technische universität
dortmund

Zerlegungsstrategien für Multilevel-Graphdrawing-Verfahren

Gereon Bartel

Algorithm Engineering Report
TR09-1-013
Dez. 2009
ISSN 1864-4503

Fakultät für Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

technische universität
dortmund

Diplomarbeit

Zerlegungsstrategien für Multilevel-Graphdrawing-Verfahren

Gereon Bartel

Februar 2009

Gutachter:
Prof. Dr. Petra Mutzel
Dipl.-Inform. Karsten Klein

Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
http://ls11-www.cs.uni-dortmund.de

Danksagung

Ich danke Prof. Petra Mutzel für ihre Betreuung und Hilfe bei der Literaturrecherche und allen anfallenden Fragen. Besonders danke ich auch Dipl.-Inform. Karsten Klein für die gute Betreuung, anregenden Diskussionen, Ideen und Lösungsvorschläge. Ich möchte außerdem Dipl.-Inform. Carsten Gutwenger für hilfreiche Gespräche und Ideen danken. Martin Gronemann gebührt Dank für die sehr erhellenden Gespräche und Diskussionen, die guten Ideen und seine Diplomarbeit. Er hat einige der hier präsentierten Ergebnisse erst möglich gemacht. Besonderer Dank geht auch an Alena Lohne, Thorsten Bartel, Thorsten Flügel und Anne Bartel die bei der Verbesserung von Rechtschreibung und Verständnis eine große Hilfe waren und alle lieben Menschen, die mir durch Kleinigkeiten direkt und indirekt beim Schreiben dieser Arbeit geholfen haben.

Zusammenfassung

In dieser Diplomarbeit werden verschiedene bekannte und neue Zerlegungsstrategien für Multilevel-Verfahren zum automatischen Zeichnen von Graphen verglichen. Der im Laufe der Arbeit entstandene *Modular Multilevel Mixer* ermöglicht eine sehr flexible Anpassung des Multilevel-Ansatzes an konkrete Anforderungen. Einige der im Laufe dieser Arbeit entstandenen Varianten des Verfahrens stellen in vieler Hinsicht eine Verbesserung gegenüber den bisherigen Verfahren dar und können als eigenständige Zeichenverfahren empfohlen werden. Außer den verfügbaren Zerlegungsstrategien werden auch die Platzierungsstrategien und kräftebasierten Verfahren in Hinblick auf Qualität und Laufzeit miteinander verglichen. Auch die verschiedenen möglichen Kombinationen dieser Module werden untersucht und bewertet. In manchen Multilevel-Verfahren werden auch weitere Schritte wie ein Aufteilen des Graphen in seine Zusammenhangskomponenten oder ein Post-Processing genutzt. Diese Verbesserungen wurden verallgemeinert und auf alle Verfahren angewandt, was die Vergleichbarkeit erhöht. Die verglichenen Verfahren entsprechen dadurch jedoch nicht genau den ursprünglichen von den jeweiligen Autoren entworfenen Verfahren. Basierend auf den besonderen Schwächen und Stärken der bekannten Verfahren werden neue Zerlegungsstrategien entwickelt. Das häufig auftretende Problem von Verdrehungen in der Zeichnung durch verlorenen Zwei-Zusammenhang in gröberen Verfeinerungsstufen wird deutlich verringert.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

(Bochum, den 10. Februar 2009, Gereon Bartel)

Inhaltsverzeichnis

Kapitel 1

Grundlagen

1.1 Aufbau der Arbeit

Zunächst werden die Grundlagen des automatischen Zeichnens von Graphen und der kräftebasierten Zeichenverfahren erläutert. Nachdem das Prinzip der Multilevel-Verfahren eingeführt wurde, werden existierende Multilevel-Verfahren vorgestellt. Der Aufbau des im Rahmen dieser Arbeit entstandenen *Modular Multilevel Mixer* wird beschrieben. Die verfügbaren Kräfteverfahren werden erklärt und die implementierten Zerlegungsstrategien und Platzierungsstrategien genauer beschrieben. Die experimentellen Ergebnisse werden detailliert betrachtet und in Hinblick auf Qualität der Zeichnungen und die Laufzeit der Algorithmen analysiert. Zum abschließenden Fazit werden auch mögliche Ansätze für zukünftige Verbesserungen dargelegt.

1.2 Automatisches Zeichnen von Graphen

Einen Überblick über große Datensätze unterschiedlichster Art zu gewinnen, ist eine häufige und nicht einfache Aufgabe in vielen Forschungs- und Wirtschaftszweigen. Für den menschlichen Betrachter[1] ist meist eine Zeichnung, also eine Visualisierung der Daten auf einer zweidimensionalen Zeichenfläche oder eine Repräsentation als dreidimensionales Gebilde am eingängigsten und kann folglich am besten erfasst werden. Sobald die Datensätze eine gewisse Größe überschreiten, ist oft nicht mehr intuitiv einzusehen, wie eine gute Visualisierung des gesamten Datensatzes aussehen sollte. Ebenfalls ist es nicht mehr möglich, eine solche Visualisierung manuell zu erzeugen. Automatische Verfahren stellen hier den einzig gangbaren Weg dar. Viele Datensätze sind als Graphen darstellbar, so dass es Sinn macht sich damit zu beschäftigen Graphen automatisch zu zeichnen.

[1] Aus Gründen der Lesbarkeit soll im Nachfolgenden auf den Zusatz der weiblichen Form verzichtet werden und die männliche Form für beide Geschlechter gleichermaßen gelten.

1.2.1 Graphen

Ein Graph besteht aus Knoten, die jeweils einzelne Daten repräsentieren. Kanten verbinden jene Knoten, deren Daten in einer für die jeweilige Anwendung wichtigen Beziehung stehen. Insbesondere für Relationen ist ein Graph also eine angemessene Darstellung, die sich oft anbietet. Es gibt verschiedene Sorten von Graphen, abhängig von den Beziehungen, die die Knoten untereinander haben können. Insbesondere können die Kanten gerichtet oder ungerichtet sein.

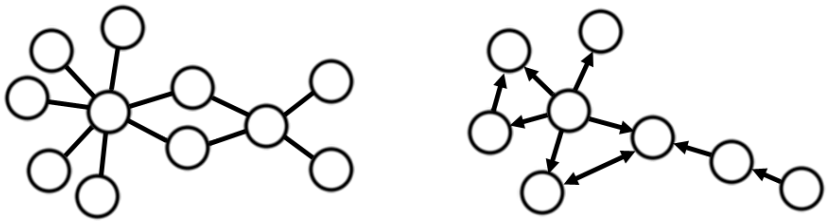

Abbildung 1.1: Ein ungerichteter und ein gerichteter Graph.

Definition 1 (Graph). Das Paar $G = (V, E)$ ist ein Graph, wenn V eine Menge von Knoten ist, und E eine Menge von Kanten $e = (v, w) \in V \times V$ ist.

Sollte bei einer Kante $e = (v, w)$ gelten, dass $v = w$ ist, so wird die Kante eine Schleife genannt. Gibt es mehrere Kanten zwischen den gleichen Knoten v und w, ist von mehrfachen Kanten die Rede.

Es gibt viele Anwendungsbereiche, in denen Graphen eine große Rolle spielen. Ein paar Beispiele sind:

- *Soziale Netzwerke:* Zehntausende Akteure, die in unterschiedlichsten Beziehungen zueinander stehen, können als Graph dargestellt werden.
- *Softwareentwicklung:* In der Planung und Entwicklung von Software werden die Beziehungen zwischen Klassen in UML-Diagrammen dargestellt.
- *Biologie:* Hier fallen enorme Mengen von experimentellen und abgeleiteten Daten an. Graphen für Signal-Transduktion, Protein-Interaktion und metabolische Netzwerke können mehrere tausend Knoten enthalten und eventuell sehr dicht sein.
- *Visualisierung von Computernetzen:* Bei der Darstellung eines Computernetzes als Graph sollen meist die Übertragungskapazitäten verdeutlicht werden. Die tatsächlichen Standorte der Rechner sind für die Positionen der Knoten im Graphen von untergeordneter Bedeutung.
- *Datenbank Design:* Die Relationen zwischen Datenbanktabellen können als Graph dargestellt werden, um die Planung zu erleichtern.

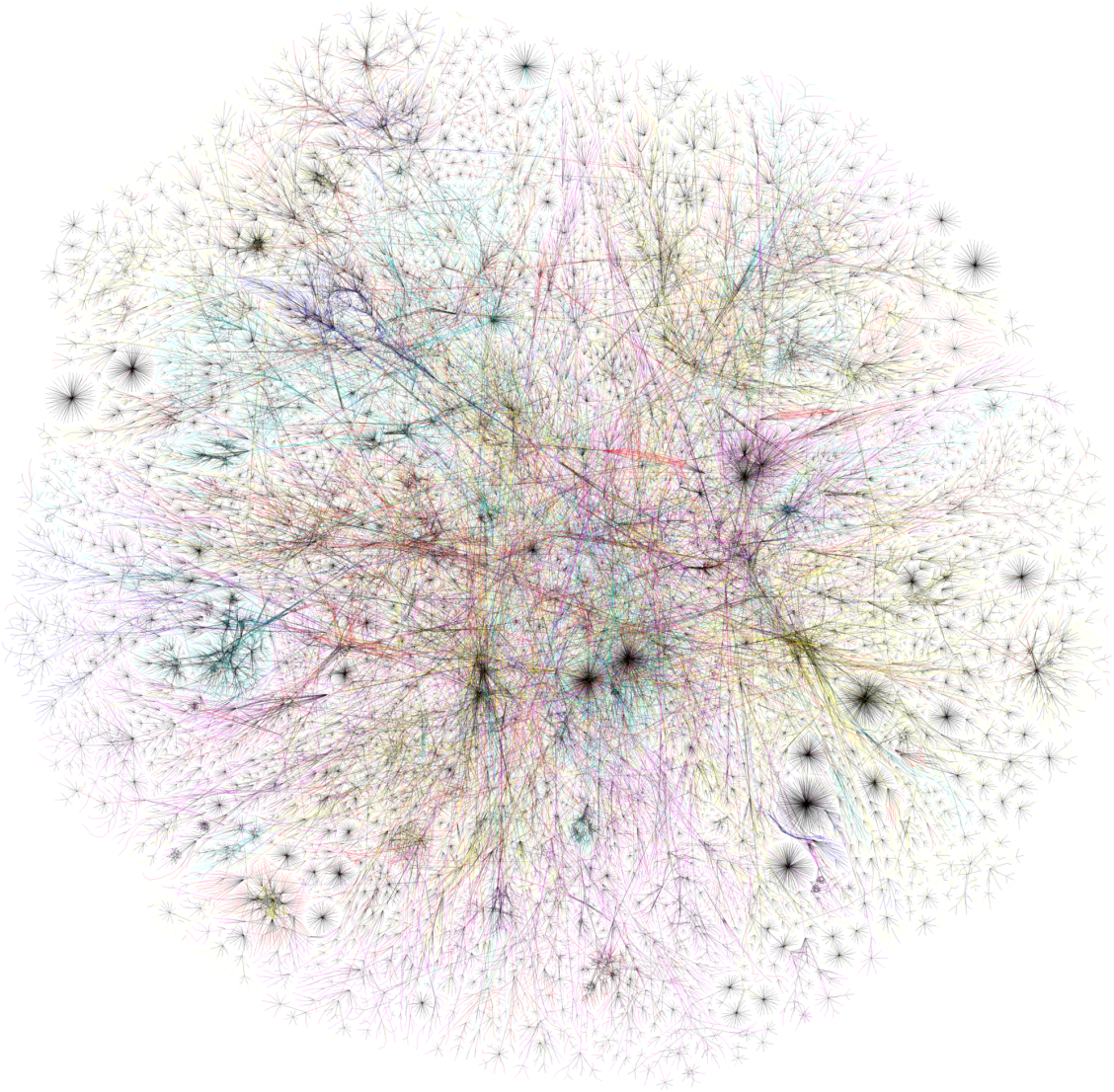

Abbildung 1.2: Graph des Internet im Jahr 2003, ca. 5 Millionen Kanten. [OPTE]

Es gibt viele weitere Anwendungsfelder, bei denen jedoch auch unterschiedliche Anforderungen an die Zeichnungen bestehen. Einige Beispiele für sehr unterschiedliche Möglichkeiten, einen Graphen zu zeichnen, sind im nächsten Abschnitt aufgeführt.

1.2.2 Aufgaben beim automatischen Zeichnen von Graphen

Eine Zeichnung eines Graphen kann sehr unübersichtlich werden, insbesondere wenn der Graph sehr groß ist. Solche Graphen können nicht mehr manuell durch Menschen, sondern müssen automatisch gezeichnet werden. Bei extremen Beispielen wie dem Graph des Internets, in dem viele Millionen Knoten vorhanden sind, wird klar, dass auch die Laufzeit des Zeichenverfahrens wichtig wird.

Für die Zeichnung eines Graphen müssen insbesondere die Positionen der Knoten festgelegt werden. Viele Verfahren erlauben zudem Kantenknicke, deren Positionen ebenfalls festgelegt werden müssen. Im weiteren Verlauf dieser Arbeit ist eine Zeichnung, auch Layout genannt, vollständig durch die Positionen der Knoten gegeben.

Definition 2 (Layout eines Graphen). Das Layout eines Graphen $G = (V, E)$ ist gegeben durch jeweils ein Koordinatenpaar (x, y) für alle Knoten $v \in V$.

Beim Zeichnen von Graphen gibt es verschiedene Ziele, die sich teilweise widersprechen. Bei den verschiedenen Zeichenverfahren wird unterschiedliches Gewicht auf das Erreichen der jeweiligen Ziele gelegt. Die wichtigsten Kriterien sind:

- *Kantenkreuzungen:* Es ist für den Betrachter schwer einer Kante zu folgen, wenn diese durch viele andere gekreuzt wird, insbesondere wenn auch Kantenknicke möglich sind.
- *Knotenüberlappungen:* Wenn in den Knoten Informationen stehen, die für den Benutzer wichtig sind, sind Knotenüberlappungen besonders ärgerlich. Auch die Struktur des Graphen ist so weniger leicht zu erkennen.
- *Schnitte von Knoten und Kanten:* Kanten, die einen Knoten schneiden ohne dort zu enden, sind schwer zu verfolgen und können zudem den falschen Eindruck erwecken, mit diesem Knoten verbunden zu sein.
- *Ähnliche Kantenlängen:* Graphen, die in etwa gleich lange und möglichst kurze Kanten enthalten, sind meist übersichtlicher.
- *Zeichenfläche:* Der Graph sollte auf einer möglichst kleinen Fläche gezeichnet werden. Dadurch kommen meist auch Details besser zur Geltung.
- *Angular Resolution:* Je größer die Winkel zwischen den Kanten sind, desto leichter lässt sich auf einen Blick erkennen, zu welchen Knoten sie führen.
- *Symmetrien:* Wenn ein Graph symmetrisch ist, sollte die Zeichnung dies auch betonen. Ebenso sollten Teile des Graphen, die die gleiche Struktur haben, auch möglichst gleich gezeichnet werden.
- *Knotenverteilung:* Gleichmäßig ohne unnötige Häufungen auf der Zeichenfläche verteilte Knoten machen einen geordneteren Eindruck.

Zusätzlich zu diesen allgemeinen Zielen gibt es noch einige, die für die jeweilige Anwendung wichtig sein können. Durch gerichtete Kanten kann eine Hierarchie symbolisiert werden, die in der Zeichnung durch entsprechende Positionierung der Knoten verdeutlicht werden sollte. In manchen Verfahren sind Kantenknicke erlaubt, um andere Bedingungen

wie zum Beispiel senkrechte und waagerechte Kanten zu erfüllen. Es ist dann natürlich wünschenswert, mit möglichst wenig Kantenknicken auszukommen. Zum besseren Verständnis sind meist die Knoten oder die Kanten mit Text versehen, der die jeweilige Bedeutung klärt. Diese zusätzlichen Elemente werden in dieser Arbeit nicht näher betrachtet.

1.2.3 Zeichenverfahren

Es gibt die unterschiedlichsten Zeichenverfahren, um Graphen mit bestimmten Eigenschaften zu zeichnen. Dabei wird auf unterschiedliche Qualitätskriterien Wert gelegt. Einige Zeichenverfahren sind nur für sogenannte planare Graphen geeignet, die sich ganz ohne Kantenkreuzungen zeichnen lassen. Um tatsächlich eine ansprechende Zeichnung ohne Kreuzungen zu erreichen werden oft Kantenknicke erlaubt. Eine besondere Art dieser Verfahren sind so genannte Orthogonale Zeichenverfahren, die zwar Kantenknicke erlauben, dafür jedoch nur senkrechte und waagerechte Kantensegmente.

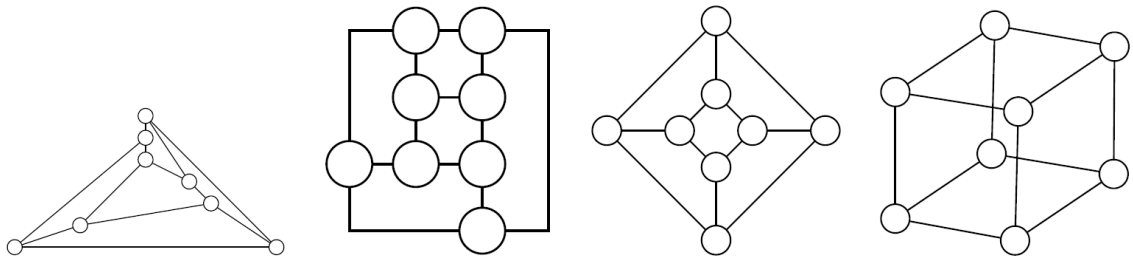

Abbildung 1.3: Unterschiedliche Zeichnungen des gleichen Graphen. (a) Eine planare Zeichnung. (b) Eine orthogonale Zeichnung. (c) Eine Zeichnung die die Symmetrie gut darstellt. (d) Eine Zeichnung mit gleich langen Kanten, die die Struktur des Graphen gut darstellt. [Hac05]

Für gerichtete Graphen kann eine hierarchische Zeichnung interessant sein, bei der Kanten nur von oben nach unten verlaufen. Das Verfahren von Sugiyama ist hier das am weitesten verbreitete [STT81].

Häufig wird jedoch auch verlangt, dass keine Kantenknicke auftreten. Solche Straight-Line-Verfahren müssen meist zusätzliche Kreuzungen in Kauf nehmen. Ein Beispiel für Straight-Line-Zeichenverfahren sind die in dieser Arbeit betrachteten Kräfteverfahren.

Graphen können auch dreidimensional dargestellt werden, wodurch viele der Probleme einer zweidimensionalen Zeichnung, wie Kreuzungen und Kantenknicke, umgangen werden können. Um sich in einer dreidimensionalen Zeichnung einen Überblick zu verschaffen, ist es meist nötig, das Gebilde drehen zu können. Eine einfache Projektion auf eine Fläche kann dennoch akzeptable Ergebnisse liefern.

Eine hier nicht weiter betrachtete Möglichkeit, große Graphen zu zeichnen, ist ein algebraischer Ansatz, bei dem der Graph in einem mehrdimensionalen Raum eingebettet

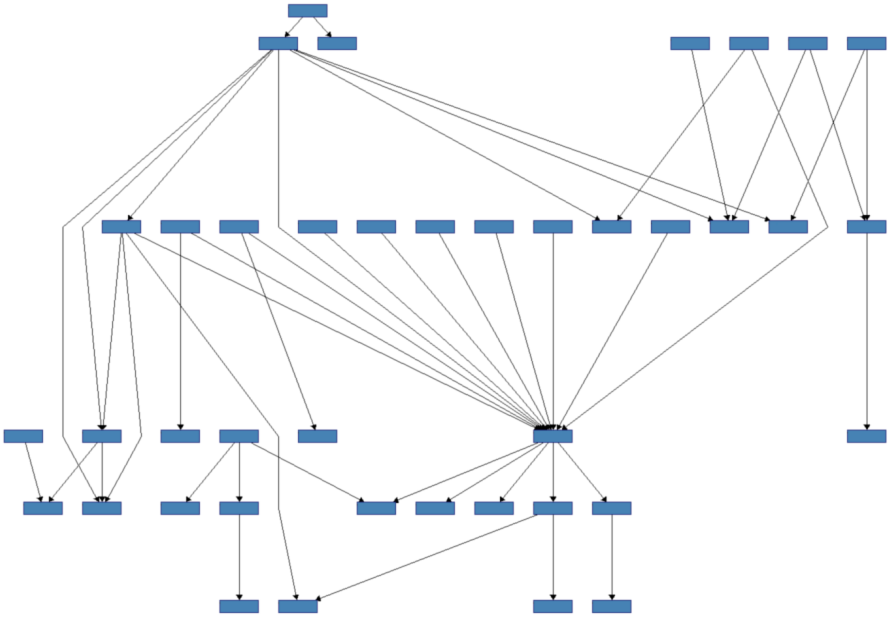

Abbildung 1.4: Hierarchische Zeichnung eines Graphen. Kanten führen nur zu Knoten die unterhalb des Startknotens liegen.

und danach auf die Ebene projiziert wird. Harel und Koren haben bei ihrem Ansatz 50 Dimensionen verwendet [HaKo02b].

Für die jeweilige Anwendung kann es vorteilhaft sein, ein Verfahren zu implementieren, das auf die Anforderungen zugeschnitten ist. UML-Diagramme sollen nach den üblichen Kriterien orthogonal gezeichnet werden. Die Hierarchie der Klassen soll dabei möglichst durch entsprechende Positionierung der Knoten verdeutlicht werden. Für andere Graphen sollen die Knoten alle auf einem oder mehreren Kreisen liegen, während die Kantenkreuzungen minimiert werden. Die Möglichkeiten sind hier unbegrenzt. Im Rahmen dieser Arbeit werden nur allgemeine Zeichenverfahren betrachtet, für die keinerlei Annahmen zur Struktur des Graphen oder der gewünschten Anwendung der resultierenden Zeichnung getroffen werden. Allgemeine Zeichenverfahren kommen oft zum Einsatz, wenn die Ressourcen für eine eigene Implementierung nicht vorhanden oder noch unklar ist, welche Kriterien für die Anwendung besonders wichtig sind. Zudem haben die hier betrachteten Multilevel-Verfahren eine hohe Qualität erreicht, so dass oft kein spezialisiertes Verfahren mehr nötig ist.

1.3 Kräftebasierte Verfahren

Eine Möglichkeit, eine Straight-Line-Zeichnung eines beliebigen Graphen zu erhalten, bieten die so genannten kräftebasierten Zeichenverfahren. Falls der Graph gerichtete Kanten enthält, werden diese wie ungerichtete Kanten behandelt. Der Name *kräftebasiertes Ver-*

fahren entstand aus der physikalischen Metapher, die derartigen Verfahren zugrunde liegt. Die Knoten werden dabei als elektrisch geladene Kugeln interpretiert, während die Kanten sich wie Federn verhalten. Alle Kugeln stoßen einander ab, während die Federn die Kugeln, mit denen sie verbunden sind, zueinander ziehen. Dieses System wird simuliert, bis sich ein Kräftegleichgewicht eingestellt hat. Die Federn sollen dafür sorgen, dass im Graphen nah beieinander liegende Knoten auch in der Zeichnung einen geringen Abstand haben. Die abstoßenden Kräfte der geladenen Kugeln sorgen dafür, dass der Graph sich möglichst entfaltet um graphentheoretisch weit entfernte Knoten auch in der Zeichnung auseinander zu bewegen.

Definition 3 (Graphentheoretischer Abstand). Der graphentheoretische Abstand zweier Knoten v_1 und v_2 entspricht der kleinsten Anzahl von Kanten die auf einem Weg zwischen den beiden Knoten liegen.

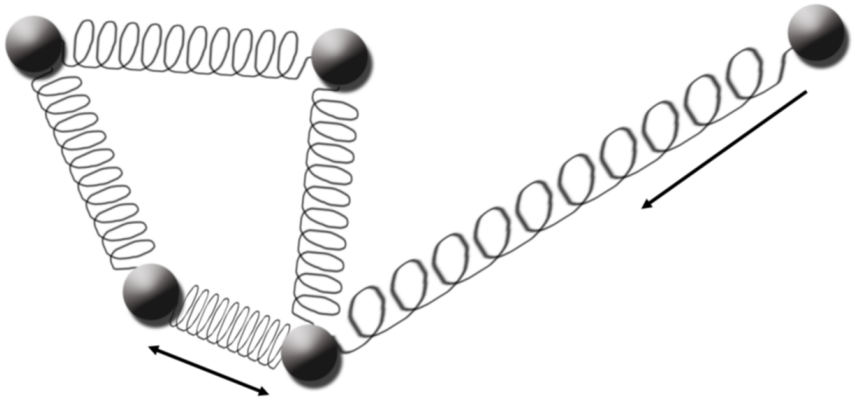

Abbildung 1.5: Gestreckte Federn ziehen die Kugeln zueinander. Gestauchte Federn drücken sie auseinander.

Die Federn ziehen die an ihnen hängenden Kugeln normalerweise nicht beliebig nah zueinander, sondern haben eine Ruhelänge. Wird diese unterschritten, erzeugen die Federn sogar abstoßende Kräfte um die gewünschte Kantenlänge zu erreichen.

Qualitätskriterien, wie zum Beispiel die Anzahl der Kreuzungen oder Symmetrien, werden meist nicht explizit modelliert. Die uniforme Kantenlänge und das Aufzeigen von Symmetrien sind Stärken der kräftebasierten Zeichenverfahren. Die Anzahl der Kreuzungen ist oft deutlich höher als bei Zeichnungen, die durch andere Verfahren erstellt wurden. So kann es leicht passieren, dass sogar planare Graphen nicht kreuzungsfrei gezeichnet werden. Auch häufen sich Schnitte von Knoten mit Kanten, die nicht bei ihnen enden, was bei den meisten anderen Zeichenverfahren unmöglich ist. Mit den aktuellsten Verbesserungen durch Approximation der weitreichenden Kräfte liegt der Vorteil von kräftebasierten Verfahren darin, besonders große Graphen in annehmbarer Zeit zeichnen zu können. Dieser Vorteil wird durch den Multilevel-Ansatz noch weiter ausgebaut.

Algorithmus 1.1 KRÄFTEBASIERTES ZEICHENVERFAHREN

Eingabe: Knotenmenge V, Kantenmenge E, Positionen $\vec{P}$, Qualitätsgrenze q
Ausgabe: Positionen $\vec{P}$

repeat
  $\vec{F}[\,] \leftarrow \emptyset$ // die Kräfte
  for all Knoten $v_1 \in$ V **do** // Kräfte berechnen
    for all Knoten $v_2 \in$ V **do** // abstoßende Kräfte
      $\vec{F}[v_1] \leftarrow \vec{F}[v_1] + (\vec{P}[v_2] - \vec{P}[v_1])$ / dist$(\vec{P}[v_1], \vec{P}[v_2])^2$
    end for
    for all Knoten v_2 aus Nachbarn von v_1 **do** // Kantenkräfte an allen Nachbarn
      $\vec{F}[v_1] \leftarrow \vec{F}[v_1] - (\vec{P}[v_2] - \vec{P}[v_1])$ / dist$(\vec{P}[v_1], \vec{P}[v_2])$
    end for
  end for
  for all Knoten $v_1 \in$ V **do** // Kräfte anwenden
    $\vec{P}[v_1] \leftarrow \vec{P}[v_1] + \vec{F}[v_1]$
  end for
until max$(\vec{F}[\,]) <$ q
return $\vec{P}$

1.3.1 Probleme bei kräftebasierten Verfahren

Ein Problem ist, dass bei exakter Berechnung bereits eine Laufzeit von $O(n^2)$ für eine einzige Iteration des Kräfteverfahrens, in der sich alle Knoten gegenseitig abstoßen, notwendig ist. Verbesserungen konnten hier durch raumaufteilende Datenstrukturen wie ein Raster (Verfahren von Fruchtermann und Reingold [FrRe91]) oder einen Quadtree (Algorithmus FM^3 von Hachul [Hac05]) erreicht werden. Die Kräfte zu den entfernteren Knoten können nun approximiert werden und eine Iteration des Verfahrens ist deutlich schneller durchführbar. Hachul hat für seinen Ansatz FM^3 bewiesen, dass eine Iteration des Kräfteverfahrens stets in $O(n \log n)$ durchgeführt wird [Hac05].

Ein weiteres Problem für die Laufzeit von kräftebasierten Verfahren ist, dass oft Teile des Graphen über weite Strecken bewegt werden müssen, während alle Knoten sich in einer Iteration nur ein kleines Stück bewegen können. Dadurch werden sehr viele Iterationen gebraucht, bis eine ausreichende Zeichnungsqualität erreicht ist. Da das Verfahren üblicherweise damit beginnt, alle Knoten nah beieinander an zufälligen Positionen zu platzieren, müssen die in der Zeichnung später ganz außen liegenden Punkte besonders weit bewegt werden.

Besonders ungünstig für ein kräftebasiertes Verfahren ist es, wenn die momentane Zeichnung des Graphen Faltungen oder Verdrehungen enthält. Ein solcher Zustand kann von Anfang an gegeben sein oder im Laufe des Verfahrens entstehen. Wie eine Verdrehung

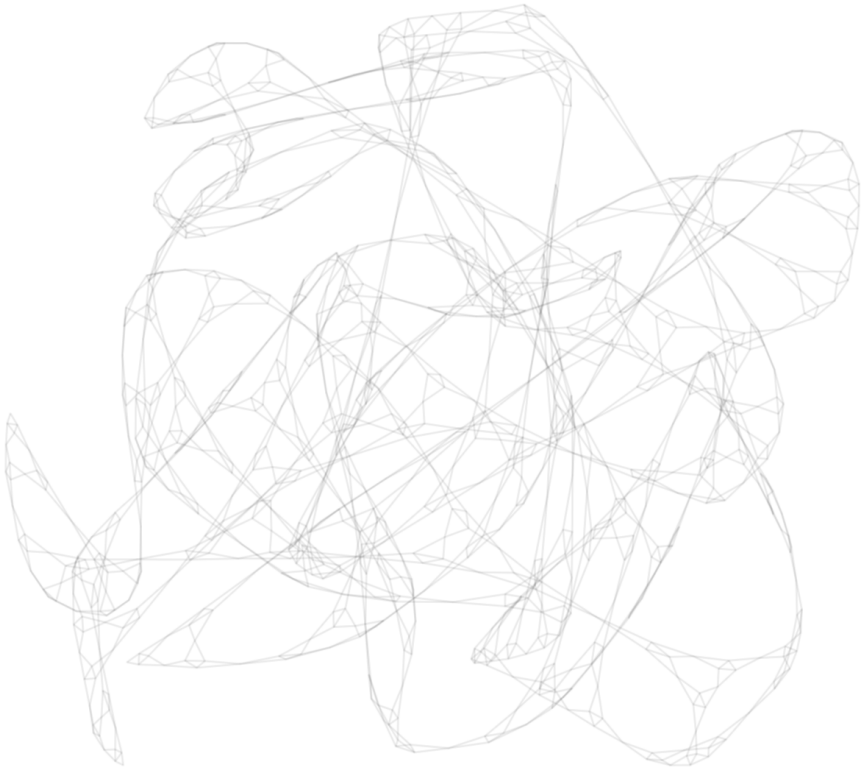

Abbildung 1.6: Ein verknoteter Graph.

oder Faltung aussieht, kann man sich leicht vorstellen, wenn man ein Stück Stoff zu Hilfe nimmt. Wenn dieses ausreichend verknotet ist, kann der Knoten nicht aufgelöst werden, indem man einfach an den Enden des Stoffstücks zieht. So ähnlich kann es passieren, dass ein Kräfteverfahren diese Verdrehungen gar nicht mehr auflösen kann, da es ein lokales Kräftegleichgewicht bereits erreicht hat.

1.4 Multilevel-Verfahren

Multilevel-Verfahren wurden entwickelt, um besonders große Graphen zeichnen zu können und einige der Probleme bei klassischen kräftebasierten Verfahren zu bekämpfen. Je größer der Graph ist, desto extremer werden die bei kräftebasierten Verfahren auftretenden Entfaltungsprobleme. Die nötige Laufzeit bis ein Kräftegleichgewicht erreicht ist, kann inakzeptabel lang werden. Ein Multilevel-Verfahren nutzt aus, dass graphentheoretisch nah beieinander liegende Knoten meist auch nah beieinander gezeichnet werden sollten, was durch das kräftebasierte Verfahren auch zu erreichen versucht wird. Mit dieser Einsicht kann man benachbarte Knoten zunächst als einen einzelnen Knoten behandeln und so mit nur geringem Rechenaufwand eine gute Vorab-Zeichnung erhalten, die sich bereits nahe an einem Kräftegleichgewicht befindet. Dazu wird zunächst auf den vereinfachten Graphen das kräftebasierte Verfahren angewandt. Dann wird eine Vorab-Zeichnung generiert indem

die zuvor verschmolzenen Knoten wieder getrennt und nahe beieinander platziert werden. Auf die Vorab-Zeichnung kann das ursprüngliche Verfahren angewandt werden, um eine gute Zeichnung für den gesamten Graphen zu erhalten.

Algorithmus 1.2 MULTILEVEL-ZEICHENVERFAHREN

Eingabe: Graph G

Ausgabe: Positionen P

1: $G \leftarrow \emptyset$ // die Verfeinerungsstufen
2: $i \leftarrow 1$
3: $G[0] \leftarrow G$
4: **repeat** // Verfeinerungsstufen erzeugen
5: $G[i] \leftarrow$Zerlegungsstrategie($G[i-1]$)
6: $i \leftarrow i + 1$
7: **until** sizeof(G[i] ≤ 3) // Graphen mit ≤ 3 Knoten sind leicht zu zeichnen
8: $P \leftarrow random$
9: **for** $i \leftarrow sizeof(G)$ .. 1 **do** // Beginnend mit der kleinsten Verfeinerungsstufe
10: $P \leftarrow$Kräfteverfahren($G[i], P$)
11: $P \leftarrow$Platzierungsstrategie($G[i], G[i-1], P$)
12: **end for**
13: $P \leftarrow$Kräfteverfahren($G[0], P$)
14: **return** P

In der Praxis werden so viele Vereinfachungen hintereinander durchgeführt, bis der Graph eine bestimmte Größenschranke erreicht hat, unter der es für das zugrunde liegende Kräfteverfahren leicht ist, den Graphen zu zeichnen. Es werden also zunächst in einer Verfeinerungsphase alle vereinfachten Graphen errechnet. Danach wird in der Kräftephase der einfachste Graph durch das Kräfteverfahren gezeichnet. Die so entstandene Zeichnung wird in der Platzierungsphase genutzt, um eine gute Ausgangszeichnung für die nächste Verfeinerungsstufe zu berechnen.

1.4.1 Verfeinerungsphase

Aus dem originalen Graphen G_0 wird eine Serie von Graphen $G_0 ... G_k$ erzeugt. Der Graph G_{i+1} wird aus G_i erzeugt, wobei der erzeugte Graph kleiner als der vorherige sein und die Struktur des ursprünglichen Graphen möglichst genau abbilden soll. Es gibt verschiedene Verfahren, mit denen der vereinfachte Graph berechnet werden kann, von denen einige in Kapitel 4 vorgestellt werden. Sie haben alle gemeinsam, dass sie durch die Verschmelzung von benachbarten Knoten beschrieben werden können.

Im vereinfachten Graphen G_{i+1} repräsentiert ein Knoten meist mehrere Knoten aus der vorherigen Verfeinerungsstufe G_i. Der Graph kann vereinfacht werden, bis die kleinste

Verfeinerungsstufe nur noch eine bestimmte Menge Knoten enthält. Oft sind das nur noch ein oder zwei Knoten, die dann den gesamten Graphen G repräsentieren.

1.4.2 Kräftephase

Beginnend mit $i = k$ wird ein Kräfteverfahren auf die Verfeinerungsstufe G_i des Graphen angewandt. Die daraus resultierende Zeichnung wird als Eingabe für die Platzierungsphase benötigt. Das eingesetzte Zeichenverfahren sollte die ursprünglich bestehende Position der Knoten nicht verwerfen, sondern auf dieser Basis eine verbesserte Zeichnung liefern. Platzierungs- und Kräftephase wechseln einander ab bis $i = 0$ erreicht ist. Die im letzten Aufruf der Kräftephase entstehende Zeichnung des Graphen G_0 ist die endgültige Zeichnung des Graphen und wird ausgegeben.

1.4.3 Platzierungsphase

Die Knoten aus der Verfeinerungsstufe G_i werden platziert. Dabei wird ausgenutzt, dass für die Knoten aus dem Graphen G_{i+1} bereits eine Zeichnung vorliegt. Die einfachste Möglichkeit die Knoten zu platzieren ist, für die neuen Knoten aus G_i, die nicht in G_{i+1} existieren, genau die Position zu übernehmen, die ihr Repräsentant in der Zeichnung hat. Die so entstandene vorläufige Zeichnung für den Graphen G_i wird nun durch die darauf folgende Kräftephase verbessert.

Vorteil eines Multilevel-Verfahrens gegenüber einem kräftebasierten Verfahren ist erstens die bessere Qualität, da der Graph zuverlässiger entfaltet wird und oft schlechte Kräftegleichgewichte vermieden werden. Die bessere Entfaltung wird hauptsächlich dadurch erreicht, dass die einfachen Verfeinerungsstufen leicht ohne Faltung gezeichnet werden können. Die Vorab-Zeichnung ohne Faltungen wird für die nächstgrößere Verfeinerungsstufe als Vorlage genommen, so dass auch diese ohne Faltungen gezeichnet werden kann. Zweitens verringert sich die Laufzeit des Algorithmus, da das Bewegen größerer Teile des Graphen und eine eventuell nötige Größenveränderung bereits auf sehr groben Verfeinerungsstufen geschehen und so deutlich weniger Zeit in Anspruch nehmen. Auch die schnelle Entfaltung verbessert die Laufzeit des Gesamtverfahrens.

1.4.4 Probleme bei Multilevel-Verfahren

Auch ein Multilevel-Verfahren arbeitet natürlich nicht absolut fehlerfrei, so dass es auch hier sinnvoll ist, dass viele verschiedene Multilevel-Ansätze existieren. Besonders gravierend wirken sich im Multilevel-Verfahren die Fehler aus, die auf einer der gröbsten Verfeinerungsstufen passieren. Diese Fehler werden im Laufe des Verfahrens für alle nachfolgenden Stufen übernommen und können meist nicht mehr ausgeglichen werden. In den meisten Fällen sind diese Fehler jedoch deutlich geringer als beim eingesetzten kräftebasierten Ver-

fahren ohne Multilevel-Ansatz. Um Fehler auf den frühen Stufen zu vermeiden wird bei manchen Multilevel-Verfahren besonders viel Zeit in die ersten Kräftephasen investiert.

Ein Problem, das durch den Multilevel-Ansatz erzeugt wird, sind zusätzliche Verdrehungen des Graphen. Im Laufe der Verfeinerungsphase kann es passieren, dass der Zwei-Zusammenhang des Graphen, oder einer bisher zwei-zusammenhängenden Komponente des Graphen verloren geht. Die dadurch entstehenden Verdrehungen wären ohne den Multilevel-Ansatz nicht aufgetreten.

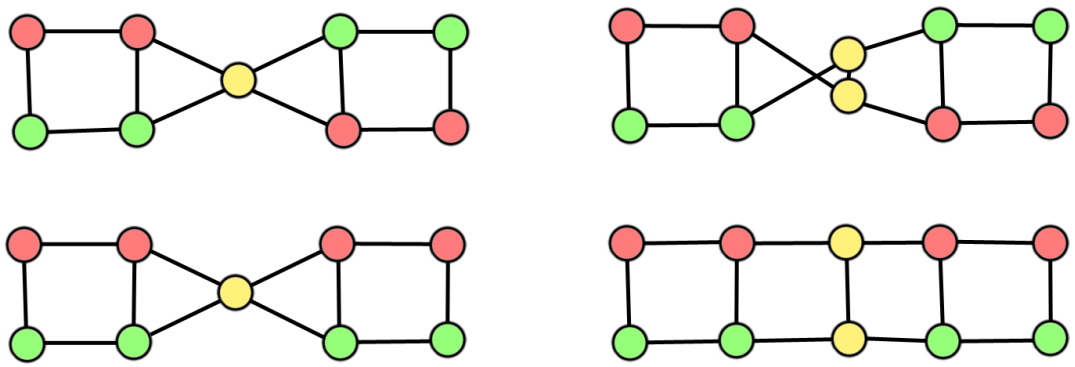

Abbildung 1.7: Zwei gleichberechtigte mögliche Zeichnungen einer nicht zwei-zusammenhängenden Verfeinerungsstufe und die daraus resultierenden Zeichnungen.

Definition 4 (Zwei-Zusammenhang). Ein Graph $G = (V, E)$ ist zwei-zusammenhängend, wenn es zwischen allen Knotenpaaren $v_1, v_2 \in V$ mindestens zwei Wege gibt, die außer den beiden Endknoten keinen Knoten gemeinsam haben.

Definition 5 (Zwei-Zusammenhangs-Komponente). Eine Menge von Knoten $V_z \subset V$ aus einem Graphen $G = (V, E)$ heißt Zwei-Zusammenhangs-Komponente, wenn es zwischen allen Knotenpaaren $v_1, v_2 \in V_z$ mindestens zwei Wege gibt, die außer den beiden Endknoten keinen Knoten gemeinsam haben. Zudem darf es keinen Knoten $v_3 \notin V_z$ geben der zu V_z hinzugenommen werden kann, ohne, dass der Zwei-Zusammenhang in V_z verloren geht.

Wenn in der entsprechenden Kräftephase der Graph aus der nicht zwei-zusammenhängenden Verfeinerungsstufe gezeichnet wird, hat das Kräfteverfahren keinen Anhaltspunkt, um zu entscheiden, wie die beiden Komponenten aneinander gefügt werden sollten, ohne eine Verdrehung des resultierenden Graphen zu erzeugen. Die Wahrscheinlichkeit, an dieser Stelle die falsche Wahl zu treffen liegt bei $\frac{1}{2}$. Der Fehler kann eventuell durch eine weitere schlechte Wahl in der Nähe oder, wenn die Verdrehung nahe am Rand des Graphen liegt, durch das Kräfteverfahren ausgeglichen werden. Wenn ein Graph an vielen Stellen den Zwei-Zusammenhang verliert, kann die Wahrscheinlichkeit, dass der Graph noch feh-

lerfrei gezeichnet wird, äußerst gering werden, so dass auch ein mehrmaliges voneinander unabhängiges Durchführen des Verfahrens keine gute Zeichnung mehr erzeugt.

Abbildung 1.8: Ein Graph mit Verdrehung.

Das Problem tritt bei allen vorgestellten existierenden Multilevel-Verfahren auf. Wie negativ es sich auf die Zeichnung auswirken kann, liegt hauptsächlich am Graphen selbst. Graphen, in denen bereits sehr dünne Stellen existieren, an denen nur wenige Knoten einen Verlust des Zwei-Zusammenhangs verhindern, sind besonders gefährdet. Eine Zerlegungsstrategie, die explizit entworfen wurde um dieses Problem zu bekämpfen, wurde im Rahmen dieser Arbeit entwickelt und wird noch vorgestellt.

1.5 Existierende Multilevel-Verfahren

Der Multilevel-Ansatz wurde bereits von einigen Autoren gewählt und mit Erfolg angewandt. Es werden nicht alle existierenden Verfahren vorgestellt, sondern nur die bekanntesten. Die Verfeinerungsstufen und Platzierungen der meisten Verfahren wurden für diese Arbeit neu implementiert um ein gleichberechtigtes modulweises Austauschen zu ermöglichen. Einige der kräftebasierten Verfahren waren im Open Graph Drawing Framework [OGDF] nicht verfügbar, da sie speziell für das jeweilige Multilevel-Verfahren entwickelt wurden. Die Implementierung weiterer kräftebasierter Verfahren hätte den Rahmen dieser Arbeit gesprengt, so dass der Vergleich ohne die spezialisierten kräftebasierten Verfahren stattfinden muss.

Dies hat natürlich zur Folge, dass der hier vorliegende Vergleich kein exakter und fairer Vergleich der jeweils vorgestellten Verfahren ist, sondern nur ein Vergleich der zu Grunde liegenden Vereinfachungs- und Platzierungsverfahren. Bei der Bewertung der experimentellen Ergebnisse sollte beachtet werden, dass das zugrunde liegende Kräfteverfahren einen enormen Einfluss auf Zeichnungsqualität und Laufzeit hat, wodurch die Ergebnisse wenig Aussagekraft für Verfahren haben, die ein anderes Kräfteverfahren als die hier getesteten nutzen. Zuletzt haben auch die vielfältigen Parameter der Kräfteverfahren einen großen Einfluss, so dass es sicher möglich ist, für bestimmte Graphen deutlich bessere Ergebnisse zu erzielen, indem die entsprechenden Parameter angepasst werden. Da die hier betrachteten Verfahren für Graphen aller Art nutzbar sein sollen, wurden Parameter gesucht, die für alle Graphen annehmbare Ergebnisse liefern. Schlechtere Ergebnisse für einzelne Graphen wurden dabei in Kauf genommen.

1.5.1 Algorithmus von Walshaw

Walshaw hat einen Algorithmus zum kräftebasierten Zeichnen von Graphen vorgestellt, bei dem ein Multilevel-Ansatz verwendet wird [Wal03]. Als Kräfteverfahren wird eine modifizierte Version des Verfahrens von Fruchtermann und Reingold [FrRe91] genutzt. Die Verfeinerungsstufen werden erzeugt, indem auf dem aktuellen Graphen eine Paarung (Matching) erzeugt wird. Diese muss nicht optimal sein, ist jedoch nicht trivial erweiterbar. Es wird versucht, zu einem zufälligen Knoten einen Nachbarn mit möglichst geringer Masse zu finden, der noch nicht für eine frühere Matching-Kante gewählt wurde. Die Positionierung der Knoten in der Platzierungsphase erfolgt durch die Übernahme der Position des Matching-Partners.

Definition 6 (Knotengrad). Der Grad eines Knotens ist definiert als die Anzahl der Kanten, die an diesem Knoten starten oder enden.

Auf einem Matching basierende Multilevel-Verfahren haben den Nachteil, dass sie sehr viele ähnliche Multilevel-Ebenen erzeugen, sobald der Graph Knoten mit extrem hohem Knotengrad enthält. Ein Matching wird stets höchstens eine zu einem Knoten gehörende Kante auswählen. Dies beeinflusst die Gesamtlaufzeit des Algorithmus negativ, da die Anzahl der Verfeinerungsstufen im Extremfall linear zu der Anzahl der Knoten verlaufen kann. Zusammen mit einem Kräfteverfahren, das ebenfalls eine mindestens lineare Laufzeit aufweist, wird die Gesamtlaufzeit bereits quadratisch. Üblicherweise soll ein Multilevel-Verfahren nur eine logarithmische Anzahl von Verfeinerungsstufen erzeugen.

1.5.2 Fast Multipole Multilevel Method (FM^3)

Die *Fast Multipole Multilevel Method* von Hachul [Hac05] zeichnet sich dadurch aus, dass mit ihr besonders große Graphen in kurzer Zeit gezeichnet werden können. Das Kräfteverfahren wurde für diesen Algorithmus neu entwickelt und liefert im Vergleich mit den anderen Verfahren eine besonders schnelle und genaue Approximation der weitreichenden abstoßenden Kräfte. Zur *Fast Multipole Multilevel Method* wurden weitere Paper [HaJu05, HaJu05b] veröffentlicht, die einen kompakteren Überblick und weiterführende Informationen liefern.

Multilevel-Erzeugung über Sonnensysteme

Die Verfeinerungen des Graphen werden erzeugt, indem die Knoten des Graphen einer Sonnensystem-Metapher entsprechend in Sonnen, Planeten und Monde eingeteilt werden. Nur die Sonnen werden in die nächste Verfeinerungsstufe übernommen. Jeder Knoten, der einen graphentheoretischen Abstand von Eins zu einer Sonne hat, wird als Planet definiert. Da ein Planet nicht zu zwei Sonnen gehören darf, muss der Abstand zwischen zwei Sonnen mindestens Drei betragen. Dadurch kann es passieren, dass Knoten weder

Planet einer Sonne sind, noch selber eine Sonne sein dürfen. Diese Knoten werden als Mond definiert und gehören zu einem benachbarten Planeten. Wie Hachul zeigt, ist es immer möglich den Graphen nach diesem Muster in Sonnensysteme einzuteilen [Hac05]. Besonderen Wert legt er auf die Beobachtung, dass in jedem Fall nur logarithmisch viele Verfeinerungsstufen erzeugt werden können. Leider kann es bei sehr dichten Graphen sein, dass nur sehr wenige Verfeinerungsstufen erzeugt werden, so dass kaum Nutzen aus dem Multilevel-Ansatz gezogen wird.

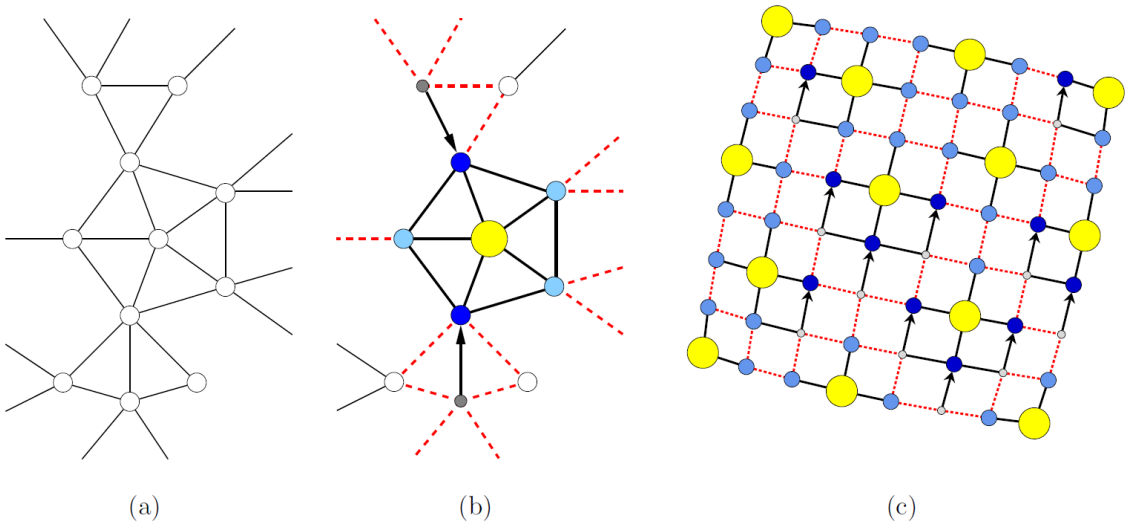

Abbildung 1.9: Graphen mit Einteilung in Sonnensysteme. Gelbe Knoten sind Sonnen, blaue Planeten und graue Monde. [Hac05]

Platzierung mit Zusatzinformationen

Um eine möglichst geschickte Platzierung der Knoten in der nächsten Verfeinerungsstufe zu erreichen, werden während der Erzeugung der Sonnensysteme weitere Informationen berechnet. Jede Kante zwischen zwei Knoten auf einer Verfeinerungsstufe n entspricht in der nächst gröberen Verfeinerungsstufe $n-1$ einem Pfad von einer Sonne über einen Planeten, eventuell Monde und einen weiteren Planeten zur anderen Sonne. Die relative Position eines Knotens zu den beiden Sonnen kann nun als Position auf diesem Intersystempfad annähernd beschrieben werden. Diese Pfadpositionen werden gespeichert. Wenn der Knoten nun platziert werden soll, wird er entsprechend der Pfadpositionen näher an der einen oder der anderen Sonne platziert. Befindet sich ein Knoten auf mehreren Intersystempfaden, wird der Durchschnitt der jeweils errechneten Positionen für den Knoten übernommen.

Variante Fast Multilevel Multipole Embedder (FMME)

Von Martin Gronemann wurde im Zeitraum dieser Arbeit eine Variante des *FM*^{3} Algorithmus entwickelt [Gro09]. Ziel der Entwicklung des *Fast Multilevel Multipole Embedder* ist die Beschleunigung des Algorithmus durch Parallelisierung. Das dafür neu implementierte Kräfteverfahren *FME* steht im OGDF zur Verfügung. In dieser Arbeit wurde das Kräfteverfahren wie viele andere in den Vergleichen genutzt und war an einigen der besten Ergebnisse beteiligt. Auch das Gesamtverfahren wird in der Arbeit mit anderen verfügbaren Verfahren verglichen. Der *Fast Multilevel Multipole Embedder* ist zudem das erste Verfahren, das direkt vom in dieser Arbeit entwickelten *Component Splitter Layout* profitiert, da diese Funktionalität so nicht ein weiteres mal implementiert werden musste.

1.5.3 Graph Drawing with Intelligent Placement (GRIP)

Gajer und Kobourov haben ein Multilevel-Verfahren namens GRIP vorgestellt [GaKo02]. Das kräftebasierte Verfahren ist für GRIP entwickelt worden, nutzt aber die Kräftefunktion von Kamada und Kawai [KaKa88]. In der letzten Kräftephase wird die Kräftefunktion von Fruchtermann und Reingold [FrRe91] genutzt.

Verfeinerungsstufen des Graphen werden hier gebildet, indem eine möglichst große unabhängige Knotenmenge (independent set) gefunden wird. Im ersten Schritt werden zunächst so lange Knoten ohne einen bereits gewählten Nachbarn ausgewählt, bis dies nicht mehr möglich ist. Die so gewählten Knoten entsprechen der ersten Verfeinerungsstufe. Die folgenden Verfeinerungsstufen werden erzeugt, indem von diesen Knoten aus eine Tiefensuche bis zur Tiefe $2^{(i-1)}$ durchgeführt wird, wobei i die Nummer der Verfeinerungsstufe ist. Jeder Knoten, der bei einer solchen Suche erreicht wurde, ist nicht in der Verfeinerungsstufe $i + 1$ enthalten. Von bereits entfernten Knoten aus wird keine Tiefensuche ausgeführt. Die Platzierungsstrategie nutzt ebenfalls die graphentheoretischen Distanzen im ursprünglichen Graphen aus.

Ein Nachteil des Verfahrens ist wie schon bei *FM*^{3}, dass bei sehr dichten Graphen nur wenige Verfeinerungsstufen erzeugt werden. Der Algorithmus kann dann kaum Vorteile aus dem Multilevel-Ansatz ziehen. Vorteilhaft ist die besonders gleichmäßige Verteilung der Knoten aus den Verfeinerungsstufen und, dass wie bei *FM*^{3} garantiert nicht mehr als logarithmisch viele Verfeinerungsstufen erzeugt werden.

1.5.4 Algorithmus von Harel und Koren

Der Algorithmus von Harel und Koren [HaKo02] nutzt als Kräfteverfahren eine Variante der Methode von Kamada und Kawai [KaKa89]. Das Verfeinerungsverfahren benötigt die kürzesten Wege zwischen allen Knotenpaaren, was eine hohe Laufzeit erzwingt. Im Verfahren von Kamada und Kawai fällt dies nicht so sehr ins Gewicht, da die kürzesten Wege

bereits für einen Preprocessing-Schritt benötigt werden und so leicht wiederverwendbar sind.

Der größte Nachteil dieses Ansatzes ist, dass der Aufbau der Verfeinerungsstufen bereits quadratische Laufzeit benötigt. Ein diesem Verfeinerungsverfahren entsprechender Ansatz wurde nicht implementiert, da der Fokus der Arbeit auf sehr großen Graphen liegt. Die mindestens quadratische Laufzeit würde sämtliche Laufzeitvergleiche dominieren. Der Nutzen einer solchen Variante ist insbesondere in Hinblick auf deutlich schnellere Algorithmen ähnlicher Qualität für große Graphen fraglich.

Kapitel 2

Der Modular Multilevel Mixer

Hauptziel dieser Arbeit war es, mit dem *Modular Multilevel Mixer* eine Umgebung zu schaffen, in der verschiedene Zerlegungsstrategien und Platzierungsverfahren fair verglichen werden können. Ein Vergleich dieser Art war bisher nicht ohne weiteres möglich, da jedes Multilevel-Verfahren zusätzlich zu einer eigenen Zerlegungsstrategie meist auch eine eigene Platzierungsstrategie und, was am wichtigsten ist, ein eigenes Kräfteverfahren oder zumindest eine eigene Variante eines bestehenden Verfahrens nutzt. Das Austauschen einer einzelnen dieser Komponenten zu Vergleichszwecken war in den bestehenden Multilevel-Verfahren nie vorgesehen. Zusätzlich zu einem ausreichend flexiblen Rahmenwerk mussten also auch alle zu vergleichenden Zerlegungsstrategien und Platzierungsverfahren neu implementiert werden, um zu diesem System kompatibel zu sein.

Es hat sich jedoch herausgestellt, dass eines der wichtigsten Unterscheidungsmerkmale der verschiedenen Multilevel-Verfahren auch im Kräfteverfahren zu finden ist. Glücklicherweise war im OGDF, das als Rahmen für die Implementierung gewählt wurde, bereits ein modularer Ansatz vorhanden, um verschiedene Zeichensysteme auszutauschen. Darunter waren auch einige kräftebasierte Verfahren, so dass ohne viel Aufwand ein Austauschen der Kräfteverfahren im *Modular Multilevel Mixer* möglich wurde. Die sich daraus ergebenden Vergleichsmöglichkeiten wurden natürlich genutzt. Des Weiteren waren die Ergebnisse mancher Varianten sowohl von der Laufzeit als auch von der Zeichnungsqualität her so überzeugend, dass der *Modular Multilevel Mixer* durchaus als einsetzbares und nützliches Zeichensystem für große Graphen angesehen werden kann.

Auch möglichst viele andere Teile des Gesamtsystems *Modular Multilevel Mixer* wurden als Module implementiert, wenn möglich als *ogdf::LayoutModule*, um im Rahmen des OGDF unabhängig eingesetzt werden zu können. Diese Module können in Kombination mit anderen Zeichenverfahren eingesetzt werden.

2.1 Komponentenübersicht

Zentrale Klasse des Systems ist der *Modular Multilevel Mixer*. Die meisten Module werden von hier aus aufgerufen, wobei auch viele Parameter modifiziert werden können. Die wichtigsten Module sind natürlich das kräftebasierte Verfahren, das als *ogdf::LayoutModule* übergeben wird, die Zerlegungsstrategie, die als Implementierung des Interfaces *MultilevelBuilder* übergeben wird, und die Platzierungsstrategie, die das Interface *InitialPlacer* implementiert. Alle Module erhalten eine Referenz auf den aktuellen *Multilevel Graph* und können diesen verändern, so dass keine Kopien notwendig werden. Aus dem *MultilevelBuilder* Interface muss nur die Funktion *void buildOneLevel(MultilevelGraph &MLG)* implementiert werden, was beliebige Freiheit in der Implementierung lässt. Für die Platzierungsstrategien muss nur die Funktion *void placeOneLevel(MultilevelGraph &MLG)* implementiert werden.

Des Weiteren wird das *Scaling Layout*, das ebenfalls ein *ogdf::LayoutModule* ist, im *Modular Multilevel Mixer* aufgerufen, um die Skalierung des Graphen zu steuern. Zuletzt wird noch das Postprocessing durchgeführt, das ebenfalls durch ein *ogdf::LayoutModule* angegeben wird.

Unabhängig von der Klasse *Modular Multilevel Mixer* gibt es noch das *Preprocessor Layout*, in dem der Graph in eine Form gebracht wird, die von allen Teilen des Algorithmus und allen Kräfteverfahren bearbeitet werden kann und das *Component Splitter Layout*, das dafür zuständig ist, dass die Algorithmen nur zusammenhängende Graphen bearbeiten müssen. Diese sind beide *ogdf::LayoutModule* und erhalten ein weiteres *ogdf::LayoutModule* als Parameter. Dieses wird für die bearbeiteten Graphen ausgeführt. Dadurch ist es besonders leicht möglich, ein kombiniertes Layout zu erstellen, das durch einen einzelnen Aufruf genutzt werden kann, ohne dass Zwischenergebnisse verwaltet werden müssen.

2.2 Multilevel Graph

Der *Multilevel Graph* ist die zentrale Datenstruktur innerhalb des *Modular Multilevel Mixers*. Das Interface *ogdf::LayoutModule*, von dem alle hier genutzten kräftebasierten Verfahren und der *Modular Multilevel Mixer* abgeleitet sind, wurde erweitert um Layoutaufrufe nicht nur mit *ogdf::GraphAttributes*, sondern auch mit einem *Multilevel Graph* zu erlauben. Dies war nötig, um eine ständige Konvertierung eines *Multilevel Graph* in *ogdf::GraphAttributes* zu vermeiden. Verglichen mit *ogdf::GraphAttributes* speichert der *Multilevel Graph* nur relativ wenig Informationen über die Darstellung des Graphen. Dafür können neben Knotenpositionen, Größen und gewünschten Kantenlängen zusätzlich alle Verfeinerungsstufen des Graphen gespeichert werden. Um unnötige Kopien zu vermeiden, ist es im Gegensatz zu *ogdf::GraphAttributes* erlaubt, Änderungen am Graphen vorzunehmen.

Die einzelnen Verfeinerungsstufen werden nicht explizit als Graphen gespeichert, sondern der ursprüngliche Graph wird modifiziert. Die Änderungen bestehen sämtlich aus Verschmelzungen zweier Knoten samt Modifikation der betroffenen Kanten. Alle Verschmelzungen werden gespeichert und mit der Information versehen, zu welchem Level sie gehören. Im späteren Verlauf des Algorithmus können die Verschmelzungen dann einzeln rückgängig gemacht werden.

Eine weitere Funktion, die im *Multilevel Graph* implementiert wurde, ist die Zerlegung des Graphen in seine Zusammenhangskomponenten, wie sie vom *Component Splitter* benötigt werden. Zum Zeitpunkt der Aufteilung des Graphen sind noch keine Verschmelzungen ausgeführt worden.

2.3 Preprocessor

Viele Zeichenverfahren setzen voraus, dass die Graphen keine mehrfachen Kanten enthalten. Auch Schleifen, also Kanten deren Start- und Zielknoten identisch sind, können oft nicht korrekt behandelt werden. In den Verfahren wird schlicht davon ausgegangen, dass der Benutzer die Eingaben diesen Vorgaben entsprechend bereitstellt, während andere die Eingabe an die Anforderungen anpassen. Um hier ein einheitliches Verhalten aller getesteten Algorithmen zu ermöglichen und alle Graphen der Testmenge bearbeiten zu können, wird ein eigenes *ogdf::LayoutModule* bereitgestellt, das diese Aufgaben übernimmt. Das *Preprocessor Layout* löscht doppelte Kanten und Schleifen, ruft dann ein anderes *ogdf::LayoutModule*, z.B. ein kräftebasiertes Verfahren, auf und fügt zuletzt die gelöschten Kanten wieder ein, da diese für den Nutzer des Zeichenverfahrens wichtig sein können. Doppelte Kanten und Schleifen haben im Kontext eines kräftebasierten Verfahrens kaum einen Einfluss auf die Zeichnung. Mehrfache Kanten können durch eine einzige Feder repräsentiert werden, während bei Schleifen keine Kräfte auf den betroffenen Knoten wirken würden. Dieses Modul kann auch in Kombination mit einem beliebigen *ogdf::LayoutModule* eingesetzt werden, was jedoch selten der Fall sein wird, da die entsprechenden Berechnungen ohne großen Aufwand implementiert werden können.

2.4 Component Splitter

Eine häufige Voraussetzung für Graphzeichenverfahren ist, dass der Graph zusammenhängend sein soll. Dies ist natürlich nicht immer gegeben. Das *Component Splitter Layout* soll diese Fälle bearbeiten, so dass die Verfahren einerseits nur zusammenhängende Graphen bearbeiten müssen, andererseits jedoch eine ästhetisch ansprechende Zeichnung für den gesamten nichtzusammenhängenden Graphen entsteht. Die Idee ist nicht neu und wurde zum Beispiel in *FM*^{3} [Hac05], der vorliegenden Implementierung des Verfahrens von Fruchtermann und Reingold [FrRe91] und *GEM* [FLM98] bereits angewandt. Da an-

dere Algorithmen diesen Ansatz jedoch nicht wählen, wäre ein Vergleich der zugrunde liegenden Algorithmen bei nicht zusammenhängenden Graphen unfair. In allen Varianten des *Modular Multilevel Mixer* und allen damit verglichenen Algorithmen wird also das *Component Splitter Layout* vorgeschaltet. Viele Verfahren sind nicht darauf ausgelegt, mit nichtzusammenhängenden Graphen zu arbeiten, und liefern mit derartigen Eingaben sehr schlechte Ergebnisse oder arbeiten gar nicht. Für solche Verfahren bietet das *Component Splitter Layout* deutliche Verbesserungen. Das Modul kann dementsprechend auch mit einem beliebigen *ogdf::LayoutModule* genutzt werden, um dort ein Zerlegen und ein abschließendes Zusammenfügen zu nutzen. Für zukünftig im Rahmen des OGDF implementierte Zeichenverfahren ist also nicht mehr nötig einen entsprechenden Algorithmus zu implementieren. Das im Rahmen dieser Arbeit entwickelte *Component Splitter Layout* ist bereits außerhalb der Arbeit genutzt worden, um den *Fast Multilevel Multipole Embedder* auf nichtzusammenhängenden Graphen einsetzen zu können.

Nachdem für jede Komponente eine Zeichnung berechnet wurde, müssen diese noch möglichst platzsparend auf der Zeichenfläche angeordnet werden. Dies geschieht mit Hilfe des bereits im OGDF enthaltenen Algorithmus *TileToRowsCCPacker*. Erwartet wird als Eingabe eine Liste von Rechtecken, die hier den umschließenden Rechtecken um die einzelnen Komponenten entsprechen. Da die Komponenten sehr ungleichmäßig geformt sein können, kann es eine große Platzersparnis ergeben, wenn eine optimale Drehung der Komponente berechnet wird. Die Drehung gilt in diesem Zusammenhang als optimal, wenn dadurch die Fläche des umschließenden Rechtecks minimiert wird. Zunächst wird beobachtet, dass die optimale Drehung einer aus Punkten und ihren Verbindungen bestehenden Zeichnung gleich der optimalen Drehung der konvexen Hülle dieser Punkte ist. Von Ausnahmefällen abgesehen enthält die konvexe Hülle deutlich weniger Punkte als die umschlossene Punktmenge, wodurch die Berechnung der Drehung bereits stark beschleunigt wird. Durch die Außenseiten der konvexen Hülle ist zudem eine beschränkte und gute Kandidatenmenge für mögliche Drehungen gegeben. Durch Auswahl einer dieser Kanten kann das umschließende Rechteck für die entsprechende Drehung eindeutig hergeleitet werden, indem eine Kante des Rechtecks (hier die untere) als genau auf dieser Hüllenkante liegend definiert wird. Um nun eine gute Drehung zu finden, wird für alle Kanten der konvexen Hülle die Fläche des resultierenden Rechtecks berechnet. Die Drehung, die zur geringsten Fläche führt, wird übernommen.

2.4.1 Konvexe Hülle

Ein wichtiger Teil des *Component Splitter Layout* ist die Berechnung der konvexen Hülle der jeweiligen Zusammenhangskomponente. Die konvexe Hülle einer Punktmenge zu berechnen ist an sich keine schwierige Aufgabe. Wenn die Größe der Punktmenge jedoch im Bereich von zehntausenden Punkten liegt, wird die asymptotische Laufzeit wichtig. Die

bisher besten Verfahren erreichen eine Laufzeit von $O(n \cdot log(h))$ wobei n die Anzahl der Punkte und h die Anzahl der Punkte auf der Hülle bezeichnet.

Definition 7 (Konvexe Hülle). Die Konvexe Hülle einer Punktmenge P ist das kleinste Polygon, das alle Punkte aus P, und alle Punkte die auf allen möglichen Verbindungslinien zwischen zwei Punkten aus P liegen, enthält.

Das ausgewählte randomisierte Verfahren von Wenger mit dem Namen *Randomized Quick Hull* (RQH) [Wen97] hat eine erwartete Laufzeit von $O(n \cdot log(h))$. Da guter Pseudocode vorlag, war die Implementierung zunächst kein Problem. Die Berücksichtigung aller möglichen auftretenden Sonderfälle hat jedoch mehr Arbeit gemacht als man erwarten würde, wenn man einen bewährten Algorithmus umsetzt.

Da die für *Randomized Quick Hull* implementierten Hilfsfunktionen es leicht ermöglichten, wurde zudem eine Heuristik implementiert, die dem restlichen Hüllalgorithmus einen günstigen Start mit deutlich kleinerer Eingabegröße verschaffen soll. Diese wurde von Devroye und Toussaint beschrieben und für alle Hüllalgorithmen empfohlen [DeTo81].

Randomized Quick Hull

Der Algorithmus *Randomized Quick Hull* verfolgt einen Divide & Conquer-Ansatz. Wie bei vielen anderen Hüllenalgorithmen wird nur eine Halbhülle berechnet, also maximal der Teil der Hülle, der aus einer bestimmten Richtung sichtbar ist. Die Eingabe für den Algorithmus besteht erstens aus zwei bereits bekannten Punkten auf der konvexen Hülle, die die äußeren Grenzen des momentan betrachteten Hüllenabschnitts definieren. Zweitens werden als Restpunktmenge die von diesem Abschnitt zu umschließenden Punkte angegeben. Drittens ist eine Referenz auf die Liste der bisher ausgegeben Punkte nützlich, damit die neuen Punkte im Ablauf der Rekursion in der richtigen Reihenfolge eingefügt werden können. Alle weiteren in diesem Hüllenabschnitt nötigen Punkte müssen in dieser Punktmenge enthalten sein. In einem Schritt wird nur ein weiterer Hüllenpunkt gefunden, das Problem wird an dieser Stelle dann in zwei Teilprobleme aufgeteilt. Die ersten Punkte auf der Hülle werden durch den Preprocessing-Schritt geliefert, der im Abschnitt *Preprocessing für die konvexe Hülle* beschrieben wird. Auch manche denkbaren Sonderfälle werden dadurch ausgeschlossen.

Zunächst werden zwei zufällige Punkte aus der Restpunktmenge gewählt. Hier sind mehrere Fälle möglich.

- Die Restpunktmenge enthält nur einen Punkt, der somit als Hüllenpunkt gewählt wird. Die Rekursion kann an dieser Stelle abgebrochen werden.

- Einer der beiden Punkte ist innerhalb des Dreiecks, das vom anderen Punkt und den beiden Hüllenpunkten aufgespannt wird. In diesem Fall wird der enthaltene Punkt

Algorithmus 2.1 RANDOMIZED QUICK HULL

Eingabe: Punkt *start*, Restpunkteliste L, Punkt *ende*, Referenz auf *Ausgabe*

Ausgabe: Hüllenpunkte zwischen *start* und *ende* werden in der Reihenfolge angehängt.

1: **repeat**
2: **if** $size(L) = 1$ **then** // Es gibt nur einen Punkt in L.
3: $Ausgabe \Leftarrow L$; **return** // L wird hinten an die Ausgabe angehängt.
4: **end if**
5: Punkte $q_1, q_2 \leftarrow random$ aus L
6: **if** $q_1 \in \triangle(start, q_2, ende)$ **then**
7: lösche q_1 aus L
8: **else if** $q_2 \in \triangle(start, q_1, ende)$ **then**
9: lösche q_2 aus L
10: **end if**
11: **until** q_1 und q_2 erfolgreich gewählt // Alle Punkte bearbeitet.
12: **if** q_2 weiter links als q_1 **then** vertausche q_1 und q_2 **endif**
13: $q \leftarrow$ Punkt aus L mit dem größten Abstand oberhalb von Gerade(q_1, q_2)
14: Listen $linksPunkte, rechtsPunkte \leftarrow \emptyset$
15: $polygon \leftarrow (start, q_1, q, q_2, ende)$
16: **while** $L \neq \emptyset$ **do**
17: Punkt $p \leftarrow random$ aus L
18: **if** $not\, p \in polygon$ **then**
19: $tempPolygon \leftarrow polygon + p$
20: **for** ein paar zufällige Punkte r aus L **do**
21: **if** $r \in tempPolygon$ **then**
22: lösche r aus L
23: **end if**
24: **end for**
25: **if** r gelöscht **or** $|tempPolygon| \leq |polygon|$ **then**
26: $polygon \leftarrow tempPolygon$
27: **end if**
28: **if** p ist oberhalb der Geraden $(start, q)$ **then**
29: $linksPunkte \leftarrow linksPunkte \cup p$
30: **else if** p ist oberhalb der Geraden $(q, ende)$ **then**
31: $rechtsPunkte \leftarrow rechtsPunkte \cup p$
32: **end if**
33: **end if**
34: lösche p aus L
35: **end while**
36: RANDOMIZEDQUICKHULL$(start, linksPunkte, q, Ausgabe)$;
37: $Ausgabe \Leftarrow q$ // q wird hinten an die Ausgabe angehängt.
38: RANDOMIZEDQUICKHULL$(q, rechtsPunkte, ende, Ausgabe)$;
39: **return**

gelöscht, da er nicht auf der konvexen Hülle liegen kann. Es werden nun zwei neue zufällige Punkte ausgewählt.

- Die beiden Punkte und die beiden Hüllenpunkte spannen ein konvexes Viereck auf. Die Suchrichtung nach dem Extrempunkt ist als senkrecht zur Geraden durch die beiden zufälligen Punkte festgelegt.

Das zufällige Festlegen der Suchrichtung verhindert, dass es Worst-Case-Verteilungen, für die die Punkte der Hülle der Reihe nach auf der Hülle gefunden werden, geben kann. Dies würde jeweils nur eine minimale Punktmenge am Rand abschneiden und die Laufzeit auf $O(n \cdot h)$ erhöhen.

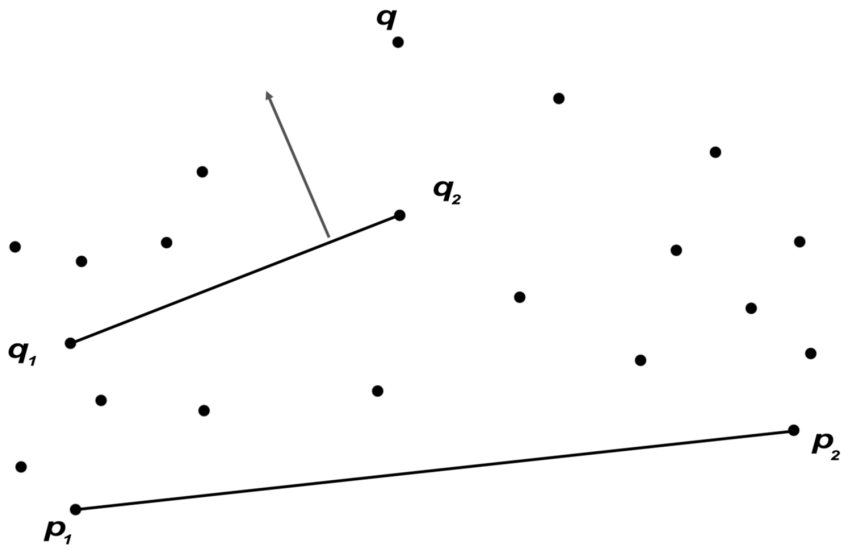

Abbildung 2.1: Extrempunkt q in der durch q_1 und q_2 festgelegten Suchrichtung.

Es wird nun der Punkt aus der Restpunktmenge gesucht, der den größten Abstand zur soeben berechneten Gerade hat und oberhalb davon liegt. Falls es mehrere solcher Punkte gibt, wird der am weitesten links liegende gewählt. Ob links oder rechts gewählt wird ist hier unwichtig, es sollte jedoch stets die gleiche Richtung sein. Diese zusätzliche Auswahl verhindert, dass mehr als zwei, von mehreren gleichzeitig auf einer Geraden liegenden Punkten, in die konvexe Hülle aufgenommen werden. Der so gefundene Extrempunkt ist auf jeden Fall Teil der konvexen Hülle.

Um die Anzahl der in weiteren Schritten zu berücksichtigenden Knoten zu reduzieren, wird ein weiterer randomisierter Schritt ausgeführt, der ebenfalls nur lineare Zeit benötigt. Alle Restpunkte werden einander zu zufälligen Paaren zugeordnet. Nun wird das Polygon betrachtet, das von den beiden anfangs gegebenen Hüllenpunkten, dem neu gefundenen Hüllenpunkt und einem der beiden zufälligen Punkte aufgespannt wird. Falls der jeweils andere der zufällig gezogenen Punkte in diesem Polygon enthalten ist, wird er gelöscht.

An dieser Stelle wurde der Algorithmus etwas modifiziert. Da bereits ein einfacher Algorithmus implementiert wurde, um zu testen ob ein Punkt in einem beliebigen Polygon enthalten ist, und ein gegebenes kleines konvexes Polygon recht leicht um einen außerhalb liegenden Punkt erweitert werden kann, wurde der Test für die zufällig gewählten Punkte verschärft. Gestartet wird nun mit der konvexen Hülle um die drei bekannten Hüllenpunkte und die beiden für die Richtungssuche genutzten Punkte. Immer wenn dieses Polygon im weiteren Verlauf um einen Punkt erweitert wird, wird stets dafür gesorgt, dass das Polygon konvex bleibt. Punkte auf dem Polygon, die einen Knick nach innen verursachen, werden also gelöscht.

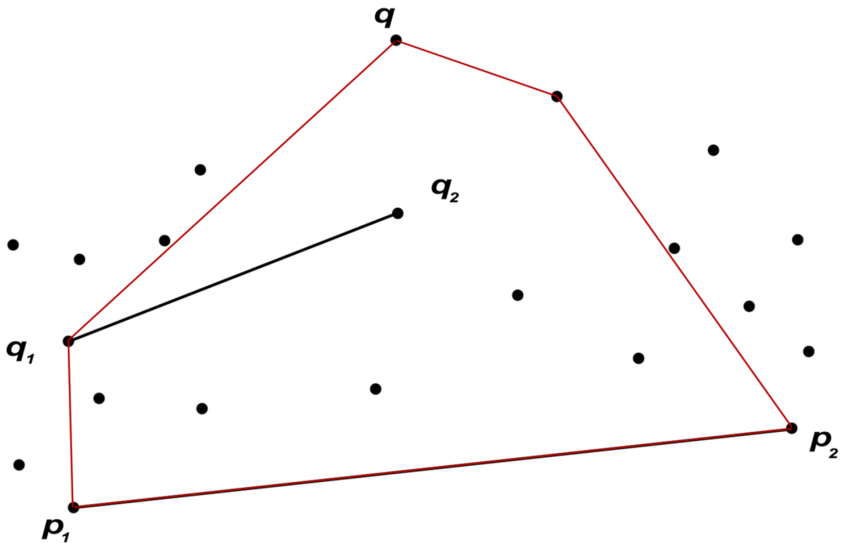

Abbildung 2.2: Beispiel für das Testpolygon.

Aus den Restpunkten wird nun stets ein zufälliger ausgewählt. Falls dieser vom Testpolygon umschlossen wird, wird er gelöscht, sonst wird er vorläufig zum Testpolygon hinzugenommen. Nun werden weitere Punkte zufällig gewählt und darauf getestet, ob sie im Polygon enthalten sind. Falls dadurch erfolgreich Punkte gelöscht werden konnten, dürfen mehr zufällige Punkte gewählt werden, ohne Erfolg gibt es nur zwei Versuche. Der vorläufig hinzu genommene Punkt wird eventuell dauerhaft im Testpolygon behalten. Dies geschieht, falls das Testpolygon dadurch dennoch mit gleich vielen oder weniger Eckpunkten auskommt. Der Punkt wird ebenfalls beibehalten, wenn in diesem Durchgang erfolgreich Punkte gelöscht wurden und die Größe des Polygons durch das Hinzufügen nicht über zehn Eckpunkte anwächst. Eine Grenze für die Größe des Polygons wurde eingeführt um keine Laufzeitprobleme durch zu aufwändige Tests zu erzeugen.

Alle Punkte, die bei diesen Tests nicht gelöscht wurden, liegen entweder oberhalb der Verbindungsgeraden zwischen dem ersten Hüllenpunkt und dem neu gefundenen Hüllenpunkt oder oberhalb der Verbindungsgeraden zwischen dem neuen Hüllenpunkt und dem

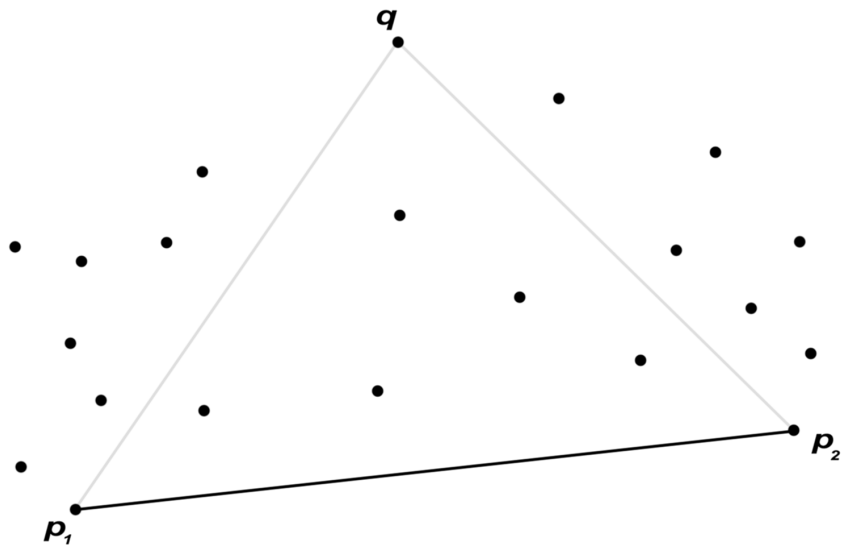

Abbildung 2.3: Aufteilung des Problems in zwei Teilprobleme an q.

zweiten zu Anfang bekannten Hüllenpunkt. Je nachdem in welcher Hälfte der Punkt liegt, wird er in eine Liste eingetragen. Die nächsten rekursiven Aufrufe von *Randomized Quick Hull* erhalten als Parameter dann den ersten beziehungsweise zweiten Hüllenpunkt, den neu gefundenen Hüllenpunkt und die jeweils dazugehörige Punktliste. Der Ablauf ist in Algorithmus 2.1 noch einmal übersichtlich dargestellt.

Preprocessing für die konvexe Hülle

Das von Devroye und Toussaint empfohlene Preprocessing hat das Potential, die Eingabegröße für einen Hüllenalgorithmus deutlich zu verkleinern [DeTo81]. Da die Laufzeit des Preprocessings linear ist, wird die asymptotische Gesamtlaufzeit der Berechnung der konvexen Hülle nicht negativ beeinflusst. Es wird ein konvexes Polygon berechnet, dessen Eckpunkte bereits gültige Punkte der konvexen Hülle sind. Alle Punkte im Inneren dieses Polygons müssen nicht mehr betrachtet werden. Zudem wird eine Zerlegung des Problems in bis zu acht Teilprobleme geliefert.

Die Eckpunkte des Polygons erhält man, indem man zunächst die Extrempunkte in acht verschiedenen Richtungen berechnet. Die Richtungen entsprechen der X- und Y-Achse sowie den beiden Diagonalen. Für jede dieser vier Achsen werden die beiden Punkte berechnet, die entlang dieser Achsen die höchste beziehungsweise niedrigste Koordinate haben. Sollten einige der Punkte mehrfach vorhanden sein, werden sie nur einmal beachtet. Aus diesen Punkten wird nun ein konvexes Polygon mit bis zu acht Eckpunkten gebildet. Betrachtet man nur eine Kante des Polygons, so trennt diese einen Teil der Eingabemenge vom Rest. Da alle Ecken des Polygons bereits gültige Hüllenpunkte sind, gilt das auch für die Endpunkte der Kante.

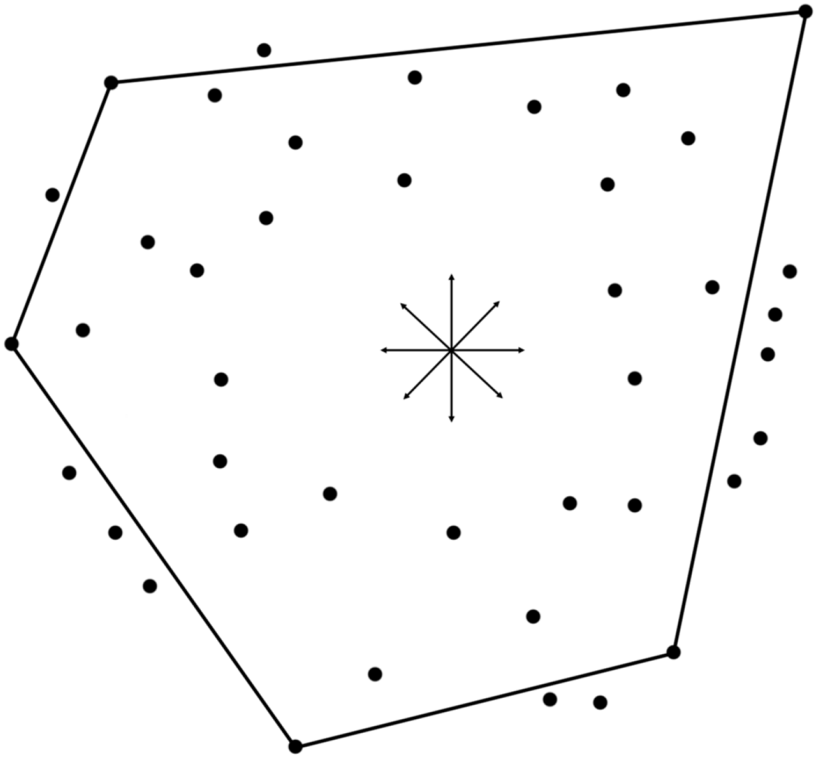

Abbildung 2.4: Die acht Richtungen und die gefundenen Extrempunkte.

Es kann nun einfach *Randomized Quick Hull* aufgerufen werden, wobei die beiden Endpunkte der Kante und die abgetrennte Punktmenge die Parameter sind. Wird für alle Kanten des Polygons der Hüllenalgorithmus aufgerufen, müssen am Ende nur noch die Ergebnisse der RQH-Aufrufe und die Eckpunkte des Polygons in die richtige Reihenfolge gebracht werden. Dies ergibt sich bei geschickter Implementierung jedoch ohne weiteren Aufwand.

Es besteht nun die Hoffnung, dass die Kanten des entstandenen Polygons kleine Teile des Graphen abtrennen und so die Eingabegröße für RQH stark reduziert werden kann. Zudem wird der Graph bereits zu Beginn in bis zu acht Teilprobleme aufgeteilt statt in zwei beim normalen Start von *Randomized Quick Hull*. Das Preprocessing wirkt besonders gut, wenn die Punkte auf der Fläche nahezu gleichverteilt sind [DeTo81]. Dies stellt im Kontext des automatischen Zeichnens von Graphen eine der angestrebten Eigenschaften einer Zeichnung dar, sollte also zumindest tendenziell auf die fertigen Zeichnungen zutreffen, die auf der Zeichenfläche angeordnet werden sollen.

2.4.2 Packen der Komponenten

Nachdem für alle Komponenten mit Hilfe der konvexen Hülle die beste Drehung und die umschließenden Rechtecke gefunden wurden, müssen diese Rechtecke noch möglichst platzsparend auf der Zeichenfläche angeordnet werden. Dies geschieht mit Hilfe des *TileToRowsCCPacker* der im OGDF bereits implementiert war. Das Seitenverhältnis der gewünschten Zeichenfläche kann im *TileToRowsCCPacker* angegeben werden, was es vereinfachte, diesen Parameter auch im *Component Splitter Layout* zur Verfügung zu stellen.

2.5 Skalierung

Eine Skalierung des gesamten Graphen zwischen den Kräftephasen kann große Auswirkungen auf die Laufzeit des Algorithmus und die Qualität der Zeichnung haben. Da in den kleinsten Verfeinerungsstufen ein Knoten eine große Anzahl von Knoten des Originalgraphens repräsentiert, wird die entstehende Zeichnung der eigentlichen Größe des Graphen nicht gerecht. Da sich von einer Verfeinerungsstufe zur nächstgrößeren abhängig vom Verfeinerungsverfahren die Anzahl der Knoten verdoppeln kann, wird auch die benötigte Zeichenfläche entsprechend wachsen. Diese Vergrößerung müsste das Kräfteverfahren nun leisten, indem es alle Knoten nach außen verschiebt. Dies kostet viele Iterationen und kann zu unerwünschten Faltungen führen, wenn Teile des Graphen übereinander gedrückt werden. Durch eine explizite Skalierung vor dem nächsten Aufruf des Kräfteverfahrens kann diese Arbeit vorweggenommen werden.

Das Ergebnis ist ein schnellerer Algorithmus, der keine zusätzlichen Faltungen im Graphen verursacht. Durch etwas stärkere Skalierung kann zudem eine Spannung im Graphen aufgebaut werden, da dieser sich nun zusammenziehen muss. Dies kann bereits bestehende Faltungen im Graphen bekämpfen und so die Zeichnungsqualität weiter erhöhen. Eine zu starke Skalierung kann den Graphen jedoch stark verzerren, was zu einer schlechteren Qualität und Laufzeit führt.

2.5.1 Skalierung in mehreren Schritten

Es gibt also bei der Größe der Skalierung sich widersprechende Wünsche. Einerseits soll eine möglichst große Skalierung zur Entfaltung des Graphen beitragen. Andererseits möchte man nahe der gewünschten Kantenlänge bleiben um am Ende des Verfahrens diese Kantenlänge auch zu erreichen. Zudem wirkt sich eine zu starke Abweichung von der idealen Kantenlänge negativ auf die Laufzeit aus, insbesondere wenn die Abweichungen sich von Verfeinerungsstufe zu Verfeinerungsstufe verstärken. Im Laufe des kräftebasierten Verfahrens wird viel Arbeit investiert, um diesen Fehler so gut es geht auszugleichen.

Ein Ansatz, diese beiden Forderungen besser zu erfüllen, ist, pro Kräftephase mehrere Skalierungsschritte auszuführen. Zunächst wird der Graph sehr groß skaliert, um mög-

lichst viele Faltungen zu entfernen. Danach wird eine Skalierung gewählt, die nahe an der durchschnittlichen gewünschten Kantenlänge liegt, um zu verhindern, dass der Graph zu stark verzerrt wird. Nach jedem Skalierungsschritt wird natürlich das Kräfteverfahren aufgerufen.

Um die Skalierung komfortabel zu ermöglichen, wurde auch das Skalieren als eigenes *ogdf::LayoutModule* implementiert. Als Parameter werden Anfangs- und Endskalierung, die Anzahl der Skalierungsschritte und das zwischendurch aufzurufende Kräfteverfahren übergeben. Diese Skalierung kann ein kräftebasiertes Zeichenverfahren bereits verbessern, da es die Entfaltungsfähigkeit erhöht. Je mehr Schritte zwischen der maximalen und der minimalen Skalierung durchgeführt werden, desto höher steigt natürlich die Laufzeit, was den erwünschten Laufzeitvorteil wieder zerstören kann. Je nachdem ob die Laufzeit oder die Qualität für die aktuelle Anwendung wichtiger ist, werden hier andere Parameter am günstigsten sein.

2.6 Postprocessing

Ziel des Postprocessings ist es, das Bild lokal zu verbessern. Obwohl während des vorherigen Verfahrens die Gesamtstruktur so gut es geht gezeichnet wurde, können insbesondere im lokalen Bereich noch „Schönheitsfehler“ auftreten. Dies kann unter anderem durch die extrem weitreichenden abstoßenden Kräfte ausgelöst werden, die ihren Zweck eher in der Entfaltung des Graphen haben. Die lokalen Änderungen werden das Gesamtbild kaum noch beeinflussen.

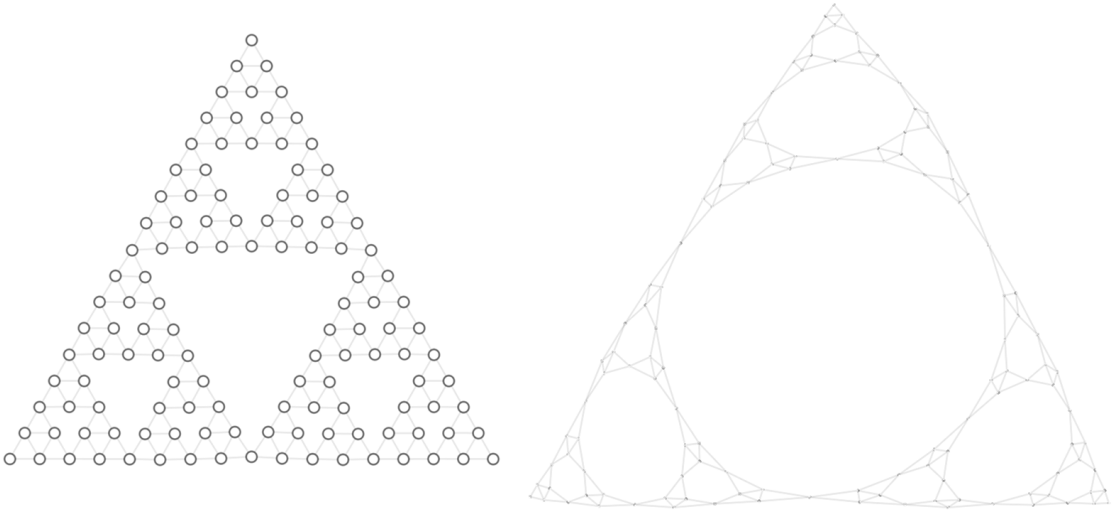

Abbildung 2.5: Der Graph Sierpinski_04.gml, mit und ohne Postprocessing gezeichnet.

Als Zeichenverfahren empfiehlt sich hier natürlich eines, das nur lokale Änderungen vornimmt bzw. deren Änderungen nur durch lokale Gegebenheiten gesteuert sind. In Frage

kommt zum Beispiel ein kräftebasiertes Verfahren welches nur die Kantenkräfte simuliert oder nur in der lokalen Umgebung abstoßende Kräfte erzeugt. Für das Postprocessing sind üblicherweise viele Iterationen notwendig. Bei zu vielen Iterationen kann ein reines Zusammenziehen der Kanten den Graphen zu stark verzerren, so dass es nützlich sein kann, das Postprocessing mit Hilfe menschlicher Aufsicht rechtzeitig abzubrechen.

Das Postprocessing wirkt besonders gut und schnell, wenn die durchschnittliche Kantenlänge in der Zeichnung bereits nah an der durchschnittlichen gewünschten Kantenlänge ist. Dies kann durch eine Skalierung erreicht werden. Eine weitere Möglichkeit die Zeichnung lokal möglichst schön zu halten ist, das Postprocessing mit wenigen Iterationen am Ende jeder Kräftephase aufzurufen. Dies kann auch die für das abschließende Postprocessing nötige Zeit deutlich verringern.

Kapitel 3

Verfügbare kräftebasierte Verfahren

Wie erwähnt sind nicht alle in den existierenden Multilevel-Verfahren genutzten kräftebasierten Verfahren im OGDF verfügbar. Auch die vorhandenen Algorithmen unterscheiden sich meist von den in anderen Verfahren genutzten, da diese oft an das Multilevel-Verfahren angepasst wurden. Kräftebasierte Verfahren zeichnen sich zudem meist durch eine enorme Anzahl von Parametern aus, die ihr Verhalten beeinflussen. Es kann also selbst in den günstigsten Fällen nicht erwartet werden, dass hier die gleichen Ergebnisse und der gleiche algorithmische Ablauf reproduziert werden, wie sie in den Referenzimplementierungen der jeweiligen Autoren vorzufinden wären. Die verfügbaren Kräfteverfahren liefern dennoch eine große Bandbreite an Variationen. Die Parameter wurden jeweils für den *Modular Multilevel Mixer* angepasst, wobei manchmal mehrere Varianten nützlich erschienen, die dann einzeln getestet wurden.

Im Grunde können alle *ogdf::LayoutModule* als Zeichenverfahren in der Kräftephase eingesetzt werden. Sinnvoll ist das jedoch nur, wenn das Zeichenverfahren inkrementell arbeitet. Die Positionen aus der vorherigen Verfeinerungsstufe müssen also nutzbringend ausgewertet werden. Dies geschieht bei den kräftebasierten Verfahren, indem vorwiegend lokale Änderungen durchgeführt werden. Das Verfahren von Davidson und Harel [DaHa96] entspricht nicht ganz einem klassischen kräftebasierten Ansatz, gewinnt aber dennoch in der Zeichnungsqualität wenn es mit einem Multilevel-Ansatz kombiniert wird, während *GEM* [FLM98] die bestehenden Positionen stärker ignoriert als die anderen Verfahren.

3.1 Algorithmus von Fruchtermann und Reingold (FR)

Das am weitesten verbreitete und in vielen Multilevel-Verfahren genutzte kräftebasierte Verfahren von Fruchtermann und Reingold [FrRe91] wurde auch im Rahmen des OGDF implementiert.

In diesem Verfahren wird ein Raster genutzt, um nicht alle abstoßenden Kräfte berechnen zu müssen und so den Algorithmus zu beschleunigen. Ein Knoten wird nur von anderen Knoten abgestoßen die sich im gleichen Sektor des Rasters oder einem direkt oder diagonal benachbarten befinden. Die Zeichnungen sind meist deutlich kompakter und weisen eine gleichmäßigere Knotenverteilung auf als viele andere kräftebasierte Verfahren.

Das im OGDF implementierte Verfahren nutzt außerdem eine eigene automatische Skalierung. Im experimentellen Vergleich wurden zwei Varianten dieses Algorithmus getestet, wobei in der einen Variante die automatische Skalierung deaktiviert wurde. Die Ergebnisse zeigen, dass die automatische Skalierung bessere Ergebnisse liefert als die im *Modular Multilevel Mixer* eingesetzten Skalierungen.

3.2 Fruchtermann Reingold FM^3-Variante (FR2)

In *FM*^{3} wurde ebenfalls das kräftebasierte Verfahren von Fruchtermann und Reingold implementiert [FrRe91]. Da durch die Einbettung in *FM*^{3} neue Möglichkeiten wie zum Beispiel die genauere Approximation der abstoßenden Kräfte eröffnet wurden, ist *FR2* als eigenständiges zu *FR* abgrenzbares Zeichenverfahren zu betrachten.

Beim Einsatz der *FR2* im *Modular Multilevel Mixer* entsteht ein Zusatzaufwand, da das Verfahren in *FM*^{3} integriert wurde und nur indirekt aufgerufen werden kann, ohne eine teilweise Neuimplementierung notwendig zu machen. Dies erhöht die Laufzeit aller Algorithmen, die dieses Verfahren in der Kräftephase nutzen, minimal. Die Erhöhung der Laufzeit fällt im Rahmen der hier präsentierten Versuche jedoch nicht ins Gewicht.

Die ursprüngliche Möglichkeit, ein Raster statt des Quadtrees für die Approximation der abstoßenden Kräfte zu nutzen, ist hier optional noch möglich. Beide Varianten wurden in den Versuchen erprobt, wobei sich hier die Variante mit Quadtree als überlegen herausgestellt hat.

3.3 Verfahren von Eades aus FM^3 (EADES)

Auch das kräftebasierte Verfahren von Eades wurde in *FM*^{3} implementiert [Ead84]. Das Kräftemodell entspricht etwa dem von *FR*. Dieses Verfahren profitiert noch mehr als *FR2* durch die von *FM*^{3} ermöglichte schnelle Approximation der abstoßenden Kräfte, die so im ursprünglichen Verfahren nicht vorgesehen war. Ohne diese Verbesserung wäre die Laufzeit des Verfahrens für große Graphen nicht akzeptabel.

Obwohl es sich um das mit Abstand älteste hier betrachtete Verfahren handelt, können die Ergebnisse im Kontext eines Multilevel-Verfahrens durchaus neben den neueren bestehen. Wie bei *FR2* wird dieses Verfahren indirekt über *FM*^{3} aufgerufen.

3.4 New Multipole Method (NMM)

Die *New Multipole Method* ist das in *FM*^{3} hauptsächlich genutzte kräftebasierte Verfahren und wurde von Hachul eigens für sein Multilevel-Verfahren entwickelt [Hac05]. Die Besonderheit der *New Multipole Method* ist die garantierte Laufzeit von $O(n \cdot log(n))$. Diese wird erreicht, indem die weitreichenden abstoßenden Kräfte zwischen den Knoten mit Hilfe eines Quadtree approximiert werden. Hierbei ist wichtig, dass der Quadtree auch schnell genug aufgebaut werden kann, was Hachul in seiner Arbeit ebenfalls beweist. Wie bei den beiden zuvor beschriebenen Verfahren wird auch die *New Multipole Method* indirekt über *FM*^{3} aufgerufen.

3.5 Fast Multipole Embedder (FME)

Der *Fast Multipole Embedder* wurde im Rahmen von Martin Gronemanns Multilevel-Verfahren *Fast Multilevel Multipole Embedder* [Gro09] erstellt und stellt im Grunde eine beschleunigte und auf parallele Berechnung ausgelegte Variante von Hachuls *New Multipole Method* [Hac05] dar.

Der *Fast Multipole Embedder* zeichnet sich insbesondere durch seine hohe Geschwindigkeit bei extrem großen Graphen auch im Vergleich zu *NMM* aus und kommt in einigen der besten Modulkombinationen zum Einsatz. Die Anzahl der Kreuzungen ist mit dem *Fast Multipole Embedder* ebenfalls oft deutlich geringer, als mit anderen Verfahren. Eine Schwäche des Verfahrens sind jedoch die Kantenlängen. Im Gegensatz zu den meisten anderen Verfahren kann der *Fast Multipole Embedder* die gewünschte Kantenlänge nur selten erreichen.

3.6 Fast Edges Only Embedder (FEOE)

Mit dem FEOE steht ein besonders simples Verfahren zur Verfügung. Es handelt sich hierbei um eine FME-Variante, die nur die Kantenkräfte berechnet und im *Fast Multipole Embedder* nur für kleinere Verbesserungen eingesetzt wird. Die Änderung zum *Fast Multipole Embedder* besteht nur in einem übergebenen Parameter, was das Verhalten jedoch drastisch ändert.

Wenn das Verfahren in der Kräftephase als Zeichenverfahren eingesetzt wird, muss die Entfaltung der Zeichnung komplett durch das Multilevel-Verfahren, also Skalierung und Platzierungsstrategie geleistet werden. Dies hat sich im Vergleich zu den anderen Verfahren als nicht konkurrenzfähig herausgestellt. Insbesondere bei dichten Graphen lassen die Ergebnisse sehr zu wünschen übrig, da sich im Vergleich mit anderen Verfahren besonders viele Knoten überlappen.

Der *Fast Edges Only Embedder* ist sehr schnell aber liefert meist nur im lokalen Bereich annehmbare Ergebnisse. Er ist damit für einen Einsatz im Postprocessing geeignet.

3.7 Verfahren von Davidson & Harel (DH)

Das Verfahren von Davidson und Harel [DaHa96] ist kein kräftebasiertes Verfahren im eigentlichen Sinne, da auch Knotenbewegungen möglich sind, die weder durch Richtung noch Betrag mit einer tatsächlich auf diesen Knoten wirkenden Kraft zusammenhängen.

Die zugrunde liegende Idee basiert auf der randomisierten Suchheuristik *Simulated Annealing*. Dementsprechend gibt es eine als Temperatur interpretierbare Variable, deren Wert im Verlauf des Algorithmus langsam absinkt. Je höher diese Temperatur ist, desto weiter bewegen sich die Knoten in eine zufällige Richtung. Da diese Bewegungen auch eine Verschlechterung der Zeichnung bewirken können, werden diese nicht immer akzeptiert. Um die Güte der Position eines Knoten zu bewerten existiert eine komplexe und an die Anforderungen detailliert anpassbare Funktion. Diese wird jedes mal aufgerufen wenn die Güte einer Knotenposition bewertet werden soll.

Eine Bewegung die den Graphen verbessert, wird immer akzeptiert. Eine, die den Graphen verschlechtert, jedoch nur mit einer gewissen Wahrscheinlichkeit, die umso geringer ist, je stärker die Verschlechterung, aber umso höher, je größer die globale Temperatur ist.

Der Algorithmus von Davidson und Harel ist bereits als eigenständiges Verfahren extrem langsam, was sich innerhalb des Multilevel-Verfahrens nicht verbessert. Zudem ist der Speicheraufwand zu extrem, um die größeren Graphen zu zeichnen. Wenn man ausreichend Zeit investiert, kann auch das Verfahren von Davidson und Harel durch einen Multilevel-Ansatz an Qualität gewinnen.

3.8 Graph Embedder (GEM)

Der Graph Embedder von Frick, Ludwig und Mehldau [FLM98] ist als eigenständiges Verfahren ohne Multilevel-Ansatz das stärkste hier zur Verfügung stehende. Auch hier wird die vorhandene Implementierung aus dem OGDF genutzt.

Da das Verfahren weniger Informationen von den initialen Positionen der Knoten übernimmt, kann es bei extrem großen Graphen leider nur wenig von einem Multilevel-Ansatz profitieren. Es wäre interessant, eine Variante von *GEM* zu erstellen, die mehr Informationen aus der ursprünglichen Platzierung übernimmt, um einen erfolgreicheren Einsatz im Rahmen eines Multilevel-Verfahrens zu ermöglichen. Für kleinere Graphen sind die Ergebnisse von *GEM* im Rahmen des Multilevel-Ansatzes sehr gut.

3.9 Mixed Force Layout (MFL)

Es wurde im Laufe der Implementierung beobachtet, dass der *Fast Multipole Embedder* [Gro09] sich durch eine hohe Geschwindigkeit und gute Fähigkeiten bei der Entfaltung des Graphen auszeichnet. Das Verfahren von Fruchtermann und Reingold [FrRe91] konnte besonders gute lokale Qualität liefern. Es besteht also die Hoffnung, diese Eigenschaften zu kombinieren indem zuerst die Entfaltung durch den *Fast Multipole Embedder* und dann die lokale Qualität durch das Verfahren von Fruchtermann und Reingold verbessert wird.

Selbst wenn diese Verbesserungen eintreten, werden sie vermutlich durch erhöhte Laufzeit erkauft sein, da jedes mal zwei Verfahren aufgerufen werden.

Kapitel 4

Zerlegungsstrategien

Das Hauptaugenmerk liegt in dieser Arbeit auf den Zerlegungsstrategien, die eingesetzt werden um die Verfeinerungsstufen für das Multilevel-Verfahren zu generieren. Die verschiedenen existierenden Strategien wurden bereits mit den Multilevel-Verfahren dargelegt. Die in dieser Arbeit umgesetzten Verfahren werden hier noch einmal im Detail beschrieben und ihre Besonderheiten hervorgehoben.

Alle in dieser Arbeit betrachteten Verfahren lassen sich darauf reduzieren, dass Knotenpaare verschmolzen werden. Einer der Knoten wird als „Elternknoten“, der andere als „Verschmelzungspartner“ festgelegt. Der Elternknoten bleibt erhalten und steht von nun an auch für den Verschmelzungspartner und alle von ihm repräsentierten Knoten. Während der Verschmelzung werden alle Kanten, die am Verschmelzungspartner starten oder enden, zum Elternknoten umgeleitet. Dadurch eventuell entstehende doppelte Kanten oder Schleifen werden gelöscht. Bei doppelten Kanten wird der Durchschnitt der gewünschten Kantenlängen gebildet und für die übrig gebliebene Kante übernommen, um beiden Kanten gerecht zu werden. Diese Änderungen werden im *Multilevel Graph* gespeichert und können einzeln rückgängig gemacht werden. Von Details abgesehen entsprechen die Zerlegungsstrategien dem dargestellten Algorithmus 4.1 und unterscheiden sich nur in den Paaren, die jeweils zum Verschmelzen ausgewählt werden.

Bei Knoten mit sehr hohem Grad kann ein negativer Effekt auftreten, der die Speicherplatzkomplexität gefährdet. Falls ein Knoten mit hohem Grad als Verschmelzungspartner gewählt wird, müssen alle Kanten zum Elternknoten verschoben werden. Diese Verschiebung muss gespeichert werden damit sie später rückgängig gemacht werden kann. Aufgetreten ist das Problem bei den vom *Matching Merger* abgeleiteten Strategien *Edge Cover Merger* und *Local Biconnected Merger*. Diese sind besonders gefährdet, weil die Verschmelzungen anhand der Kanten ausgewählt werden und gleichzeitig verlangt wird, dass eine bestimmte Anzahl von Verschmelzungen durchgeführt wird. Wenn sehr viele der Kanten am gleichen Knoten enden ist unvermeidlich, dass dieser Knoten in fast jeder

Algorithmus 4.1 ZERLEGUNGSSTRATEGIE

Eingabe: Graph G

Ausgabe: Graphen G_0 ... G_n, Knotenzuordnung Q

```
1:  G_0 ← G
2:  Q ← ∅
3:  i ← 1
4:  repeat
5:    Pairs ← selectPairsFrom(G_{i-1}) // Zu verschmelzende Knotenpaare auswählen
6:    for all (a, b) ∈ Pairs do
7:      moveEdges(a, b) // verschiebt die Kanten von b zu a
8:      if a ∉ G_i then
9:        G_i ⇐ a // a einfügen
10:     end if
11:     if b ∈ G_i then
12:       G_i ⇒ b // b löschen
13:       Pairs ⇒ (b, *) // Nicht mehr mögliche Vermelzungen verhindern
14:     end if
15:     Q ⇐ (a, b) // Zuordnung speichern
16:   end for
17:   i ← i + 1
18: until G_i.size() ≤ 3
```

Verfeinerungsstufe sehr oft modifiziert wird. Das Problem kann jedoch leicht vermieden werden, indem stets der Knoten mit höherem Knotengrad als Elternknoten gewählt wird.

Die verschiedenen Zerlegungsstrategien können im *Modular Multilevel Mixer* beliebig eingesetzt werden und sind mit allen kräftebasierten Verfahren und allen Platzierungsstrategien nutzbar. Manche Kombinationen sind jedoch speziell aufeinander abgestimmt und erzeugen so besonders gute Ergebnisse.

4.1 Random Merger (RM)

Als einfachstes Verfahren zur Erzeugung der Vereinfachungsstufen wurde der *Random Merger* implementiert. Es wird per Zufall ein Knoten ausgewählt und mit einem zufälligen Nachbarn verschmolzen, bis der Graph um einen einstellbaren Faktor geschrumpft ist. Der größte Vorteil dieses Verfahrens ist, dass die Anzahl der Verfeinerungsstufen im Algorithmus gut gesteuert werden kann. Dies wäre auch bei den meisten anderen Zerlegungsstrategien möglich, führt bei diesen jedoch zu einem gewissen Verlust der Gleichmäßigkeit. Nachteilig wirkt sich aus, dass die Verschmelzungen nicht gleichmäßig über den Graphen verteilt sind, was die Zeichnungsqualität abhängig vom Zufall stark schwanken

lässt. Eine komplexere Zerlegungsstrategie wird sich stets zumindest gegen den *Random Merger* behaupten müssen. Der *Random Merger* setzt also eine Mindestanforderung an die Qualität nützlicher Zerlegungsstrategien.

4.2 Matching Merger (MM)

Der *Matching Merger* wurde entsprechend der Multilevel-Strategie aus Walshaws Verfahren implementiert [Wal03]. Um den nächst einfacheren Graphen zu erhalten wird zunächst folgendermaßen eine Paarung (Matching) berechnet: Es wird aus den noch verfügbaren Knoten zufällig einer ausgewählt und ist somit im weiteren Verlauf nicht mehr wählbar. Aus allen noch wählbaren Nachbarn wird ein weiterer zufällig ausgewählt und kann ebenfalls in späteren Schritten nicht mehr gewählt werden. Die Kante zwischen den beiden Knoten wird zum Matching hinzugenommen. Diese Schritte werden wiederholt, bis keine Knoten mehr wählbar sind. Das Matching kann nun nicht mehr durch einfaches Hinzunehmen von Kanten vergrößert werden. Alternativ kann die Auswahl des Nachbarn davon abhängig gemacht werden, wie viele Knoten aus vorherigen Verfeinerungsstufen durch den Nachbarn repräsentiert werden. Dies wird hier auch als die Masse des Knotens bezeichnet. Es wird nun stets der Nachbar mit der kleinsten Masse ausgewählt. Falls dafür mehrere Knoten in Frage kommen, wird wieder zufällig ausgewählt. Walshaw erwähnt, dass die resultierenden Zeichnungen in der zweiten Variante qualitativ besser sind [Wal03]. Die Implementation des *Matching Merger* stellt beide Varianten bereit.

Nachdem das Matching berechnet wurde, können die zusammengehörenden Knoten verschmolzen werden. Welcher der beiden Knoten als Repräsentant im nächsten Verfeinerungsschritt erhalten bleibt, wird ebenfalls zufällig bestimmt. Der *Matching Merger* zeichnet sich dadurch aus, dass die Verschmelzungen besonders gleichmäßig passieren.

Da stets nur eine Matching-Kante an einem Knoten gewählt werden kann, wird der *Matching Merger* stets mindestens dem höchsten im Graphen vorkommenden Knotengrad entsprechend viele Verfeinerungsstufen erzeugen. Dies führt bei hohem Knotengrad dazu, dass die Anzahl der Stufen linear zu der Anzahl der Knoten ist, was für die Gesamtlaufzeit des Algorithmus sehr negative Auswirkungen hat.

4.3 Solar Merger (SM)

Der *Solar Merger* berechnet die Vereinfachungsstufen wie in Hachuls *FM*^{3} [Hac05] nach dem Modell von Sonnensystemen. Die Knoten des Graphen werden dazu in Sonnen, Planeten und Monde eingeteilt. Dabei gilt stets, dass alle zu einer Sonne benachbarten Knoten zu Planeten werden. Ein Planet darf zudem nur zu einer einzigen Sonne benachbart sein. Dadurch ergibt sich, dass Sonnen graphentheoretisch stets mindestens einen Abstand von Drei haben, die nächste Sonne jedoch höchstens einen Abstand von Fünf. Möglicherweise

gibt es im Graph Knoten, die weder Planet noch Sonne werden können. Solche Knoten werden als Mond definiert und sind stets zu einem Planeten benachbart. Um die nächste Verfeinerungsstufe zu erreichen, werden alle zu einem Sonnensystem gehörigen Planeten und Monde mit der Sonne verschmolzen. Dadurch, dass alle Knoten in der Nachbarschaft in der nächsten Verfeinerungsstufe mit der Sonne verschmolzen werden, können insbesondere für sehr dichte Graphen nur wenige Verfeinerungsstufen erstellt werden. Dies kann sich nachteilig auf die Zeichenqualität auswirken, da so kaum vom Multilevel-Ansatz Gebrauch gemacht wird.

Um mit Platzierungsstrategien, welche die durch den *Solar Merger* generierten Zusatzinformationen nicht ausnutzen können, dennoch möglichst gute Ergebnisse zu erzielen, werden zuerst die Planeten und dann die Monde mit der Sonne verschmolzen. Dies sorgt dafür, dass in der Platzierungsphase zunächst die Monde wahrscheinlich etwa in der Mitte zwischen zwei Sonnen platziert werden. Erst danach werden die Planeten zwischen den Monden und ihren Sonnen platziert. Im umgekehrten Fall könnte die Position des Mondes bei der Platzierung der Planeten noch nicht berücksichtigt werden.

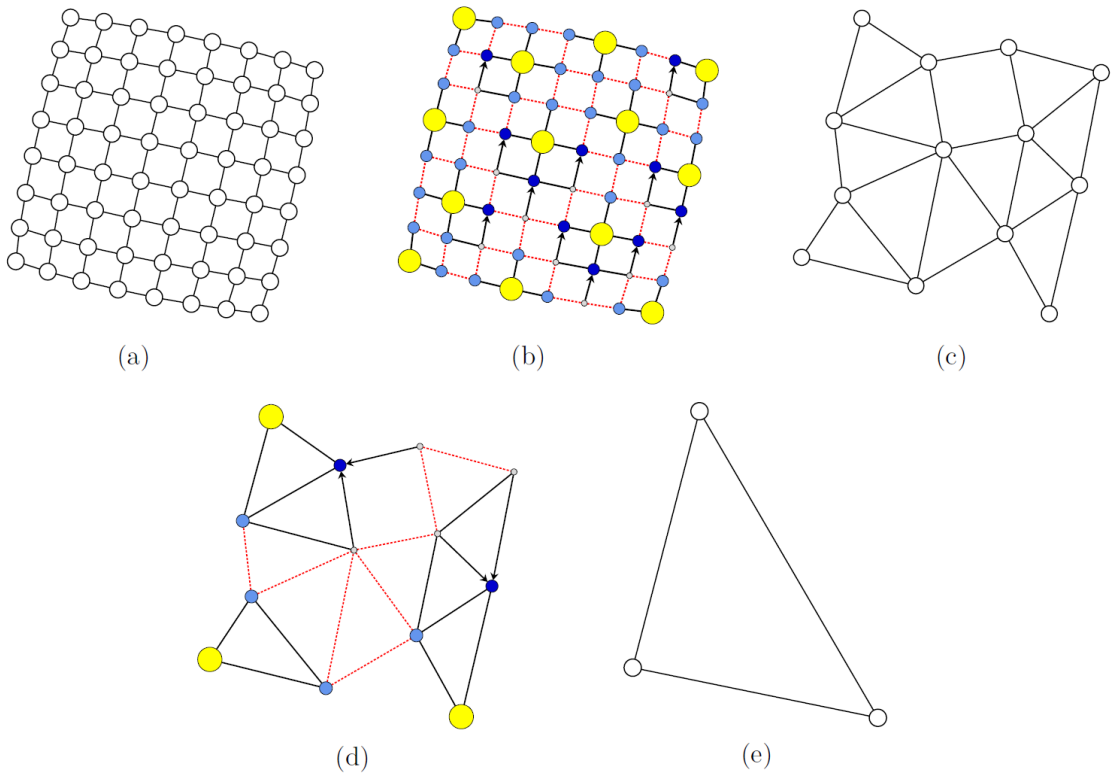

Abbildung 4.1: Entstehung der Verfeinerungsstufen beim Solar Merger. [Hac05]

Um festzulegen, welche Knoten des Graphen Sonnen werden, gibt es zwei Möglichkeiten: In der ersten Variante wird stets ein Knoten aus den restlichen ausgewählt. Falls der Knoten noch eine Sonne sein kann, wird er auch eine. Der zufällig ausgewählte Knoten

wird aus der Kandidatenmenge entfernt, unabhängig davon, ob er Sonne werden konnte oder nicht. Dies wird wiederholt, bis alle Knoten betrachtet wurden.

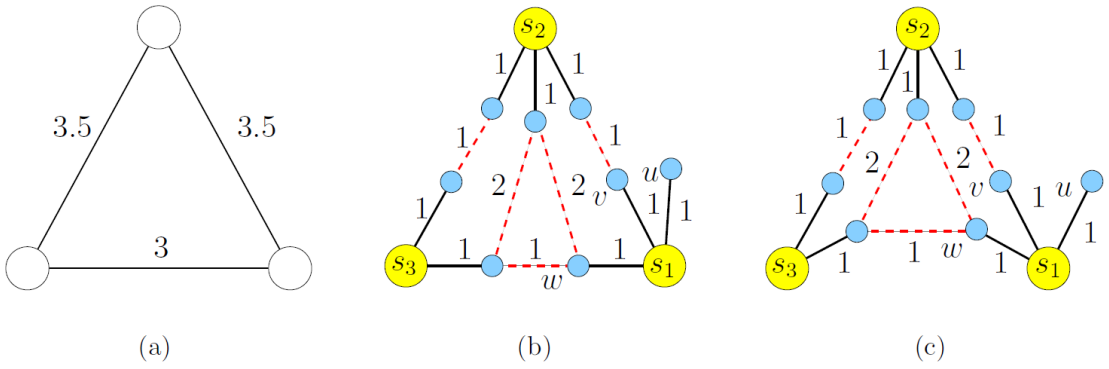

Abbildung 4.2: Intersystemkanten und Platzierung der Knoten. [Hac05]

Die zweite Variante ist etwas komplexer. Da ein Problem dieser Zerlegungsstrategie die oft zu geringe Anzahl von Verfeinerungsstufen ist, sollen nun Knoten ausgewählt werden, die möglichst wenige Knoten des Ursprungsgraphen repräsentieren. Dadurch kann die Anzahl der Vereinfachungsstufen eventuell erhöht werden. Jeder Knoten im originalen Graph erhält eine Masse von Eins. Die Masse eines Knotens entspricht der Summe der Massen der direkt von ihm repräsentierten Knoten und seiner ursprünglichen Masse. Die Masse einer Sonne entspricht also genau der Anzahl von Knoten aus dem ursprünglichen Graphen, die von ihm repräsentiert werden. Die in etwa zu erwartende Masse einer Sonne nach der Verschmelzung ist leicht zu berechnen, wenn angenommen wird, dass die Sonne keine Monde erhalten wird. Im Gegensatz zur ersten Variante werden nun zehn Knoten, die noch Sonne werden könnten, zufällig ausgewählt. Von diesen wird der Knoten mit der geringsten erwarteten Sonnenmasse ausgewählt und zur Sonne gemacht.

Definition 8 (Intersystemkante). Eine Kante, die zwischen Monden oder Planeten zweier unterschiedlicher Sonnen verläuft, wird *Intersystemkante* genannt.

Aus einer gegebenen *Intersystemkante* kann ein eindeutiger *Intersystempfad* konstruiert werden. Dies geschieht, indem von den beiden durch die Kante verbundenen Knoten ein Weg zu ihrer Sonne gesucht wird. Diese führt stets von einem Mond zu dem dazugehörigen Planeten und von einem Planeten direkt zu seiner Sonne. Es gibt mehrere *Intersystempfade* zwischen den gleichen zwei Sonnen, wenn es mehrere entsprechende *Intersystemkanten* gibt. In der nachfolgenden Verfeinerungsstufe werden alle zwischen zwei Sonnen bestehenden *Intersystempfade* gemeinsam durch eine einzige Kante repräsentiert. Diese hat dann als angestrebte Kantenlänge den Durchschnitt der Längen der *Intersystempfade*.

Durch den *SolarMerger* werden noch zusätzliche Informationen gespeichert, die später beim Platzieren der Knoten helfen sollen. Dazu werden ebenfalls die *Intersystempfade*

genutzt. Viele der mit einer Sonne verschmolzenen Planeten und Monde liegen vorher auf einem oder mehreren der *Intersystempfade*. Für diese Knoten wird gespeichert, wo sie auf dem Pfad in Relation zur Pfadlänge liegen. Es muss hier also nur die Zielsonne und eine relative Entfernung im Wertebereich von Null bis Eins für jede Zielsonne gespeichert werden. Liegt der Knoten auf mehreren *Intersystempfaden* zur gleichen Zielsonne, wird der Durchschnitt der relativen Entfernungen gespeichert. Die angestrebten Längen aller Kanten, aus denen ein *Intersystempfad* besteht, werden bei der Berechnung der Pfadlänge und der relativen Position berücksichtigt.

4.4 Independent Set Merger (ISM)

Der *Independent Set Merger* entspricht der Zerlegungsstrategie bei GRIP [GaKo02]. Dieses Verfahren ist vielversprechend, da es eine besonders gleichmäßige Verschmelzung ermöglicht und selbst die Repräsentanten der höchsten Verfeinerungsstufen in Hinblick auf den originalen Graphen sehr regelmäßig verteilt sind. Nachteilig wirkt sich jedoch aus, dass wie beim *Solar Merger* für dichte Graphen nur sehr wenige Verfeinerungsstufen erzeugt werden.

Definition 9 (Unabhängige Knotenmenge). Eine Knotenmenge heißt unabhängig, wenn keiner der enthaltenen Knoten mit einem anderen Knoten der Menge durch eine Kante verbunden ist.

Definition 10 (Maximale unabhängige Knotenmenge). Eine unabhängige Knotenmenge heißt maximal, wenn kein Knoten des Graphen hinzugenommen werden kann, ohne dass die Knotenmenge die Unabhängigkeit verliert.

Für die erste Verfeinerungsstufe wird eine maximale unabhängige Knotenmenge (maximal independet set) berechnet. Diese Menge wird erzeugt, indem aus der Menge der verfügbaren Knoten so lange zufällig ein von den bisher gewählten Knoten unabhäniger Knoten ausgewählt wird, bis dies nicht mehr möglich ist. Die so gewählten Knoten sind dann die Knoten der ersten Verfeinerungsstufe.

Die weiteren Verfeinerungsstufen entstehen nicht, indem auf dem vorherigen Level wieder eine maximale unabhängige Menge erzeugt wird. Stattdessen entsteht Verfeinerungsstufe $n+1$ durch Tiefensuche im originalen Graphen von den Knoten der Verfeinerungsstufe n aus. Diese Suche wird jedoch nur bis zur Tiefe $2^{(n-1)}$ durchgeführt. Alle Knoten aus der Verfeinerungsstufe n, die bei einer solchen Suche gefunden werden, sind nicht in der Verfeinerungsstufe $n+1$ und es wird von ihnen aus auch keine Tiefensuche gestartet.

Für die Laufzeit dieser Strategie gibt es keine bewiesene subquadratische obere Schranke, so dass es möglich ist, dass die Laufzeit im Worst-Case $O(n^2)$ beträgt. Laut Gajer und Kobourov zeigt die Praxis jedoch ein deutlich subquadratisches Verhalten [GaKo02]. Die

Laufzeitmessungen im Rahmen dieser Arbeit, bei denen ein quadratisches Verhalten insbesondere bei den großen getesteten Graphen schnell aufgefallen wäre, haben das bestätigt.

Gajer und Kobourov haben die Tiefensuche stets bei der Tiefe $b^{(n-1)}$ mit $b = 2$ abgebrochen. Um die Qualität des *Independent Set Merger* steuern zu können, kann in dieser Implementierung die Basis b in dieser Formel verändert werden. Sinnvoll sind hier nur Werte über Eins. Ein Wert unter Zwei wird zur Folge haben, dass weniger Knoten bei der Tiefensuche gefunden werden und so dafür sorgen, dass mehr Verfeinerungsstufen erzeugt werden. Ein höherer Wert verringert die Anzahl der erzeugten Verfeinerungsstufen.

4.5 Edge Cover Merger (ECM)

Der *Edge Cover Merger* wurde im Rahmen dieser Arbeit entwickelt um den *Matching Merger* zu verbessern. Das Hauptproblem beim *Matching Merger* ist, dass bei hohem Knotengrad zu wenige Verschmelzungen stattfinden, da es nicht genügend Paarungen gibt. Um dieses Problem zu bekämpfen, werden beim *Edge Cover Merger* zusätzlich die Kanten einer Kantenüberdeckung berücksichtigt, sobald die Paarungskanten verbraucht sind. Die Kantenüberdeckung wird jedoch nur für die Knoten berechnet, die im Matching noch nicht berücksichtigt wurden. Für jede Kante aus der Kantenüberdeckung muss ein bisher nicht markierter Knoten markiert werden. Die Kantenüberdeckung wird also möglichst klein gehalten.

Zudem wurde wie beim *Random Merger* eingeführt, dass die relative Größe der Verfeinerungsstufen durch einen Faktor gesteuert werden kann. Die Paarungskanten werden bevorzugt, da sie sehr gleichmäßig über den Graph verteilt sind. Die Reihenfolge in der das Matching beziehungsweise die Kantenüberdeckung abgearbeitet wird, wird zufällig bestimmt.

4.6 Local Biconnected Merger (LBCM)

Beim *Local Biconnected Merger* handelt es sich um eine Variante des *Edge Cover Merger*, die entwickelt wurde, um das bei Multilevel-Verfahren auftretende Problem des im Laufe des Verfahrens verlorenen Zwei-Zusammenhangs zu bekämpfen. Die Auswahl der zu verschmelzenden Knoten erfolgt wie beim *Edge Cover Merger*. Zusätzlich wird jedoch geprüft, ob lokal der Zwei-Zusammenhang des Graphen verloren geht. Echter Zwei-Zusammenhang ist hier nicht das gewünschte Kriterium, da durchaus Fälle denkbar sind in denen der Zwei-Zusammenhang global erhalten bleibt, lokal jedoch Verdrehungen existieren.

Definition 11 (Lokaler Zwei-Zusammenhang). Lokaler Zwei-Zusammenhang ist gegeben, wenn von allen Nachbarn der beiden zu verschmelzenden Knoten alle anderen Nachbarn erreichbar sind, ohne einen Weg zu nutzen, der einen der beiden zu verschmelzenden

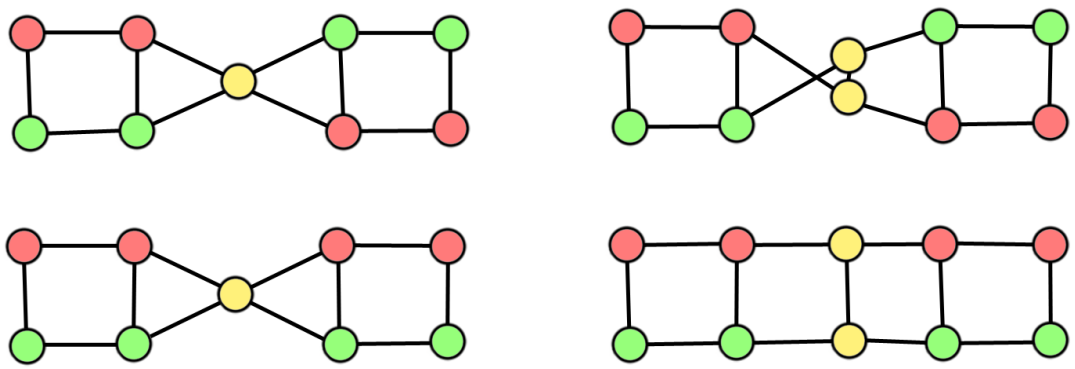

Abbildung 4.3: Durch den verloreren Zwei-Zusammenhang sind beide Zeichnungen auf der linken Seite gleichermaßen für eine Verfeinerungsstufe möglich. Auf der rechten Seite sind die resultierenden Zeichnungen des Graphen zu sehen.

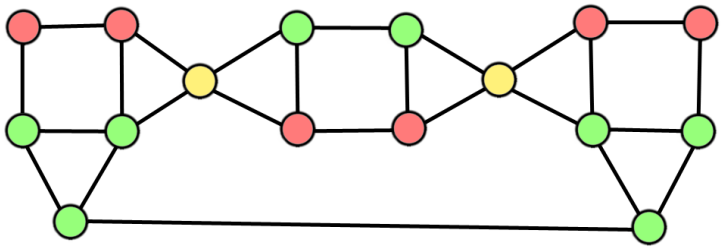

Abbildung 4.4: Es ist eine Verdrehung möglich, obwohl der Graph ist zwei-zusammenhängend ist.

Knoten enthält. Bei der dafür durchgeführten Breitensuche dürfen nicht alle Knoten des Graphen besucht werden, sondern nur ein fest definierter Anteil.

Der Anteil der besuchbaren Knoten wird hier auf $50 + 2 \cdot log(n)$ gesetzt, um zwar bei kleinen Graphen ausreichend Spielraum zu haben, bei großen Graphen jedoch nur wenig Laufzeit zu benötigen und den Bereich nicht zu sehr auszudehnen.

Dies reicht in der Praxis noch nicht aus, da auch bei gegebenem Zwei-Zusammenhang Verdrehungen möglich sind. Möglich ist das dadurch, dass die Platzierung keine Informationen darüber hat, welche Positionen eine Verdrehung erzeugen werden, und bei zwei-zusammenhängenden Graphen oft mehrere gleichberechtigte Möglichkeiten bestehen, den Graph in einem Kräftegleichgewicht zu zeichnen. Welche dieser Möglichkeiten vom kräftebasierten Verfahren gewählt wird, hängt nur von der weitgehend zufälligen Platzierung der Knoten ab.

Problematisch kann zudem die modifizierte Kantenlänge sein, da diese den Graphen trotz gegebenem Zwei-Zusammenhang instabil machen kann. Die so entstehenden extrem unterschiedlich langen Kanten können dafür sorgen, dass in einigen der stärkeren Verfeine-

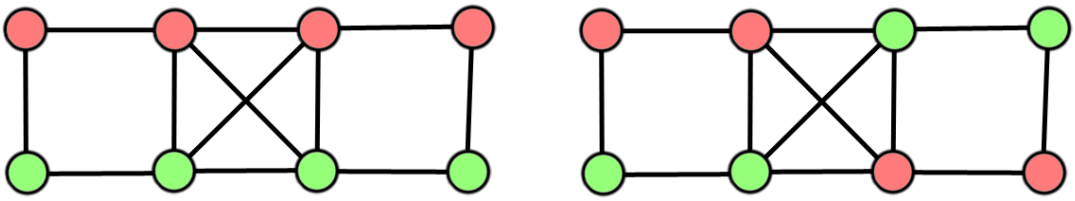

Abbildung 4.5: Beide Zeichnungen sind auf dieser Verfeinerungsstufe gleich gut. Auf späteren Verfeinerungsstufen werden im einen Fall wieder Verdrehungen entstehen.

Abbildung 4.6: Möbius Schleife [SCHL]

rungsstufen selbst für geometrisch einfache Gebilde keine gute Zeichnung gefunden wird. Die extrem langen Kanten können auch komplexere Strukturen so in die Länge ziehen, dass alle Knoten auf der Kante liegen. Diese Form ist nicht mehr sehr widerstandsfähig gegen Verdrehungen und es können so sogar neue Verdrehungen entstehen. Wenn sie nur geringfügig angewandt wird kann eine Modifikation der Kantenlänge aber auch hilfreich sein um die graphentheoretisch gleichen Fälle auseinander zu halten und so Verdrehungen unwahrscheinlicher zu machen.

Verdrehungen sind natürlich nicht immer zu vermeiden. Zum Beispiel kann eine Möbiusschleife nicht ohne Verdrehung gezeichnet werden. Auch theoretisch vermeidbare Verdrehungen werden durch den hier gewählten Ansatz nicht in jedem Fall vermieden, da erstens der Zufall eine sehr große Rolle im Ablauf des Algorithmus spielt und zweitens der Graph Strukturen enthalten kann, die eine lokale Zusammenschnürung bewirken können. An solchen Punkten ist eine Verdrehung wahrscheinlich.

Solche Zusammenschnürungen entstehen auch durch die weitreichenden Kräfte einiger kräftebasierter Verfahren, die sich für ausreichend große Graphen so stark aufaddieren können, dass der restliche Graph in die Länge gezogen wird. Die Wahl des Zeichenverfahrens für die Kräftephase hat also ebenfalls einen Einfluss auf die Anzahl der Verdrehungen.

Für die Laufzeit ist es wichtig, dass die gesamte Breitensuche nur sublineare Zeit benötigt, da sonst die Laufzeit der Zerlegungsstrategie bei den nötigen $n-3$ bis $n-1$ Verschmelzungen auf $O(n^2)$ anwachsen würde. Auch Initialisierungen von linear großen Arrays für einfache Knotenmarkierungen sind hier nicht möglich, so dass zum Beispiel Hash-Arrays zum Einsatz kommen müssen, die nur für tatsächlich betrachtete Knoten Daten speichern.

Kapitel 5

Platzierungsstrategien

Ein weiteres wichtiges Modul im *Modular Multilevel Mixer* ist die Platzierungsstrategie. Damit eine Platzierungsstrategie als Modul einsetzbar ist, muss das Interface *InitialPlacer* implementiert werden. Die Platzierungsstrategie hat im Vergleich mit der Zerlegungsstrategie oder dem Kräfteverfahren meist einen relativ geringen Einfluss auf die Zeichnung. Trotzdem hat sie einen Einfluss auf die Laufzeit des Verfahrens, da deutlich weniger Iterationen des Kräfteverfahrens nötig sind, wenn die Knoten bereits nahe der optimalen Position platziert wurden.

Im Detail geschieht die Platzierung eines Knotens meist einzeln, ohne den Rest der in dieser Platzierungsphase zu platzierenden Knoten zu beachten. Die Position des Knotens wird also aus der Zeichnung der vorherigen Verfeinerungsstufe berechnet. Insbesondere sind es natürlich die Positionen des Knotens, der den zu platzierenden Knoten bisher repräsentiert hat, und die Positionen der benachbarten Knoten, die die wichtigsten Hinweise auf eine gute Platzierung geben. Von einer Zerlegungsstrategie werden zusätzliche Informationen generiert, die in der Platzierungsphase genutzt werden können.

Um Probleme im Kräfteverfahren zu vermeiden, wird zu allen Platzierungen eine geringe zufällige Abweichung addiert. Insbesondere ergeben sich bei vielen Kräfteverfahren Probleme, wenn zwei Knoten die gleiche Position und so einen Abstand von Null zueinander haben.

5.1 Random Placer (RP)

Der Knoten wird an einer zufälligen Position im Umkreis um die bisherige Zeichnung platziert. Als Umkreis wird der kleinste Kreis definiert, der als Mittelpunkt den Schwerpunkt der Zeichnung, also den Durchschnitt aller Knotenpositionen, hat und alle Knoten enthält. In einigen Fällen kann der *Random Placer* Verdrehungen erzeugen, die mit einem anderen Platzierungsverfahren nicht aufgetreten wären. Dafür kann er jedoch auch dafür sorgen, dass die Zeichnung besser gestreckt wird und der Graph sich besser entfaltet.

In den meisten Fällen ist die Zeichnungsqualität sehr gut. Oft ist es nicht möglich, den *Random Placer* von anderen Platzierungsverfahren zu unterscheiden, indem man nur das Endergebnis des Verfahrens betrachtet. Der erwünschte Effekt einer reduzierten Laufzeit durch Platzierung nahe der energieminimalen Position des Knotens ist beim *Random Placer* nicht zu erwarten.

5.2 Zero Placer (ZP)

Als das einfachste Platzierungsverfahren ist der *Zero Placer* auch das verbreitetste. Der Knoten wird an der Stelle platziert, an der sich sein Repräsentant aus der vorherigen Vereinfachungsstufe befindet. Wichtig ist es hier, ein geringes Zufallsrauschen zu addieren, da viele Kräfteverfahren, wenn zwei Knoten einen Abstand von Null haben, unerwünschte Ergebnisse liefern. Der *Zero Placer* liefert gute Ergebnisse und kann als Messlatte für die Qualität aller komplizierteren Platzierungsverfahren angesehen werden. Ein Parameter des *Zero Placer* ist die Reichweite des Zufallsrauschens. Der Standardwert ist Eins, während alternativ auch ein Wert von 100 genutzt wurde.

5.3 Solar Placer (SP)

Diese Platzierungsstrategie entspricht derjenigen aus Hachuls *FM*^{3} [Hac05]. Die Platzierung entsteht hier nicht ausschließlich durch Berechnungen am aktuellen Layout sondern nutzt zusätzlich Informationen, die im *Solar Merger* generiert wurden. Wenn ein Knoten platziert werden soll, sind zwei Fälle zu unterscheiden: Wenn für den Knoten keine Informationen gespeichert wurden, er also in der entsprechenden Verfeinerungsstufe auf keinem *Intersystempfad* liegt, wird er nahe der Position der Sonne eingefügt. Die Position wird zufällig leicht abgewandelt um zu verhindern, dass der Abstand zwischen dem neu platzierten Knoten und der Sonne Null wird.

Im anderen Fall lag der Knoten auf einem oder mehreren *Intersystempfaden*. Die relative Position auf diesen Pfaden aus Sicht seiner Sonne liegt also vor. Für die Platzierung des Knotens wird nun angenommen, dass der Pfad entlang der direkten Verbindungsgeraden zwischen den beiden Sonnen verläuft. Gibt es nur einen für den Knoten relevanten Pfad, wird er an der relativen Position zwischen den beiden Sonnen platziert. Lag der Knoten auf mehreren Pfaden, werden die gewünschten Positionen aller Pfade einzeln berechnet. Der Knoten wird nun im Durchschnitt dieser Knoten platziert.

Diese Platzierungsstrategie ist nur im Zusammenspiel mit dem *Solar Merger* erfolgreich. Ohne diese Zusatzinformationen verhält sich der *Solar Placer* leider wie ein *Zero Placer*. Schon allein aus diesem Grund kann keine allgemein beste Platzierungsstrategie angegeben werden. Bei anderen Kombinationen von Platzierungs- und Zerlegungsstrategien ist der Einfluss der Zerlegungsstrategie jedoch nicht so extrem.

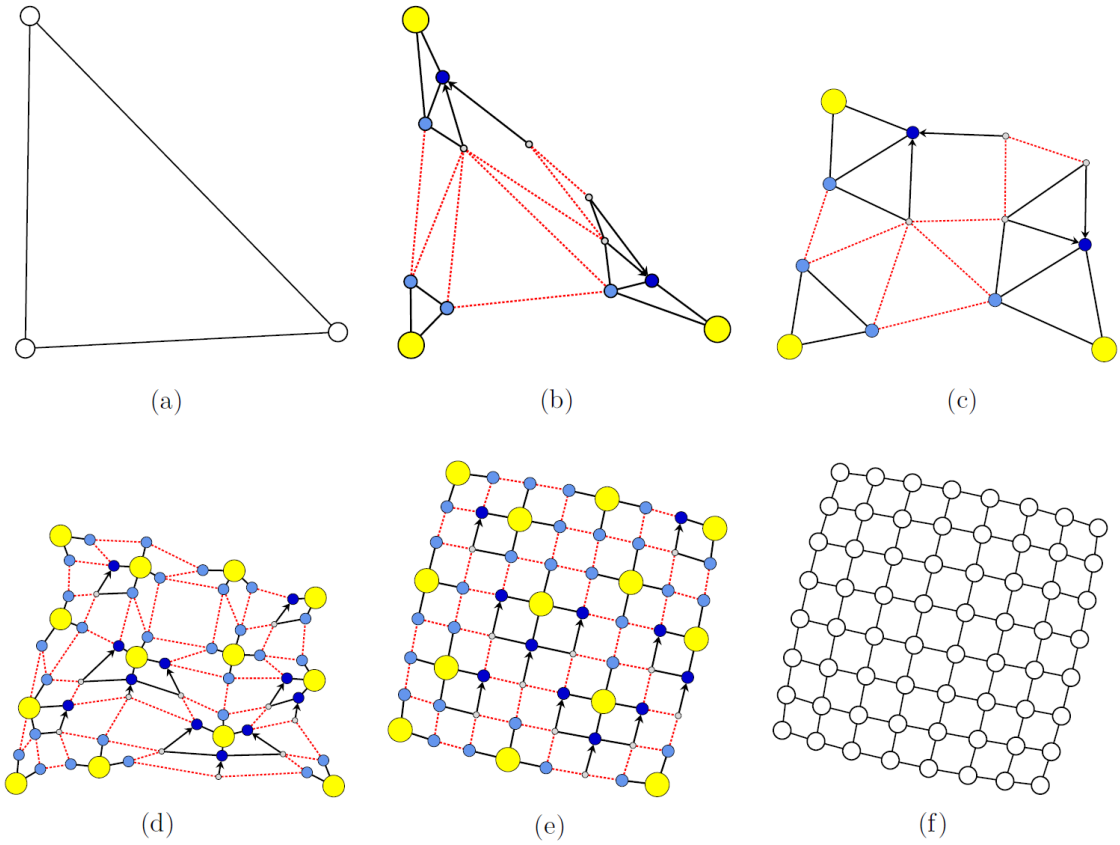

Abbildung 5.1: Platzierungs und Kräftephasen beim Solar Merger. [Hac05]

5.4 Barycenter Placer (BP)

Der *Barycenter Placer* ist ebenfalls ein weit verbreitetes Platzierungsverfahren, das zum Beispiel in GRIP genutzt wird [GaKo02]. Der zu platzierende Knoten wird im Schwerpunkt aller seiner Nachbarn platziert. Optional kann der Einfluss der Nachbarn N_i entsprechend der angestrebten Kantenlänge L_i gewichtet werden. Das Gewicht des Nachbarn N_i entspricht $\frac{1}{L_i}$. In beiden Varianten nimmt der *Barycenter Placer* die vermutete Wirkung der Kantenkräfte teilweise vorweg und kann so erreichen, dass das Kräfteverfahren eventuell deutlich weniger Iterationen benötigt, um zu einem Kräftegleichgewicht zu gelangen.

5.5 Median Placer (MP)

Als Alternative zum *Barycenter Placer* wurde der *Median Placer* implementiert. Der Knoten wird hier im Median aller seiner Nachbarn platziert. Dazu werden die Mediane auf der X-Achse und der Y-Achse einzeln berechnet, um die neuen Koordinaten zu erhalten. Falls es zwei Mediane auf einer Achse gibt, der Median also nicht eindeutig ist, wird der Durchschnitt der beiden als neuer Wert übernommen. Im Vergleich zum *Barycenter Pla-*

cer fallen Ausreißer unter den Nachbarn nicht so stark ins Gewicht. Diese Vorgehensweise macht unter der Annahme Sinn, dass der Ausreißer noch weit von seiner energieminimalen Position entfernt ist und er es ist, der noch bewegt werden sollte. In der Praxis sind *Barycenter Placer* und *Median Placer* nahezu nicht voneinander zu unterscheiden.

5.6 Circle Placer (CP)

Der *Circle Placer* wurde in der Hoffnung entworfen, den Graph stärker zu spannen um die Entfaltung zu begünstigen. Die neu platzierten Knoten werden hier zunächst wie im *Barycenter Placer* im Schwerpunkt der Nachbarn platziert, danach jedoch auf den wie beim *Random Placer* definierten Umkreis projiziert. Alle neu platzierten Knoten befinden sich also auf dem Kreis außerhalb der restlichen Zeichnung und erzeugen so Kräfte, die den Graph spannen.

Die daraus resultierende Zeichnungsqualität ist leider nicht sehr überzeugend, da der Graph zu sehr auf die Kreisform konzentriert wird und sich fast alle Knoten am Ende des Verfahrens auf oder nahe dem Umkreis befinden. Da ein Kreislayout nicht das Ziel war, muss das Experiment als gescheitert angesehen werden. Die Experimente mit dem *Circle Placer* haben gezeigt, dass es durchaus möglich ist, zuverlässig viel schlechtere Ergebnisse als der *Random Placer* zu erzielen, was intuitiv nicht unmittelbar einzusehen ist.

Kapitel 6

Ergebnisse der Experimente

Nachdem alle Module zur Verfügung standen, konnten die Vergleichstests durchgeführt werden. Hier sollte insbesondere herausgefunden werden, welche Zerlegungsstrategie eine besonders gute Qualität oder eine besonders gute Laufzeit aufweist. Zudem sollten gute Kombinationen von Zerlegungs- und Platzierungsstrategie gefunden werden.

Auch die weiteren Module, also Kräfteverfahren, Skalierung und Postprocessing, wurden untersucht, um am Ende einige besonders gute Kombinationen präsentieren zu können. Diese wurden dann mit den vorliegenden Verfahren *Fast Multilevel Multipole Embedder* von Martin Gronemann [Gro09] und *Fast Multipole Multilevel Method* von Hachul [Hac05] verglichen.

6.1 Die Testumgebung

Alle Tests wurden auf einem Intel Core 2 Duo 8500 mit 3,16 GHz Taktfrequenz durchgeführt. Das Betriebssystem war Windows Vista.

Vom zweiten Prozessorkern konnte der *Fast Multipole Embedder* als einziges Kräfteverfahren profitieren. Dies hat natürlich nur Einfluss auf seine Laufzeit und nicht auf die anderen Testkriterien. Bei den Gesamtverfahren profitieren natürlich alle Varianten des *Modular Multilevel Mixer* die den *Fast Multipole Embedder* einsetzen und der *Fast Multilevel Multipole Embedder*. Für genauere Informationen über die Geschwindigkeitszuwächse, die der *Fast Multilevel Multipole Embedder* durch die Nutzung mehrerer Prozessoren erzielen kann, sei auf die Diplomarbeit von Martin Gronemann [Gro09] verwiesen.

Bei allen Tests wurde vor dem Aufruf des eigentlichen Verfahrens das *Preprocessor Layout* und das *Component Splitter Layout* ausgeführt, damit alle Graphen von allen Verfahren bearbeitet werden konnten, und Verfahren ohne eigene Behandlung nichtzusammenhängender Graphen nicht unnötig benachteiligt wurden.

6.1.1 Testfälle und mögliche Kombinationen

Für die experimentelle Auswertung stand eine große Anzahl von Modulen zur Verfügung, die hier noch einmal kurz aufgelistet werden. Außerdem werden die in den Tabellen und Abbildungen genutzten Abkürzungen für die Module festgelegt.

Inklusive Varianten wurden elf Kräfteverfahren genutzt. Da die meisten Kräfteverfahren noch eine große Anzahl von Parametern anbieten, könnte diese Liste nahezu beliebig verlängert werden:

- *FR FM3*: Die Quadtree-Variante des Verfahrens von Fruchtermann und Reingold [FrRe91], wie es in Hachuls *FM*^{3} [Hac05] implementiert ist.
- *FR FM3 grid*: Die Raster-Variante des Verfahrens von Fruchtermann und Reingold, wie es in *FM*^{3} implementiert ist.
- *Eades*: Das Verfahren von Eades, wie es in *FM*^{3} implementiert ist.
- *NMM*: Die *New Multipole Method* aus *FM*^{3}.
- *DH*: Das Verfahren von Davidson und Harel. Es ist auf Grund der hohen Laufzeit in den Tests nicht enthalten.
- *GEM*: Der *Graph Embedder* von Frick, Ludwig und Mehldau [FLM98].
- *FR*: Die OGDF-Implementierung des Verfahrens von Fruchtermann und Reingold. Das automatische Skalieren wurde deaktiviert.
- *FR as*: Die *FR*-Variante mit aktivierter automatischer Skalierung.
- *FME*: Der *Fast Multipole Embedder* von Martin Gronemann [Gro09].
- *FEOE*: Die *FME*-Variante *Fast Edges Only Embedder* liefert keine ausreichende Qualität um außerhalb des Postprocessings eingesetzt zu werden.
- *MFL*: Das *Mixed Force Layout* ruft erst *FME* und dann *FR* auf.

Es wurden acht Zerlegungsstrategien implementiert. Weitere zwei ergeben sich als Variante. Zudem ist es auch als Zerlegungsstrategie anzusehen, überhaupt keine Verfeinerungsstufen zu generieren.

- *NullM*: Hier werden keine Verfeinerungsstufen generiert. Dies wird genutzt, um die Kräfteverfahren mit ihren Multilevel-Varianten zu vergleichen.
- *RM*: Der *Random Merger* wählt zufällig Knoten aus und verschmilzt sie mit einem Nachbarknoten.

- *RM2*: Im Gegensatz zu *RM* wird der Graph hier nur um den Faktor $1,1$ statt um den Faktor 2 verkleinert.
- *MM*: Der *Matching Merger* berechnet ein Matching und verschmilzt die so gefundenen Knotenpaare.
- *MM2*: Bei dieser Variante von *MM* wird versucht, Knoten mit möglichst geringer Masse für das Matching zu wählen.
- *ECM*: Der *Edge Cover Merger* kann zusätzlich zu den Matchingkanten auch Kanten aus einer Kantenüberdeckung auswählen.
- *LBCM*: Der *Local Biconnected Merger* soll unnötige Verdrehungen des Graphen vermeiden.
- *SM*: Der *Solar Merger* wählt Knoten als Sonnen wie in Hachuls *FM*^{3} und bevorzugt Sonnen mit geringer Masse.
- *SM2*: Diese *SM*-Variante wählt die Sonnen zufällig.
- *ISM*: Der *Independent Set Merger* berechnet die Verfeinerungsstufen ausgehend von den Knoten einer maximalen unabhängigen Menge durch parallele Breitensuche.
- *ISM2*: Bei dieser Variante von *ISM* erhöht sich die Tiefe der Breitensuche um den Faktor $1,1$ statt um den Faktor 2.

Inklusive zwei Varianten gibt es acht Platzierungsstrategien:

- *MP*: Beim *Median Placer* werden die Knoten im Median der Positionen der Nachbarknoten platziert.
- *BP*: Der *Barycenter Placer* berechnet statt des Medians die durchschnittliche Position.
- *BP2*: Diese *Barycenter Placer*-Variante gewichtet die Positionen der Nachbarn entsprechend der gewünschten Kantenlänge.
- *SP*: Der *Solar Placer* nutzt die vom *Solar Merger* generierten Zusatzinformationen, um die Knoten zu positionieren.
- *ZP*: Der *Zero Placer* platziert die Knoten an der Position des Repräsentanten aus der vorherigen Verfeinerungsstufe.
- *ZP100*: Eine *Zero Placer*-Variante, bei der das Zufallsrauschen eine Ausdehnung von 100 in jede Richtung hat, statt nur Eins wie bei *ZP*.

- *RP*: Der *Random Placer* platziert die Knoten zufällig im Umkreis um die bisherige Zeichnung.
- *CP*: Der *Circle Placer* wird in den Experimenten nicht genutzt, da die Qualität nicht überzeugend ist.

Die Skalierungsschemata haben ebenfalls einen großen Einfluss auf die Qualität der Zeichnung. Es ist leicht einzusehen, dass hier noch viel mehr Möglichkeiten bestehen, indem die Parameter verändert werden. Um die Anzahl der Tests im Rahmen zu halten, wurden nur die folgenden sieben implementiert:

- *no sc*: Keine Skalierung.
- *HD 2-sc*: Zwei Skalierungsschritte, erst auf das Zehnfache der gewünschten Kantenlänge, dann auf das Fünffache.
- *ND 2-sc*: Zwei Skalierungsschritte, erst auf das Dreifache der gewünschten Kantenlänge, dann genau auf die gewünschte Kantenlänge.
- *ND 4-sc*: Vier Skalierungsschritte, beginnend mit dem Vierfachen der gewünschten Kantenlänge. Am Ende wird die Größe der Zeichnung wieder genau so gewählt, dass die durchschnittliche Kantenlänge der gewünschten entspricht.
- *factor 2*: Die Zeichnung wird um den Faktor Zwei vergrößert.
- *factor 3*: Die Zeichnung wird um den Faktor Drei vergrößert.
- *factor 4*: Die Zeichnung wird um den Faktor Vier vergrößert.

Dazu kommen noch sechs Postprocessing-Ansätze. Auch hier sind durch andere Parameter noch viele Varianten möglich:

- *no Post*: Postprocessing wird nicht durchgeführt.
- *Little Post every*: Nach jeder Iteration werden 30 Iterationen Postprocessing mit *FEOE* durchgeführt.
- *Big Post final*: Am Ende des Algorithmus wird noch einmal halb so viel Zeit mit *FEOE* verbracht, wie für den Algorithmus benötigt wurde.
- *Big Post every*: Nach jedem Schritt 100 *FEOE*-Iterationen und am Ende noch einmal die Hälfte der benötigten Zeit.
- *FR Post every*: Nach jedem Schritt wird das Verfahren von Fruchtermann und Reingold aufgerufen.

- *FR Post final*: Am Ende des Algorithmus wird einmal das Verfahren von Fruchtermann und Reingold aufgerufen.

Im weiteren Verlauf der Arbeit werden die Module meist nur noch mit den jeweiligen Abkürzungen bezeichnet. Alle Tabellen und Abbildungen enthalten nur die Abkürzungen der Module.

Rein rechnerisch ergeben sich aus der Anzahl der Module bereits 40.656 mögliche Kombinationen. Es ist zeitlich nicht möglich, diese alle mit einer bedeutsamen Anzahl von Graphen auszuprobieren, selbst wenn ein Aufruf nur wenige Sekunden benötigt. Auch die Auswertung der Ergebnisse wäre nicht machbar. Um dennoch für alle Module Aussagen treffen zu können, wurden die Tests so zusammengestellt, dass in einem Test stets nur ein oder zwei der Module ausgetauscht werden konnten. Aus den Ergebnissen eines Tests wurden dann Kombinationen von Modulen abgeleitet, die in den späteren Tests so weiter verwendet wurden.

6.1.2 Getestete Qualitätskriterien

Um herauszufinden, welche Module besonders gut zueinander passen, wurden die Zeichnungen nach unterschiedlichen Kriterien ausgewertet. Als Besonderheit ist hier zu beachten, dass nachdem die Entfernung von der durchschnittlichen Kantenlänge berechnet wurde, der Graph so skaliert wird, dass die durchschnittliche Kantenlänge Fünf beträgt. Ein Zeichenverfahren, das die Kantenlänge nicht einhält, könnte durch eine besonders große Zeichnung andere Kriterien wie zum Beispiel die Knotenüberlappungen beeinflussen. Eine extrem große Zeichnung wird keine Knotenüberlappungen aufweisen, könnte jedoch auch leicht für alle anderen Verfahren erzeugt werden, so dass die Anzahl der Knotenüberlappungen keine Aussagekraft für die Qualität der Zeichnungen mehr hätten. Die nachträgliche Skalierung erhöht also die Vergleichbarkeit der Zeichnungen. Da der *Fast Multipole Embedder* eine besonders hohe Abweichung von der gewünschten Kantenlänge aufweist und außerdem nach der Skalierung eine besonders kleine Zeichenfläche nutzt, entstehen hier leider auch extrem viele Knotenüberlappungen.

Folgende Kriterien werden geprüft:

- *Entfernung von der gewünschten Kantenlänge*: Die gewünschte Kantenlänge ist für alle Zeichnungen Fünf, während der Radius der Knoten Eins beträgt. Nicht alle Verfahren konnten dieser Anforderung gerecht werden, so dass die Entfernung der durchschnittlichen Kantenlänge von der gewünschten ein Qualitätskriterium darstellt.

- *Standardabweichung der Kantenlängen*: Ein wichtiges Qualitätskriterium bei Kräfteverfahren ist eine möglichst gleiche Länge der Kanten. Dies wird durch die Standardabweichung der Kantenlängen gemessen.

- *Zeichenfläche*: Die Zeichenfläche wird durch die Fläche der konvexen Hülle der Zeichnung gemessen.
- *Winkelabweichung*: Die Zeichnung ist übersichtlicher, wenn zwischen den Kanten an den Knoten möglichst große Winkel liegen. Da die Winkel am größten sind, wenn die Kanten gleichmäßig um den Knoten verteilt sind, ist der Durchschnitt der Standardabweichungen der Winkel an den Knoten ein nützliches Maß.
- *Anzahl der Kreuzungen*: Die Anzahl der Kreuzungen ist eines der wichtigsten Kriterien, wie später noch dargelegt wird. Sie ist leicht zu berechnen, wozu aber quadratische Laufzeit nötig ist.
- *Knotenüberlappungen*: Auch für das Zählen der Knotenüberlappungen ist viel Laufzeit nötig, hier jedoch nur quadratisch in der Anzahl der Knoten statt der Kanten.
- *Knoten/Kante-Überlappungen*: Wenn eine Kante einen Knoten schneidet, kann das so aussehen als würde die Kante an diesem Knoten enden. Auch hier ist die Laufzeit der Zählung nicht zu vernachlässigen.
- *Laufzeit*: Die Laufzeit sollte natürlich möglichst gering sein. Eine hohe Laufzeit kann jedoch akzeptabel sein, wenn dadurch die Qualität erhöht werden kann. Die meisten getesteten Varianten sind für die Zeichenqualität optimiert worden, was Abstriche bei den Laufzeiten zur Folge hatte.
- *Knotenverteilung*: Die Gleichmäßigkeit der Knotenverteilung auf der Zeichenfläche wird durch die Summe der quadrierten Knotenabstände gemessen. Auch hier ist wie bei allen anderen Kriterien ein kleiner Wert besser.

6.1.3 Die getesteten Graphen

Zur Verfügung standen viele Graphen, die bereits von Hachul für den Test seines Multilevel-Verfahrens genutzt wurden. In seiner Arbeit [Hac05] sind also weitere Zeichnungen einiger der besonders großen Graphen zu finden. Dies beinhaltet auch Zeichnungen durch andere Verfahren. Zudem wurden einige große Graphen aus der Biologie genutzt, die teilweise sehr dicht sind. Insgesamt stehen hier sehr verschiedene Graphen zur Verfügung, die auch in Hinblick auf Parameter wie die Dichte sehr unterschiedlich sind. Die Graphen sind so ausgewählt, dass keine Modulkombination einen erkennbaren Vorteil hat.

Es wurden zudem einige langgestreckte Grids generiert, die gut als Beispiel für die durch den *Local Biconnected Merger* zu bekämpfenden Verdrehungen geeignet sind. Die Grids wurden einzeln getestet.

Die Graphen wurden entsprechend der Anzahl der Kanten in drei Größenklassen eingeteilt. Die Anzahl der Kanten wurde deshalb gewählt, weil die Laufzeit der Tests durch die Auswertung der Kantenkreuzungen dominiert wird.

- *kleine Graphen*: Graphen mit bis zu 5.000 Kanten, die in Tests mit mehreren Durchläufen und mehreren variablen Modulen genutzt wurden. Auch diese Tests haben oft über zwölf Stunden gedauert.

- *große Graphen*: Die Graphen mit bis zu 50.000 Kanten wurden nur eingesetzt, wenn Module einer einzelnen Kategorie getestet wurden. Auch ohne Wiederholungen haben diese Tests oft einen ganzen Tag gedauert.

- *extrem große Graphen*: Einige Graphen haben noch deutlich mehr als 50.000 Kanten. Hier konnten für die Tests die Anzahl der Kreuzungen und andere Kriterien auf Grund der enormen dafür nötigen Laufzeit nicht ausgewertet werden. Es wurden die Laufzeiten der Zeichenverfahren gemessen und Zeichnungen für den direkten Vergleich generiert.

6.2 Auswirkung der Skalierung

Die Skalierung hat je nach eingesetztem Kräfteverfahren eine sehr unterschiedliche Wirkung. Manche Verfahren, wie zum Beispiel *FR as*, werden kaum durch die Skalierung beeinflusst. Es kann auch keine für alle Kräfteverfahren optimale Skalierung geben. Hier soll zunächst herausgefunden werden welches Verfahren mit welcher Skalierung am besten zusammenspielt. Nachdem diese Daten vorliegen, kann die Skalierungsmethode für das jeweilige Kräfteverfahren festgelegt und die Menge der möglichen Kombinationen für weitere Tests verringert werden. Der Test wurde an den kleinen Graphen durchgeführt.

Welche Skalierung für das Gesamtverfahren die beste ist, hängt natürlich auch von der Wahl der Zerlegungsstrategie und der Platzierungsstrategie ab. Um alle Kombinationen testen zu können, wäre jedoch eine größere Zeitspanne notwendig, als im Rahmen dieser Diplomarbeit für Tests zur Verfügung steht. Für zukünftige Arbeiten könnte es interessant sein, diese langwierigen Tests durchzuführen. In diesem Test wurde als Zerlegungsstrategie *ECM* und als Platzierungsstrategie *BP2* gewählt.

Als wichtigstes Qualitätskriterium werden in diesem und den späteren Tests die Kantenkreuzungen angesehen. Die Kreuzungen können, im Gegensatz zu anderen Kriterien wie zum Beispiel Knotenüberlappungen und Kantenlänge, nicht leicht durch ein nachträgliches Postprocessing des Graphen modifiziert werden. Am Beispiel der Knotenüberlappungen ist dies besonders leicht einzusehen. Am Ende des Zeichenverfahrens kann eine modifizierte Kräftephase durchgeführt werden, während der sich Knoten nur abstoßen, wenn sie sich gerade überlappen. Andere Kräfte wirken hier nicht oder nur sehr schwach. Ein solches Verfahren würde die Knotenüberlappungen auf Null reduzieren ohne große Änderungen an der Zeichnung vorzunehmen. Eine ähnliche Behandlung für Kantenkreuzungen, die nur geringe Änderungen an der Zeichnung vornimmt, ist nicht möglich.

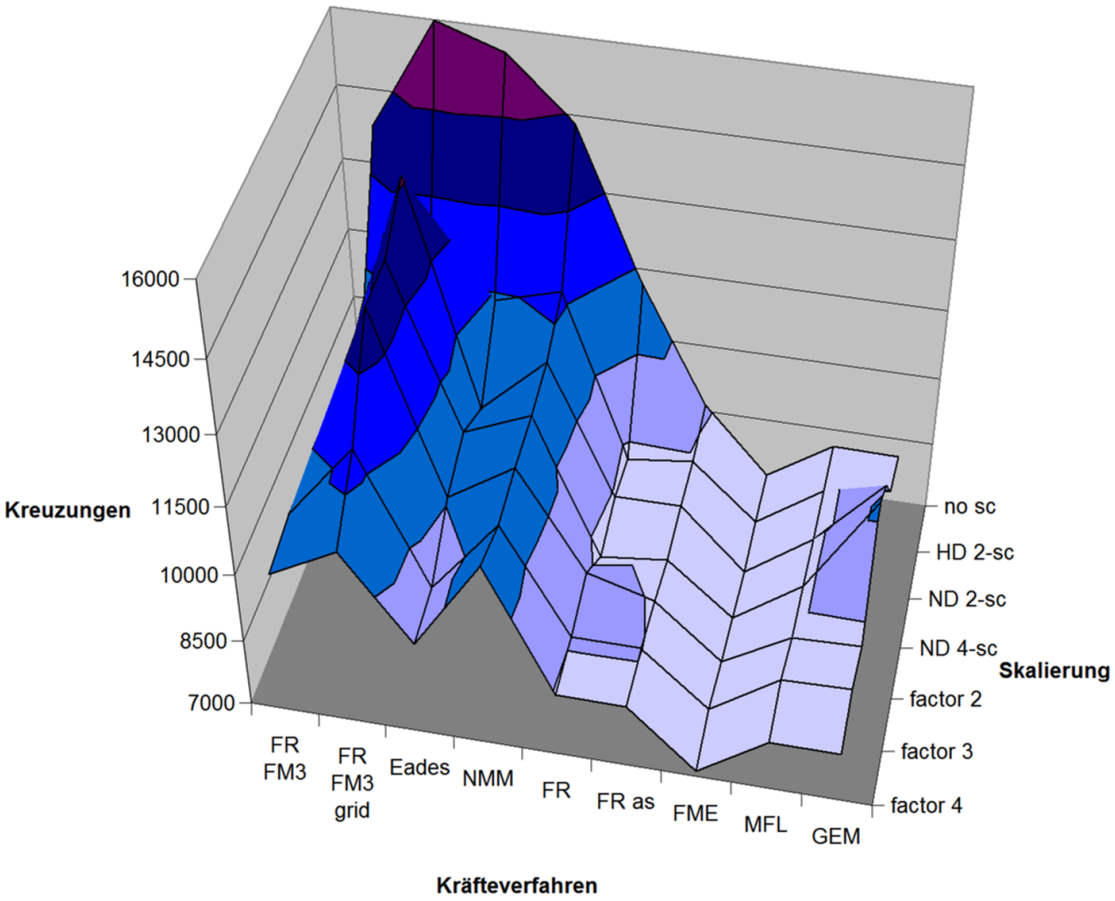

Abbildung 6.1: Durchschnittliche Anzahl der Kreuzungen bei unterschiedlichen Kräfteverfahren und Skalierungen.

	no sc	HD 2-sc	ND 2-sc	ND 4-sc	factor 2	factor 3	factor 4
FR FM3	13768	11041	10478	10383	10329	10352	10158
FR FM3 grid	29971	10954	14635	13875	13905	12001	10889
Eades	15547	11362	9962	10470	10084	9172	9064
NMM	14328	11751	11203	11053	10977	10822	11058
FR	11162	8195	8392	8074	8847	8529	8406
FR as	8648	8362	8398	8232	8437	8526	8410
FME	7194	7137	7050	7087	7243	7335	7134
MFL	8107	8120	8074	8110	8081	8309	8109
GEM	8090	8333	9576	10292	8133	8347	8090

Tabelle 6.1: Durchschnittliche Anzahl der Kreuzungen bei unterschiedlichen Kräfteverfahren und Skalierungen.

In Abbildung 6.1 ist die durchschnittliche Anzahl der Kreuzungen aufgetragen. Um die Übersichtlichkeit zu erhöhen, wurde der extreme Wert, der bei der Kombination von *FR FM3 grid* und *no sc* aufgetreten ist, bei 16.000 Kreuzungen abgeschnitten. Die genauen Werte können aus Tabelle 6.1 entnommen werden. Es wird sichtbar, dass das Kräfteverfahren einen größeren Einfluss auf die Anzahl der Kreuzungen hat als die Skalierung. Als bestes Kräfteverfahren in Bezug auf die Kantenkreuzungen ist *FME* hervorgegangen. *FME* erzielt den besten Wert in Kombination mit *ND 2-sc*, die Unterschiede zu den anderen Werten sind jedoch gering. Interessant ist, dass *GEM* die besten Ergebnisse unter anderem ohne Skalierung erzeugt, während viele andere Verfahren ohne Skalierung besonders schlechte Ergebnisse zeigen.

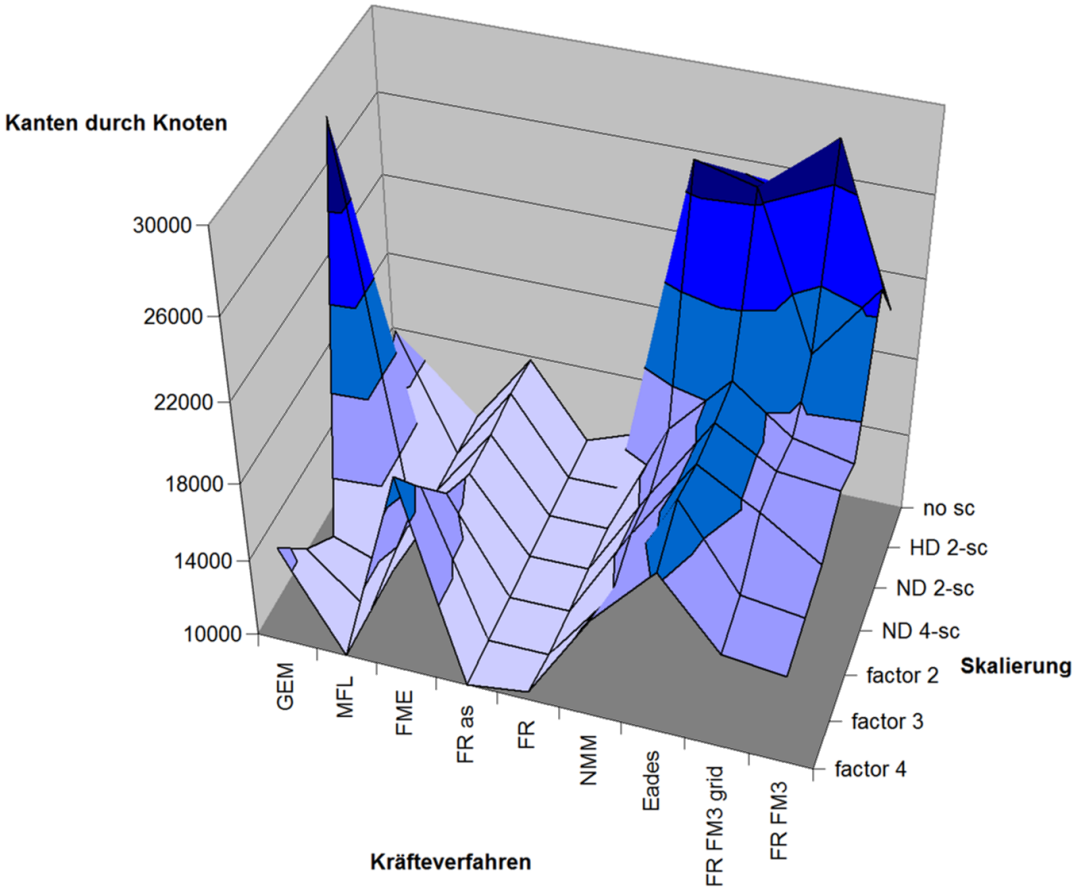

Abbildung 6.2: Durchschnittliche Anzahl von Kanten, die einen Knoten schneiden, bei unterschiedlichen Kräfteverfahren und Skalierungen.

Damit die Auswahl der für das jeweilige Kräfteverfahren festgelegten Skalierung nicht nur auf einem Kriterium basiert, wurde zudem betrachtet, wie viele Kanten durchschnittlich einen Knoten schneiden, an dem die Kante nicht beginnt oder endet. In Abbildung 6.2 kann die Anzahl der Schnitte zwischen Knoten und Kanten verglichen werden. Die gemessenen Werte können in Tabelle 6.2 eingesehen werden. Eine zu starke Skalierung

	no sc	HD 2-sc	ND 2-sc	ND 4-sc	factor 2	factor 3	factor 4
FR FM3	20178	23046	16353	17139	15563	15133	14547
FR FM3 grid	27779	19245	17018	17437	17506	15589	14874
Eades	25236	26689	19275	19189	19236	19669	18298
NMM	19047	27379	16129	14816	14917	13885	15075
FR	11070	9889	9919	9712	10003	9947	10431
FR as	9917	9810	9826	9759	9887	9933	9825
FME	13831	13968	13784	13803	14906	17421	20099
MFL	9654	9653	9639	9638	9732	9802	9801
GEM	10688	16086	13884	29907	11164	12731	15051

Tabelle 6.2: Durchschnittliche Anzahl von Kanten, die einen Knoten schneiden, bei unterschiedlichen Kräfteverfahren und Skalierungen.

scheint hier bei vielen Kräfteverfahren von Nachteil zu sein. Die Varianten des Verfahrens von Fruchtermann und Reingold schneiden hier am besten ab.

Eine weitere interessante Beobachtung ist, dass die in *FM*^{3} implementierten Kräfteverfahren bei beiden Kriterien deutlich schlechter abschneiden als die anderen Verfahren. Dies ist möglicherweise auf ein Postprocessing zurückzuführen das in *FM*^{3} integriert ist und durch den indirekten Aufruf der Kräfteverfahren nach jeder Kräftephase angewandt wird. Wenn die in *FM*^{3} eingesetzten Kräfteverfahren als *ogdf::LayoutModule* verfügbar gemacht werden, können diese eventuell bessere Ergebnisse erzielen. Die dafür nötigen Umbauarbeiten wären jedoch über den Rahmen dieser Diplomarbeit hinausgegangen.

Nach den Ergebnissen dieses Tests wurde die Auswahl der Skalierungen für die jeweiligen Kräfteverfahren festgelegt. In allen weiteren Tests sind die Skalierungen wie folgt gewählt, falls nichts anderes angegeben wurde:

- FR FM3 : factor 4
- FR FM3 grid : factor 4
- Eades : factor 4
- NMM : factor 3
- GEM : no sc
- FR : ND 4-sc
- FR as : ND 4-sc
- FME : ND 2-sc
- MFL : ND 2-sc

6.3 Vergleich mit den Single Level Algorithmen

Ziel dieses Tests war es die weit verbreitete und intuitiv einsehbare Meinung zu bestätigen, dass alle kräftebasierten Verfahren von einem Multilevel-Ansatz profitieren. Bei diesem Test wurde als Zerlegungsstrategie *ECM* und als Platzierungsstrategie *BP2* eingesetzt. Die Skalierung wurde abhängig vom Kräfteverfahren so gewählt wie im vorherigen Abschnitt angegeben. Der Test wurde an den großen Graphen durchgeführt.

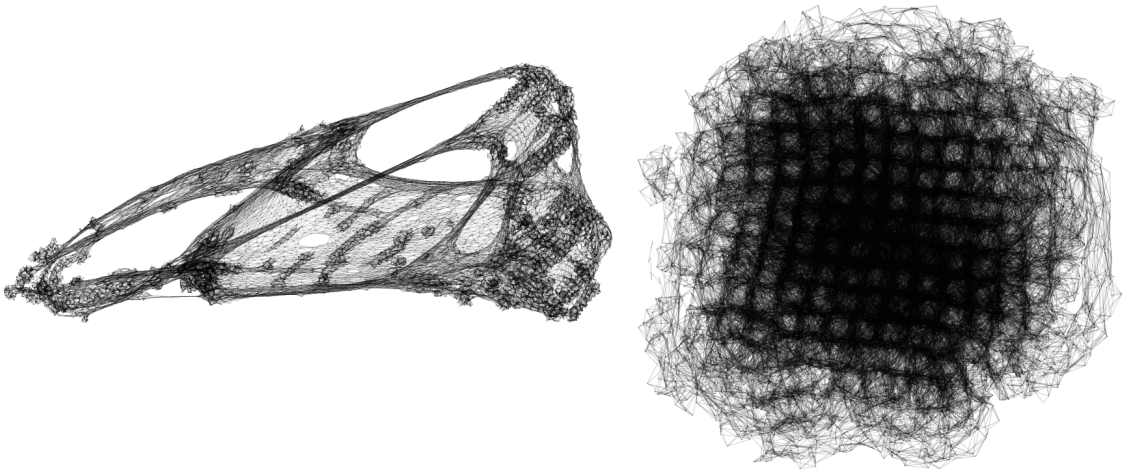

Abbildung 6.3: Zeichnungen eines Graphen durch FR as mit und ohne Multilevel-Ansatz.

Die Werte in Tabelle 6.3 zeigen, dass bei allen Verfahren die Ergebnisse mit Multilevel-Ansatz besser sind als ohne. Zu beachten ist, dass die Verfahren ohne Multilevel-Ansatz sehr viel unterschiedlichere Ergebnisse erzeugen. Einigen Verfahren kann man an den Werten direkt ansehen, dass die Ergebnisse nicht sehr gut sind. Andere Verfahren, wie zum Beispiel *FR* und *FR as* erzeugen Zeichnungen von hoher Qualität, die teilweise weniger Kreuzungen enthalten als die Zeichnungen anderer Kräfteverfahren in Kombination mit dem Multilevel-Ansatz. Während bei *FR* die Laufzeit ohne Multilevel-Ansatz extrem hoch ist, war *FR as* sogar schneller.

Auch in der Qualität werden kaum schlechtere Ergebnisse erzielt, wenn man die hier getesteten Qualitätskriterien betrachtet. Die Werte täuschen hier allerdings, denn eine Zeichnung mit wenig Kreuzungen muss die Struktur des Graphen noch nicht gut darstellen. Schaut man sich die erzeugten Zeichnungen in Abbildung 6.3 an, ist das Multilevel-Verfahren klar überlegen. Hier sieht man, dass vom Menschen wahrgenommene Ästhetik nur unzureichend durch mathematische Auswertungen erfasst werden kann. Alle Aussagen, die nur durch eine Auswertung von Messreihen begründet sind, sollten in dieser Arbeit und beim automatischen Zeichnen von Graphen allgemein also mit Vorsicht betrachtet werden.

	Kreuzungen	Standard-abweichung der Winkel	Standard-abweichung der Kantenlängen	Entfernung von Kantenlänge	Laufzeit (s)	Zeichenfläche	Knoten-überlappungen	Kanten durch Knoten	Knoten-verteilung
FR FM3	743.953	64,71	2,350	0,02	19,11	2.733.940	60.968	787.946	21.006.700
FR FM3 grid	652.731	65,92	2,198	0,02	13,75	2.759.890	63.410	597.014	34.618.800
Eades	692.039	65,17	4,276	0,03	18,78	2.962.350	70.306	1.555.697	24.800.500
NMM	1.032.629	64,65	2,035	0,02	19,10	1.522.290	65.302	807.432	17.826.900
FR	525.163	68,03	3,153	0,83	38,16	354.051	67.728	286.590	47.939.200
FR as	525.565	65,55	2,918	9,42	33,87	308.426	55.824	250.724	37.495.400
FME	445.714	66,38	1,546	21,66	19,68	99.343	139.338	384.504	15.606.500
MFL	505.510	64,76	2,786	13,71	29,40	289.591	48.925	230.258	17.755.300
GEM	526.696	64,34	3,031	20,66	17,93	358.667	75.976	343.560	23.184.400
FR FM3	31.637.266	57,42	2,280	0,02	4,73	95.789	3.014.609	20.592.730	5.641.370
FR FM3 grid	68.388.507	51,20	2,303	0,03	0,94	75.807	5.050.693	34.102.615	12.360.400
Eades	51.055.337	54,70	3,433	0,03	4,78	110.767	6.589.415	45.787.354	6.455.290
NMM	34.593.085	55,88	2,102	0,02	4,76	91.672	3.013.084	20.778.681	5.409.080
FR	605.121	66,18	3,211	1,52	202,78	257.158	61.486	286.367	17.041.700
FR as	633.600	65,86	3,155	12,05	19,39	173.015	60.454	285.093	14.582.900
FME	1.274.580	67,03	1,803	57,52	4,25	30.395	461.859	1.475.456	14.848.100
MFL	582.688	66,12	3,034	16,99	12,02	133.572	61.377	294.603	14.124.600
GEM	67.150.250	53,64	2,395	34,33	3,89	67.417	6.666.709	33.226.870	18.649.800

Tabelle 6.3: Die Werte aller Kriterien für die Verschiedenen Kräfteverfahren. Die oberen Werte wurden mit dem Multilevel-Ansatz erzielt, die unteren ohne.

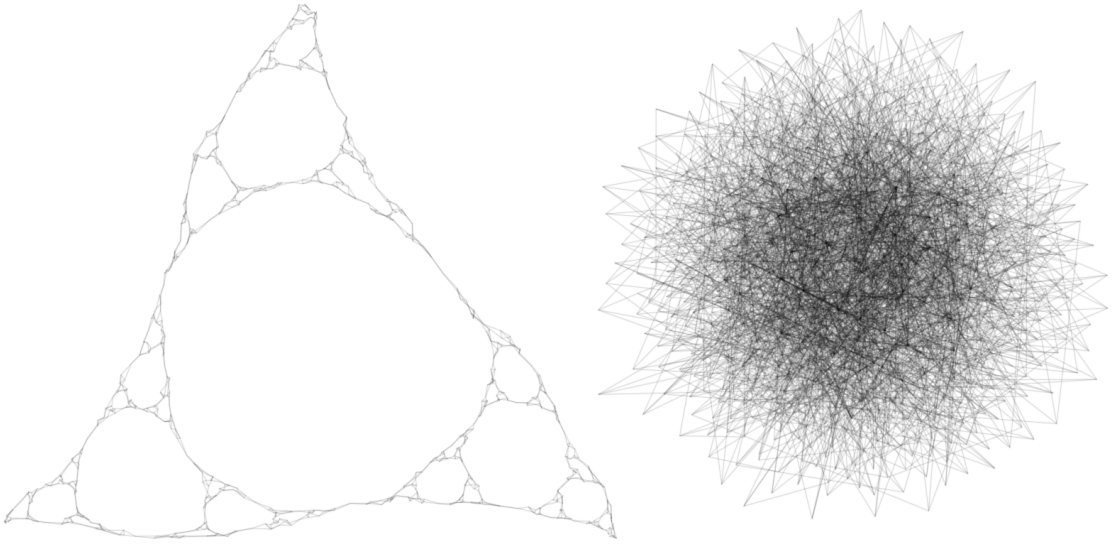

Abbildung 6.4: Zeichnung des Graphen sierpinski_06.gml durch DH. Links mit Multilevel-Ansatz und rechts ohne.

Das Verfahren von Davidson und Harel [DaHa96] kann wie erwähnt nicht alle Graphen zeichnen und ist auch deutlich langsamer als die anderen hier vorgestellten Verfahren. In den Tests wurde es also nicht verwendet. In Abbildung 6.4 kann man jedoch sehen, dass auch dieses Verfahren deutlich von einem Multilevel-Ansatz profitiert. Dies ist jedoch mit Geduld verbunden, da für diese Zeichnung ungefähr zehn Minuten gebraucht wurden. Alle anderen Verfahren zeichnen diesen Graph innerhalb weniger Sekunden in höherer Qualität. Der Vergleichslauf ohne Multilevel-Ansatz konnte so viele Iterationen durchführen, bis genau so viel Zeit genutzt wurde.

6.4 Vergleich der Kräfteverfahren

Mit den Daten des vorherigen Tests ist natürlich auch ein Vergleich der Kräfteverfahren untereinander möglich. Interessant ist im Rahmen dieser Arbeit der Vergleich der Verfahren im Kontext eines Multilevel-Ansatzes. Da die in Tabelle 6.3 zusammengetragene Datenmenge in einer einzigen Abbildung nicht übersichtlich darstellbar ist, wurden die Verfahren in kleinen Gruppen miteinander verglichen.

Da für die verschiedenen Kräfteverfahren unterschiedliche Skalierungen gewählt wurden, ist der Laufzeitvergleich hier nicht sehr aussagekräftig. Die Skalierungen wurden nach der Zeichenqualität ausgewählt und führen teilweise zu mehrfachen Aufrufen des Kräfteverfahrens. Ein aussagekräftigerer Test zur Laufzeit wurde an den extrem großen Graphen mit einigen Verfahren durchgeführt.

In den Abbildungen wurden die Achsen der jeweiligen Qualitätskriterien jeweils so skaliert, dass der höchste gemessene Wert außen und der kleinste innen aufgetragen werden

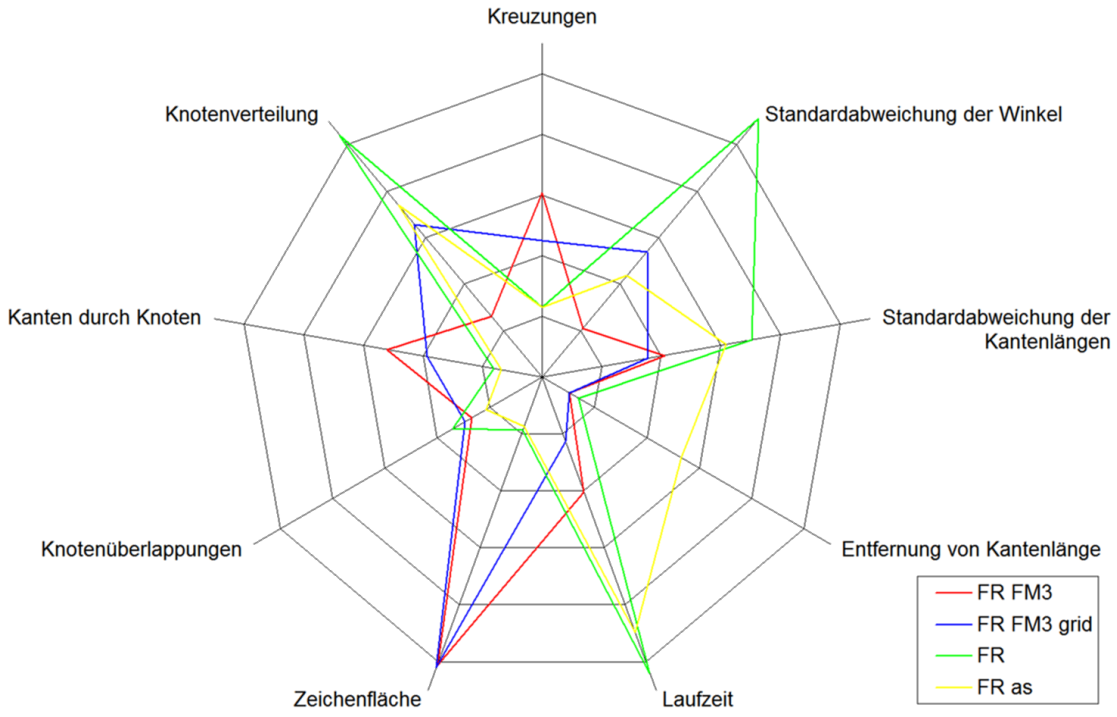

Abbildung 6.5: Varianten des Verfahrens von Fruchtermann und Reingold im Vergleich. Die Werte sind besser, je näher sie am Mittelpunkt aufgetragen sind.

konnte. Die exakten Werte können aus Tabelle 6.3 im vorherigen Abschnitt entnommen werden. Alle Abbildungen wurden entsprechend der gleichen Skalierung erzeugt. Verfahren, die nicht in einer gemeinsamen Abbildung zu finden sind, kann man also vergleichen, indem die Datenpunkte übertragen werden.

Einige der Kriterien haben sich als nicht sehr aussagekräftig herausgestellt. Insbesondere für die Standardabweichung der Winkel wurden für alle Kräfteverfahren nahezu die gleichen Werte gemessen. Andere Kriterien wie zum Beispiel die Knotenüberlappungen sind leicht durch einen Postprocessing-Schritt zu verbessern.

Vom Verfahren von Fruchtermann und Reingold [FrRe91] standen besonders viele Varianten zur Verfügung. Diese sind in Abbildung 6.5 im direkten Vergleich zu sehen. Hier fällt besonders der Unterschied zwischen den beiden Varianten aus FM^3 einerseits und den beiden Varianten aus OGDF andererseits auf. Die FM^3-Varianten können hier nur im Bereich der Knotenverteilung und bei der Laufzeit bessere Ergebnisse als die Konkurrenten vorweisen. Die OGDF-Varianten können hier also klar als überlegen gelten und erzielen auch im Gesamtvergleich insbesondere bei der Anzahl der Kreuzungen, Knotenüberlappungen und Kanten durch Knoten vorbildliche Ergebnisse.

In Abbildung 6.6 werden die vier Varianten der Kräfteverfahren verglichen, die mit FM^3 zur Verfügung standen. Am auffälligsten ist hier, dass alle vier Verfahren die Kantenlänge exakt einhalten. Dies erhärtet die zuvor geäußerte Vermutung, dass hier ein Postprocessing

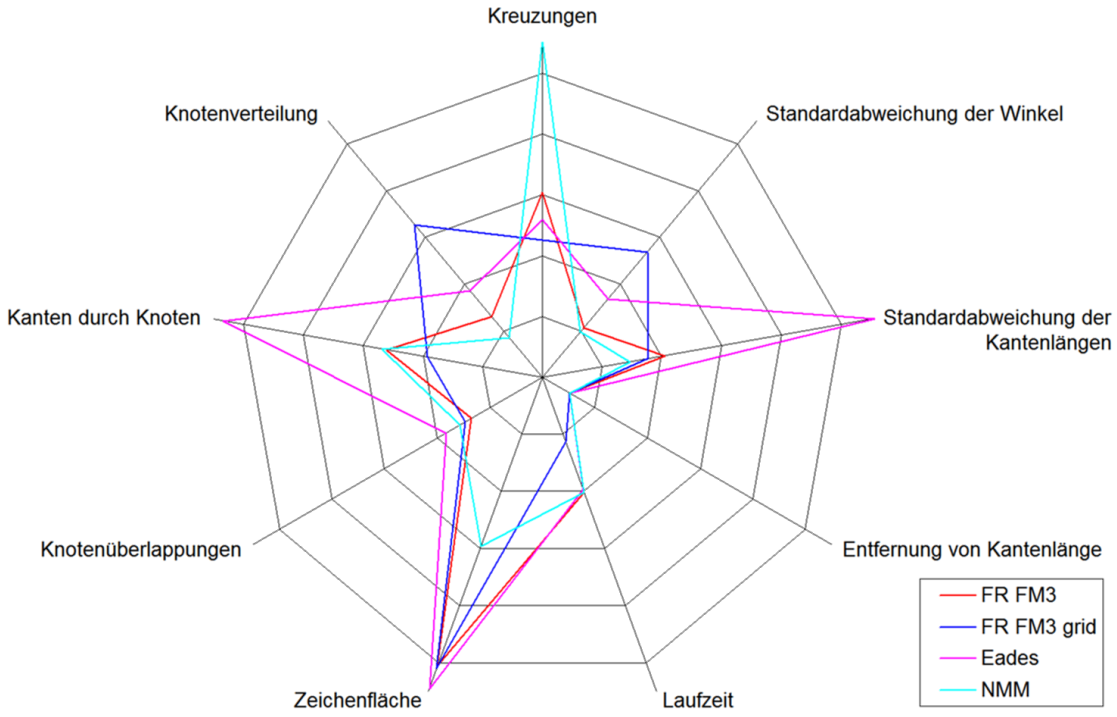

Abbildung 6.6: Die verschiedenen in *FM*^{3} implementierten Kräfteverfahren im Vergleich. Die Werte sind besser, je näher sie am Mittelpunkt aufgetragen sind.

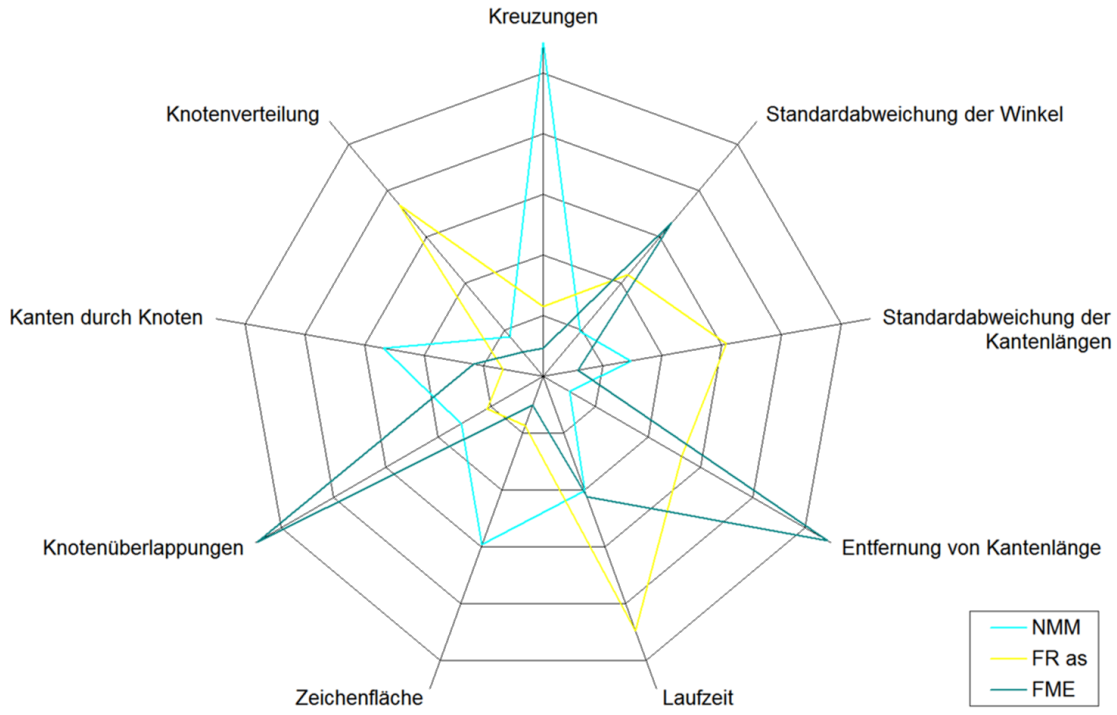

Abbildung 6.7: Vergleich der vielversprechendsten Verfahren. Die Werte sind besser, je näher sie am Mittelpunkt aufgetragen sind.

stattfindet, durch das die gewünschte Kantenlänge im Durchschnitt garantiert wird. Gute Ergebnisse werden hier bei der Knotenverteilung und den Knotenüberlappungen erzielt. Die Kantenkreuzungen sind eher als schlecht zu bewerten, insbesondere bei *NMM*, das hier den insgesamt schlechtesten Wert zeigt. *Eades* erreicht hier in gleich drei Kategorien die schlechtesten Ergebnisse.

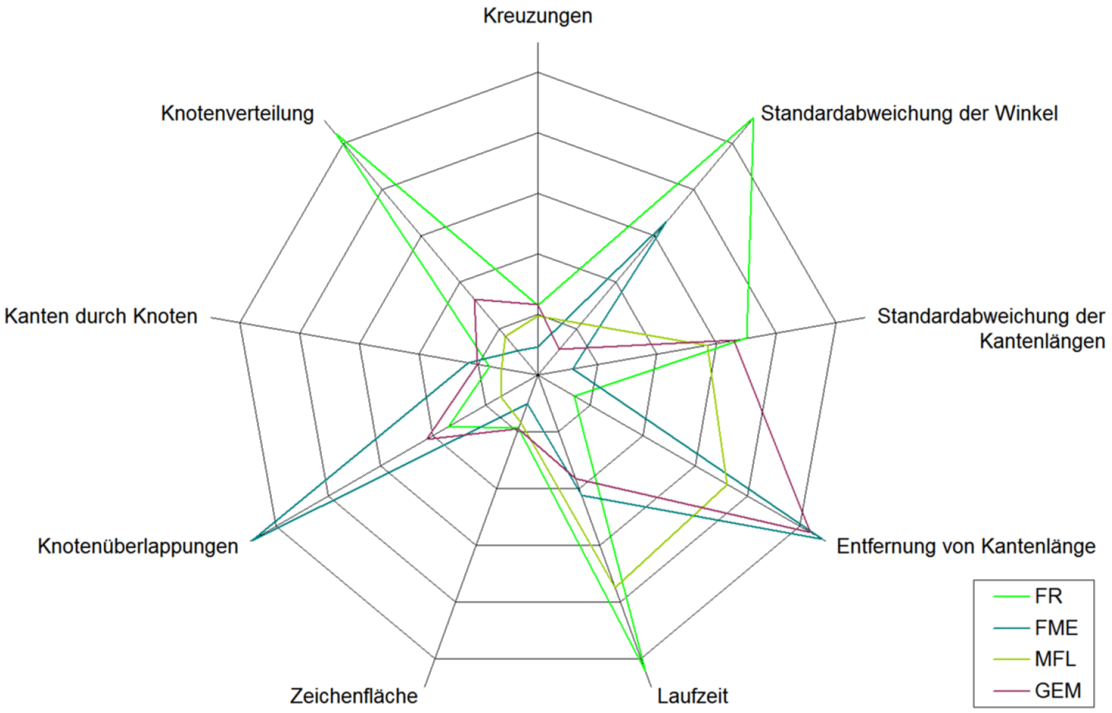

Abbildung 6.8: Vergleich weiterer Verfahren. Die Werte sind besser, je näher sie am Mittelpunkt aufgetragen sind.

Die Verfahren *FR as* und *NMM* sind besonders interessant für den Vergleich mit *FME*. *FR as*, weil es auch im Multilevel-Kontext das am weitesten verbreitete Verfahren ist, und *NMM*, weil *FME* eine beschleunigte Variante dieses Verfahrens darstellt. In Abbildung 6.7 wird deutlich, dass die Verfahren jeweils sehr unterschiedliche Stärken und Schwächen besitzen. Die hohe Abweichung von der gewünschten Kantenlänge bei *FME* ermöglicht offenbar, dass die Kantenlängen sich dafür kaum unterscheiden. Das Kriterium der gleichen Kantenlänge wird hier also besonders gut erfüllt. Nach der Skalierung auf die gewünschte Kantenlänge ist ebenfalls die Zeichenfläche die kleinste unter allen Verfahren. Dadurch wird jedoch die Anzahl der Knotenüberlappungen stark erhöht, so dass *FME* hier das schlechteste Ergebnis erzielt. In der Anzahl der Kantenkreuzungen ist *FME* ungeschlagen und kann das eindeutig beste Ergebnis vorweisen. Interessanterweise scheinen die besonderen Stärken und Schwächen von *NMM* und *FME* genau entgegengesetzt zu sein. Das zeigt, wie schwierig es ist, das Verhalten eines kräftebasierten Verfahrens in nur kleinen Teilen zu verändern, ohne auch den Rest des Verfahrens zu beeinflussen.

In Abbildung 6.8 werden auch die übrigen Verfahren verglichen. Auch hier zeigt sich, dass die Verfahren sehr unterschiedliche Stärken haben, so dass die Wahl des Verfahrens von den konkreten Anforderungen abhängig gemacht werden sollte. Interessant ist, dass bei *MFL* offenbar erfolgreich die Stärken und Schwächen der beteiligten Verfahren ausgeglichen werden konnten und so zuverlässig bei allen Kriterien Ergebnisse im Mittelfeld des Wertebereichs erzeugt wurden. In vielen Kriterien sind die Ergebnisse sogar vorbildlich. Hier kann festgehalten werden, dass durch geschickte Kombination vorhandener Verfahren ohne großen Aufwand neue und qualitativ überzeugende Verfahren generiert werden können.

6.5 Vergleich der Zerlegungs- und Platzierungsstrategien

Für den Vergleich der Zerlegungs- und Platzierungsstrategien wurden zwei Tests durchgeführt. Zunächst wurden alle Zerlegungs- und Plazierungsstrategien miteinander kombiniert. Dieser Test wurde natürlich auf den kleinen Graphen durchgeführt, da die Testzeit sonst bei weitem zu hoch gewesen wäre. Auch hier wäre ein größerer Test interessant, da die Ergebnisse nicht einfach für die großen Graphen verallgemeinert werden können. Zudem wäre auch eine größere Anzahl von Wiederholungen interessant, da die Ergebnisse noch stark vom Zufall beeinflusst sind. Das Ausmaß der Zufallsstreuung kann in etwa abgeschätzt werden, wenn man die unterschiedlichen Ergebnisse beim *NullM* betrachtet. Theoretisch müsste hier bei allen Platzierungsstrategien der gleiche Wert stehen, da diese mangels Verfeinerungsstufen nie zum Einsatz kommen. Für einen solchen Test auf großen Graphen mit vielen Wiederholungen müssten jedoch einige Wochen Rechenzeit eingeplant werden.

Alle drei Abbildungen 6.9, 6.10 und 6.11 sind sehr ähnlich. In Abbildung 6.9 sind die Kreuzungen aufgetragen. Hier kann man sehr gut erkennen, dass die Zerlegungsstrategien in zwei Kategorien eingeteilt werden können. Dass der *NullM* weit aus dem Rahmen fällt und deutlich schlechtere Ergebnisse liefert als der Rest der Verfahren, wird an dieser Stelle niemanden mehr überraschen. Besonders wenig Kreuzungen entstehen, wenn man *MM*, *MM2*, *RM2*, *ECM* oder *LBCM* anwendet. Die Zerlegungsstrategien *SM*, *SM2*, *ISM*, *ISM2* und *RM* haben verglichen damit deutlich mehr Kreuzungen erzeugt. Diese Einteilung in Zerlegungsstrategien von hoher und geringerer Qualität lässt sich an Abbildung 6.10 und etwas weniger eindeutig an Abbildung 6.11 ebenfalls nachvollziehen. Da die Kriterien hier deutlich übereinstimmen, kann eine klare Empfehlung für die bessere Kategorie der Zelegungsstrategien gegeben werden.

Wie in den Abbildungen 6.10 und 6.11 zu sehen ist, schwankt der Wert von *NullM* deutlich. Auf Grund dieser zufälligen Schwankungen gibt es keine eindeutig beste Zerlegungsstrategie.

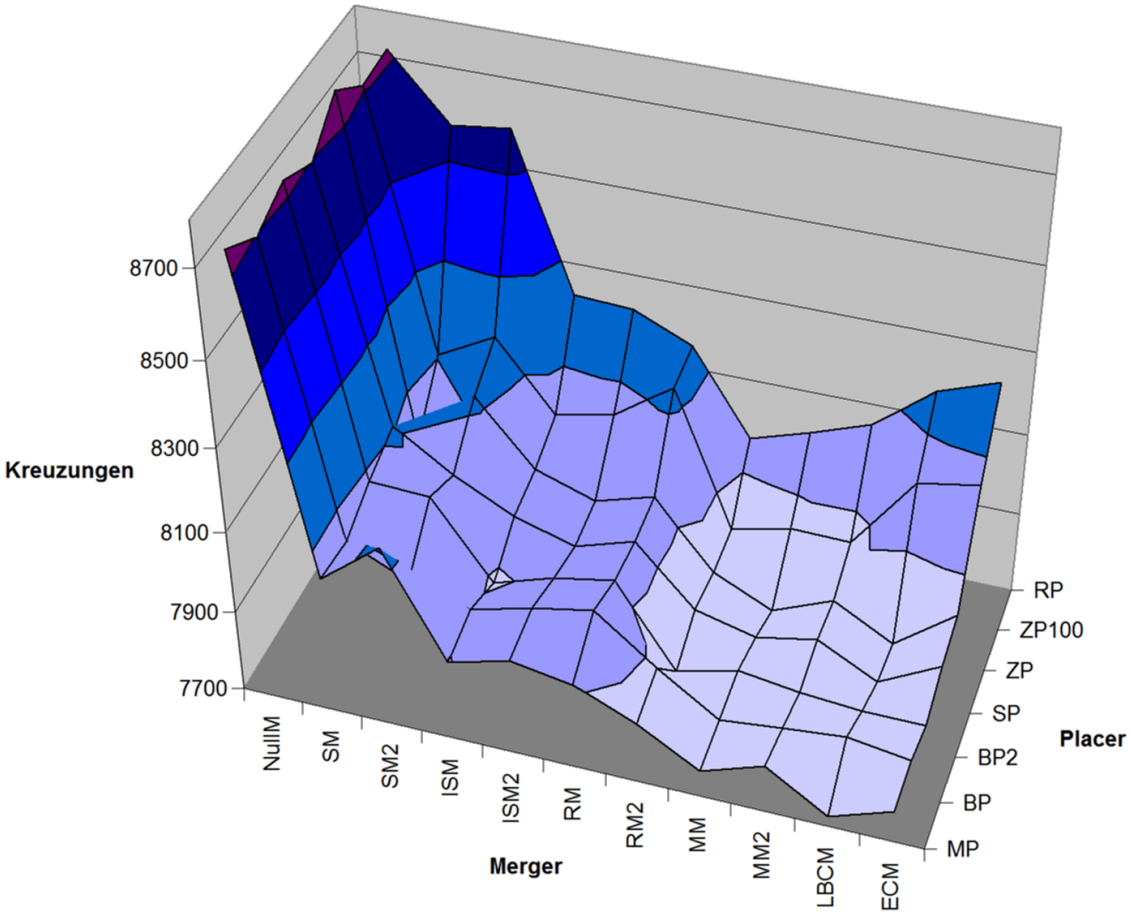

Abbildung 6.9: Anzahl der durch die verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie erzeugten Kreuzungen.

	NullM	SM	SM2	ISM	ISM2	RM	RM2	MM	MM2	LBCM	ECM
MP	8751	8035	8145	7897	7936	7912	7849	7766	7817	7725	7779
BP	8695	8024	7958	7921	7960	7989	7880	7781	7799	7814	7791
BP2	8734	8074	8067	7885	7930	7958	7758	7797	7777	7766	7763
SP	8693	8114	8025	7952	7908	7952	7811	7774	7804	7727	7804
ZP	8772	7986	8125	7973	7924	7965	7803	7737	7797	7718	7829
ZP100	8710	8106	8180	8015	8046	8143	7813	7848	7846	8030	8057
RP	8716	8571	8589	8224	8217	8155	7952	7998	8049	8162	8214

Tabelle 6.4: Anzahl der durch die verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie erzeugten Kreuzungen.

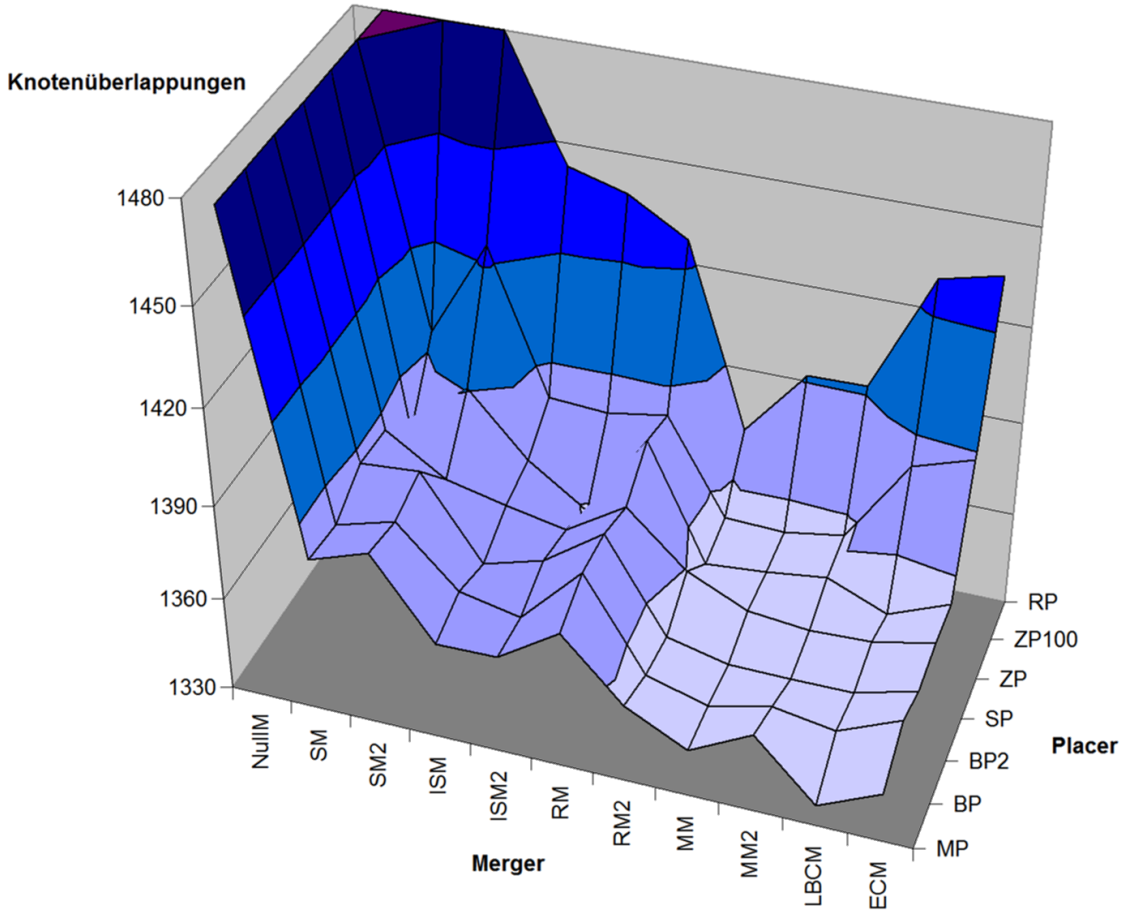

Abbildung 6.10: Anzahl der durch die verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie erzeugten Knotenüberlappungen. Extreme Werte wurden bei 1480 Knotenüberlappungen abgeschnitten.

	NullM	SM	SM2	ISM	ISM2	RM	RM2	MM	MM2	LBCM	ECM
MP	1608	1379	1386	1361	1361	1374	1355	1345	1355	1337	1346
BP	1607	1378	1383	1365	1361	1380	1351	1346	1350	1347	1355
BP2	1605	1386	1387	1361	1367	1380	1350	1346	1346	1346	1351
SP	1602	1385	1373	1368	1364	1376	1359	1350	1348	1349	1353
ZP	1593	1373	1391	1371	1358	1386	1349	1351	1353	1345	1353
ZP100	1606	1394	1425	1380	1379	1382	1352	1352	1355	1382	1388
RP	1616	1503	1525	1443	1438	1427	1370	1392	1392	1429	1434

Tabelle 6.5: Anzahl der durch die verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie erzeugten Knotenüberlappungen.

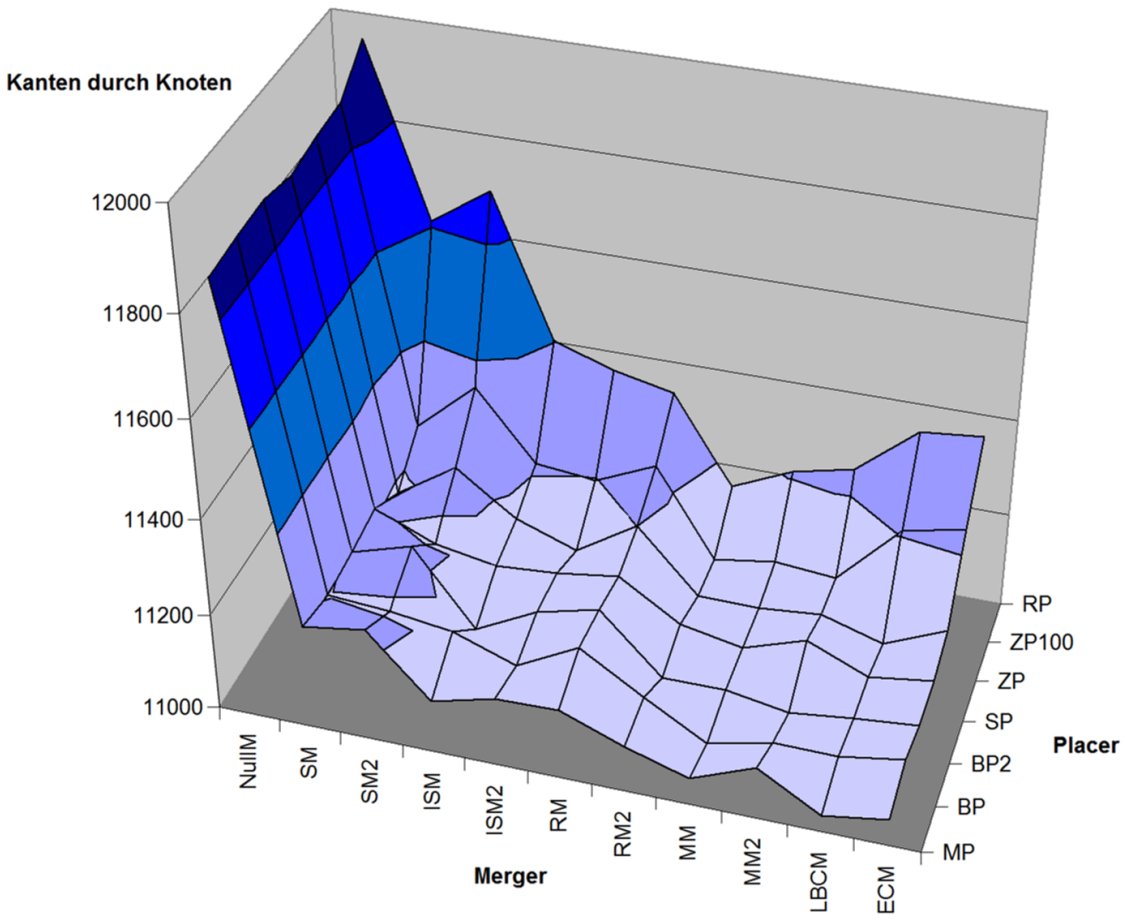

Abbildung 6.11: Anzahl der durch die verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie erzeugten Schnitte von Kanten mit Knoten.

	NullM	SM	SM2	ISM	ISM2	RM	RM2	MM	MM2	LBCM	ECM
MP	11875	11215	11235	11110	11143	11146	11096	11056	11108	11033	11056
BP	11884	11197	11189	11171	11124	11190	11111	11038	11069	11067	11089
BP2	11888	11206	11244	11091	11154	11185	11063	11063	11042	11056	11070
SP	11869	11220	11168	11145	11154	11172	11094	11069	11111	11057	11076
ZP	11882	11176	11254	11167	11123	11201	11071	11070	11083	11044	11099
ZP100	11887	11246	11352	11210	11197	11252	11071	11090	11080	11208	11235
RP	11952	11611	11690	11403	11361	11334	11153	11210	11239	11342	11354

Tabelle 6.6: Anzahl der durch die verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie erzeugten Schnitte von Kanten mit Knoten.

Für die Platzierungsstrategien lassen sich keine derart klaren Ergebnisse ablesen. Die Ergebnisse der verschiedenen Platzierungsstrategien sind kaum zu unterscheiden. Es gibt jedoch zwei deutliche Ausnahmen: *RP* und *ZP100* verschlechtern die Zeichnungsqualität, indem sie die Positionen der vorherigen Verfeinerungsstufe zu wenig übernehmen und stattdessen dem Zufall viel Platz einräumen.

Ein überraschendes Ergebnis an dieser Stelle war, dass auch die Kombination von *SM* mit *SP* in der Qualität keine überdurchschnittlichen Ergebnisse gebracht hat. Die Kombination konnte sich von anderen Verfahren nicht absetzen, wodurch zumindest in Hinblick auf die Qualität der Implementierungsaufwand nicht lohnenswert ist. Der *Solar Placer* ist auch verglichen mit anderen Platzierungsstrategien in Kombination mit *SM* nicht erfolgreicher. Ein Geschwindigkeitsgewinn durch *SP* ist ebenfalls nicht festzustellen.

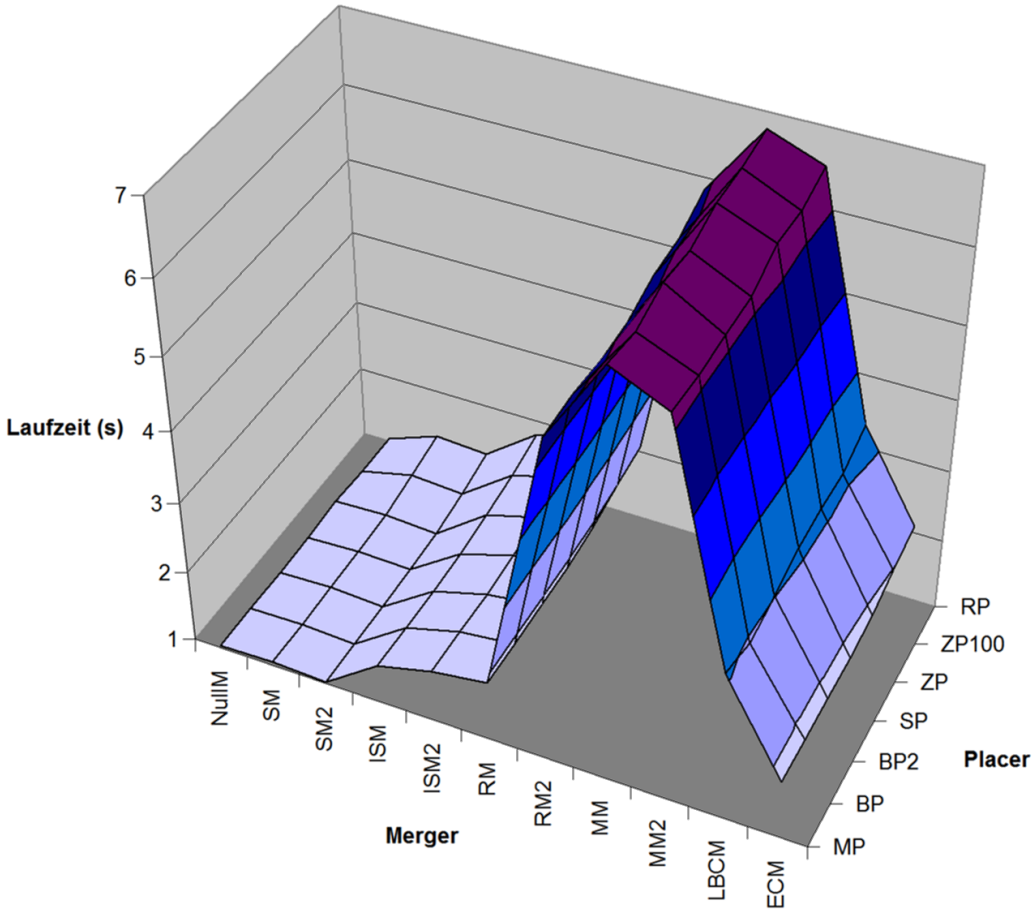

Abbildung 6.12: Die von verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie benötigte Laufzeit.

In Abbildung 6.12 zeigen sich Unterschiede, die mit der vorherigen Einteilung nicht übereinstimmen. Dies ist insbesondere durch die hohen Laufzeiten von *RM2*, *MM* und *MM2* gegeben. Die entsprechenden Verfahren generieren entweder immer wie der *RM2* oder in bestimmten Szenarios wie die *Matching Merger*-Varianten besonders viele Verfei-

	NullM	SM	SM2	ISM	ISM2	RM	RM2	MM	MM2	LBCM	ECM
MP	1,032	1,053	1,016	1,548	1,730	1,843	5,436	6,547	6,205	3,087	1,818
BP	1,035	1,057	1,025	1,531	1,740	1,828	5,461	6,469	6,161	2,914	1,810
BP2	1,030	1,060	1,024	1,522	1,735	1,832	5,415	6,609	6,185	3,011	1,809
SP	1,034	1,071	1,046	1,542	1,765	1,843	5,439	6,551	6,195	3,128	1,818
ZP	1,078	1,110	1,066	1,601	1,805	1,889	5,603	6,706	6,409	3,153	1,873
ZP100	1,033	1,202	1,157	1,678	1,878	1,996	5,630	6,714	6,387	3,132	1,984
RP	1,033	1,309	1,264	1,797	2,010	2,075	5,844	6,795	6,518	3,318	2,090

Tabelle 6.7: Die von verschiedenen Kombinationen von Zerlegungs- und Platzierungsstrategie benötigte Laufzeit.

nerungsstufen, wodurch die Anzahl der Kräftephasen stark erhöht wird. Von der vorherigen, nur auf der Qualität beruhenden Empfehlung sind nach Ausschluß der besonders laufzeitintensiven Verfahren also nur noch *ECM* und *LBCM* übrig, falls man auf Qualität und Laufzeit gleichermaßen Wert legt. Die Laufzeit von *LBCM* ist jedoch ebenfalls deutlich höher als die der schnellsten Verfahren.

Es hat sich hier gezeigt, dass *ISM2* nicht viel langsamer ist als *ISM*. Dies lässt darauf schließen, dass trotz deutlich geringerer Schrittweite bei der Breitensuche nur unwesentlich mehr Verfeinerungsstufen erzeugt werden. Die Ursache liegt vermutlich darin, dass trotz geringerer Schrittweite die Knoten gleichzeitig erreicht werden, so dass viele Verfeinerungsstufen keine Veränderungen enthalten. Verfeinerungsstufen bei denen keine Verschmelzung stattfindet, werden bereits in der Verschmelzungsphase entfernt.

Falls man auf besonders hohe Geschwindigkeit Wert legt, ist jedoch der *Solar Merger* in beiden Varianten zu empfehlen. Die enorme Geschwindigkeit ist darin begründet, dass besonders wenige Verfeinerungsstufen erzeugt werden. Hier wäre für zukünftige Tests interessant, ob eine *ECM*-Variante, die weniger Verfeinerungsstufen generiert, als schnelle Zerlegungsstrategie eine gute Wahl ist.

Für die Platzierungsstrategien lässt sich in Abbildung 6.12 wenig Variation erkennen. Ein Blick in die Tabelle 6.7 zeigt jedoch, dass *ZP100* und *RP* etwas langsamer sind als die anderen Platzierungsstrategien.

Der niedrigste Wert für die Kreuzungen ist bei der Kombination von *LBCM* mit *MP* zu finden, während auch nach den anderen Kriterien außer der Laufzeit hier sehr gute Ergebnisse erzielt werden. In einem langwierigeren Test auf den großen Graphen wurden nun alle Platzierungsstrategien in Kombination mit *LBCM* und alle Zerlegungsstrategien in Kombination mit *MP* getestet. So soll sich zeigen ob auch für große Graphen diese Kombination sehr gut ist und ob allgemein die Ergebnisse der kleinen Graphen auch bei großen Graphen zu erwarten sind.

Die Varianten des *Matching Merger* wurden hier nicht getestet, da die Laufzeit zu extrem wird. Das Zeichnen des Graphen dg_1087.gml hat bereits fast zehn Stunden gedauert, obwohl dieser nur etwas über 7000 Knoten und Kanten enthält. Da die meisten der Kanten jedoch an einem zentralen Knoten enden wurden tausende von Verfeinerungsstufen generiert. Auch *RP* wurde aus Laufzeitgründen nicht getestet.

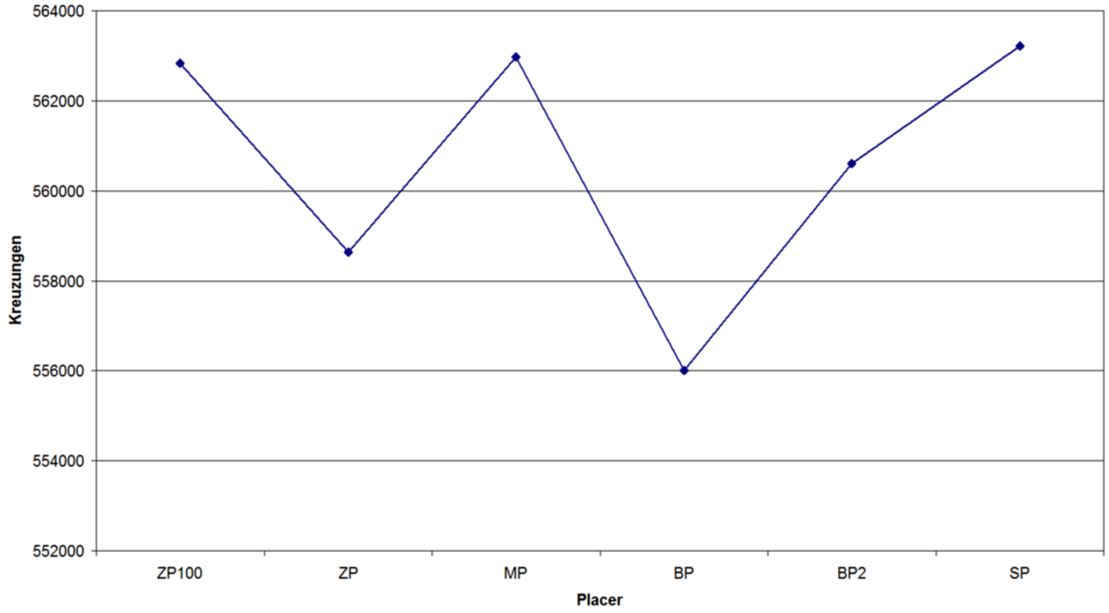

Abbildung 6.13: Kreuzungen bei verschiedenen Platzierungsstrategien in Kombination mit LBCM.

Platzierungsstrategie	ZP100	ZP	MP	BP	BP2	SP
Kreuzungen	562839	558635	562976	555999	560608	563210
Laufzeit (s)	41,58	37,96	37,48	37,46	38,27	44,81

Tabelle 6.8: Kreuzungen und Laufzeit bei verschiedenen Platzierungsstrategien in Kombination mit LBCM.

In Abbildung 6.13 ist die Anzahl der Kreuzungen der erzeugten Zeichnungen von verschiedenen Platzierungsstrategien auf den großen Graphen abzulesen. Als Zerlegungsstrategie wurde hier *LBCM* genutzt. Im Gegensatz zu den Zeichnungen der kleinen Graphen ist *MP* hier nicht unter den besten Platzierungsstrategien. Die wenigsten Kreuzungen werden von *ZP* und *BP* erzeugt. Die Unterschiede zwischen den Platzierungsstrategien sind hier aber nach wie vor sehr gering. Die in Abbildung 6.14 aufgetragenen Laufzeiten zeigen *SP* und *ZP100* als weniger schnell. Die deutlich höhere Laufzeit von *SP* im Vergleich zu *ZP* ist an dieser Stelle nicht nachvollziehbar, da sich beide Platzierungsstrategien im Zusammenspiel mit *LBCM* gleich verhalten. Dass ein derart deutlicher Laufzeitunterschied

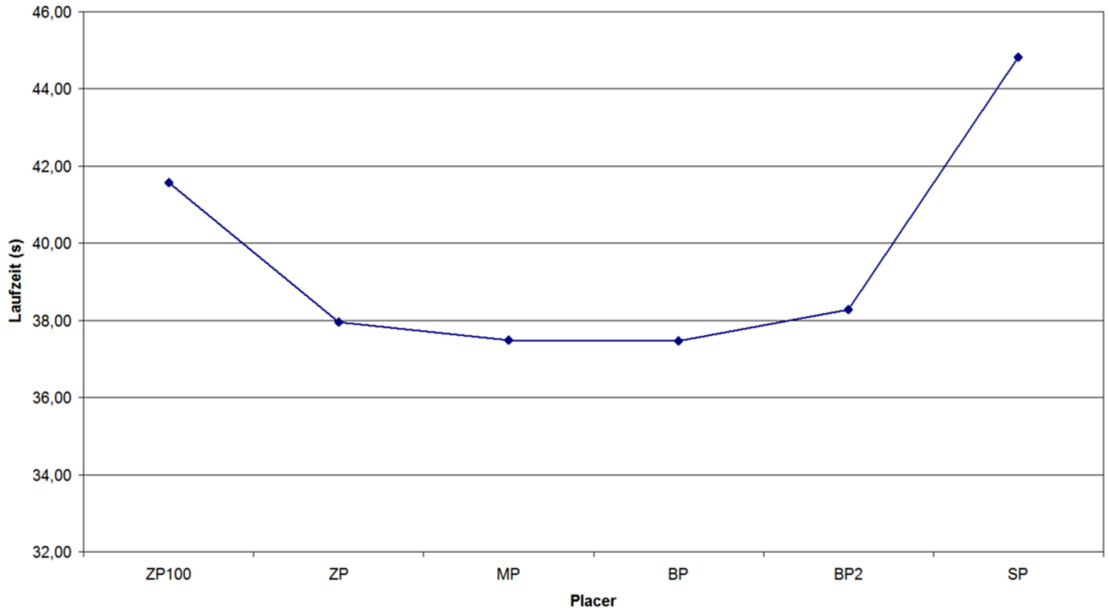

Abbildung 6.14: Laufzeiten bei verschiedenen Platzierungsstrategien in Kombination mit LBCM.

möglich ist, nur weil *SP* etwas mehr Verwaltungsaufwand benötigt, erscheint möglich, wenn man die große Anzahl zu platzierender Knoten berücksichtigt. Die gemessenen Werte aus beiden Abbildungen sind in Tabelle 6.8 aufgeführt.

Zerlegungs-strategie	RM	RM2	ECM	LBCM	SM	SM2	ISM	ISM2
Kreuzungen	566471	563029	653652	562976	573855	573271	570083	572347
Laufzeit	15,47	47,28	15,37	37,48	8,22	8,03	11,42	14,19

Tabelle 6.9: Kreuzungen und Laufzeit bei verschiedenen Zerlegungsstrategien in Kombination mit MP.

Der Vergleich der verschiedenen Zerlegungsstrategien auf den großen Graphen hat die auf den kleinen Graphen erzielten Ergebnisse bestätigt. Wie in Abbildung 6.15 zu sehen ist, sind die Zerlegungsstrategien qualitativ wieder in zwei Gruppen einzuteilen. *LBCM*, *ECM* und *RM* erzielen hier eindeutig die besseren Ergebnisse. Auch die in Abbildung 6.16 sichtbaren Ergebnisse der Laufzeiten decken sich mit denen auf den kleinen Graphen. Die *Solar Merger*-Varianten sind hier wieder eindeutig die schnellsten. In Tabelle 6.9 sind die Werte aus beiden Abbildungen zu sehen.

Die Ergebnisse sind als Bewertung der ursprünglich von den jeweiligen Autoren vorgestellten Verfahren nur eingeschränkt geeignet, da in den jeweiligen Implementationen andere Parameter gewählt wurden und hier nicht betrachtete weitere Ansätze zur Verbes-

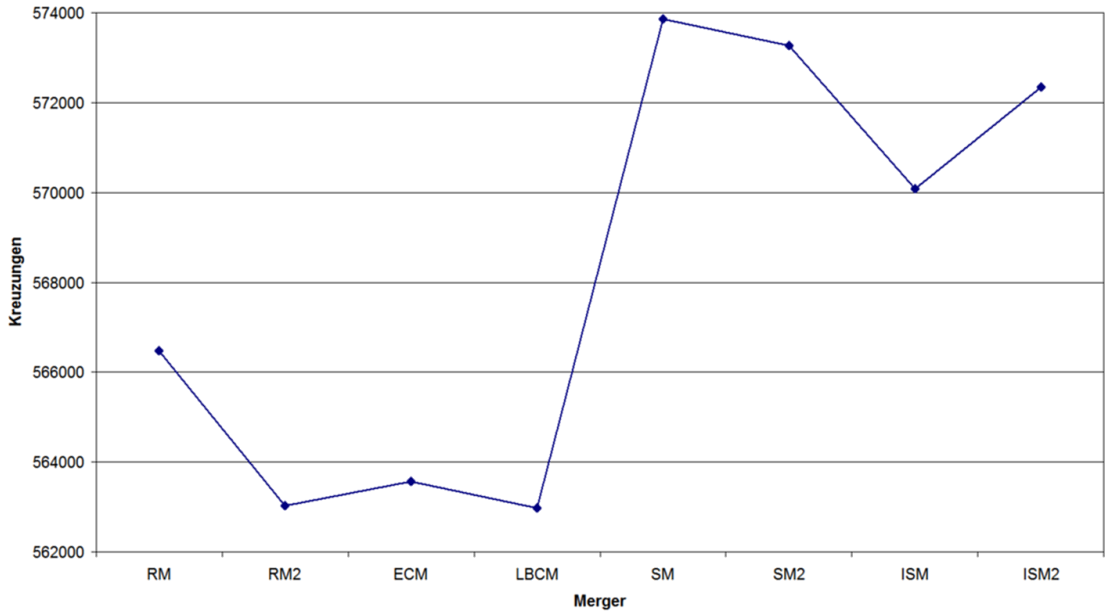

Abbildung 6.15: Kreuzungen bei verschiedenen Zerlegungsstrategien in Kombination mit MP.

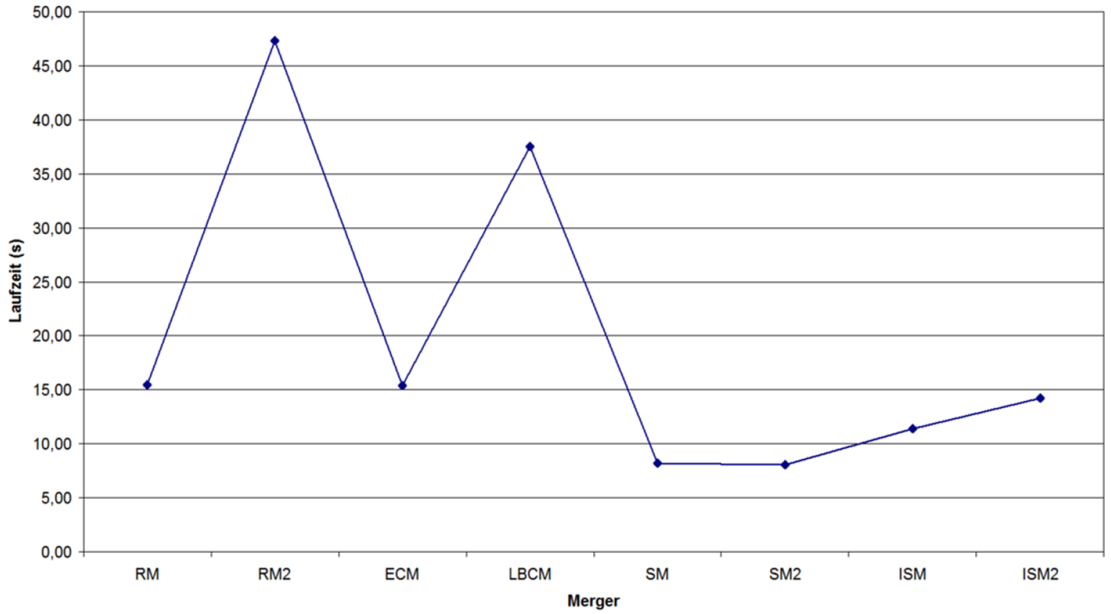

Abbildung 6.16: Laufzeiten bei verschiedenen Zerlegungsstrategien in Kombination mit MP.

serung der Zeichenqualität enthalten sind. Nur die besonders starken Aussagen wie die extreme Laufzeit des *Matching Merger* können übertragen werden.

6.6 Auswirkungen des Postprocessings

Nachdem die meisten Module getestet wurden, konnten im Folgenden besonders vielversprechende Kombinationen für weitere Tests genutzt werden. Als letztes Modul muss noch das Postprocessing festgelegt werden. Das ausgewählte Kräfteverfahren ist in allen Fällen *FME*, da hier erstens die beste Geschwindigkeit und zweitens auch die wenigsten Kantenkreuzungen zu erwarten waren. Die Tests für die Skalierung wurden unter anderen Gesichtspunkten noch einmal durchgeführt. Für folgende Kombinationen wurde auf den großen Graphen getestet, welches Postprocessing die besten Ergebnisse liefert:

- *FME, ISM, BP, factor 2*: Als Nachahmung des Verfahrens GRIP [GaKo02] wurde die Skalierung so gewählt, dass die Anzahl der Kreuzungen möglichst gering ist.
- *FME, SM, SP, factor 4*: Diese Modul-Kombination ist darauf ausgelegt, eine besonders geringe Laufzeit zu erreichen. Die Skalierung mit einem hohen Faktor hat sich im Vortest als zweckmäßig erwiesen.
- *FME, LBCM, BP, no sc*: In Kombination mit dem *Local Biconnected Merger* bietet es sich natürlich an, die Kreuzungen so gut es geht zu minimieren. Eine aktive Skalierung hat sich hier als kontraproduktiv erwiesen. Hohe Laufzeit wurde dabei in Kauf genommen.
- *FME, ECM, BP, no sc*: Diese Variante soll hohe Qualität bei akzeptabler Laufzeit liefern. Auch hier war eine Skalierung nicht nützlich.

Diese Kombinationen werden im Rest der Arbeit durch die Zerlegungsstrategie identifiziert, da sie sich in diesem Punkt eindeutig unterscheiden lassen.

	no Post	Little Post every	Big Post final	Big Post every	FR Post every	FR post final
ECM	515845	545898	539147	541337	599243	590028
LBCM	512301	545450	963207	833237	614140	594387
SM2	554332	564344	556907	511868	597716	633608
ISM	546954	537896	541865	548974	604041	607048

Tabelle 6.10: Anzahl der Kreuzungen bei unterschiedlichen Postprocessin-Strategien.

Die Anzahl der Kreuzungen wird durch ein Postprocessing in fast keinem Fall positiv beeinflusst, wie man in Abbildung 6.17 erkennen kann. Wenn man die genauen Werte in

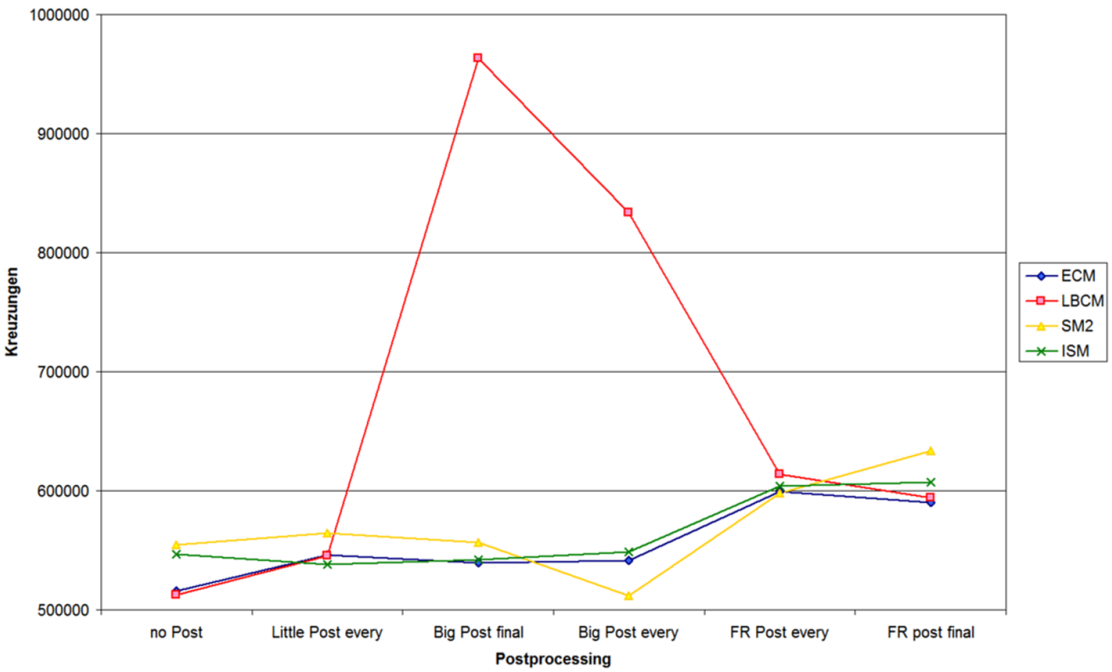

Abbildung 6.17: Anzahl der Kreuzungen bei unterschiedlichen Postprocessing-Strategien.

Tabelle 6.10 betrachtet, sieht man, dass nur für *ISM* und *SM2* eine einzelne Ausnahme existiert in der die Anzahl der Kreuzungen leicht sinkt. Für die eher qualitätsorientierten Ansätze *LBCM* und *ECM* kommt ein Postprocessing aus dieser Auswahl also nicht in Frage, da dieses Manko auch durch die Werte in den anderen Kriterien nicht ausgeglichen wird.

	no Post	Little Post every	Big Post final	Big Post every	FR Post every	FR post final
ECM	449236	711350	706195	727555	483291	475972
LBCM	456576	700459	1081315	890012	488667	487922
SM2	485494	781836	1076247	782110	477074	499782
ISM	484972	751606	904649	747139	488214	493054

Tabelle 6.11: Anzahl der Kanten, die Knoten schneiden, bei unterschiedlichen Postprocessing-Strategien.

In Abbildung 6.18 sieht man, dass auch die Anzahl der Kanten, die einen Knoten schneiden, durch ein Postprocessing in den meisten Fällen vergrößert wird. Auch für *ISM* zeichnet sich nun ab, dass die hier vorgestellten Postprocessing-Ansätze die Qualität nicht verbessern.

Die Anzahl der Knotenüberlappungen konnte, wie Abbildung 6.19 zeigt, durch ein Postprocessing mit *FR* verringert werden. Dies ist jedoch bei weitem nicht genug um die

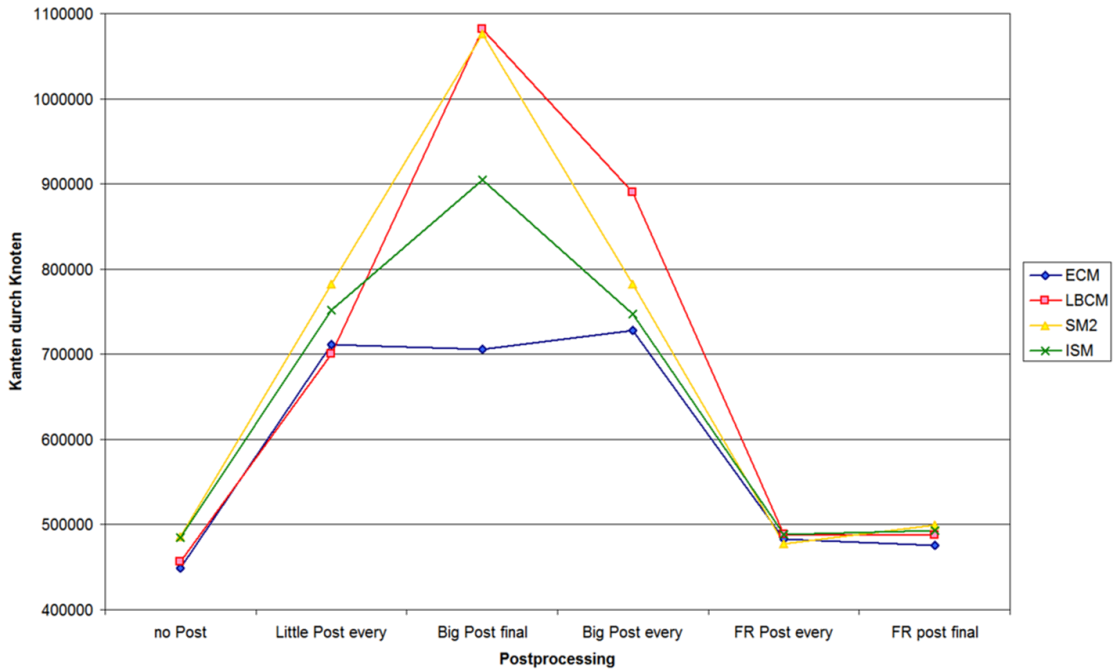

Abbildung 6.18: Anzahl der Kanten, die Knoten schneiden, bei unterschiedlichen Postprocessing-Strategien.

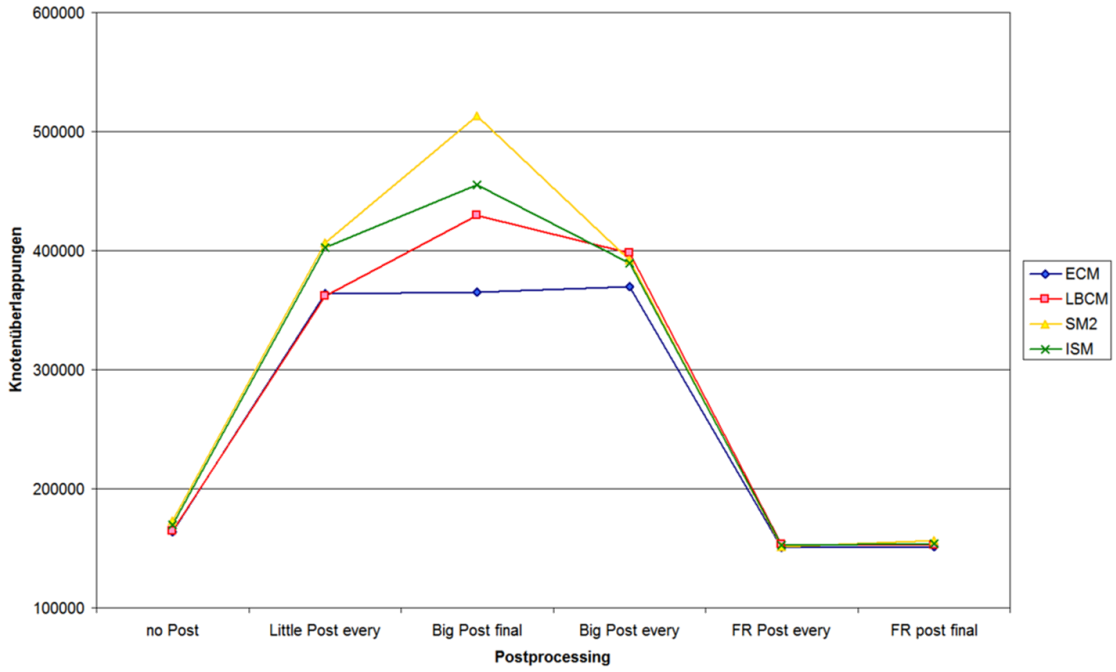

Abbildung 6.19: Anzahl der Knotenüberlappungen bei unterschiedlichen Postprocessing-Strategien.

	no Post	Little Post every	Big Post final	Big Post every	FR Post every	FR post final
ECM	163956	363733	365004	369690	150817	150993
LBCM	164305	361524	429512	397848	153159	152729
SM2	172783	406349	512896	391937	151447	156882
ISM	169620	402859	455366	389290	152436	153655

Tabelle 6.12: Anzahl der Knotenüberlappungen bei unterschiedlichen Postprocessing-Strategien.

Nachteile in den anderen Kriterien auszugleichen. Das besonders schlechte Abschneiden der Variante *Big Post Final* in allen Qualitätskriterien fällt natürlich besonders ins Auge. Es zeigt sich hier, dass ein so einfacher Postprocessing-Ansatz, wie er hier gewählt wurde, nicht erfolgreich ist. Die fürs Postprocessing ausgewählten Kräfteverfahren sollten für diesen Zweck speziell abgestimmt werden.

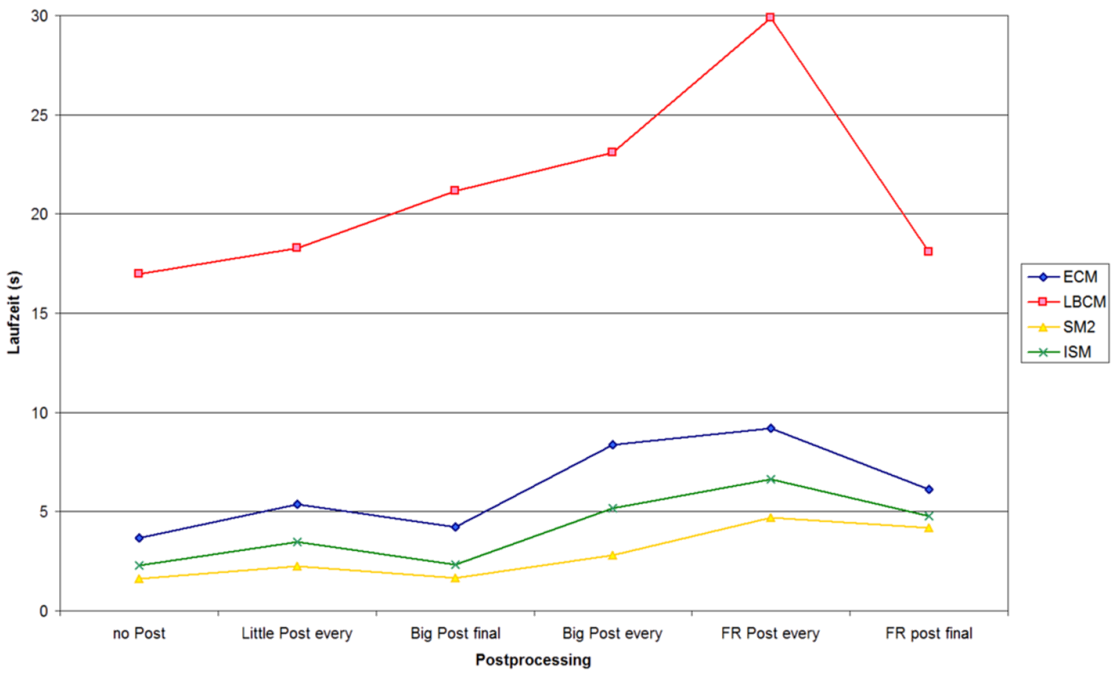

Abbildung 6.20: Laufzeiten bei unterschiedlichen Postprocessing-Strategien.

Abbildung 6.20 zeigt, dass sich auch die Laufzeiten der Gesamtverfahren nicht durch ein Postprocessing verbessert haben, was aber auch nicht zu erwarten war. Die Verfahren werden die folgenden Tests also so wie am Anfang dieses Abschnitts beschrieben und ohne ein Postprocessing absolvieren. Abbildung 6.21 zeigt jedoch, dass für bestimmte Graphen das Postprocessing dennoch sehr gute Ergebnisse liefern kann.

	no Post	Little Post every	Big Post final	Big Post every	FR Post every	FR post final
ECM	3,65347	5,35128	4,21283	8,38721	9,18804	6,12278
LBCM	16,9623	18,2823	21,1745	23,0868	29,8985	18,0619
SM2	1,61324	2,23598	1,64342	2,82096	4,71266	4,19893
ISM	2,29564	3,47378	2,31951	5,15757	6,63307	4,78337

Tabelle 6.13: Laufzeiten bei unterschiedlichen Postprocessing-Strategien.

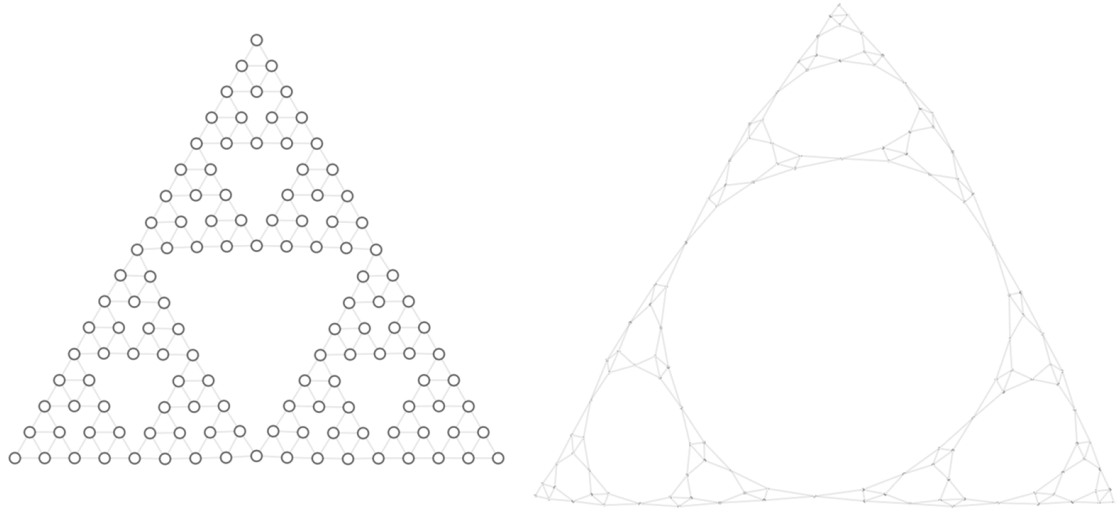

Abbildung 6.21: Der Graph Sierpinski_04.gml, mit und ohne Postprocessing gezeichnet.

6.7 Verdrehungen

Ein in dieser Arbeit ausführlich betrachtetes Problem stellen die durch verlorenen lokalen Zwei-Zusammenhang entstandenen Verdrehungen dar. Besonders gut zu beobachten sind diese bei langgestreckten Grids ab einem Seitenverhältnis von 10:1. Alle bisherigen Multilevel-Verfahren haben Probleme, diese ohne Verdrehungen zu zeichnen. Der *Local Biconnected Merger* wurde im Rahmen dieser Arbeit entwickelt, um dieses Problem zu bekämpfen. In diesem Test sollte herausgefunden werden, ob eine deutliche Verbesserung der Zeichenqualität erreicht werden konnte.

Abbildung 6.22: Ein Grid das verdreht gezeichnet wurde.

Abbildung 6.23: Zeichnung des Grid ohne Verdrehung.

Die Kantenkreuzungen sind hier als Kriterium geeignet, da eine Verdrehung eine große Anzahl neuer Kreuzungen erzeugt. An der durchschnittlichen Anzahl der Kreuzungen lässt sich hier also gut ablesen, ob das Grid häufig verdreht gezeichnet wurde.

Zerlegungs-strategie	RM	RM2	MM	MM2	ECM	LBCM	SM	SM2	ISM	ISM2
Kreuzungen	3220	2773	2747	2941	2785	2105	2806	3102	3050	3141

Tabelle 6.14: Anzahl der Kreuzungen bei verschiedenen Zerlegungsstrategien. Als Platzierungsstrategie wurde hier MP gewählt.

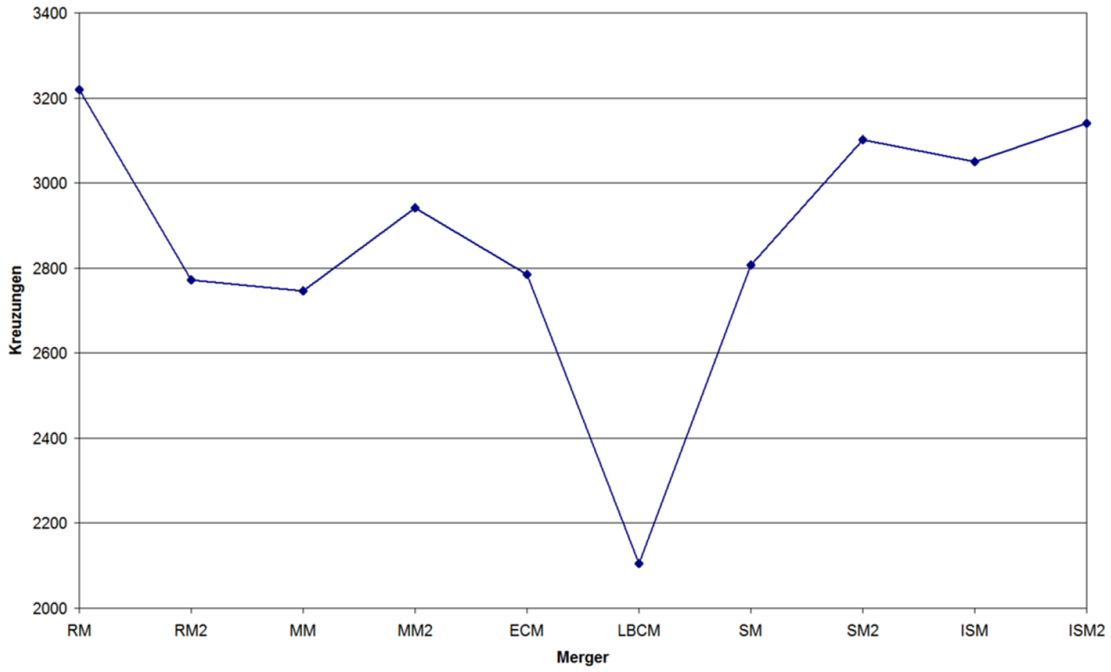

Abbildung 6.24: Anzahl der Kreuzungen bei verschiedenen Zerlegungsstrategien.

Im ersten Test wurden die verschiedenen Zerlegungsstrategien verglichen. Als Platzierungsstrategie wurde hier *MP* gewählt. Verglichen mit dem späteren Test fällt in Abbildung 6.24 auf, dass in allen Fällen mit über 2.000 eine hohe durchschnittliche Anzahl von Kreuzungen zu beobachten ist. Dies liegt daran, dass manche der hier getesteten Grids zusätzliche Kanten enthalten, die es unmöglich machen, das Grid ganz ohne Kreuzungen

zu zeichnen. Es ist dennoch gut zu erkennen, dass durch *LBCM* mit Abstand am wenigsten Kreuzungen erzeugt werden, was darauf zurückzuführen ist, dass Verdrehungen hier deutlich seltener auftreten.

Zeichenverfahren	ECM	LBCM	SM2	ISM	FM3	FMME
Kreuzungen	1569	24	1441	1268	1442	917

Tabelle 6.15: Kreuzungen verschiedener Zerlegungsstrategien. Als Platzierungsstrategie wurde hier MP gewählt.

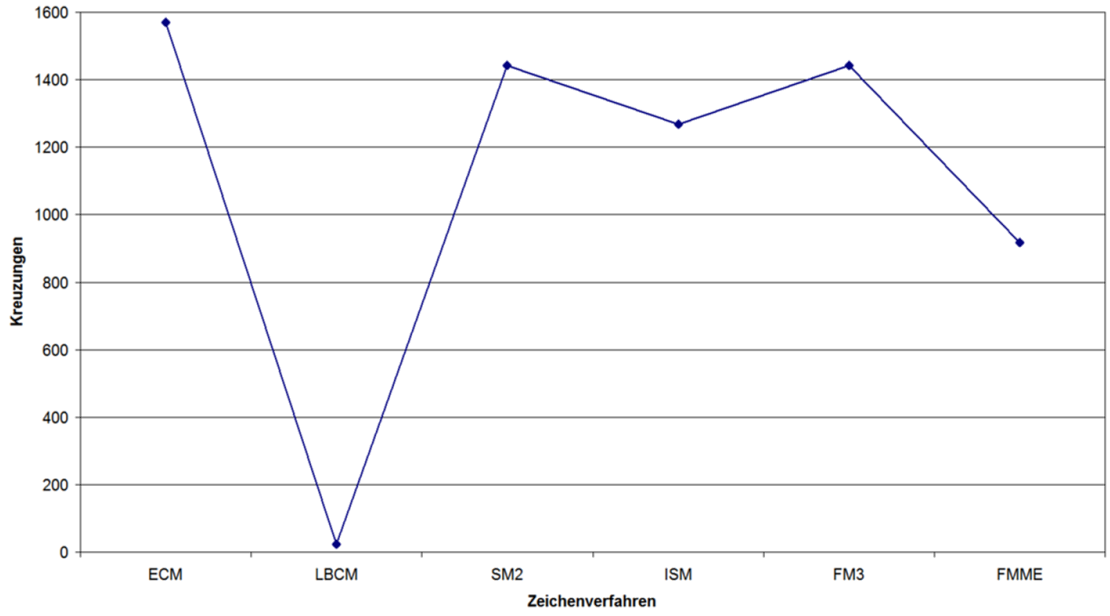

Abbildung 6.25: Auf den langen Grids durch verschiedene Zeichenverfahren erzeugte Kreuzungen.

Im zweiten Test wurden die besten Varianten des *Modular Multilevel Mixer* und die beiden vorliegenden Gesamtverfahren getestet. Hier kamen nur planare Graphen zum Einsatz. Es ist also möglich alle Graphen völlig ohne Kreuzungen zu zeichnen. Alle Kreuzungen entstehen also aus vermeidbaren Verdrehungen. Wie aus Tabelle 6.15 entnommen werden kann, hat es *LBCM* nicht immer geschafft, die Verdrehungen zu vermeiden. Das Verfahren kann eine Verdrehungsfreiheit also nicht garantieren, sondern die Anzahl der Verdrehungen nur deutlich verringern. In Abbildung 6.25 ist die Verbesserung im Vergleich zu den anderen Verfahren deutlich zu sehen.

Wie aus anderen Tests bekannt ist, benötigt *LBCM* jedoch eine hohe Laufzeit, um diese guten Ergebnisse zu erzielen. *ISM* erzeugt von den anderen getesteten *Modular Multilevel Mixer*-Varianten im letzten Test die besten Ergebnisse. In einem weiteren Test wurde nun überprüft, ob *ISM* nicht eine ähnlich gute Qualität liefert, wenn genau so viel Laufzeit zur

Verfügung steht. Hier konnte eine Verbesserung auf 1.320 Kreuzungen festgestellt werden. Dies ist im Vergleich mit den durchschnittlich 24 Kreuzungen von *LBCM* noch immer sehr viel. Es kann also festgehalten werden, dass *LBCM* die Anzahl der Verdrehungen auf langgestreckten Grids und ähnlichen Graphen deutlich reduzieren kann. Diese Verbesserung wird nicht einfach durch höhere Laufzeit erreicht, sondern durch die grundlegend andere Zerlegungsstrategie, die den Verlust des lokalen Zwei-Zusammenhangs verhindert.

6.8 Aussagen zur theoretischen Laufzeit

Die theoretische Worst-Case-Laufzeit für den *Modular Multilevel Mixer* ist nicht besonders gut, insbesondere wenn man den Vergleich mit Hachuls *FM*^{3} vor Augen hat. In Hachuls Dissertation [Hac05] wurde eine Laufzeit von $O(|V| \cdot log(|V| + |E|))$ bewiesen.

Diese gute theoretische Laufzeit kann schon nicht erreicht werden, sobald in der Kräftephase ein anderes Verfahren als *NMM* eingesetzt wird, da für kein anderes Verfahren eine subquadratische obere Schranke bewiesen wurde. Es gibt jedoch noch weitere Punkte, die eine höhere Laufzeit erzeugen können. Da für jede Kante, die aus dem Graphen entfernt wird, geprüft werden muss, ob die Kante bei der Verschmelzung mit einer anderen vereinigt werden soll, müssen in vollständigen Graphen für jede entfernte Kante $O(|V|)$ Kanten geprüft werden. Da im Laufe der Verfeinerungsphase nahezu alle Kanten gelöscht werden, ergibt dies eine Laufzeit von $O(|V| \cdot |E|)$.

Außerdem müssen alle verschobenen Kanten gespeichert werden, wodurch auch der Speicherplatzbedarf im Worst-Case $O(|V| \cdot |E|)$ betragen kann. Dies ist möglich, wenn bei nahezu jeder der $O(|V|)$ Verschmelzungen $O(|E|)$ Kanten verschoben werden.

Ein weiteres Problem ist, dass die Anzahl der Verfeinerungsstufen nur durch $O(|V|)$ beschränkt ist. Diese Schranke wird vom *Matching Merger* in extremen Fällen sogar erreicht.

Insgesamt muss im Worst-Case damit gerechnet werden, dass die Verfeinerungsphase einen Laufzeit- und Speicherplatzbedarf von $O(|V| \cdot |E|)$ hat. Jeder der $O(|V|)$ Kräftephasen kann je nach Kräfteverfahren eine Worst-Case-Laufzeit von $O(|V|^3)$ haben, da bereits eine Iteration theoretisch eine Laufzeit von $O(|V|^2)$ benötigen kann. Die Erwartete Anzahl der Iterationen wird oft mit $O(log(|V|))$ abgeschätzt, es gibt aber keinen Grund den Worst-Case nicht auf $O(|V|)$ zu schätzen. Die Worst-Case-Gesamtlaufzeit des *Modular Multilevel Mixer* kann je nach Modulauswahl also $O(|V|^4)$ betragen, wobei Speicherplatz in der Größenordnung von $O(|V| \cdot |E|)$ benötigt wird.

Die praktische Laufzeit ist jedoch in den meisten hier getesteten Fällen stets schneller als quadratisch und auch der Worst-Case des Speicherplatzbedarfs tritt bei geschickter Implementierung in der Praxis nicht auf. Beim Speicherplatz kann man lineares Verhalten im Bereich von $O(|V| + |E|)$ erwarten. Die subquadratische Laufzeit in den getesteten Fällen lässt sich daraus ableiten, dass die meisten Varianten des *Modular Multilevel Mixer*

eine geringere Laufzeit haben als *FM*^{3}, für den eine subquadratische Laufzeitschranke bekannt ist. Der Laufzeitvergleich ist im nächsten Abschnitt zu finden.

6.9 Vergleich der Gesamtverfahren

Als letzter Test wurde der Vergleich mit den zur Verfügung stehenden Gesamtverfahren *FM*^{3} und *FMME* durchgeführt. Die hier getesteten Modulkombinationen sind die gleichen wie beim Postprocessing-Test. Es werden also die Zerlegungsstrategien *ECM*, *LBCM*, *SM2* und *ISM* verwendet.

	ECM	LBCM	SM2	ISM	FM3	FMME
Standard-abweichung der Winkel	61,4499	61,7808	62,7839	62,7431	61,2291	62,5109
Entfernung von Kantenlänge	23,4206	23,6917	35,6515	32,6458	0,0248	30,1739
Knotenüber-lappungen	163956	164305	172783	169620	71619	68037
Laufzeit (s)	3,65347	16,9623	1,61324	2,29564	6,77759	1,30357
Kanten durch Knoten	449236	456576	485494	484972	624601	268042
Zeichenfläche	60526	64584	821881	766418	2120090	1035780
Kreuzungen	515845	512301	554332	546954	1167695	554192
Standard-abweichung der Kantenlängen	1,61739	1,66636	1,89948	1,84529	1,90728	3,12993
Knoten-verteilung	14631000	15456800	17717800	17714600	16061700	17237600

Tabelle 6.16: Die verschiedenen Zeichenverfahren im Vergleich.

Zunächst wurde die Qualität der Verfahren auf den großen Graphen getestet. In Abbildung 6.26 sind die Werte aller Verfahren im Vergleich zu sehen. Die genauen Werte können der Tabelle 6.16 entnommen werden. Auch hier zeigt sich wieder, dass die Verfahren sehr unterschiedliche Stärken und Schwächen haben. Die kleinste Anzahl von Kreuzungen wird bei den *Modular Multilevel Mixer*-Varianten erreicht, dicht gefolgt von *FMME*, während *FM*^{3} hier deutlich schlechtere Ergebnisse liefert. Auch bei der Laufzeit ist nur *LBCM* langsamer als *FM*^{3}. Es war nicht zu erwarten, dass der *Modular Multilevel Mixer* bei der Laufzeit so gut abschneidet, da die Verwaltung der Module und die dafür nötige Flexibilität keine auf die Laufzeit ausgelegte Optimierung erlaubt. Da der größte Teil der

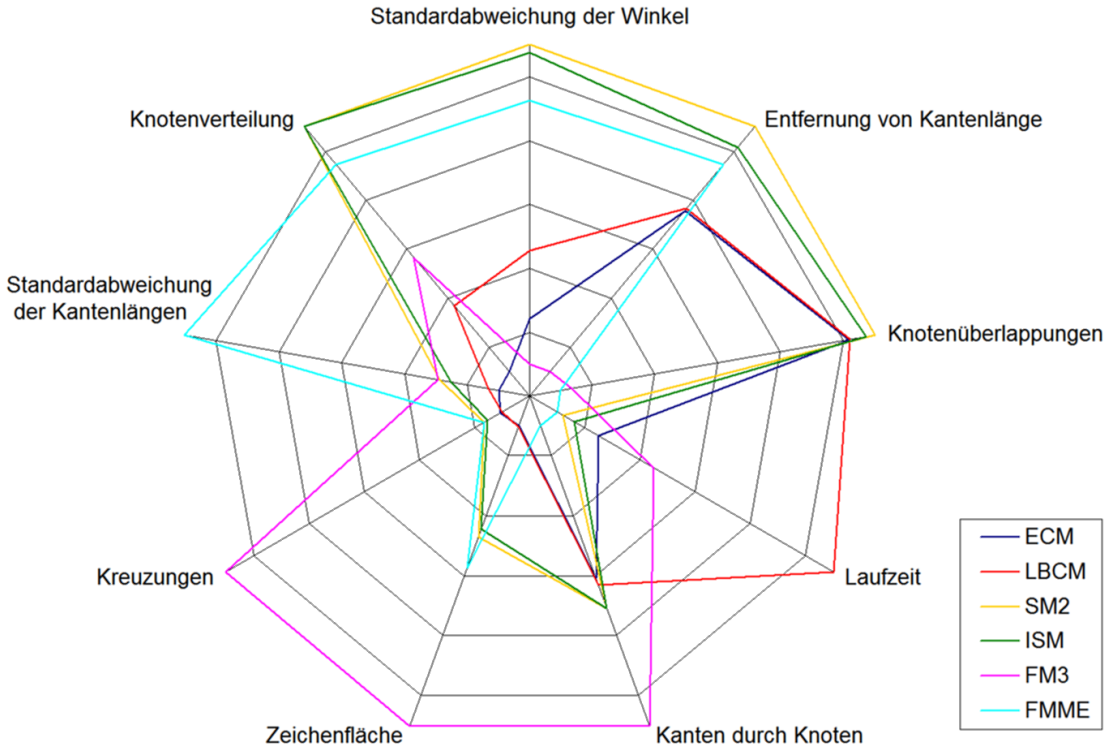

Abbildung 6.26: Vergleich der besten Modulkombinationen mit FM^3 und FMME. Die Werte sind besser, je näher sie am Mittelpunkt aufgetragen sind.

Rechenzeit jedoch in der Kräftephase anfällt, ist die gute Laufzeit natürlich dadurch zu erklären, dass *FME* als Kräfteverfahren im *Modular Multilevel Mixer* genutzt wurde. Die beste Laufzeit wird klar von *FMME* erreicht.

Bei den Knotenüberlappungen schneiden alle *Modular Multilevel Mixer*-Varianten schlecht ab, da ein gutes Postprocessing fehlt. Bei der Anzahl der Kanten, die einen Knoten schneiden, liegen alle *Modular Multilevel Mixer*-Varianten im Mittelfeld. Das beste Ergebnis erreicht hier *FMME* während *FM*^{3} den schlechtesten Wert erzeugt.

Zeichenverfahren	ECM	LBCM	SM2	ISM	FM3	FMME
Laufzeit (s)	24,3401	350,866	14,8308	24,4331	71,2782	10,2502

Tabelle 6.17: Laufzeiten der Verfahren auf extrem großen Graphen.

Die Laufzeit ist insbesondere für die extrem großen Graphen interessant. In Abbildung 6.27 wurde der Wert für *LBCM* abgeschnitten, da sonst keine Unterschiede zwischen den schnellen Verfahren mehr zu erkennen wären. Auch auf den extrem großen Graphen hat *FMME* eindeutig das beste Laufzeitverhalten gezeigt. In der Tabelle 6.17 kann man ablesen, dass der *LBCM* fünf mal so viel Zeit benötigt hat wie *FM*^{3}. Die Laufzeit wird

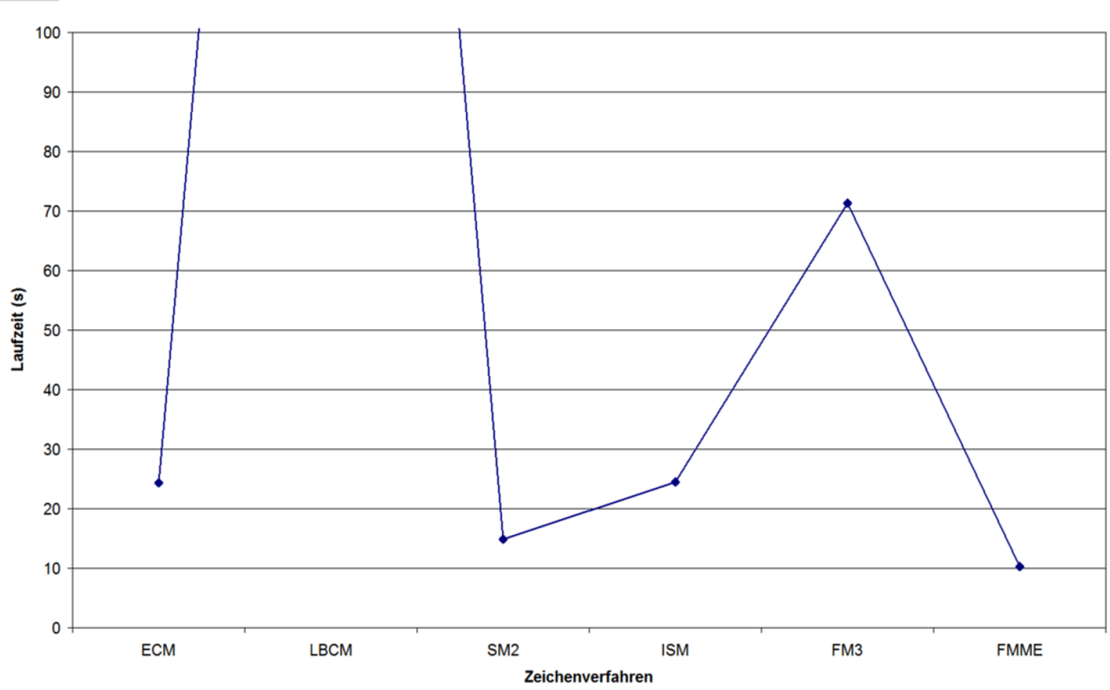

Abbildung 6.27: Laufzeiten der verschiedenen Verfahren auf den extrem großen Graphen.

also bei den extrem großen Graphen nicht zu hoch, so dass hier im Falle von extremen Verdrehungen *LBCM* eine gute Alternative ist.

Die extrem großen Graphen mit teils weit über 50.000 Kanten können zwar durch die Verfahren problemlos und auch recht schnell gezeichnet werden, eine Auswertung der wichtigsten Qualitätskriterien, wie zum Beispiel der Kantenkreuzungen, ist jedoch nicht mehr in annehmbarer Zeit möglich. Ein Vergleich der Qualität ist dennoch möglich, indem man sich die resultierenden Zeichnungen anschaut. Da dies jedoch sehr subjektiv ist, wird der Leser vielleicht eine andere Meinung dazu haben, welche Zeichnung die beste ist.

Da sich in den vorherigen Tests gezeigt hat, dass die berechenbaren Qualitätskriterien keine zuverlässige Aussage darüber erlauben, welche Zeichnung tatsächlich die übersichtlichste ist, ist eine Bewertung durch den Menschen nach wie vor die zuverlässigste. Bei extrem großen Graphen ist das Ziel meist die Hervorhebung ihrer Struktur. Die folgenden Bilder zeigen ein paar Zeichnungen von extrem großen Graphen durch verschiedene Kräfteverfahren. Hier wird deutlich, dass der *Modular Multilevel Mixer*, auch verglichen mit *FMME* und *FM*^{3}, eine gute Zeichenqualität liefert.

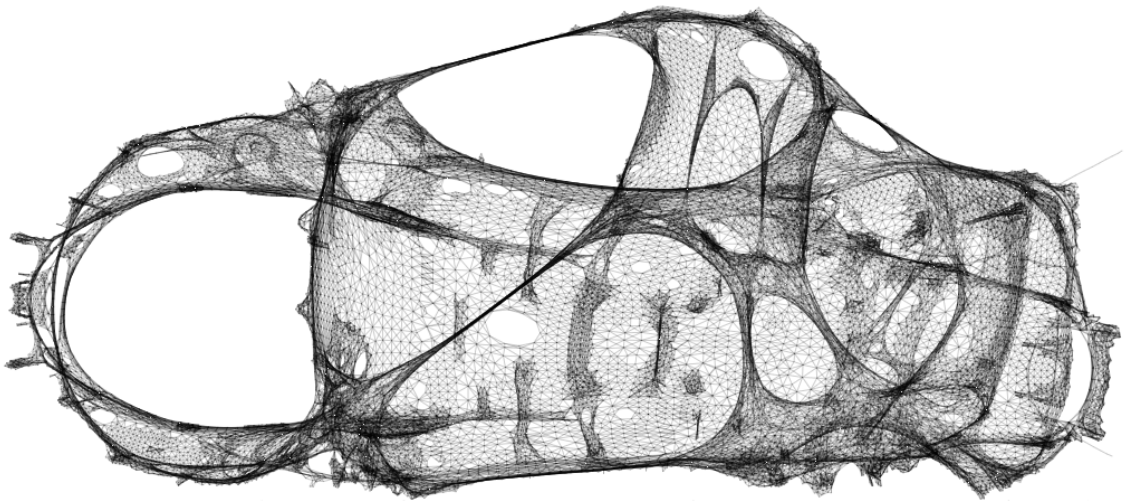

Abbildung 6.28: Zeichnung der größten Komponente von fe_body.gml durch FM^3.

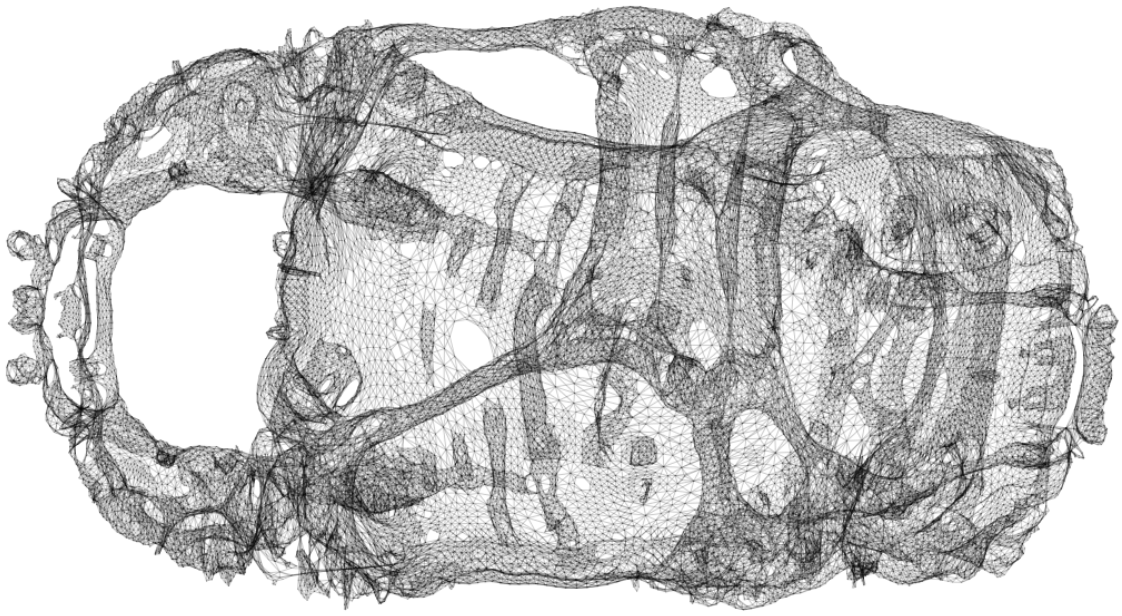

Abbildung 6.29: Zeichnung der größten Komponente von fe_body.gml durch ECM. Die Proportionen des Autos werden hier besser dargestellt.

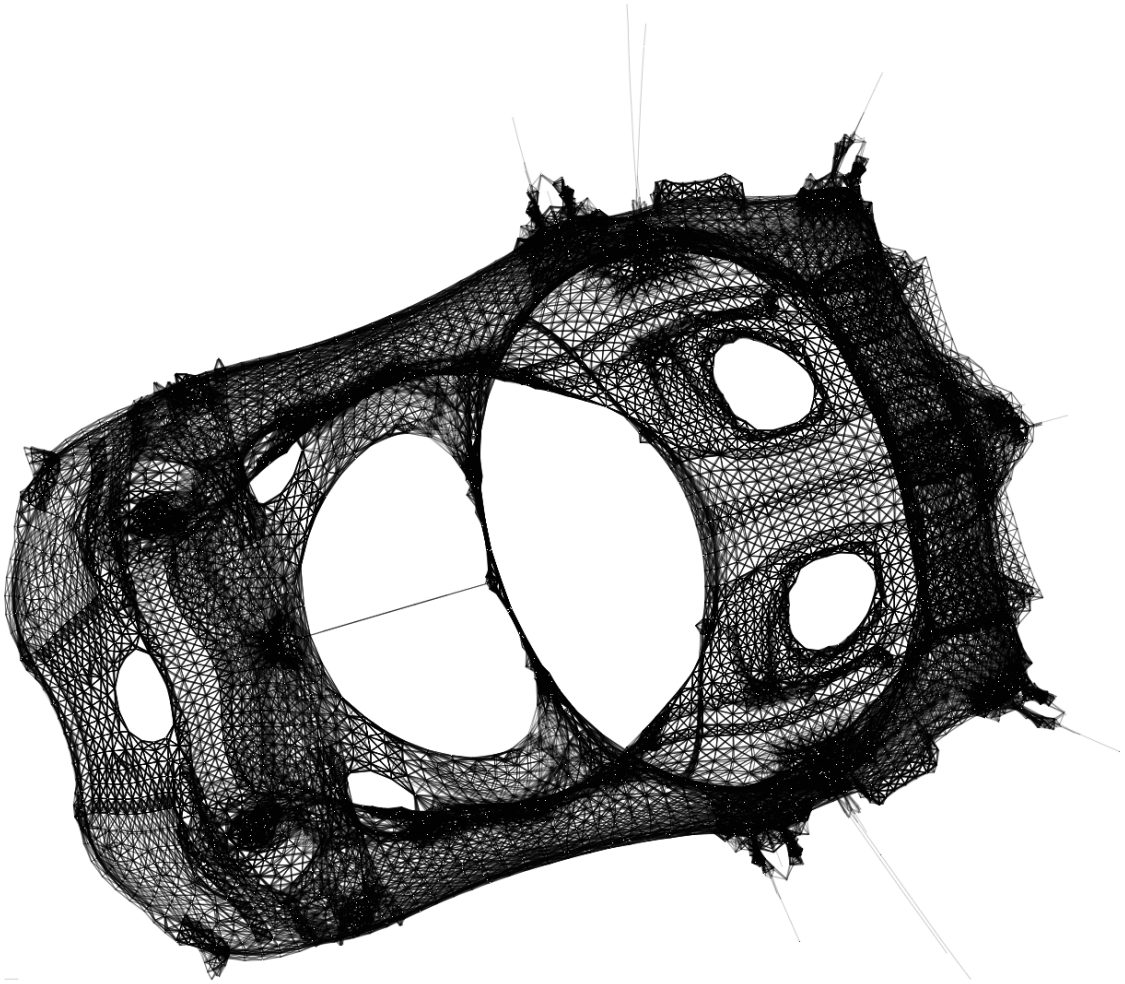

Abbildung 6.30: Zeichnung des Graphen bcsstk31.gml durch FM^3.

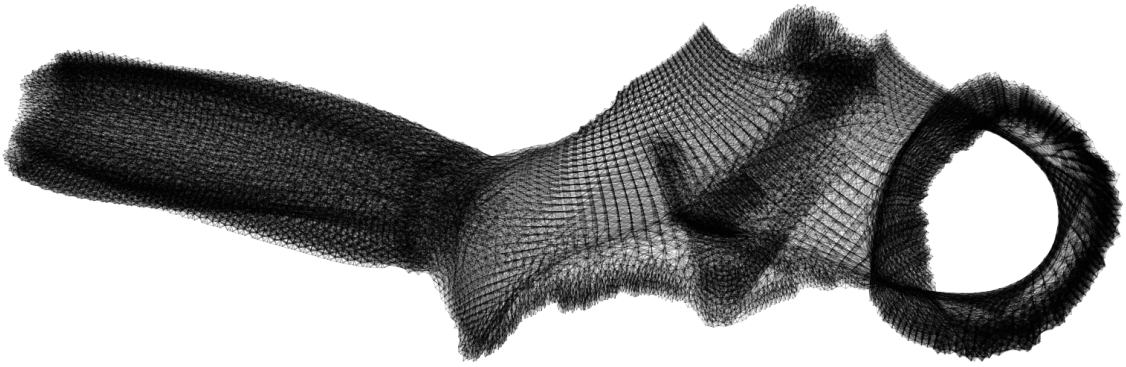

Abbildung 6.31: Zeichnung des Graphen brack2.gml durch FM^3.

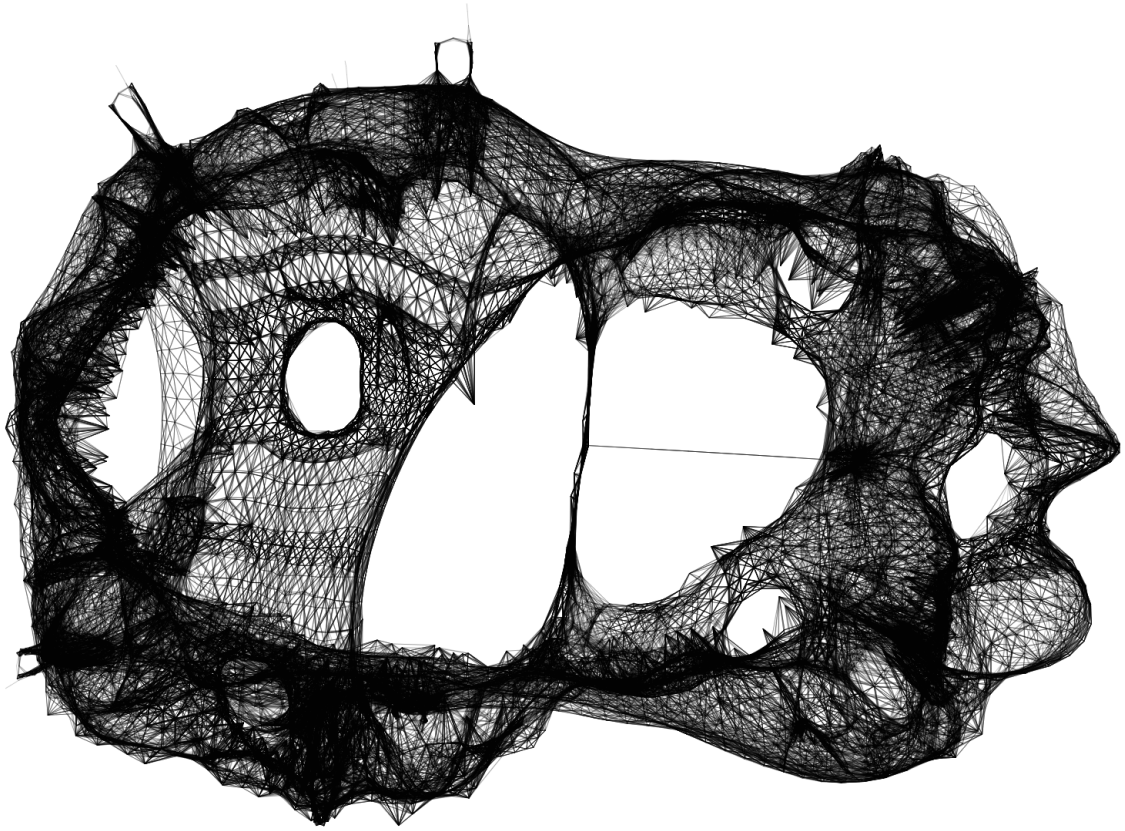

Abbildung 6.32: Zeichnung des Graphen bcsstk31.gml durch ECM. Diese Zeichnung ist schlechter als die von FM^3.

Abbildung 6.33: Zeichnung des Graphen brack2.gml durch ECM.

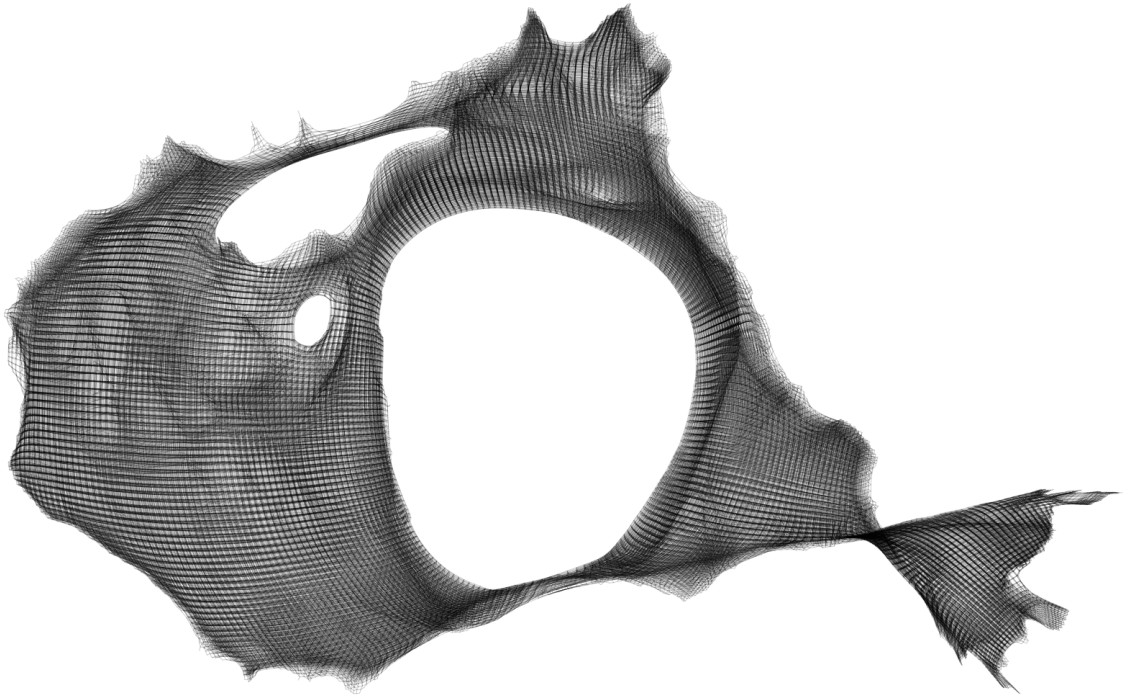

Abbildung 6.34: Zeichnung des Graphen fe_ocean.gml durch FM^3. Hier fällt die unnötige Verdrehung auf.

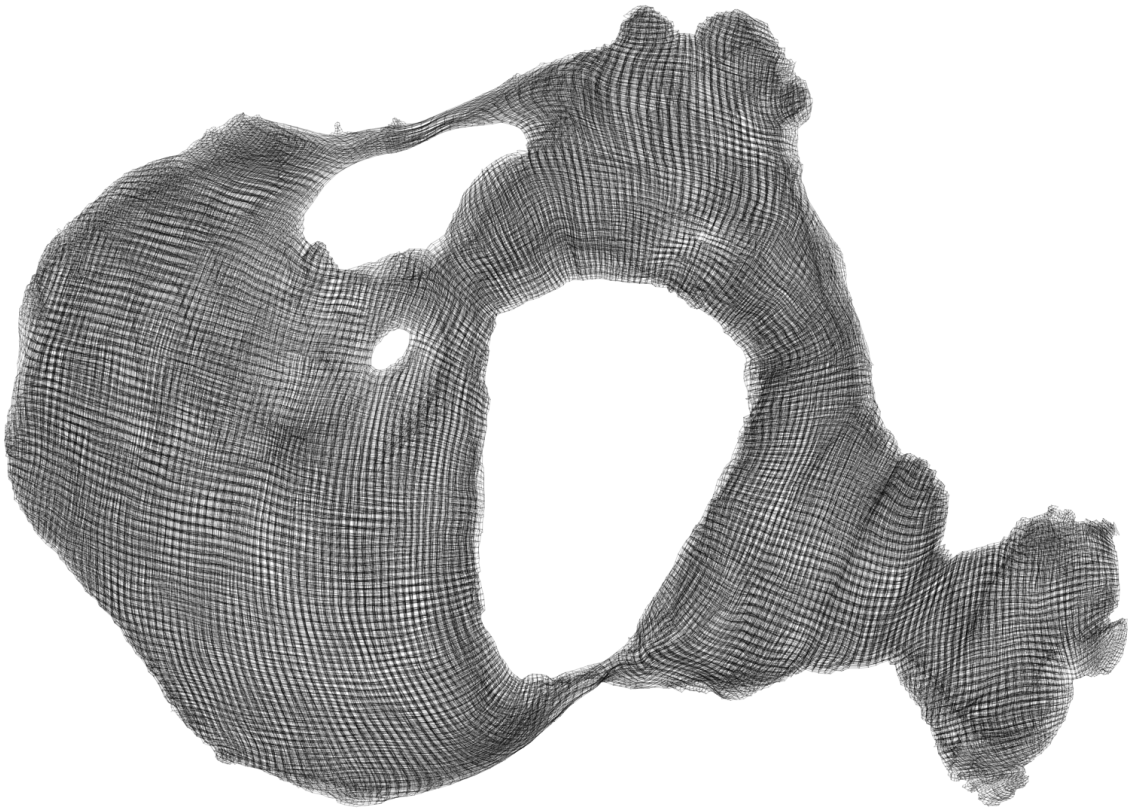

Abbildung 6.35: Zeichnung des Graphen fe_ocean.gml durch ECM.

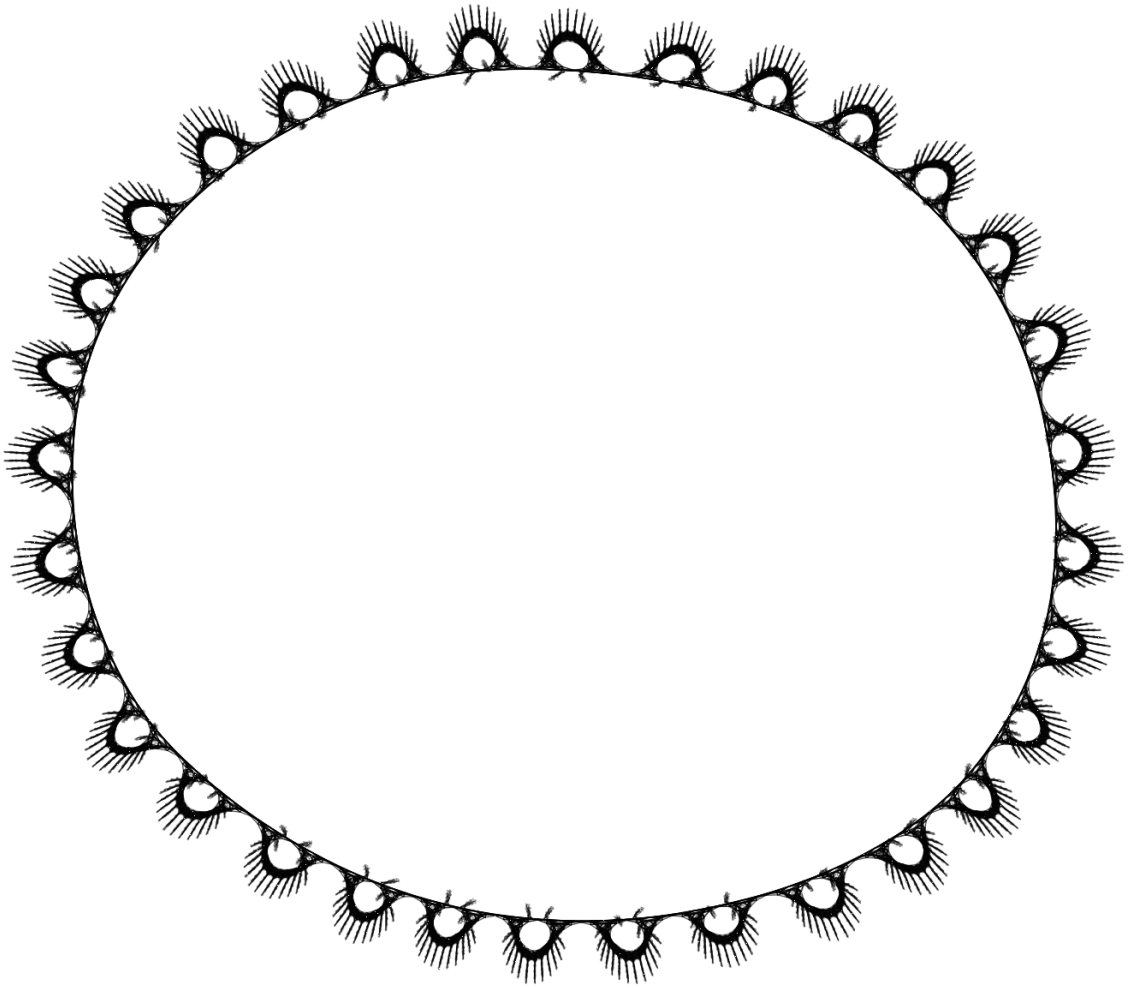

Abbildung 6.36: Zeichnung des Graphen finan512.gml durch FM^3

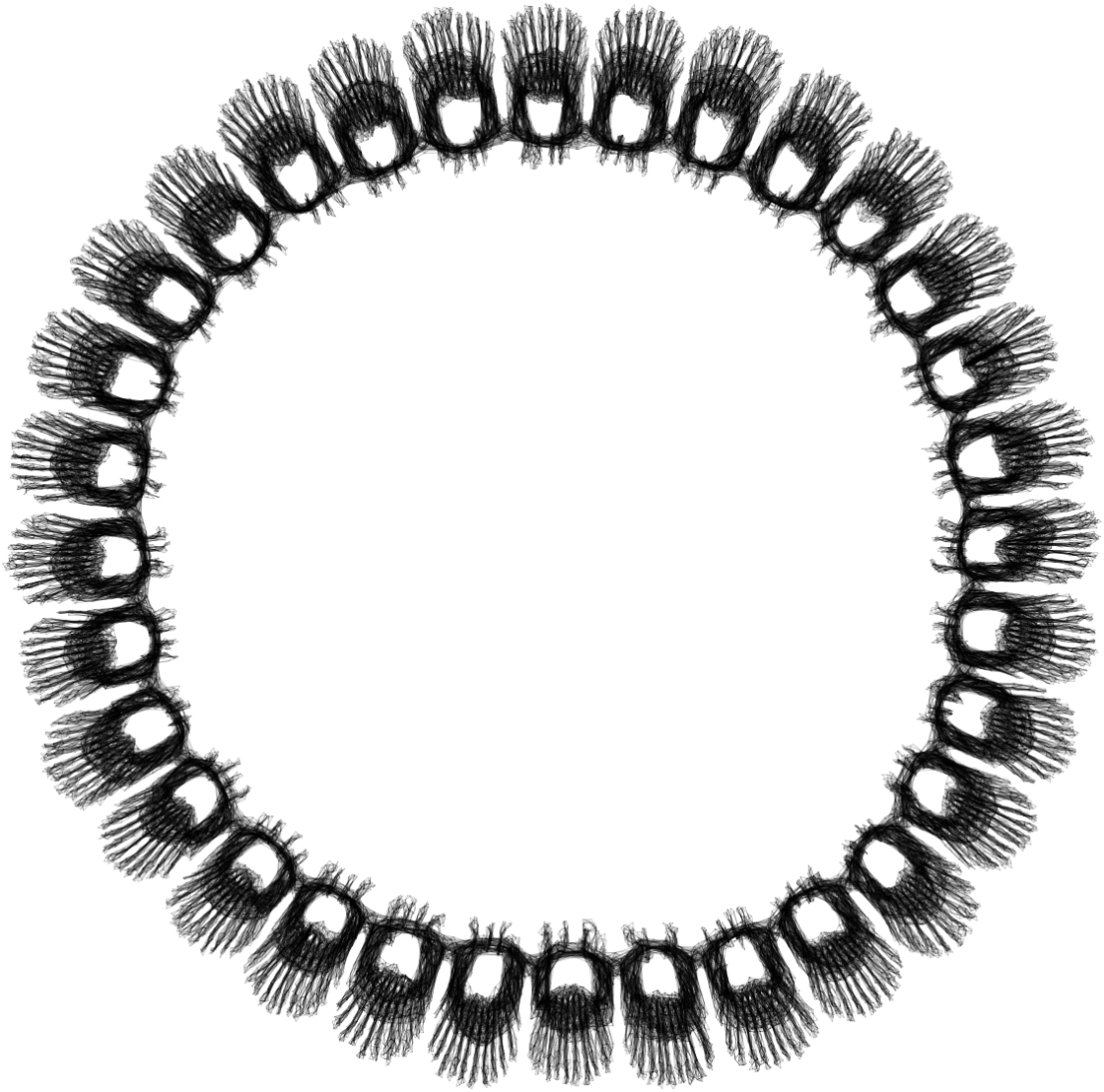

Abbildung 6.37: Zeichnung des Graphen finan512.gml durch ECM. Durch die weniger stark gestreckte Zeichnung sind hier die Details viel besser zu erkennen.

Kapitel 7

Fazit und Ansätze für weitere Arbeiten

Der Vergleich der verschiedenen Zerlegungsstrategien, Platzierungsstrategien und Kräfteverfahren hat keinen eindeutigen Sieger in allen Kriterien hervorgebracht. Jede Kombination hat ihre eigenen Stärken und Schwächen, so dass oft nur die genaueren Anforderungen des Anwenders den Ausschlag geben können, welches Verfahren bevorzugt werden sollte. Der hier durchgeführte Vergleich kann dem Anwender jedoch eine Hilfe bei der Auswahl der für ihn passenden Verfahren sein. Die Laufzeiten der hier getesteten Verfahren sind mit wenigen Ausnahmen auch für extrem große Graphen im Rahmen von wenigen Minuten anzusiedeln. Hier sollte der Anwender sich also nicht nur vom schnellsten Verfahren beeindrucken lassen, da mit etwas zusätzlicher Rechenzeit oft deutlich bessere Ergebnisse erzielt werden.

Auch kann allgemein festgehalten werden, dass die Qualität der Zeichnungen durch jeweils an die aktuellen Kriterien und den zu zeichnenden Graphen angepasste Parameter deutlich erhöht werden kann. Dieser zusätzliche Arbeitsaufwand sollte bei wichtigen Graphen nicht gescheut werden. Der im Rahmen dieser Arbeit entstandene *Modular Multilevel Mixer* stellt eine voll funktionsfähige Umgebung zum schnellen und qualitativ hochwertigen Zeichnen von Graphen dar. Entsprechend der präsentierten Ergebnisse der Experimente kann der *Modular Multilevel Mixer* leicht an gegebene Anforderungen und einzelne Qualitätskriterien angepasst werden.

Ein gutes Beispiel dafür, dass starke Verbesserungen der Qualität auf Kosten der Laufzeit möglich sind, ist der *Local Biconnected Merger*. Durch diesen neuen Ansatz konnte die Anzahl der Kreuzungen in vielen Fällen stark reduziert und die Menge der durch kräftebasierte Verfahren mit gutem Ergebnis zu zeichnenden Graphen erweitert werden. Auch für Graphen, bei denen die entsprechenden Verdrehungen selten sind, ist der *Local Biconnected Merger* in der Praxis vorteilhaft, da nun weniger Versuche notwendig sind, um eine Zeichnung ohne Verdrehungen zu erhalten.

Als zukünftige Arbeit, um den in dieser Arbeit verfolgten Multilevel-Ansatz weiter zu verbessern, gibt es mehrere Ansatzpunkte. So können weitere neue Zerlegungsstrategien die Zeichnungsqualität sicher noch verbessern.

Auf der Ebene der Kräfteverfahren wäre ein Verfahren interessant, das ohne weitreichende abstoßende Kräfte auskommt. Für die Laufzeit würde dies einen enormen Gewinn bedeuten, während auch für die Qualität im lokalen Bereich Vorteile zu erwarten sind. Verluste bei der Darstellung der globalen Struktur des Graphen können – eventuell durch geschickte Skalierung zwischen den Kräftephasen – vermieden werden. Der größte Vorteil bei diesem Ansatz wäre der Wegfall der weitreichenden abstoßenden Kräfte, die im Multilevel-Ansatz problematisch sind und mehr Probleme verursachen, als sie durch Entfaltung des Graphen lösen können. Selbst wenn dieses Verfahren nicht geeignet wäre, die globale Struktur des Graphen darzustellen, wäre ein produktiver Einsatz im Postprocessing sicher möglich.

Beim Postprocessing ist allgemein noch Arbeit nötig, da man von einem Zeichenverfahren zumindest erwarten sollte, dass Knoten sich nicht überlappen. Dies ist aber im Rahmen dieser Arbeit sowie bei den beobachteten Multilevel-Verfahren nicht der Fall. Insbesondere beim *Modular Multilevel Mixer* besteht hier noch Verbesserungspotential. Ein gezieltes Postprocessing mit sehr lokalen starken abstoßenden Kräften würde dieses Problem zuverlässig bekämpfen, ohne die anderen Qualitätskriterien stark zu beeinflussen.

Weitere Verbesserungen sind auch in Hinblick auf den *Local Biconnected Merger* möglich. Eine leichte Veränderung, deren Auswirkungen auf Qualität und Laufzeit noch nicht untersucht wurde, wäre den *Local Biconnected Merger* nur auf den kleinsten Verfeinerungsstufen zu nutzen. Auch eine Kombination des lokalen Zwei-Zusammenhangs-Tests mit anderen Zerlegungsstrategien wurde noch nicht untersucht und könnte die Qualität der Zeichnungen weiter verbessern.

Ein Laufzeitproblem ergibt sich aus der Menge der durch den *Local Biconnected Merger* erzeugten Verfeinerungsstufen. Dieses Problem sollte sich leicht bekämpfen lassen, indem einige der Stufen zusammengefasst werden, was die Laufzeit des Gesamtverfahrens deutlich verbessern würde. Ob dies Qualitätseinbußen zur Folge hat sollte dabei auch untersucht werden.

Für die Qualität der resultierenden Zeichnung wäre es interessant, die Gleichmäßigkeit der Verfeinerungsstufen beim *Local Biconnected Merger* zu verbessern. Bei der momentanen Implementierung kann es leicht passieren, dass die Verschmelzungen sehr unregelmäßig über den Graphen verteilt sind. Hierzu sind jedoch zunächst genauere Erkenntnisse über den Einfluss der Regelmäßigkeit der Verfeinerungsstufen auf die Qualität der Zeichnung nötig.

Das Potential des Multilevel-Ansatzes ist noch nicht ausgeschöpft. Nach den in dieser Arbeit vorgestellten sind auch in Zukunft weitere Verbesserungen in Qualität und Laufzeit zu erwarten.

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[DaHa96] Davidson, R. und D. Harel: *Drawing graphics nicely using simulated annealing.* ACM Trans. Graph., 15(4):301–331, 1996.

[DeTo81] Devroye, L. und G.-T. Toussaint: *A Note on Linear Expected Time Algorithms for Finding Convex Hulls.* COMPUTING, 26:361–366, 1981.

[Ead84] Eades, P.: *A heuristic for graph drawing.* Congressus Numerantium, 42:149–160, 1984.

[FLM98] Frick, A., A. Ludwig und H. Mehldau: *A Fast Adaptive Layout Algorithm for Undirected Graphs.* In: *Graph Drawing (Proc. GD '94) – LNCS 894*, Seiten 388–403, 1998.

[FrRe91] Fruchtermann, T.-M.-J. und E.-M. Reingold: *Graph drawing by force-directed placement.* Software – Practice and Experience, 21:1129–1164, 1991.

[GaKo02] Gajer, P. und S.-G. Kobourov: *GRIP: Graph Drawing with Intelligent Placement.* Journal of Graph Algorithms and Applications, 6(3):203–224, 2002.

[Gro09] Gronemann, M.: *Engineering the Fast-Multilevel-Multipole Method for multicore and SIMD architectures.* Diplomarbeit, Technische Universität Dortmund, Fachbereich Informatik – Lehrstuhl für Algorithm Engineering, Februar 2009.

[Hac05] Hachul, S.: *A Potential–Field–Based Multilevel Algorithm for Drawing Large Graphs.* Dissertation, Universität zu Köln, Mathematisch–Naturwissenschaftliche Fakultät, 2005.

[HaJu05] Hachul, S. und M. Jünger: *Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm.* Lecture Notes in Computer Science, 3383:285–295, 2005.

[HaJu05b] Hachul, S. und M. Jünger: *Large-Graph Layout with the Fast Multipole Multilevel Method.* ACM Transactions on Graphics, Seiten 203–224, 2005.

[HaKo02] Harel, D. und Y. Koren: *A Fast Multi-Scale Method for Drawing Large Graphs.* Journal of Graph Algorithms and Applications, 6(3):179–202, 2002.

[HaKo02b] Harel, D. und Y. Koren: *Graph Drawing by High–Dimensional Embedding.* Lecture Notes in Computer Science, 2528:207–219, 2002.

[KaKa88] Kamada, T. und S. Kawai: *Automatic display of network structures for human understanding.* Technical Report 88–007, Department of Information Science, University of Tokyo, 1988.

[KaKa89] Kamada, T. und S. Kawai: *An Algorithm for Drawing General Undirected Graphs.* Information Processing Letters, 31(1):7–15, 1989.

[OGDF] *Open Graph Drawing Framework.* http://ogdf.net.

[OPTE] *The Opte Project.* http://www.opte.org/maps/.

[SCHL] *Seltsame Schleifen.* http://www.seltsame-schleifen.com.

[STT81] Sugiyama, K., S. Tagawa und M. Toda: *Methods for Visual Understanding of Hierarchical System Structures.* IEEE Transactions on Systems, Man, and Cybernetics, SMC-11:109–125, 1981.

[Wal03] Walshaw, C.: *A Multilevel Algorithm for Force-Directed Graph-Drawing.* Journal of Graph Algorithms and Applications, 7(3):253–285, 2003.

[Wen97] Wenger, R.: *Randomized Quick Hull.* Algorithmica, 17, 1997.