knoten dürfen

sich weit bewegen, um schnell zum Endergebnis zu gelangen und lokale Minima zu vermeiden. Im Laufe der Zeit senkt man die Temperatur. Es werden nur noch immer kleinere Knotenbewegungen erlaubt, um genauer das Minimum zu treffen und damit die Qualität des Ergebnisses zu verbessern.

FRUCHTERMAN und REINGOLD arbeiten mit einer Temperatur für den gesamten Graphen, für jeden Knoten gilt dieselbe Temperatur. Der Temperaturverlauf wird über eine vordefinierte Formel festgelegt. Hier können verschiedene Temperaturverläufe verwendet werden. FRUCHTERMAN und REINGOLD lassen erst die Temperatur indirekt proportional zur Zahl der bereits berechneten Iterationen abnehmen. Später stellen sie fest, daß es besser ist, mehrphasig zu arbeiten – erst mit einer hohen, schnell abnehmenden Temperatur und dann mit einer niedrigen, konstanten Temperatur.

Im GEM-Verfahren kommt eine andere Idee zur Anwendung. Statt einer globalen Temperatur für den ganzen Graphen wird hier jedem Knoten eine eigene lokale Temperatur zugewiesen. Diese wird im Laufe des Verfahrens nach bestimmten Kriterien angepaßt. So beinhaltet das GEM-Verfahren Heuristiken, um zu erkennen, ob ein Knoten ins Schwingen geraten ist, oder ob der Graph rotiert. Wenn dies der Fall ist, so werden die betroffenen Knoten selektiv abgekühlt. Wenn dagegen ein Knoten mehrmals hintereinander eine Bewegung in dieselbe Richtung gemacht hat, so wird seine Temperatur erhöht, um diese zielgerichtete Bewegung zu unterstützen.

Während ein globaler Temperaturverlauf leichter zu implementieren ist als eine lokale Knotentemperatur, so hat dieser Ansatz das Problem, den richtigen Verlauf zu finden. Läßt man die Temperatur zu schnell sinken, so gelingt es dem Verfahren nicht, den Graphen weit genug zu transformieren und bis zu einem Minimum zu gelangen. Sinkt die Temperatur dagegen zu langsam, so vergeudet man Zeit.

Verwendet man lokale Knotentemperaturen, so regelt sich die Temperatur von selbst. Außerdem stellt dies die einzige Möglichkeit dar, Schwingungen und Rotationen des Graphen zu erkennen und zu verhindern. Deshalb sind lokale Temperaturen zu bevorzugen.

5.1.4 Berechnung der Bewegung

Unabhängig davon, ob man globale oder lokale Temperaturen verwendet, ist zu entscheiden, wie man in Abhängigkeit von der Temperatur T_v und der Kraft F_v eines Knoten v dessen Bewegung δ_v berechnet.

In diesem Zusammenhang ist es notwendig, Kraft, Temperatur und Bewegung zueinander in Relation zu stellen. Physikalisch gesehen kann man diese Größen nicht direkt vergleichen, weil sie verschiedene Einheiten haben, die Kraft Newton (N), die Temperatur Kelvin (K) und die Bewegung Meter (m). Genaugenommen muß man Korrekturfaktoren verwenden, um Abhängigkeiten zu formulieren, etwa $|\delta_v| = \min\{\alpha|F_v|, \beta T_v\}$, wobei α die Einheit $\frac{m}{N}$ und β die Einheit $\frac{m}{K}$ hat. Diese Faktoren werden hier – wie in den meisten anderen Arbeiten zu diesem Thema auch – weggelassen, da in der Simulation alle Größen einfache Zahlen sind. Es ist jedoch möglich – und in vielen Fällen auch notwendig, Korrekturfaktoren ohne Einheit zu verwenden, um die Größenordnung von Kraft, Temperatur und Bewegung aneinander anzupassen.

Ziemlich eindeutig ist die Wahl der Bewegungsrichtung. Der Knoten wird in dieselbe Richtung bewegt, in die die Kraft weist. Mathematisch bedeutet dies

$$\frac{\delta_v}{|\delta_v|} = \frac{F_v}{|F_v|}, \quad \text{also} \quad \delta_v = |\delta_v| \cdot \frac{F_v}{|F_v|}.$$

Zu bestimmen bleibt noch die Größe der Bewegung $|\delta_v|$. Hierzu gibt es verschiedene Möglichkeiten:

- Die Verfahren von EADES und KAMADA/KAWAI verwenden überhaupt keine Temperatur, bei ihnen gilt

$$|\delta_v| = |F_v|.$$

- FRUCHTERMAN und REINGOLD verwenden normalerweise den Betrag der Kraft. Nur wenn diese größer als die Temperatur ist, so wird sie auf die Temperatur verkürzt:

$$|\delta_v| = \min\{|F_v|, T_v\}$$

- FRICK schließlich berücksichtigt die Länge des Kraftvektors überhaupt nicht, die Größe der Bewegung ist nur von der Temperatur abhängig:

$$|\delta_v| = T_v$$

Die Methode von EADES und KAMADA/KAWAI scheidet von vornherein aus, wenn man Temperaturen verwenden will. Bei den anderen beiden Verfahren erscheint die Wahl von FRUCHTERMAN und REINGOLD eigentlich die natürlichere. Es stellt sich allerdings heraus, daß es tatsächlich günstiger ist, wie im GEM-Verfahren vorzugehen und den Knoten auch bei kleinen Kräften entsprechend der Temperatur zu bewegen. Dies wirkt sich zwar negativ auf die Gesamtlaufzeit des Verfahrens aus, hat aber den Vorteil, daß lokale Minima besser vermieden werden.

5.1.5 Erkennung von Schwingungen und Rotationen

Bei naiven Implementierungen von Springembedder-Verfahren geschieht es leicht, daß der Graph oder Teile davon beginnen, zu schwingen oder zu rotieren. Dies geschieht durch die Diskretisierung, die durch die Knotenbewegung implizit vorgenommen wird. Wenn ein Knoten über die für ihn aktuell beste Position hinaus bewegt wird, so hat dies zur Folge, daß seine nächste Bewegung mit großer Wahrscheinlichkeit in die andere Richtung geht. Ist auch diese wieder zu lang, so wird er in der nächsten Iteration wieder zurückbewegt. Dieses Verhalten tritt vor allem im Endstadium der Simulation auf, wenn alle Knoten schon ziemlich nahe an ihrer optimalen Position sind.

Die negativen Folgen von Schwingungen und Rotationen sind einerseits eine längere Simulationsdauer, andererseits ein qualitativ schlechteres Ergebnis. Deshalb sollen hier einige Heuristiken vorgestellt werden, die Schwingungen und Rotationen erkennen und verhindern. Zur Erkennung, ob ein Knoten schwingt oder rotiert, ist es notwendig, die Bewegungen oder die auf die Knoten wirkenden Kräfte über mehrere Iterationen zu vergleichen.

Im GEM-Verfahren, aus dem die wesentlichen Ideen stammen, wird der letzte Kraftvektor mit dem aktuellen verglichen. Der Winkel zwischen den beiden gibt

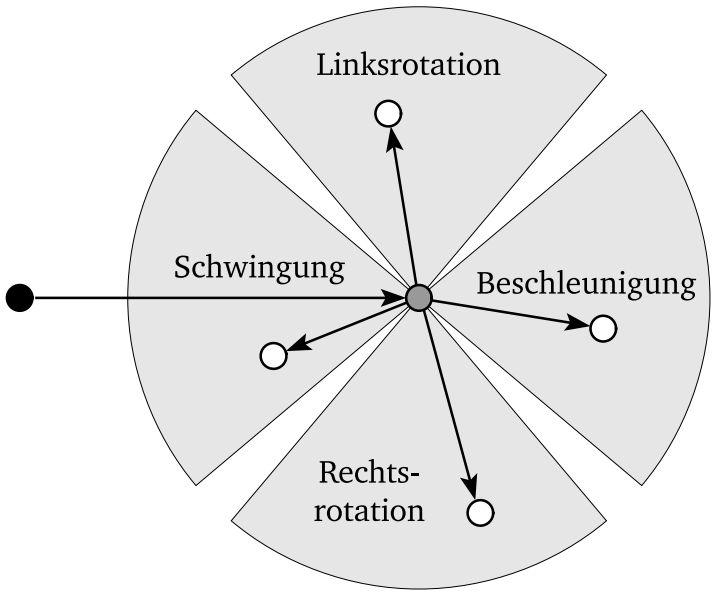


Abbildung 5.3: Erkennung von Schwingungen und Rotationen

Aufschluß über Schwingungen und Rotationen. In Abbildung 5.3 sind die möglichen Fälle dargestellt. Über eine Heuristik, die die folgenden drei Fälle berücksichtigt, wird die Temperatur angepaßt:

Beschleunigung: Hat der Kraftvektor in etwa dieselbe Richtung wie derjenige in der letzten Iteration, so hat die vorherige Bewegung nicht ausgereicht. Deshalb wird die Temperatur erhöht, um größere Bewegungen zu erlauben und den Knoten schneller an die Position zu bringen, an den ihn die Kräfte ziehen.

Schwingung: Hat der Kraftvektor in etwa die dem letzten Kraftvektor entgegengesetzte Richtung, so war die letzte Bewegung zu groß. Der Knoten ist über sein Ziel hinausgeschossen und muß ein Stück zurückbewegt werden – aber eine geringere Distanz, als in der letzten Iteration, sonst kann der Graph in Schwingungen geraten. Deshalb wird in dieser Situation die Temperatur erniedrigt, um den Knoten zu bremsen.

Rotation: Die Rotationsheuristik ist etwas komplexer als die anderen beiden. Hier reicht es nicht aus, die Temperatur des Knotens zu erniedrigen, wenn sein

Kraftvektor in dem in Abbildung 5.3 auf der vorherigen Seite angedeuteten Winkel zum vorherigen Kraftvektor steht. Eine einzelne Bewegung nach diesem Muster läßt noch nicht unbedingt auf eine Rotation im Graphen schließen. Vielmehr ergibt sich dies erst, wenn der Knoten wiederholt eine Kurve nach dem beschriebenen Muster vollzieht. Deshalb wird am Knoten ein Zähler gespeichert, der zu Beginn der Simulation auf 0 initialisiert wird und dann bei jeder Linksrotation erhöht und bei jeder Rechtsrotation erniedrigt wird. Erst wenn der Betrag des Zählers einen Schwellwert überschreitet, wird die Temperatur des Knotens erniedrigt.

In den ersten beiden Punkten erscheint die Argumentation von FRICK sehr einleuchtend – und die Geschwindigkeit seines Verfahrens gibt ihm auch recht. Die verwendete Rotationsheuristik dagegen scheint etwas unausgegoren. Der genannte Zähler speichert die Differenz der Linksbewegungen gegenüber den Rechtsbewegungen. Es ist nicht ganz einsichtig, inwiefern diese Information, die über das ganze Verfahren verteilt entstanden ist, Aussagen über das momentane Rotationsverhalten eines Knoten machen kann. Außerdem ist in manchen Fällen eine Rotation – zumindest eines Teilgraphen – sogar notwendig, um die Knoten zueinander in die richtige Lage zu bringen. Solche Bewegungen sollten eigentlich nicht unterbunden werden.

5.1.6 Erkennung von Translationen

Neben Schwingungen und Rotationen kann – speziell bei dem Grundalgorithmus des GEM-Verfahrens – noch ein weiterer Effekt auftreten, der die Konvergenz des Verfahrens beeinträchtigt. Durch ein ungünstiges Zusammenspiel der berechneten Kräfte mit dem Simulationsalgorithmus ist es möglich, daß sich der ganze Graph oder Teile davon beliebig weit in eine Richtung bewegen, obwohl die Kräfte dies nicht unbedingt fordern. Dieses Verhalten soll mit Translation bezeichnet werden. Ein Beispiel dazu sieht man in Abbildung 5.4 auf der nächsten Seite. Erst wird der linke Knoten betrachtet. Da die Kante zu lang ist, wird er nach rechts bewegt. Nun wird der rechte Knoten betrachtet. Jetzt ist aber die Kante zu kurz, und der

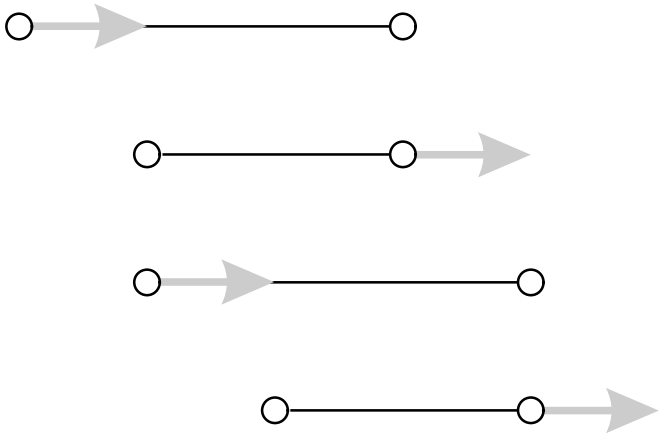


Abbildung 5.4: Translation eines einfachen Graphen

Knoten wird auch nach rechts bewegt. Jetzt ist die Kante wieder zu lang. Dies kann sich beliebig lange wiederholen. Verstärkt wird der Effekt noch durch die in Abschnitt 5.1.5 vorgestellte Heuristik zur Erkennung von Beschleunigungssituationen.

Glücklicherweise läßt sich dieses Verhalten leicht verhindern, wenn die Knoten nicht immer in derselben Reihenfolge betrachtet werden. Nehmen wir etwa an, daß im obigen Beispiel die Knoten nicht strikt abwechselnd bewegt werden, sondern manchmal auch derselbe Knoten zweimal hintereinander. Wenn dies geschieht, so dreht sich die Richtung der Translation um. Durch die zufällige Reihenfolge wird dann im Mittel jeder Knoten richtig bewegt.

Auf diese Art und Weise können zwar Schwingungen entstehen, die eigentlich der Konvergenz entgegenwirken. Diese werden jedoch von der bereits vorgestellten Heuristik erkannt und blockiert.

5.1.7 Kriterien zur Terminierung

Wie entscheidet man, nach wievielen Iterationen man die Simulation anhält und den Graphen dem Benutzer als Endergebnis präsentiert? Die Anzahl der Iterationen entspricht direkt der Laufzeit des Verfahrens, und offensichtlich hängt auch die Qualität des Ergebnisses wesentlich von dieser Entscheidung ab. Je länger man das Verfahren laufen läßt, desto besser wird das Ergebnis sein. Allerdings ist zu

berücksichtigen, daß sich die Simulation nach einer gewissen Zeit in der Nähe eines lokalen Energieminimums einpendelt und die Qualität des Layouts nicht mehr entscheidend besser wird. Diesen Zustand gilt es zu erkennen.

EADES und FRUCHTERMAN und REINGOLD beschäftigen sich überhaupt nicht näher mit dieser Fragestellung, sie verwenden eine konstante Anzahl von Iterationen. Dies hat zur Folge, daß in vielen Fällen die Qualität des Ergebnisses leidet, weil sich das Layout noch nicht ausreichend stabilisiert hat. In anderen Fällen wird Laufzeit verschenkt, weil das Verfahren schon viel früher ein vergleichbares Ergebnis hätte liefern können.

KAMADA und KAWAI machen die Terminierung von einer Schranke für die Kräfte abhängig. Sobald es keine Kräfte mehr gibt, deren Betrag über dieser Schranke liegt, wird das Verfahren abgebrochen. Durch den mathematischen Hintergrund ist hier sogar garantiert, daß dieses Terminierungskriterium irgendwann eintreten wird.

FRICK macht die Terminierung des Verfahrens von der Temperatur der Knoten abhängig. Dazu definiert er die Temperatur des Graphen als die Summe der Knotentemperaturen

$$T = \sum_{v \in V} T_v.$$

Diese läßt sich effizient berechnen. Nach der Initialisierung mit der Summe der Starttemperaturen der einzelnen Knoten wird sie nach jeder Temperaturänderung eines Knoten v in $\mathcal{O}(1)$ aktualisiert:

$$T^{i+1} = T^i + T_v^{i+1} - T_v^i.$$

Einen typischen Temperaturverlauf beim Layout eines mittelgroßen Graphen sieht man in Abbildung 5.5 auf der nächsten Seite. Das kurze konstante Stück am Anfang entspricht der ersten Iteration. Solange noch kein Knoten das zweite Mal bewegt wurde, kann auch die Temperatur nicht angepaßt werden, weil kein vorhergehender Kraftvektor vorhanden ist. Anschließend, etwa in der nächsten Iterationen steigt die Gesamttemperatur bedingt durch die Beschleunigungsheuristik an. In der dritten Iteration haben dann die ersten Knoten eine günstige Position

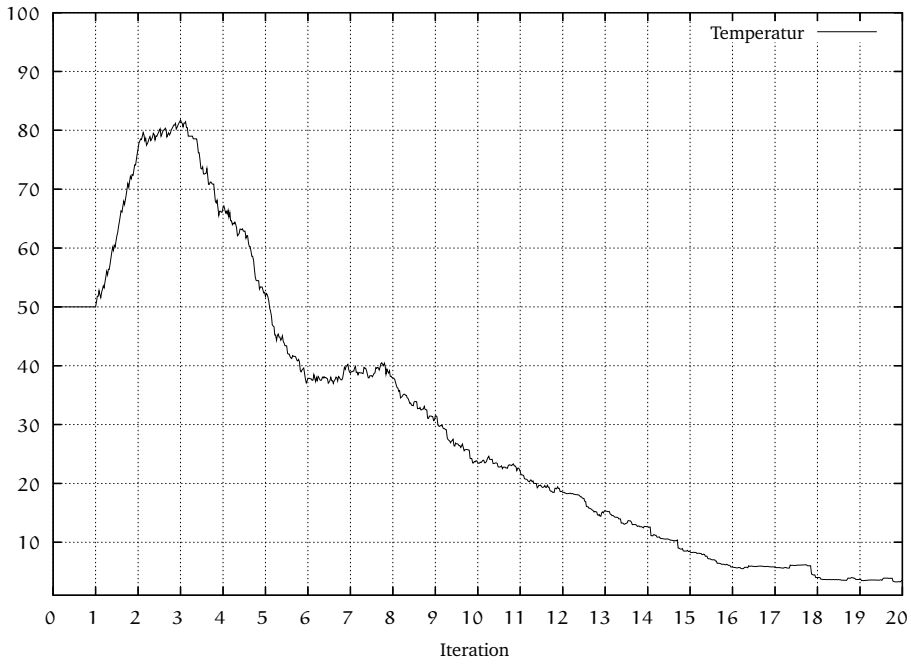


Abbildung 5.5: Typischer Temperaturverlauf

gefunden und bremsen durch die Schwingungsheuristik den Anstieg der Temperatur, die anschließend schnell zu fallen beginnt und nahezu monoton abnimmt.

Sobald die Temperatur unter eine bestimmte Schranke gefallen ist, wird die Simulation beendet. Da beim GEM-Verfahren im Gegensatz zu KAMADA/KAWAI die Konvergenz nicht nachgewiesen ist, muß – sozusagen als Notbremse – zusätzlich eine feste Schranke für die Anzahl der Iterationen eingebaut werden, um die Termination sicherzustellen. Im Normalfall sollte diese Schranke jedoch nicht erreicht werden. Im allgemeinen terminiert das Verfahren auch ohne Beschränkung auf eine bestimmte Anzahl von Iterationen – nur nachgewiesen werden konnte es bisher nicht.

5.1.8 Anwendung auf Untergraphen

Manchmal ist es nicht erwünscht, den ganzen Graphen neu zu zeichnen, sondern vielmehr nur einen Teil davon, etwa um nach dem Einfügen einiger neuer Knoten diese auch in einer bereits bestehenden Zeichnung gut zu plazieren.

Die Lösung dieser Aufgabenstellung ist ziemlich einfach. Man berücksichtigt im Simulationsverfahren nur die Knoten, die neu gezeichnet werden sollen. Alle anderen werden nur zur Berechnung der Kräfte herangezogen. Auf diese Art und Weise bleiben alle „alten“ Knoten an derselben Stelle, die neuen werden regulär nach dem Verfahren gezeichnet. Wichtig ist, daß durch diese Anwendung auch berücksichtigt wird, wie der Untergraph mit dem Rest des Graphen verbunden ist und an welcher Stelle noch Platz ist.

5.2 Wahl der Kräfte

Nachdem die meisten Aspekte zur Durchführung der Simulation untersucht wurden, fehlen noch die Kräfte, auf die die Simulation angewendet wird. Prinzipiell sind beliebige Kräfte denkbar – je nach den Anforderungen der konkreten Problemstellung. Voraussetzung ist jedoch, daß sich die Kräfteformeln effizient berechnen lassen. Die Kräfteberechnung findet in der innersten Schleife des Verfahrens statt, so daß deren Laufzeit sehr starken Einfluß auf die Gesamtlaufzeit hat.

KAMADA/KAWAI und DAVIDSON/HAREL sprechen in ihren Arbeiten nicht von Kräften sondern von Energieniveaus. Physikalisch gesehen stehen diese beiden Größen jedoch in einer direkten Beziehung. So entspricht die auf einen Knoten wirkende Kraft dem Gefälle im Energieniveau. Um einen besseren Vergleich der verschiedenen Verfahren zu ermöglichen, wird hier nur von Kräften gesprochen, die den gegebenenfalls verwendeten Energieformeln entsprechen.

5.2.1 Optimale Kantenlängen

Alle Verfahren beinhalten anziehende bzw. abstoßende Kräfte, die paarweise zwischen zwei Knoten wirken und den gewünschten Abstand regeln. Typische Parameter dieser Grundkräfte sind Wunschwerte für die Länge $l_{u,v}$ einer Kante $e = \{u, v\}$. Nicht alle Verfahren unterstützen explizit unterschiedliche optimale Längen für verschiedene Kanten. Allerdings kann man dies bei jedem der behandelten Verfahren annähernd erreichen, indem man für verschiedene Kanten verschiedene Parameter benutzt.

Die optimalen Kantenlängen können vom Benutzer vorgegeben sein, entweder explizit für jede einzelne Kante oder als ein Wert für alle Kanten. Wenn sie nicht gegeben sind, so kann ein geeigneter Wert in Abhängigkeit von der Struktur des Graphen und der zur Verfügung stehenden Bildschirmfläche gewählt werden. KAMADA und KAWAI etwa schlagen als optimale Länge für alle Kanten den Quotienten aus Bildschirmdurchmesser und Durchmesser des Graphen vor. FRUCHTERMAN und REINGOLD dagegen verwenden

$$l_{u,v} = C\sqrt{\frac{A}{|V|}}$$

als optimale Kantenlänge, wobei A die Fläche des Bildschirms ist und C eine experimentell ermittelte Konstante.

5.2.2 Grundkräfte

Wenn man zwei Knoten betrachtet, die durch eine Kante verbunden sind, so sollen sie weder zu nahe beisammen noch zu weit auseinander liegen. Deshalb muß bei kurzen Kanten eine abstoßende Kraft wirken, bei langen Kanten dagegen eine anziehende Kraft. Offensichtlich soll, wenn eine Kante im Verfahren optimale Länge hat, zwischen den beiden Endknoten keine Kraft wirken. Die optimale Kantenlänge entspricht also einer Nullstelle der Gesamtkraft, die insgesamt monoton steigend ist. Diese Eigenschaft erfüllen alle bekannten Verfahren. Unterschiede gibt es in den Details.

Wie bereits berichtet, verwendet EADES zwei getrennte Kräfteformeln. Einerseits wirkt paarweise zwischen allen Knoten eine abstoßende (*engl.* repelling) Kraft mit dem Betrag F_r, andererseits zwischen Knoten, die durch Kanten verbunden sind, eine anziehende (*engl.* attracting) Kraft mit dem Betrag F_a. Es kommen folgende Formeln zum Einsatz:

$$F_a(u,v) = c_1 \ln \frac{d(u,v)}{c_2}, \qquad F_r(u,v) = \frac{c_3}{d(u,v)^2}.$$

Wie stehen nun die drei Parameter c_1, c_2 und c_3 in Beziehung zu der optimalen Kantenlänge? Mathematisch muß für $d(u,v) = l_{u,v}$ gelten:

$$F_a(u,v) - F_r(u,v) = 0,$$

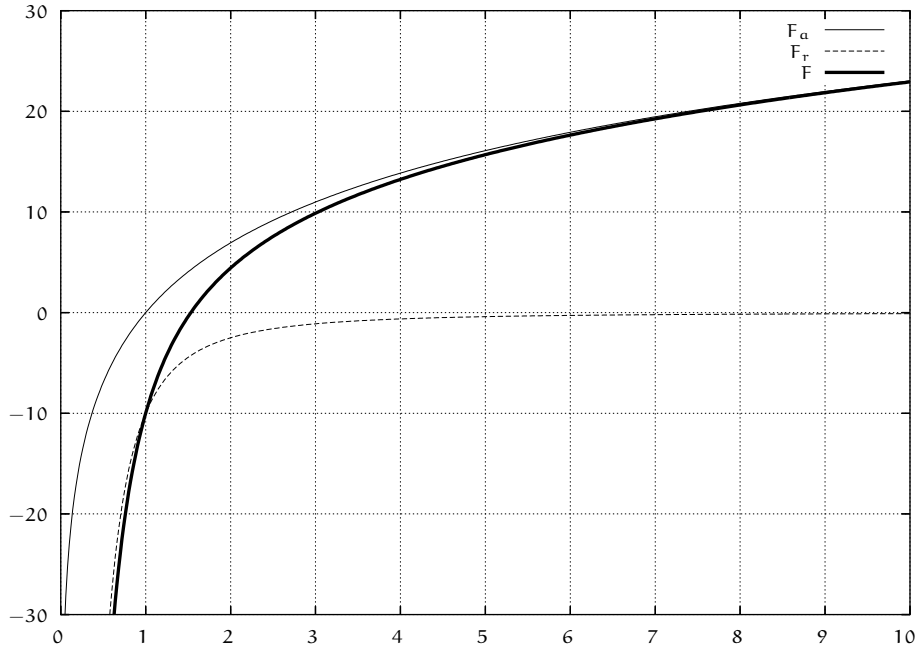


Abbildung 5.6: Grundkräfte bei EADES ($c_1 = 10$,$c_2 = 1$,$c_3 = 10$)

also

$$c_1 \ln \frac{l_{u,v}}{c_2} - \frac{c_3}{l_{u,v}^2} = 0.$$

Leider ist es mit herkömmlichen algebraischen Mitteln nicht ohne weiteres möglich, diese Formel nach $l_{u,v}$ aufzulösen. Tatsächlich ergibt sich

$$l_{u,v} = c_2 e^{\frac{1}{2}\Omega\left(\frac{2c_3}{c_1 c_2^2}\right)},$$

wobei Ω die LAMBERT'sche Ω-Funktion ist (siehe [Cha91]). Mit der für sie charakteristischen Eigenschaft $\Omega(x) \cdot e^{\Omega(x)} = x$ läßt sich das Ergebnis leicht bestätigen. Für die im Beispiel in Abbildung 5.6 gewählten Parameter gilt $l_{u,v} \approx 1,531584394$.

FRUCHTERMAN und REINGOLD verwenden dasselbe Grundprinzip wie EADES, aber andere Kraftformeln, die die optimale Kantenlänge direkt beinhalten. Konkret ziehen sie folgende Formeln mit verschiedenen Exponenten e in Betracht:

$$f_a(u,v) = \frac{d(u,v)^e}{l_{u,v}} \qquad\qquad f_r(u,v) = \frac{l_{u,v}^e}{d(u,v)}.$$

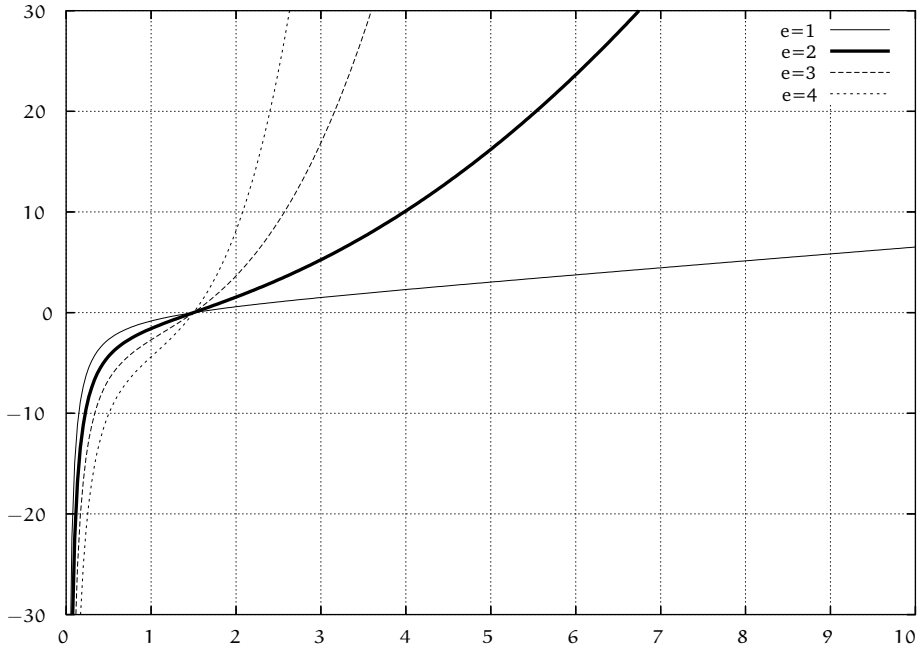


Abbildung 5.7: Verschiedene Varianten bei FRUCHTERMAN und REINGOLD

Einige dieser Funktionen sind in Abbildung 5.7 dargestellt. FRUCHTERMAN und REINGOLD wählen für ihr Verfahren $e = 2$. Der Kraftverlauf dafür ist in Abbildung 5.8 auf der nächsten Seite zum Vergleich mit den anderen Formeln noch einmal genauer abgebildet. Insgesamt ergeben diese Kräfteformeln ähnliche Ergebnisse wie die von EADES, sind aber deutlich schneller zu berechnen, da sie keinen teuren Logarithmus enthalten.

Die Wahl des Exponenten begründen FRUCHTERMAN und REINGOLD wie folgt: Bei $e = 1$ läßt sich das Ergebnis schneller berechnen als mit höheren Exponenten, aber die Simulation bleibt leicht in schlechten lokalen Minima „hängen“ – vor allem bei komplexen Graphen. Für $e = 2$ ergibt sich ein signifikant besseres Ergebnis. Höhere Exponenten liefern im wesentlichen dasselbe Ergebnis, sind angeblich aber teurer zu berechnen.

Diese Aussage scheint auf den ersten Blick einleuchtend, wenn man nur von der Berechnung des Betrages der Kraft ausgeht. Berücksichtigt man allerdings die Berechnung des gesamten Vektors, so ergibt sich ein anderes Bild. Betrachten wir etwa die Kraftberechnung für $e = 3$: Sei $\Delta_{u,v}$ der Abstandsvektor von u und v,

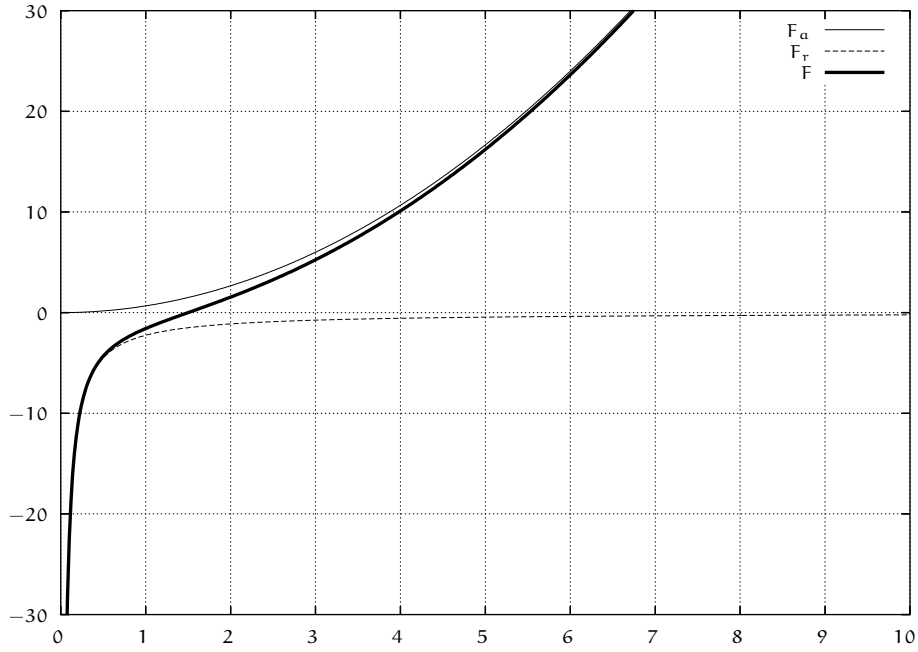


Abbildung 5.8: Grundkräfte bei FRUCHTERMAN und REINGOLD ($l_{u,v} = 1.5$, $e = 2$)

dann ist $d(u,v) = |\Delta_{u,v}|$. Der auf u wirkende Kraftvektor Φ_u läßt sich wie folgt berechnen:

$$\begin{aligned}
\Phi_u &= \frac{\Delta_{u,v}}{|\Delta_{u,v}|} \cdot (F_a(u,v) - F_r(u,v)) \\
&= \frac{\Delta_{u,v}}{|\Delta_{u,v}|} \cdot \left(\frac{d(u,v)^3}{l_{u,v}} - \frac{l_{u,v}^3}{d(u,v)} \right) \\
&= \frac{\Delta_{u,v}}{|\Delta_{u,v}|} \cdot \left(\frac{|\Delta_{u,v}|^3}{l_{u,v}} - \frac{l_{u,v}^3}{|\Delta_{u,v}|} \right) \\
&= \Delta_{u,v} \cdot \left(\frac{|\Delta_{u,v}|^2}{l_{u,v}} - \frac{l_{u,v}^3}{|\Delta_{u,v}|^2} \right).
\end{aligned}$$

$|\Delta_{u,v}|^2$ läßt sich schneller aus $\Delta_{u,v}$ berechnen als $|\Delta_{u,v}|$, da auf das teure Wurzelziehen verzichtet werden kann. Damit ist die Kräfteberechnung für $e = 3$ sogar schneller als für $e = 2$, wie man auch aus dem Laufzeittest in Abbildung 5.11 auf Seite 42 ersehen kann. Dies hat offensichtlich auch FRICK erkannt und seine Kräfteformeln ähnlich gewählt:

$$F_a(u,v) = \frac{d(u,v)^3}{l_{u,v}^2} \qquad\qquad F_r(u,v) = \frac{l_{u,v}^2}{d(u,v)}.$$

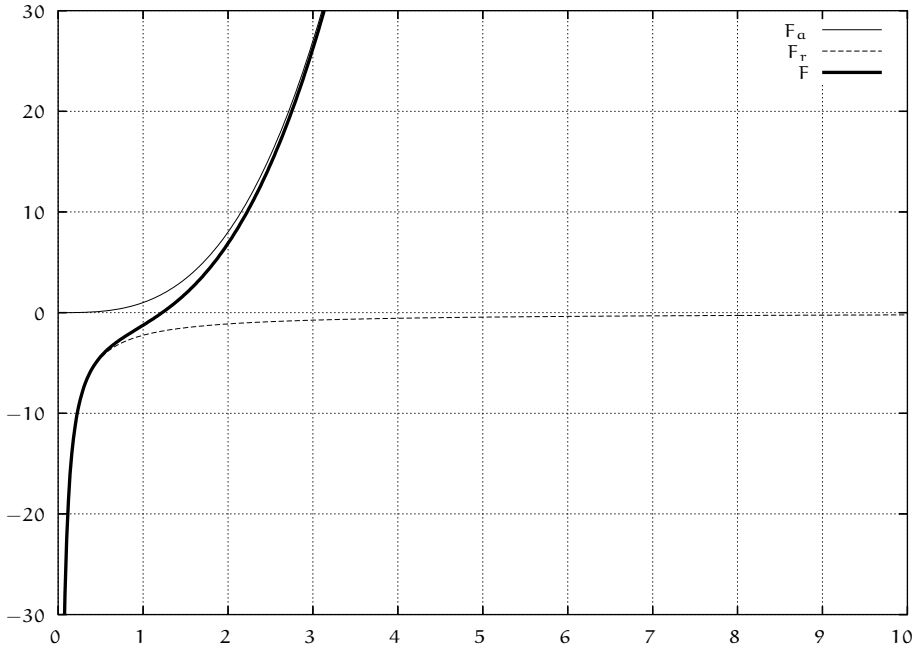


Abbildung 5.9: Grundkräfte beim GEM-Verfahren ($l_{u,v} = 1.5$)

Wenn man diese Kräfteformeln mit $l_{u,v}$ multipliziert, so ergeben sich dieselben Kräfte wie bei FRUCHTERMAN und REINGOLD mit $e = 3$.

KAMADA und KAWAI schließlich haben ein anderes Grundprinzip: Bei ihnen gibt es nicht nur eine optimale Kantenlänge, also einen optimalen Abstand zwischen adjazenten Knoten, sondern einen optimalen Abstand für jedes Knotenpaar (u, v). Dieser optimale Abstand $l_{u,v}$ entspricht dem graphentheoretischen Abstand der beiden Knoten und wird in der Initialisierungsphase für alle Knotenpaare vorberechnet – etwa mit dem Floyd-Algorithmus in $\mathcal{O}(n^3)$. KAMADA und KAWAI sprechen nicht von Kräften, sondern nur vom Energieniveau des Graphen. Als Gesamtenergie des Graphen setzen sie dabei an:

$$E_G = \sum_{\substack{u,v \in V \\ u \neq v}} \frac{1}{2} k_{u,v} (d(u,v) - l_{u,v})^2.$$

Man kann also dem Knotenpaar (u, v) direkt die Energie

$$E_{u,v} = \frac{1}{2} k_{u,v} (d(u,v) - l_{u,v})^2$$

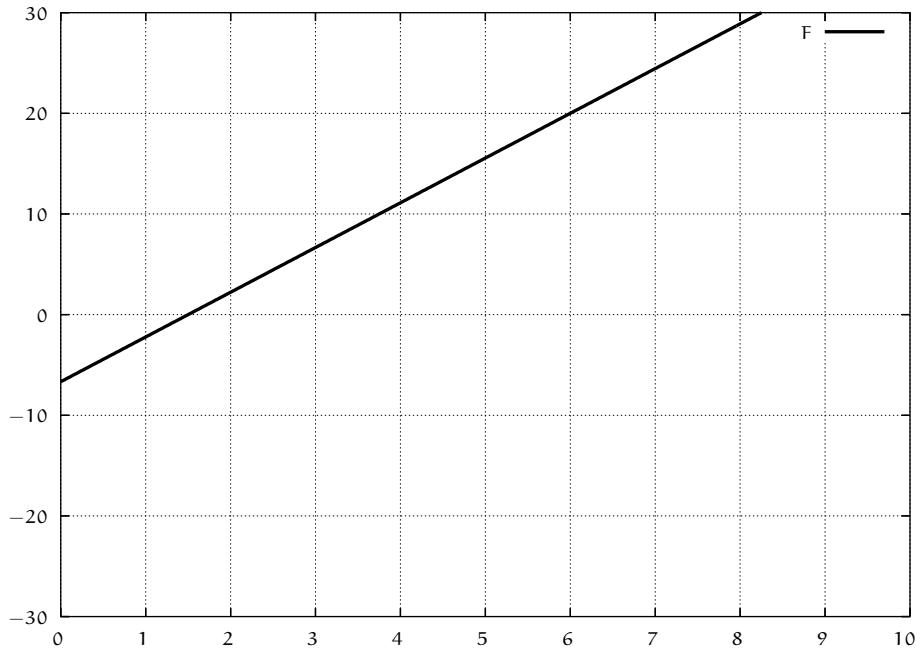


Abbildung 5.10: Grundkräfte bei KAMADA und KAWAI ($l_{u,v} = 1.5$, $K = 10$)

zuordnen, die der Kraft

$$F(u, v) = k_{u,v}(d(u, v) - l_{u,v})$$

entspricht. Dies ist genau das in der Realität für Federn geltende HOOK'sche Gesetz. Die Länge der Feder ist – wie zu erwarten war – $l_{u,v}$, die Härte $k_{u,v}$ wird wie folgt festgelegt:

$$k_{u,v} = \frac{K}{l_{u,v}^2},$$

wobei K eine Konstante des Verfahrens ist. Offensichtlich führen allerdings verschiedene Werte von K nur zu einer Skalierung der Gesamtenergie und sollten also vom Simulationsalgorithmus unabhängig sein. Die von KAMADA und KAWAI gewählten Kräfte führen zu einem qualitativ sehr guten Ergebnis und sind auch einigermaßen schnell zu berechnen. Allerdings ist die nicht unerhebliche Laufzeit der Vorverarbeitung zu berücksichtigen.

In einem praktischen Vergleich wurden die Laufzeiten der verschiedenen Kräfteformeln bei jeweils bestmöglicher Optimierung auf verschiedenen Rechnerplatt-

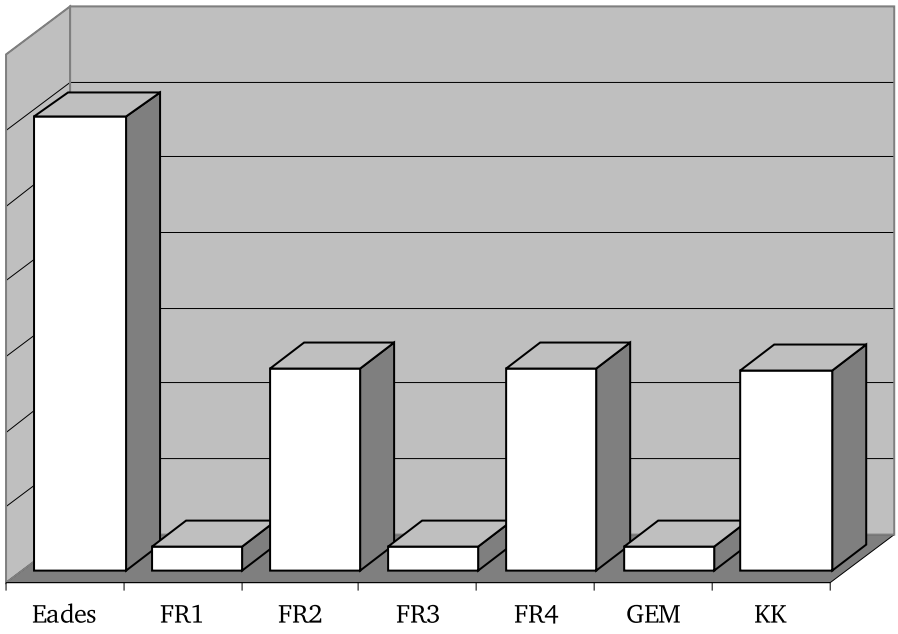


Abbildung 5.11: Laufzeit der Kräfteberechnungen

formen gegeneinander getestet. Das relative Ergebnis ist in Abbildung 5.11 dargestellt. Deutlich sichtbar ist der Unterschied, der dadurch entsteht, ob teure Operationen benötigt werden oder nicht. Bei den Kräften FR1, FR3 und GEM werden nur die Grundrechenarten benötigt. FR2, FR4 und KK benötigen je eine Quadratwurzelberechnung, Eades zusätzlich noch einen natürlichen Logarithmus. Aus dem deutlichen Unterschied folgt, daß man möglichst auf diese teuren Operationen verzichten sollte, vor allem da es auch schnellere Formeln gibt, die gute Ergebnisse liefern, etwa diejenigen aus dem GEM-Verfahren.

5.2.3 Constraints

In vielen Anwendungen werden an ein Layoutverfahren gewisse Nebenbedingungen gestellt. Es ist etwa vorgegeben, wie die relative Lage zweier Knoten zueinander sein soll, in etwa: Der Knoten u soll „rechts von“ v liegen. Solche Bedingungen werden meist als „Constraints“ bezeichnet.

Grundsätzlich gibt es zwei verschiedene Ansätze, Constraints zu behandeln. Der erste Ansatz, der etwa von SCHIRMER in [Sch96] verfolgt wird, garantiert,

daß Constraints erfüllt werden, sofern sie sich nicht gegenseitig widersprechen. Dazu wird das initiale Layout so gewählt, daß alle Constraints erfüllt sind. Dann wird bei jeder Iteration des Verfahrens sichergestellt, daß keines der Constraints verletzt wird. Dazu muß der Simulationsalgorithmus in nicht unerheblicher Weise abgeändert werden. So muß bei jeder Bewegung eines Knotens überprüft werden, ob dadurch ein gegebenes Constraint verletzt wird.

Die andere Möglichkeit ist, neue Kräfte einzuführen, die die Constraints unterstützen, und zu hoffen, daß sie stark genug sind, um die Constraints durchzusetzen. Natürlich kann die Erfüllung der Constraints dadurch nicht garantiert werden, was die Anwendung dieser Möglichkeit in bestimmten Anwendungen von vornherein ausschließt. Der große Vorteil dieser Methode ist die größere Flexibilität. So ergeben sich keine Änderungen am Simulationsalgorithmus, lediglich die Kräfte müssen angepaßt werden, was insgesamt eine relativ kleine Änderung darstellt. Neue Arten von Constraints lassen sich schnell durch neue Kräfte einfügen.

Abstrakt modelliert man Constraints leicht als „optimale Richtung“ für Kanten. Selbst wenn ein Constraint zwischen zwei Knoten gegeben ist, die nicht verbunden sind, so kann man die gegebene Aufgabenstellung leicht in eine neue transformieren, indem man eine temporäre Kante einfügt. Diese Kanten dürfen natürlich bei der Berechnung der Grundkräfte nicht berücksichtigt werden.

Um eine Kante durch Kräfte in eine bestimmte Richtung zu bringen, können etwa die von SUGIYAMA und MISUE in [SM94] vorgestellten magnetischen Kräfte verwendet werden. Die Kante wird als Magnetnadel modelliert, die sich in einem Magnetfeld ausrichtet. Dementsprechend wirken auf die adjazenten Knoten Kräfte im rechten Winkel zur Kante, wie in Abbildung 5.12 auf der nächsten Seite dargestellt. Für den Betrag der Kraft verwenden SUGIYAMA und MISUE folgende Formel mit verschiedenen Werten für c_1 und c_2:

$$F_c = d(u,v)^{c_1} \cdot \alpha^{c_2}.$$

Um verschiedene Richtungen für die Kanten zu modellieren, wird für jede Kante das Magnetfeld in eine andere Richtung orientiert. Wichtig ist noch, daß in Verbindung mit Constraints die Erkennung und Beseitigung von Rotationen abge-

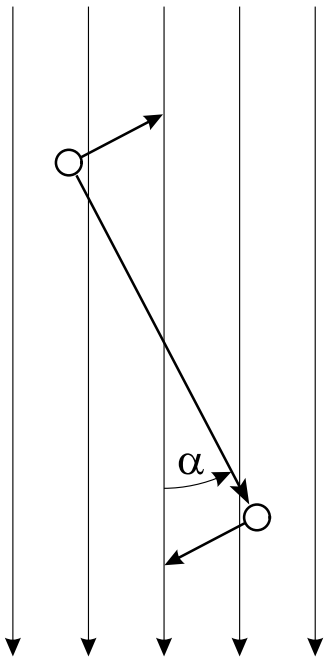


Abbildung 5.12: Magnetische Kräfte

schaltet werden muß. Hier ist eine Rotation ja ausdrücklich erwünscht, damit sich die Kante – und mit ihr der ganze Graph – in die gewünschte Richtung drehen kann.

5.2.4 Flächenbeschränkung

Eine ganz spezielle Art eines Constraints stellt eine Beschränkung der gegebenen Zeichenfläche – etwa auf die Bildschirmgröße – dar. Auch hier gibt es verschiedene Ansätze.

EADES und KAMADA/KAWAI behandeln eine Beschränkung der Zeichenfläche nicht explizit. Sie schlagen nur vor, die optimale Kantenlänge entsprechend zu wählen, damit die Größe des entstehenden Layouts in etwa der Größe der Zeichenfläche entspricht. Dies ist jedoch nur sehr schwer vorauszusehen, so daß in vielen Fällen der Graph über den Rand der Zeichenfläche hinausragt und in anderen Fällen Platz verschenkt wird.

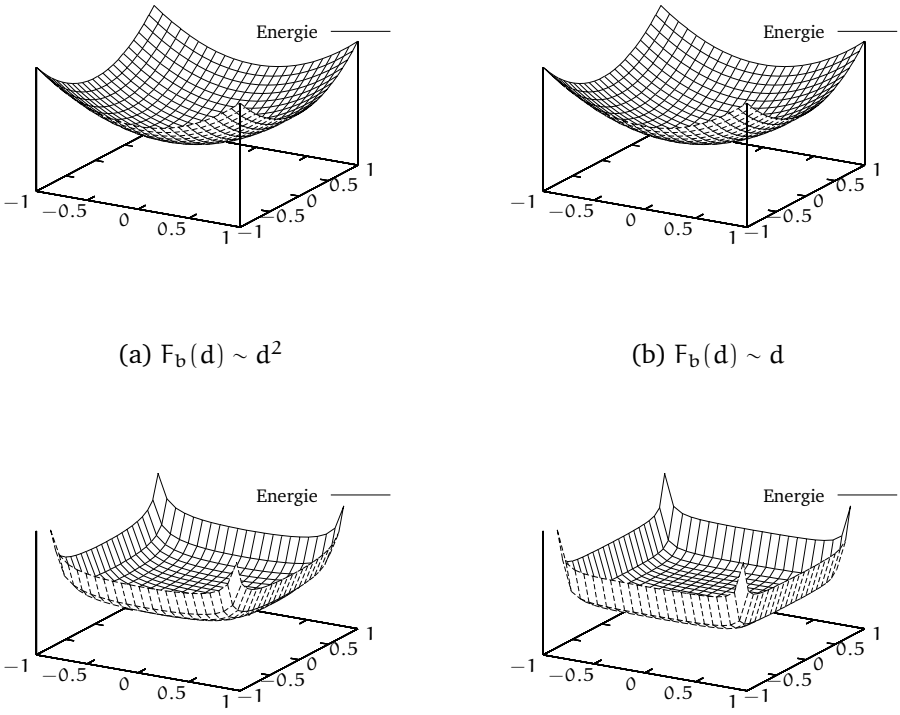
Bessere Ideen zur Beschränkung der Zeichenfläche stammen von DAVIDSON und HAREL. Deren erster Versuch war es, rund um die Zeichenfläche zusätzliche

„Dummyknoten“ anzuordnen, deren Positionen nicht verändert werden. Dies hat jedoch den Nachteil, daß die Anzahl dieser Knoten sowohl von der Größe der Zeichenfläche als auch von der Größe des Graphen anhängig ist. Es sind meist entweder zuwenig solche Knoten vorhanden, so daß Knoten des Graphen diese Barriere zu leicht überwinden können, oder zu viele Knoten, so daß die Berechnung der Kräfte zu aufwendig wird.

Deshalb verwenden DAVIDSON und HAREL statt Dummyknoten eine Energiebarriere, die verhindert, daß die Knoten den Rand überqueren. Grundsätzlich sind hier verschiedene Kräfte denkbar, die verschiedenen Einfluß auf das Layout haben. In Abbildung 5.13 auf der nächsten Seite sind die Energieniveaus verschiedener Kräfte dargestellt. Die beiden in den Abbildungen 5.13(a) und 5.13(b) dargestellten Kräfteformeln haben einen zu großen Einfluß, wenn Knoten noch weit vom Rand entfernt sind, so daß oft zwischen Rand und Graph noch zu viel Platz bleibt. Die anderen beiden Funktionen lösen dieses Problem besser, allerdings drängen sich dann oft viele Knoten an den Rand der Zeichenfläche. DAVIDSON und HAREL verwenden die Formel aus Abbildung 5.13(d).

FRUCHTERMAN und REINGOLD verfolgen eine andere Möglichkeit. Sie schlagen vor, den Rand der Zeichenfläche als vier unbewegliche Wände zu modellieren, die einen auftreffenden Knoten nach den bekannten physikalischen Formeln reflektieren. Dafür ergeben sich mehrere Möglichkeiten, die in Abbildung 5.14 auf der nächsten Seite dargestellt sind.

Elastischer Stoß Der Knoten wird an der Wand vollständig reflektiert, wobei der Reflexionswinkel analog zur Realität gleich dem Aufschlagswinkel ist. Es ist noch zu berücksichtigen, daß unter Umständen mehrere Wände getroffen werden. Dieser Ansatz hat zwei Nachteile. Erstens ist eine mehrfache Reflexion an mehreren Wänden aufwendig zu berechnen und zweitens werden Schwingungen des Graphen unterstützt. Die Kraft, die den Knoten an die Wand drängt, wird mit großer Wahrscheinlichkeit auch noch bei der nächsten Iteration existieren und den Knoten wieder an die Wand stoßen. Es ergibt sich derselbe Effekt wie bei einem hüpfenden Gummiball.



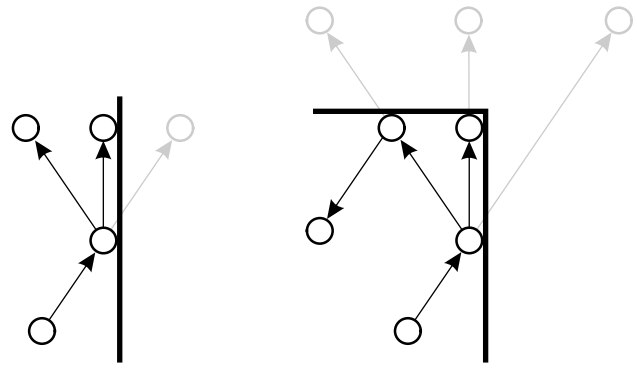
(a) $F_b(d) \sim d^2$

(b) $F_b(d) \sim d$

(c) $F_b(d) \sim \frac{1}{d^2}$

(d) $F_b(d) \sim \frac{1}{d^3}$

Abbildung 5.13: Verschiedene Energiebarrieren für den Rand



(a) Eine Wand

(b) Zwei Wände

Abbildung 5.14: Elastischer und unelastischer Stoß

Unelastischer Stoß Der Knoten bleibt an der Stelle stehen, an der er auf die Wand trifft. Dies hat den Nachteil, daß die Bewegungsfreiheit des Knotens zu stark eingeschränkt wird. Da eine erneute Bewegung des Knotens erst möglich wird, wenn auf den Knoten eine Kraft wirkt, die der Wand abgewandt ist, besteht die Gefahr, daß Knoten gegen die Wand gedrängt werden und dort an der Stelle, an der sie das erste Mal auftreffen, „hängenbleiben“. Eine Bewegung parallel zum Rand wird damit unterbunden.

Komponentenweise unelastischer Stoß Es wird nur die Komponente der Kraft beschränkt, die senkrecht zur Wand wirkt. Dies ist ein guter Kompromiß zwischen den ersten beiden Möglichkeiten. Die Bewegungsmöglichkeiten des Knotens werden nicht zu stark eingeschränkt, aber Schwingungen werden vermieden. Außerdem ist die Berechnung ziemlich einfach, es muß nur bei jeder Bewegung komponentenweise überprüft werden, ob sich der Knoten noch in der Zeichenfläche befindet, und gegebenenfalls die Koordinaten angepaßt werden.

Eine Erweiterung der Flächenbeschränkung auf beliebige Regionen, etwa Polygone, ist theoretisch leicht möglich, bringt jedoch in der Praxis einige Schwierigkeiten mit sich:

- Die Berechnung der Energiebarrieren bzw. der Reflexionen wird aufwendiger, da anstatt der parallel zum Koordinatensystem orientierten Wände beliebige Richtungen auftreten können.

- Bei Regionen mit konkaven Winkeln muß zusätzlich noch beachtet werden, daß auch Kanten die Zeichenfläche nicht verlassen wie in Abbildung 5.15 auf der nächsten Seite.

FRUCHTERMAN und REINGOLD schlagen als weitere Möglichkeit vor, den Graphen ohne besondere Vorkehrungen zu zeichnen und anschließend entsprechend zu skalieren, damit er die Bildschirmfläche gut ausfüllt. Dies hat jedoch zur Folge, daß die optimalen Kantenlängen und die Knotengrößen verändert werden. Deshalb ist diese Strategie nicht immer anwendbar.

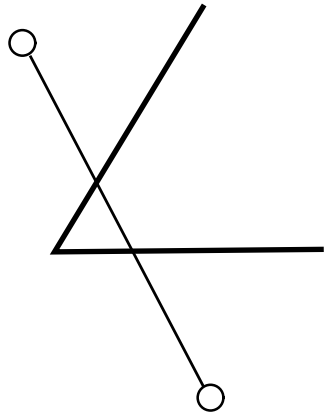


Abbildung 5.15: Konkave Regionen

5.3 Beschleunigungen

Die bisher beschriebenen Verfahren können durch bestimmte Maßnahmen noch beschleunigt werden. Ein große Beschleunigung stellt bereits die Verwendung von globalen oder lokalen Temperaturen in Verbindung mit Erkennungsheuristiken für Schwingungen und Rotationen dar.

Weitere Maßnahmen zielen auf die Beschleunigung der Kräfteberechnungen. Diese stellen den kritischen Punkt der Verfahren dar, weil sie in jeder Operation ausgeführt werden müssen. Einfachere Kräfteformeln ohne Verwendung von trigonometrischen Funktionen oder Wurzelberechnungen sind wesentlich schneller zu berechnen. Weitere Geschwindigkeitsverbesserungen erhält man durch Verwendung von Ganzzahlarithmetik, die sowohl von FRUCHTERMAN/REINGOLD als auch von FRICK vorgeschlagen wird. Da Ganzzahlberechnungen auf vielen Rechnerplattformen schneller ausgeführt werden können, lassen sich die Kräfte schneller berechnen. Allerdings sind solche Berechnungen wesentlich schwieriger zu handhaben, da leichter Über- oder Unterläufe auftreten.

Eine weitere Beschleunigung der Kräfteberechnung wird von FRUCHTERMAN und REINGOLD vorgeschlagen, die sogenannte Gittervariante ihres Verfahrens. Dabei werden abstoßende Kräfte zwischen Knoten nur berücksichtigt, falls ihr Abstand eine bestimmte Schranke nicht überschreitet. Mit Hilfe von 2D-Hashing können die zu berücksichtigenden Knoten leicht ermittelt werden. Dadurch senken

FRUCHTERMAN und REINGOLD die Komplexität der Kräfteberechnung auf $\mathcal{O}(m)$. Der Nachteil dieser Beschleunigung ist eine geringe Qualitätsminderung des Ergebnisses. Die entstehenden Layouts weisen unter Umständen leichte Verzerrungen auf.

Zusätzlich zur Vereinfachung der Kräfteformeln können auch neue Kräfte eingeführt werden, die die Konvergenz der Simulation beschleunigen. Im GEM-Verfahren werden zwei neue Kräfte verwendet, die die Simulation schneller zu einem Minimum führen und lokale Minima besser meiden. Das eine ist eine Gravitationskraft, die alle Knoten in Richtung des Schwerpunktes des Graphen zieht. Dabei wird jedem Knoten eine Masse zugewiesen, die von seinem Knotengrad abhängt. Dabei wird davon ausgegangen, daß Knoten mit vielen Nachbarn eher im Zentrum des Graphen sitzen sollen. Diese bekommen daher eine größere Masse als solche mit wenigen Nachbarn. Die andere verwendete Kraft ist eine Zufallskomponente, sogenanntes „weißes Rauschen“. Dieses hilft, lokale Minima zu vermeiden und liefert somit bei kürzerer Laufzeit bessere Ergebnisse. Diese Kraft kann nur sinnvoll eingesetzt werden, wenn Temperaturen verwendet werden, da sonst das entstehende Layout durch die Zufallsbewegungen verschlechtert wird.

5.4 Auftretende Probleme

5.4.1 Unzusammenhängende Graphen

Bisher wurde nicht erwähnt, wie mit unzusammenhängenden Graphen umgegangen werden soll. Bei naiv implementierten Springembedder-Verfahren driften die einzelnen Zusammenhangskomponenten eines Graphen beliebig weit auseinander, da zwischen ihnen nur abstoßende Kräfte wirken. Die existierenden Verfahren lösen dieses Problem durch unterschiedliche Methoden.

KAMADA und KAWAI schlagen vor, zuerst die Zusammenhangskomponenten der Graphen einzeln zu zeichnen und dann nebeneinander auf die Zeichenfläche zu positionieren. Diese Vorgehensweise hat einige Vorteile. So wird durch die Teilung des Graphen in seine Zusammenhangskomponenten die Laufzeit des Gesamtverfahrens gesenkt, weil die Kräfteberechnung mindestens quadratische Laufzeit hat.

Außerdem haben die einzelnen Zusammenhangskomponenten aufeinander keinen Einfluß, so daß ihr jeweiliges Layout nicht deformiert wird. Andererseits ist nicht klar, nach welcher Strategie die Zusammenhangskomponenten nach ihrem Layout positioniert werden sollen. Allgemein kann man nicht erwarten, daß eine flächenoptimale Plazierung erreichbar ist, da dieses Problem $\mathbb{NP}$-vollständig ist.

FRUCHTERMAN und REINGOLD lösen das Problem durch die in Abschnitt 5.3 beschriebene Gittervariante ihres Verfahrens. Dadurch, daß Kräfte von weiter voneinander entfernt liegenden Knoten nicht berücksichtigt werden, driften unzusammenhängende Graphen auch nur so weit auseinander, bis die Kräfte nicht mehr wirken.

Auch beim GEM-Verfahren löst sich das Problem von selbst, da durch die in Abschnitt 5.3 beschriebene Gravitationskraft die Zusammenhangskomponenten zusammengehalten werden.

5.4.2 Aufeinanderliegende Knoten

Im Laufe der Simulation kann es vorkommen, daß zwei Knoten exakt aufeinander zu liegen kommen. In diesem Fall kann die übliche Grundkraft nicht berechnet werden, weil die Richtung zwischen den beiden Knoten nicht definiert ist. Zusätzlich ergibt sich bei einigen der Kräfteformeln eine Division durch Null. Deshalb ist dieser Fall gesondert zu behandeln. In der Literatur gibt es zwei Strategien, wie in diesem Fall zu verfahren ist. FRUCHTERMAN und REINGOLD nehmen an, die beiden Knoten hätten einen winzigen Abstand in zufälliger Richtung und berechnen die Kräfte genauso wie immer. FRICK dagegen ignoriert in dieser Situation einfach die Kräfte zwischen den beiden Knoten. Er geht davon aus, daß die beiden Knoten in der nächsten Iteration nicht mehr aufeinanderliegen. Dies ist durch die Zufallskraft im GEM-Verfahren nahezu garantiert.

Kapitel 6

Berücksichtigung von Knotengrößen

Die bisher vorgestellten Verfahren behandeln Knoten als Punkte – also als geometrische Objekte ohne Ausdehnung. In der Praxis ist dies oft eine zu starke Einschränkung. Beispielsweise tragen in vielen Anwendungen die Knoten eines Graphen Beschriftungen, die zu unterschiedlichen Größen führen. Auch die bisherige Modellierung von Kanten als Strecken – also Linien ohne Breite – ist nicht in allen Fällen ausreichend. So können etwa auch für Kanten Beschriftungen gegeben sein, oder der Graph enthält Mehrfachkanten, die dann mit einem gewissen Abstand nebeneinander gezeichnet werden müssen und damit mehr Platz benötigen.

Es sollen von nun an Graphen betrachtet werden, deren Knoten und Kanten Ausdehnungen besitzen. Diese Information soll in beliebiger Weise durch die Attributfunktion des Graphen gegeben sein. Der Einfachheit halber gehen wird davon ausgegangen, daß die Knoten als achsenparallele Rechtecke gezeichnet werden und daß jede Kante eine konstante Breite hat. Eine Erweiterung auf Knoten mit beliebiger Form, z. B. auf Polygone oder auf Kanten mit variierender Breite wäre möglich, bringt konzeptionell jedoch kaum neue Erkenntnisse und verlangsamt die Simulation wesentlich, da die Berechnungen komplizierter werden.

Die zusätzlich gegebenen Informationen sollen soweit möglich beachtet werden, um die Qualität des Layouts zu verbessern. Natürlich ist es auch möglich, die Knotengrößen und Kantendicken zu ignorieren und die Knoten auf den Punkten, die eines der bereits vorgestellten Verfahren errechnet hat, zu zentrieren. Tatsächlich führt dies bei vielen Graphen zu ansprechenden Layouts. Voraussetzung dafür ist jedoch, daß sich die Größen der Knoten nicht zu stark unterscheiden und

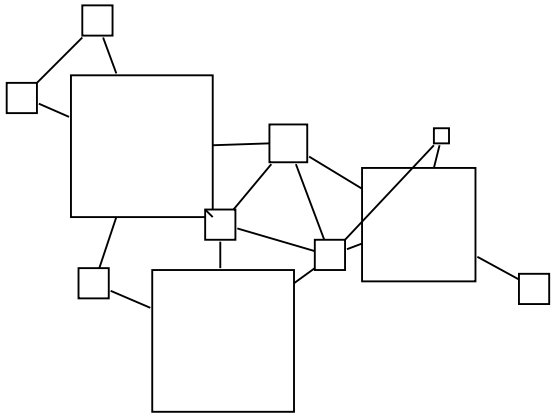


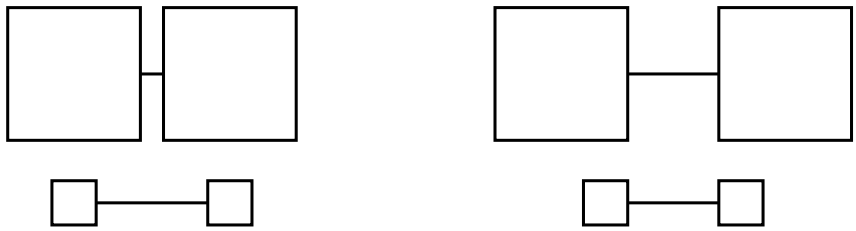
Abbildung 6.1: Layout ohne Berücksichtigung der Knotengröße

im Vergleich zu den Kantenlängen klein sind. Bei Graphen mit stark variierenden Knotengrößen dagegen bleibt für die kleinen Knoten zu viel Platz, und die großen überschneiden sich gegenseitig, wie in Abbildung 6.1 zu sehen.

Es stellt sich erneut die Frage, was überhaupt ein schönes Layout für einen solchen Graphen mit erweiterten Attributen ist. Die bereits vorgestellten Ästhetikkriterien bleiben natürlich weiterhin gültig, soweit sie anwendbar sind, müssen aber gegebenenfalls angepaßt und erweitert werden. Dadurch ergeben sich dann veränderte Anforderungen an das Verfahren, vor allem an die verwendeten Kräfte.

6.1 Kantenlängen

Ein Kriterium war die Forderung nach gleichmäßig langen Kanten bzw. nach Kanten mit vorgegebener Länge. Dieses Kriterium bleibt weiterhin bestehen, muß aber etwas angepaßt werden. Es stellt sich die Frage, wie sich die Länge einer Kante überhaupt bestimmt, wenn die Knoten eine echte Größe besitzen. Es gibt die Möglichkeit, vom Mittelpunkt des einen Knoten zum Mittelpunkt des anderen Knoten zu messen oder von Rand zu Rand. In Abbildung 6.2 auf der nächsten Seite werden beide Möglichkeiten gegenübergestellt. Es sind jeweils zwei verschieden große Knotenpaare dargestellt, zwischen denen Kanten gleicher Länge verlaufen. In dem einen Fall wurde die Länge der Kante zwischen den beiden Knotenmittelpunkten gemessen, in dem anderen Fall zwischen den Knotenrändern.



(a) Messung ab der Mitte (b) Messung von Rand zu Rand

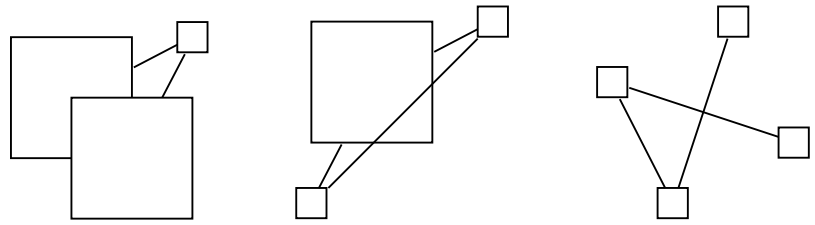
Abbildung 6.2: Bestimmung der Kantenlänge

Hier wird deutlich, daß das menschliche Auge die Kanten eher als gleich lang empfindet, wenn vom Rand aus gemessen wird. Diese Empfindung sollte beim Algorithmus berücksichtigt werden. Der erste Schritt bei der Anpassung des Verfahrens auf echte Knotengrößen besteht also darin, bei einer Beibehaltung der sonstigen Parameter die Länge von Kanten und den Abstand von Knoten nicht mehr ab der Knotenmitte zu messen, sondern vom Knotenrand aus. Dadurch ändern sich auch die wirkenden Kräfte, da diese ja direkt von der aktuellen Kantenlänge abhängen.

Tatsächlich ergibt sich durch diese kleine Anpassung ein wesentlich besseres Layout als vorher. Es entstehen auch aus Graphen mit stark variierenden Knotengrößen Layouts, deren Kantenlängen dem Anwender uniform erscheinen. Die Laufzeit des Verfahrens nimmt kaum zu, die Berechnung der Kräfte ist nur unwesentlich aufwendiger.

6.2 Überschneidungen

Ein Problem, das weiterhin besteht, sind viele Überschneidungen im Layout des Graphen. Während bisher nur Überschneidungen zwischen Kanten zu betrachten waren, müssen jetzt auch Überlappungen von Knoten und Überschneidungen von Knoten und Kanten berücksichtigt und möglichst verhindert werden. In Abbildung 6.3 auf der nächsten Seite sind die verschiedenen Möglichkeiten von Schnitten abgebildet.



(a) Knoten/Knoten (b) Knoten/Kante (c) Kante/Kante

Abbildung 6.3: Überschneidungen

6.2.1 Kanten/Kanten-Schnitte

Überschneidungen von Kanten untereinander ergeben sich sowohl in Graphen mit Größen als auch in solchen ohne. Die Maßnahmen um Kantenschnitte zu minimieren sind deshalb dieselben. Man versucht nicht explizit, die Schnittanzahl zu minimieren, sondern vertraut darauf, daß sich durch die Verteilung der Knoten automatisch nur eine geringe Zahl an Schnitten ergibt. Diese Lösung ist nicht optimal, liefert aber im allgemeinen gute Ergebnisse.

6.2.2 Knoten/Knoten-Schnitte

Während einige Kantenschnitte meist akzeptabel sind, sollten Überlappungen von Knoten immer vermieden werden, soweit dies die Vorgaben für die Größe der Zeichenfläche erlauben. Hierzu müssen gesonderte Maßnahmen ergriffen werden. Wie schon bei der Realisierung von Constraints gibt es zwei grundsätzliche Möglichkeiten.

Die erste Methode garantiert, daß keine Überlappungen auftreten, indem beginnend von einer überschneidungsfreien Zeichnung bei jeder Knotenbewegung verhindert wird, daß neue Überlappungen auftreten. So einfach, wie sich dies anhört, ist es leider nicht. Das Problem besteht darin, eine überschneidungsfreie Zeichnung zur Initialisierung zu finden. Solange genügend Platz vorhanden ist, ist dies leicht möglich. Wenn allerdings die Zeichenfläche stark eingeschränkt ist, wird dies schwieriger. Außerdem wird durch die Vermeidung von Überlappungen

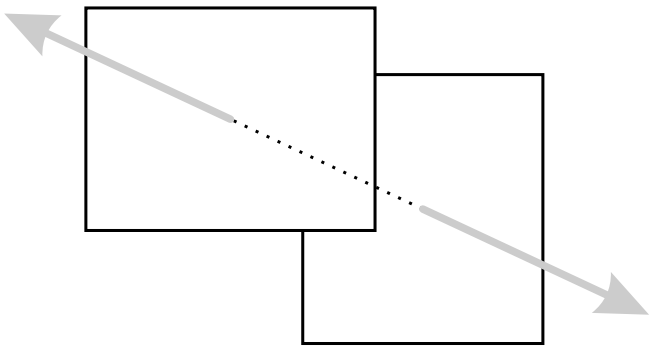


Abbildung 6.4: Kräfte zwischen überlappenden Knoten

weitgehend verhindert, daß Knoten Positionen tauschen und damit einen Weg aus einem lokalen Minimum finden. Dies sind nichttriviale Probleme, die eine umfassende Implementierung schwierig machen.

Die zweite Methode ist da unproblematischer. Sie ist ziemlich einfach zu implementieren, kann dafür aber kein überschneidungsfreies Layout garantieren. Die Idee ist, noch weitere Kräfte einzuführen, die zwischen zwei überlappenden Knoten eine starke Abstoßung hervorrufen. Wie die Richtungen der zusätzlichen Kräfte gewählt werden, ist aus Abbildung 6.4 ersichtlich. Auf beide Knoten wirkt eine abstoßende Kraft in Richtung der Verbindungslinie der Mittelpunkte. Die Berechnung des Betrag der Kraft ist fast nebensächlich, solange sie gegenüber den anderen Kräften stark genug ist.

6.2.3 Knoten/Kanten-Schnitte

Neben Knotenüberlappungen und Kantenschnitten treten auch Schnitte zwischen Knoten und Kanten auf, die ebenfalls vermieden werden sollen.

Dafür gibt es dieselben Gegenmaßnahmen wie bei den Knotenüberlappungen. Mit der gleichen Begründung ist auch hier eine kräftebasierte Strategie vorzuziehen. Die natürliche Wahl der Kräfte ist in Abbildung 6.5 auf der nächsten Seite skizziert. Man stellt fest, auf welcher Seite der Kante der Knotenmittelpunkt liegt und läßt auf den Knoten eine abstoßende Kraft senkrecht zur Kante wirken. Dieselbe Kraft läßt man in entgegengesetzter Richtung auf die Kante wirken. Da Kanten in unserer Simulation überhaupt nicht auftauchen, werden die auf sie wirkenden

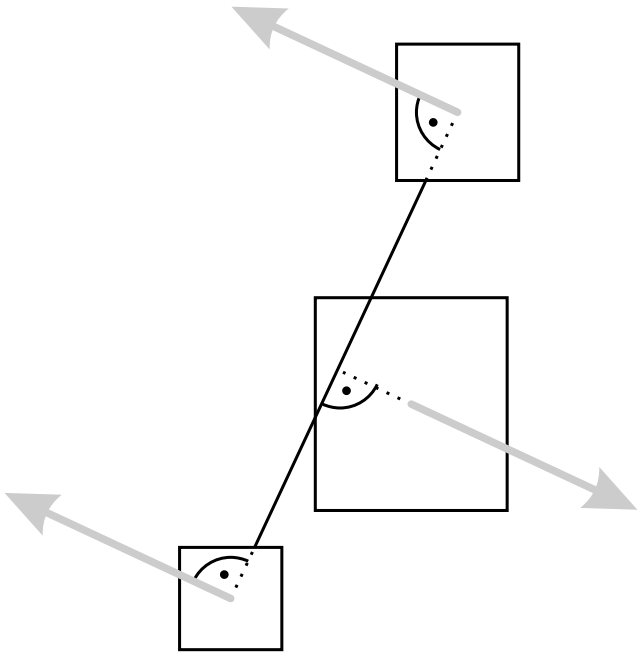


Abbildung 6.5: Kräfte zwischen Knoten und Kanten

Kräfte stattdessen den beiden Endknoten zugeordnet. Zu bemerken ist noch, daß bei diesen Überschneidungen sowohl die Ausmaße des Knotens als auch die Dicke der Kante zu berücksichtigen sind.

Der Nachteil an diesen Kräften ist, daß dadurch die Komplexität der Kräfteberechnung insgesamt steigt. Während ohne Kräfte zwischen Kanten und Knoten alle Berechnungen für eine Iteration in $\mathcal{O}(n^2)$ durchgeführt werden können, steigt jetzt die Komplexität auf $\mathcal{O}(n^2 + nm)$, was bei dichten Graphen $\mathcal{O}(n^3)$ entspricht – unter der Voraussetzung, daß man die im nächsten Abschnitt beschriebene Konstruktion für Mehrfachkanten verwendet.

6.3 Mehrfachkanten

Ein positiver Nebeneffekt der Behandlung von Knoten/Kanten-Schnitten ist die Berücksichtigung von Mehrfachkanten, die bisher nicht gesondert behandelt wurden. Wenn zwischen zwei Knoten mehrere Kanten vorhanden sind – die durchaus selbst verschiedene Breiten haben dürfen – so kann man sie einfach zu einer dicken

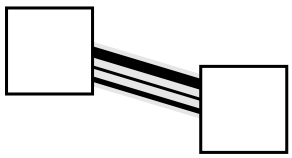


Abbildung 6.6: Modellierung von Mehrfachkanten

Kante zusammenfassen, wie in Abbildung 6.6 gezeigt. Der Rest ergibt sich durch die Schnittbehandlung. Am Schluß müssen dann nur noch die richtigen Koordinaten für die einzelnen Kanten festgelegt werden.

6.4 Entwirrung

Durch die Abmessungen der Knoten ist es oft nicht mehr möglich, daß sich Knoten übereinander oder über Kanten bewegen. Gerade am Anfang der Simulation ist dies aber essentiell, damit sich das anfänglich meist ziemlich chaotische Layout des Graphen entwirren kann. Dies wird verhindert, weil die Knoten und Kanten nur noch schwer Positionen tauschen können. Die Simulation endet viel leichter in einem schlechten lokalen Minimum.

Eine mögliche Lösung ist die Verwendung eines mehrphasigen Verfahrens. In einer ersten Phase wird der Graph gezeichnet, ohne auf die Knotengrößen Rücksicht zu nehmen. Damit sich möglichst wenig Überschneidungen ergeben, wird einerseits auf eine Flächenbeschränkung verzichtet und andererseits mit einem Vielfachen der optimalen Kantenlänge gerechnet.

In einem zweiten Schritt werden dann, wie oben beschrieben, die Knotengrößen und Kantendicken berücksichtigt und die richtigen optimalen Kantenlängen verwendet. Die Knoten ziehen sich dann näher zusammen, aber nur dort, wo es ihre Größe erlaubt. Die Einbettung des Graphen bleibt weitgehend bestehen.

Kapitel 7

Implementierung

Für die Implementierung des Verfahrens wurde ein möglichst allgemeiner Weg gewählt. Das Hauptziel war neben Geschwindigkeit und Qualität eine hohe Flexibilität. Es sollte möglich sein, für neue Entwicklungen einzelne Komponenten des Gesamtsystems auszutauschen, damit Veränderungen schnell vorgenommen werden können.

Der erste Schritt dazu war eine komplette Trennung des Verfahrens in den Simulationsalgorithmus und die Kräfteberechnungen. Auf diese Art und Weise ist es leicht möglich, neue Kräfte zu verwenden um neuen Anforderungen gerecht zu werden. Andererseits können alle Kräfte von Verbesserungen im Simulationsalgorithmus profitieren.

7.1 Simulationsalgorithmus

Für die Implementierung des Simulationsalgorithmus wurden in einigen Tests die verschiedenen vorhandenen Implementierungen gegenübergestellt und miteinander verglichen. Die Ergebnisse stimmten im wesentlichen mit den Resultaten der empirischen Tests in [B+95] überein, weshalb hier auf eine Wiedergabe detaillierter Testergebnisse verzichtet wird. Insgesamt ergab sich, daß das Verfahren von KAMADA und KAWAI und das GEM-Verfahren schneller sind als ihre Konkurrenten. Die Qualität des Layouts bei KAMADA und KAWAI übertrifft die des GEM-Verfahrens leicht, aber nicht signifikant, wobei des GEM-Verfahren schlechte lokale Minima besser vermeiden kann.

Es wurden die verschiedenen Aspekte der Verfahren detailliert betrachtet und alle Alternativen aus Kapitel 5 miteinander verglichen. Dabei wurde deutlich, daß das GEM-Verfahren mit seinen lokalen Temperaturen und verschiedenen Heuristiken zur Erkennung von Schwingungen und Rotationen eine höhere Flexibilität aufweist als die stark fixierte Optimierungsroutine von KAMADA und KAWAI.

Aus diesem Grund orientiert sich die Implementierung in den meisten Punkten am GEM-Verfahren. Sie wurde jedoch so modular aufgebaut, daß für weitergehende Entwicklungen mit geringem Aufwand einzelne Komponenten des Systems gegen andere ausgetauscht werden können. So war es ein leichtes, die Verbesserung aus Abschnitt 5.1.8 einzubauen, die es erlaubt, anstatt ganzer Graphen nur Teile davon neu zu zeichnen.

7.2 Kräfte

Bei der Auswahl der Kräfte wurde großer Wert darauf gelegt, daß sie schnell berechnet werden können. Die Geschwindigkeit der Kräfteberechnungen ist für Springembedder-Verfahren extrem wichtig, da diese in der innersten Schleife des Verfahrens durchgeführt werden. Um so bedeutsamer ist es, auf teure Rechenoperationen wie Logarithmus- oder Quadratwurzelberechnung zu verzichten.

Die verschiedenen Varianten für die Grundkräfte zur Regelung der optimalen Kantenlängen wurden bereits in Abschnitt 5.2.2 ausführlich verglichen. Als Ergebnis blieben für eine Wahl die fast identischen Kräfte von FRUCHTERMAN und REINGOLD für $e = 3$ und diejenigen des GEM-Verfahrens, bei denen Qualität und Laufzeit kaum Unterschiede aufweisen. Schließlich wurden die Kräfte des GEM-Verfahrens gewählt. Weiterhin wurden auch dessen Gravitations- und Zufallskraft übernommen, allerdings durch einen Parameter abschaltbar gemacht.

Schließlich enthält das Verfahren noch die bereits detailliert erläuterten Kräfte zur Behandlung von Constraints und Flächenbeschränkungen und die in Kapitel 6 entwickelten Kräfte zur Unterstützung echter Knotengrößen. Diese letzten Kräfte sind leider nicht so effizient zu berechnen wie die anderen. Einerseits steigt durch sie die Gesamtkomplexität der Kräfteberechnung auf $\mathcal{O}(n \cdot m)$. Andererseits wird

zur Berechnung der Abstände die Quadratwurzelberechnung benötigt. Daher sind auch diese Kräfte abschaltbar, damit sie nur verwendet werden, wenn es die zu zeichnenden Graphen erfordern.

7.3 Animation

Wie schon einige andere Algorithmen im Graphlet-System unterstützt diese Implementierung die Animation des Simulationsablaufes. Diese Funktionalität ist vom theoretischen Standpunkt weniger wichtig, für das Testen neuer Kräfte und zur Wahl der richtigen Parameter ist sie allerdings ein wertvolles Hilfsmittel. Mit ihrer Hilfe war es oft nicht nur möglich, festzustellen, daß eine bestimmte Parameterwahl ungünstig war, sondern auch, warum.

Kapitel 8

Ausblick

Das große Problem, das Springembedder-Verfahren mit sich bringen, ist die große, fast unüberschaubare Anzahl an Parametern. Um diese Situation zu verbessern, ist es notwendig, Verfahren zu entwickeln, die die Parameter des Verfahrens anhand bestimmter Vorgaben des Benutzers automatisch wählen.

Ansätze dieser Art sind in den meisten Verfahren bereits vorhanden, die etwa die optimale Kantenlänge aus der gegebenen Zeichenfläche berechnen oder, wie das Verfahren von KAMADA und KAWAI, den optimalen Abstand zweier Knoten anhand des graphentheoretischen Abstands von Knoten bestimmen.

Notwendig sind vor allem Verfahren, die in Abhängigkeit von den gegebenen Graphen bestimmen, in welchem Verhältnis die verschiedenen verwendeten Kräfte zueinander stehen müssen, damit die Ästhetikkriterien möglichst gut erfüllt werden. So sind z. B. in großen Graphen stärkere Kräfte notwendig, um Constraints zu erfüllen, als in kleinen Graphen. In vielen Fällen ist nicht von vornherein klar, wie diese Eigenschaften genau zusammenhängen. Deshalb ist es schwierig, die Parameter richtig zu wählen, um die Qualität und die Geschwindigkeit des Verfahrens zu optimieren.

Eine mögliche Lösung des Problems könnte es sein, die Parameter des Verfahrens variabel zu machen und anhand einer Kontrolle des aktuellen Simulationsverlaufs zu verändern, so daß sie sich an die aktuelle Situation anpassen. Ein Beispiel dafür ist die Verwendung von lokalen Temperaturen, die sich anhand des aktuellen Simulationsverlaufs selbst anpassen. Ähnliche Strategien könnten auch für andere Parameter möglich sein.

Anhang A

Verwendung des Algorithmus

Im folgenden wird im Detail beschrieben, wie der Algorithmus im Graphlet-System verwendet wird. Zum Verständnis aller Ausführungen sind Grundkenntnisse in Graphscript (siehe [Him99d]) notwendig.

A.1 Aufrufsyntax

Der Aufruf des Algorithmus orientiert sich an der Standardsyntax für Layoutalgorithmen:

```
layout_use editor ?-parameter wert ...?
```

Der Name des Algorithmus-Kommandos lautet `layout_use`, in Anlehnung an „Universal Spring Embedder“. Es folgt der Editor, in dem das Layout durchgeführt werden soll, und eine beliebige Anzahl von Parameterangaben. Jede Parameterangabe besteht aus dem Parameternamen, gefolgt von einem Parameterwert. Ein typischer Aufruf könnte so aussehen:

```
layout_use $editor                    \
        -nodes         [$graph nodes] \
        -respect_sizes true           \
        -animation     true
```

Dieser Befehl zeichnet den gesamten Graphen unter Berücksichtigung von Knotengrößen neu. Der Simulationsablauf wird animiert dargestellt. Für alle anderen Parameter gelten die Standardwerte.

A.2 Typen der Argumente

Die Parameterwerte des Algorithmus verwenden verschiedene Typen. Die meisten Werte sind von einem der folgenden Graphscript Basistypen:

Typ	Beschreibung	Beispiele
float	Rationale Zahl	`2.1`, `6e4`, `7.91e+16`
integer	Ganze Zahl	`0`, `17`, `-42`
boolean	Wahrheitswert	`false`, `true`
string	Zeichenkette	`"Zeichenkette"`

Weiterhin werden zwei zusammengesetzte Attributtypen verwendet:

`list` Dieser Typ wird verwendet, um geordnete Mengen von Attributwerten zu übergeben. Mit folgender Syntax wird etwa die Menge der zu betrachtenden Knoten übergeben:

```
layout_use $editor -nodes { $n1 $n2 $n3 }
```

`array` Dies wird benötigt, um Informationen zu übergeben, die spezifisch für einzelne Knoten oder Kanten sind. Man kann etwa folgendes verwenden, um alle Kanten nach rechts zu richten:

```
set direction 0

foreach edge [$graph edges] {
    set edge_direction($edge) $direction
}

layout_use $editor -edge_directions edge_direction
```

A.3 Simulationsparameter

Im folgenden werden die einzelnen Parameter des Verfahrens beschrieben. Hinter jedem Parameter sind in Klammern sein Typ und – falls vorhanden – ein Standardwert angegeben.

A.3.1 Initialisierung

Folgende Parameter bestimmen den Startzustand der Simulation.

`-start_placement` (string, random)

Strategie zur Berechnung der Anfangskoordinaten (siehe Abschnitt 5.1.2). Es sind folgende Werte erlaubt:

Wert	Bedeutung
`none`	Aktuelle Koordinaten
`random`	Gleichverteilte Zufallskoordinaten
`topological`	Topologische Plazierung

`-start_temp` (float, 1.0)

Temperatur, auf die alle Knoten initialisiert werden.

A.3.2 Terminierung

Folgende Parameter bestimmen, nach welchen Kriterien die Simulation beendet wird. Es ist ausreichend, wenn eines der Kriterien erfüllt ist. Bei einer Angabe von 0 wird das jeweilige Kriterium ignoriert. Zusätzlich ist es dem Benutzer jederzeit möglich, durch Drücken einer Schaltfläche die Simulation zu beenden.

`-stop_temp` (float, 0.02)

Abbruch, sobald die Gesamttemperatur unter diese Schranke fällt (siehe Abschnitt 5.1.7).

`-stop_iter` (integer, 0)

Abbruch nach dieser Anzahl von Knotenbewegungen.

`-stop_time` (integer, 0)

Abbruch nach dieser Laufzeit (in Sekunden).

A.3.3 Temperatursteuerung

Folgende Parameter bestimmen, nach welchen Kriterien die Temperatur der Knoten angepaßt wird (siehe Abschnitt 5.1.5).

`-acceleration_detection` (float, 0.4)

Stärke der Beschleunigungserkennung. Mit diesem Faktor wird die Temperaturänderung in Beschleunigungssituationen multipliziert.

`-oscillation_detection` (float, 0.4)

Stärke der Schwingungserkennung. Mit diesem Faktor wird die Temperaturänderung in Schwingungssituationen multipliziert.

`-rotation_detection` (float, 0.9)

Stärke der Rotationserkennung. Mit diesem Faktor wird die Temperaturänderung in Rotationssituationen multipliziert.

`-min_temp` (float, 0.0001)

Minimaltemperatur. Dieser Wert wird nie unterschritten. Damit wird verhindert, daß ein Knoten durch numerische Ungenauigkeiten die Temperatur `0` erhält. Dies würde jede weitere Bewegung verhindern.

`-max_temp` (float, 3.0)

Maximaltemperatur. Dieser Wert wird nie überschritten.

A.3.4 Untergraphen

`-nodes` (node list)

Bestimmt die Knoten des Untergraphens, auf den die Simulation angewendet werden soll. Falls nichts angegeben wird, wird der gesamte Graph neu gezeichnet.

A.3.5 Animation

`-animation` (boolean, false)

Gibt an, ob die Simulation animiert dargestellt werden soll.

`-animation_start` (boolean, true)

Gibt an, ob die Animation direkt gestartet werden soll oder erst auf Knopfdruck des Anwenders. Wird nur verwendet, wenn `-animation true` angegeben wurde.

`-animation_speed` (integer, 50)

Anzahl der Knotenbewegungen zwischen zwei Animationsbildern. Wird nur verwendet, wenn `-animation true` angegeben wurde.

A.4 Kräfteparameter

Es folgen die Parameter zur Steuerung der verschiedenen Kräfte. Die einzelnen Stärken haben keine absolute Bedeutung sondern sind nur in Relation zueinander gesehen sinnvoll.

A.4.1 Kantenlängen

Die folgenden Parameter regeln die gewünschten Kantenlängen und die Abstoßung unverbundener Knoten (siehe Abschnitt 5.2.2). Falls weder `-edge_length` noch `-edge_lengths`, dafür aber eine Zeichenfläche angegeben wurde, so wird automatisch eine passende Kantenlänge berechnet.

`-base_force` (float, 1.0)

Stärke der anziehenden und abstoßenden Kräfte.

`-edge_lengths` (float edge array)

Optimale Längen der einzelnen Kanten.

`-default_edge_length` (float, 32)

Optimale Länge der Kanten, für die mit `-edge_lengths` keine Länge angegeben wurde.

A.4.2 Kantenrichtungen

`-direction_force` (float, 1.0)

Stärke der Richtungskräfte.

`-edge_directions` (float edge array)

Wunschrichtung für die einzelnen Kanten im Bogenmaß.

`-default_edge_direction` (float)

Wunschrichtung für alle Kanten, für die mit `-edge_directions` keine Richtung vorgegeben wurde.

A.4.3 Flächenbeschränkung

`-respect_frame` (boolean, false)

Gibt an, ob garantiert werden soll, daß die Knoten innerhalb der Zeichenfläche bleiben.

`-frame_force` (float, 0.0)

Stärke der flächenbeschränkenden Kraft.

`-min_x` (float, 0.0)

Linker Rand der Zeichenfläche.

`-min_y` (float, 0.0)

Oberer Rand der Zeichenfläche.

`-max_x` (float, $+\infty$)

Rechter Rand der Zeichenfläche.

`-max_y` (float, $+\infty$)

Unterer Rand der Zeichenfläche.

A.4.4 Berücksichtigung von Größen

`-respect_sizes` (boolean, false)

Gibt an, ob die Größen von Knoten und Kanten berücksichtigt werden sollen.

`-node_dist` (float, 32)

Mindestabstand zwischen zwei Knoten oder Knoten und Kanten. Wird nur verwendet, wenn `respect_sizes` auf `true` gesetzt ist.

`-node_node_force` (float, 1.0)

Stärke der abstoßenden Kraft bei Knotenüberlappungen.

`-node_edge_force` (float, 1.0)

Stärke der abstoßenden Kraft bei Knoten/Kanten-Schnitten.

`-multi_edge_dist` (float, 3.0)

Abstand von Mehrfachkanten.

A.4.5 Sonstige Kräfte

`-random_force` (float, 1.0)

Stärke der Zufallskomponente.

`-gravity_force` (float, 1.0)

Stärke der Gravitationskraft.

`-respect_mass` (boolean, true)

Gibt an, ob den Knoten verschiedene Massen zugewiesen werden sollen.

Abbildungsverzeichnis

Algorithmusverzeichnis

Literaturverzeichnis

[B+95] BRANDENBURG, FRANZ JOSEF et al.: *An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms*. In: *GD95* [GD95], Seiten 76–87.

[Bac95] BACHL, WALTER: *Entwicklung und Implementierung eines hierarchischen Spring Embedders*. Diplomarbeit, Universität Passau, Fakultät für Mathematik und Informatik, 1995.

[Bra98] BRANDENBURG, FRANZ JOSEF: *Skript zur Vorlesung Layout von Graphen*. Universität Passau, Wintersemester 1997/98.

[Cha91] CHAR, BRUCE W. (Herausgeber): *Maple V library reference manual*. Springer, 1991.

[dB+99] BATTISTA, GIUSEPPE DI et al.: *Graph drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[DH91] DAVIDSON, RON und DAVID HAREL: *Drawing Graphs Nicely Using Simulated Annealing*. Technischer Bericht, The Weizmann Institute of Science, Rehovot, Israel, Dept. of Applied Mathematics and Computer Science, 1991.

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

[F+94] FRICK, ARNE et al.: *A Fast Adaptive Layout Algorithm for Undirected Graphs*. In: *GD94* [GD94], Seiten 388–403.

[FPR99] FORSTER, MICHAEL, ANDREAS PICK und MARCUS RAITNER: *GTL – Graph Template Library, a STL based Graph Library*. Universität Passau, 1999.

[FR91] FRUCHTERMAN, THOMAS M. J. und EDWARD M. REINGOLD: *Graph Drawing by Force-directed Placement*. Software – Practice and Experience, 21(11):1129–1164, 1991.

[Fri97] FRICK, ARNE: *Visualisierung von Programmabläufen*. Doktorarbeit, Universität Karlsruhe, Fakultät für Informatik, 1997.

[GD94] *Proc. Workshop on Graph Drawing '94*, Band 894 der Reihe *Lecture Notes in Computer Science*. Springer, 1995.

[GD95] *Proc. Workshop on Graph Drawing '95*, Band 1027 der Reihe *Lecture Notes in Computer Science*. Springer, 1996.

[GD96] *Proc. Workshop on Graph Drawing '96*, Band 1190 der Reihe *Lecture Notes in Computer Science*. Springer, 1997.

[GD97] *Proc. Workshop on Graph Drawing '97*, Band 1353 der Reihe *Lecture Notes in Computer Science*. Springer, 1997.

[GD98] *Proc. Workshop on Graph Drawing '98*, Band 1547 der Reihe *Lecture Notes in Computer Science*. Springer, 1998.

[GJ88] GAREY, MICHAEL R. und DAVID S. JOHNSON: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1988.

[Him99a] HIMSOLT, MICHAEL: *Coding Standards for Graphlet*. Universität Passau, 1999. `http://www.fmi.uni-passau.de/Graphlet/codingstandards/`.

[Him99b] HIMSOLT, MICHAEL: *The GML File Format*. Universität Passau, 1999. `http://www.fmi.uni-passau.de/Graphlet/GML/`.

[Him99c] HIMSOLT, MICHAEL: *Graphlet C++ Programmer Manual*. Universität Passau, 1999. `http://www.fmi.uni-passau.de/Graphlet/cppmanual/`.

[Him99d] HIMSOLT, MICHAEL: *The Graphscript Language*. Universität Passau, 1999. `http://www.fmi.uni-passau.de/Graphlet/graphscript/`.

[Kam89] KAMADA, TOMIHISA: *Visualizing Abstract Objects and Relations – A Constraint-Based Approach*, Band 5 der Reihe *Series in Computer Science*. World Scientific, 1989.

[KK89] KAMADA, TOMIHISA und SATORU KAWAI: *An Algorithm for Drawing General Undirected Graphs*. Information Processing Letters, 31:7–15, 1989.

[RT81] REINGOLD, EDWARD M. und J. S. TILFORD: *Tidier Drawings of Trees*. IEEE Trans. Software Engineering, 7(2):223–228, 1981.

[San96] SANDER, GEORG: *Visualisierungstechniken für den Compilerbau*. Doktorarbeit, Universität des Saarlandes, Technische Fakultät, 1996.

[Sch96] SCHIRMER, ROBERT: *Eine Erweiterung des Algorithmus von Fruchterman und Reingold zum Zeichnen von Graphen um die Berücksichtigung von Constraints*. Diplomarbeit, Universität Passau, Fakultät für Mathematik und Informatik, 1996.

[Sch97] SCHIRMER, ROBERT: *Additional Information on CFR & ICSE*. Nicht veröffentlicht, 1997.

[SM94] SUGIYAMA, KOZA und KAZUO MISUE: *A Simple and Unified Method for Drawing Graphs: Magnetig-Spring Algorithm*. In: *GD94* [GD94], Seiten 364–375.

[SR82] SUPOWIT, KENNETH J. und EDWARD M. REINGOLD: *The Complexity of Drawing Trees Nicely*. Acta Informatica, 18:377–392, 1982.

[Wat88] WATANABE, H.: *Heuristic Graph Displayer for G-Base*. Technischer Bericht TR-17, Ricoh Software Research Center, 1988.

Eidesstattliche Erklärung

Diese Diplomarbeit wurde selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet. Die Diplomarbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Passau, den 27. September 1999

MICHAEL FORSTER