

Computing Crossing Numbers: Berechnen von Kreuzungszahlen

Markus Chimani

Lehrstuhl für Algorithm Engineering, Fakultät für Informatik
Technische Universität Dortmund
markus.chimani@tu-dortmund.de

Abstract: In diesem Artikel betrachten wir das Problem der sogenannten *Kreuzungszahl* eines Graphen, d.h. die Anzahl von Kantenkreuzungen die unbedingt notwendig ist wenn man einen Graphen zeichnet. Das Problem ist *NP*-schwer und hat sich in den letzten Jahrzehnten auch als äußerst herausfordernd aus Sicht der graphentheoretischen und algorithmischen Forschung, sowie der Praxis, herausgestellt. Dennoch zeigen wir, dass sich Verfahren entwickeln lassen, die das Problem für viele praxisrelevante Graphen in annehmbarer Zeit beweisbar optimal lösen. Der Schlüssel dazu ist eine geschickte Kombination aus Graphentheorie, kombinatorischer Algorithmik, sowie algebraischen Methoden, insbesondere der Mathematischen Programmierung.

1 Einführung

Das Kreuzungszahlproblem ist ein notorisch schweres Problem aus dem Grenzgebiet zwischen Graphentheorie, Topologie und Algorithmik. Die Fragestellung selbst ist überraschend einfach: *Wie viele Kantenkreuzungen sind nötig um einen gegebenen Graph in der Ebene zu zeichnen?*

Diese minimale Anzahl ist die sogenannte *Kreuzungszahl* des Graphen. Die Kreuzungszahl, bzw. Varianten derselben, findet sich in diversen Aufgabenstellungen wieder, angefangen bei direkten Zeichenproblemen—die Anzahl der Kreuzungen ist eines der wichtigsten Kriterien beim visuellen Verstehen von Diagrammen [Pur97]—bis hin zum Chipdesign, wenn es um das Routing der Datenleitungen geht [BL84]. Obwohl das Problem nun schon seit über 60 Jahren ausführlich untersucht wird (siehe [Vrt08] für eine umfassende Bibliographie), sind selbst einfach anmutende Fragestellungen immer noch unbeantwortet, bzw. entziehen sich der wissenschaftlichen Beweisbarkeit: Was ist die Kreuzungszahl von vollständigen Graphen? Lässt sich die Kreuzungszahl eines beliebigen Graphen in polynomieller Zeit approximieren? Wie schwer ist es die Kreuzungszahl eines Graphen zu berechnen, bei dem das Entfernen einer einzigen Kante schon einen planaren (also kreuzungsfreien) Graphen ergeben würde? All diese Fragen stellen noch immer offene Forschungsgebiete dar, oftmals existieren noch nicht einmal fundierte Vermutungen.

Ein zentraler Punkt ist, dass das Kreuzungszahl-Problem zur Klasse der *NP*-schweren Probleme gehört, d.h. es existiert kein polynomieller Algorithmus der das Problem exakt löst,

es sei denn $P=NP$. Nichtsdestotrotz kann es in diversen Anwendungen interessant sein, ein solches exaktes Ergebnis zu berechnen. Die in diesem Artikel zusammengefasste Dissertation beschäftigt sich damit, die Kreuzungszahl von Graphen exakt zu bestimmen. Dies geschieht mit Hilfe von Methoden der Mathematischen Programmierung, insbesondere durch Formulierungen als *Ganzzahliges Lineares Programm* (ILP), in Kombination mit graphentheoretischen Modellen und kombinatorischen Algorithmen. Dadurch gelingt es zum ersten Mal die Kreuzungszahlen von praxisrelevanten Graphen trotz der Problemkomplexität in vertretbarem Zeitaufwand beweisbar optimal zu berechnen.

Im folgenden Abschnitt werden wir die Geschichte des Problems sowie die wichtigsten bekannten Ergebnisse Revue passieren lassen. Die Abschnitte 3 und 4 beschäftigen sich mit der mathematischen Formulierung einerseits, und dem algorithmischen Lösen derselben andererseits. In Abschnitt 5 schließlich beschreiben wir, wie unsere Preprocessingstrategien und heuristischen Verfahren die exakten Algorithmen beschleunigen; wir berichten über die erfolgreiche Anwendung der exakten Algorithmen und skizzieren, wie wir mit diesen Ansätzen auch Varianten des Kreuzungszahlproblems bearbeiten können. Aus Platzgründen sei insbesondere bei diesen Themen auf die vollständige Dissertation [Chi08] verwiesen. Diverse Teilaspekte der Dissertation erschienen in [CG07, BCE⁺08, CG08, CGM09, CJS08, CMB08, CMS08].

2 Geschichte und Vorarbeiten

Die Geschichte der Kreuzungszahl geht auf die Zeit des zweiten Weltkriegs zurück, als Pál Turán in einem Arbeitslager in einer Ziegelfabrik arbeiten musste [Tur77]. Hochöfen und Lagerstätten waren jeweils direkt über Schienenstränge miteinander verbunden, entlang derer er mit Ziegeln beladene Wagen schieben musste. Wenn sich zwei Schienenstränge kreuzten, sprangen die Wagen leicht aus ihrer Führung; das Wiederaufsetzen und ggf. neu beladen stellte natürlich einen sehr anstrengenden Zusatzaufwand dar. Turán fragte sich, wie ein Schienennetz aussehen müsse um möglichst wenige Kreuzungen zu benötigen.

Das obige Beispiel stellt graphentheoretisch die Frage nach der Kreuzungszahl eines vollständigen bipartiten Graphen dar. Zarankiewicz's Versuch dieses Problem zu lösen [Zar53] stellt gleichzeitig die erste Veröffentlichung zu Kreuzungszahlen dar. Sein Beweis enthielt aber eine Lücke [Guy69], und sein Konstruktionsverfahren liefert daher nur eine obere Schranke, von der allerdings vermutet wird, dass sie optimal ist. Doch widersetzt sich selbst diese alte Fragestellung noch immer jeglichem Beweisversuch.

Mitte der 80er Jahre zeigten Garey und Johnson [GJ83] dass es NP -vollständig ist zu testen, ob ein Graph eine Zeichnung mit maximal k Kreuzungen erlaubt. Seit kurzem weiß man, dass das Problem *fixed parameter tractable* ist [Gro01, KR07]: die Fragestellung lässt sich für *konstantes* k in linearer Zeit beantworten. Natürlich ist die Laufzeit dabei exponentiell von k abhängig, und die involvierten Konstanten sind so hoch, dass es ein offenes Problem darstellt ob praxistaugliche Varianten derartiger Algorithmen existieren.

Es ist unbekannt, ob sich das Kreuzungszahlproblem zumindest bis zu einem fixen Faktor approximieren lässt. Alle derzeitigen Ergebnisse beschränken sich auf Graphen mit

konstantem Maximalgrad. Algorithmen mit konstanten Approximationsfaktoren existieren derzeit nur für eingeschränkte Graphklassen die auf Oberflächen mit Genus 1 kreuzungsfrei gezeichnet werden können [GHLS07, HS07], bzw. bei denen das Entfernen einer einzigen Kante oder eines einzigen Knotens einen planaren Graphen garantiert [CM09, CHM09]. Für allgemeinere Graphen existieren nur Algorithmen deren Gütegarantien abhängig von der Graphengröße sind und dabei die *Summe* aus Knoten- und Kreuzungszahl betrachten [EGS00].

In der Praxis verwendet man daher zumeist Heuristiken, die in der Regel gute Ergebnisse liefern ohne dies jedoch garantieren zu können. Der derzeit beste Algorithmus dieser Art ist die *Planarisierungsheuristik* [DGL⁺97, GM04], die wir in unserem exakten Verfahren auch zur Generierung von starken oberen Schranken einsetzen.

3 ILP Formulierungen

Im Rahmen der Dissertation werden zwei verschiedene ILP Formulierungen für das Problem entwickelt: Während die neuere (OECM) in der Praxis für Realgraphen performanter ist, besitzt die ältere (SECM) Vorteile bezüglich der Erweiterbarkeit auf bestimmte alternative Kreuzungszahlvarianten. Wir werden uns in Folge vorrangig OECM widmen.

Sei \mathcal{CP} die Menge aller Kantenpaare die sich in einer Lösung potentiell kreuzen könnten. Den Kern beider Formulierungen bilden die Variablen

$$x_{\{e,f\}} \in \{0, 1\} \quad \forall \{e, f\} \in \mathcal{CP}, \quad (1)$$

die 1 sind, falls sich die Kanten e und f in der Lösung kreuzen, und 0 sonst. Basierend auf diesen Variablen ist die Zielfunktion einfach

$$\min \sum_{\{e,f\} \in \mathcal{CP}} x_{\{e,f\}}. \quad (2)$$

Das Problem mit dieser Variablenmenge ist auch als *NP-schwere Realisierbarkeitsproblem* bekannt [Kra91]: Selbst wenn wir einen vielleicht optimalen ganzzahligen Lösungsvektor \bar{x} betrachten, ist es *NP-schwer* zu testen ob \bar{x} eine gültige Lösung darstellt. Das Grundproblem dabei ist, dass die Reihenfolge von Kreuzungen auf den Kanten undefiniert ist, wenn wir z.B. für drei Kanten e, f, g die Variablen $x_{e,f} = x_{e,g} = 1$ gesetzt haben. Eine der beiden Kreuzungsreihenfolgen auf e mag zu einer optimalen Lösung führen, während die andere dies nicht zulässt. Daher müssen wir erweiterte Formulierungen betrachten, um das Problem als ILP beschreiben zu können: In der älteren SECM (*subdivision-based exact crossing minimization*) Formulierung betrachten wir anstelle des gegebenen Graphen einen modifizierten Graph, bei dem jede Kante durch eine Kette von ausreichend vielen Kanten (sogenannten *Segmenten*) ersetzt wird. Für jedes Segment fordern wir, dass maximal eine Kreuzung auf ihm liegt. Dies erlaubt einen Realisierbarkeitstest in linearer Zeit, in dem man die in \bar{x} gegebenen Kreuzungen durch Hilfsknoten im Graphen ersetzt, und den resultierenden Graphen auf Planarität testet. Es kann keine Mehrdeutigkeiten bei der

Kreuzungsreihenfolge geben, jedoch ist es notwendig einen Graphen mit $m := |E|$ Kanten in einen mit bis zu $O(m^2)$ vielen Kanten zu verwandeln, d.h. statt $O(m^2)$ Variablen benötigen wir nun bis zu $O(m^4)$.

Die alternative OEM (ordering-based exact crossing minimization) Formulierung löst das Problem anders: Zunächst betrachten wir statt des gegebenen ungerichteten Graphen eine beliebige *Orientierung* desselben, d.h. jeder Kante wird eine der zwei möglichen Richtungen zugeordnet. Danach führen wir zusätzliche Variablen

$$y_{e,f,g} \in \{0, 1\} \quad \forall (e, f, g) \in E^3; \{e, f\}, \{e, g\} \in \mathcal{CP} \quad (3)$$

ein, die 1 sein sollen falls e sich sowohl mit f als auch mit g kreuzt, und die Kreuzung mit f —gemäß der Orientierung von e —vor der Kreuzung mit g liegt. Durch diese Variablen lassen sich Kreuzungen eindeutig anordnen und wir können die Realisierbarkeit einer gegebenen Lösung wieder durch entsprechendes Ersetzen der Kreuzungen durch Hilfsknoten testen. Natürlich müssen wir nun die x - und y -Variablen miteinander verknüpfen und sicherstellen, dass letztere eine *lineare Ordnung* der Kreuzungen beschreiben. Wir verzichten an dieser Stelle auf exakte Formalismen und fordern

$$x_{\{e,f\}} \geq y_{e,f,g}, \quad x_{\{e,g\}} \geq y_{e,f,g} \quad (4)$$

$$1 + y_{e,f,g} + y_{e,g,f} \geq x_{\{e,f\}} + x_{\{e,g\}} \quad (5)$$

$$y_{e,f,g} + y_{e,g,f} \leq 1 \quad (6)$$

$$y_{e,f,g} + y_{e,g,h} + y_{e,h,f} \leq 2 \quad (7)$$

für alle offensichtlich vernünftig gewählten Kanten e, f, g, h . Ungleichungen (4) und (5) garantieren, dass x - und y -Variablen zueinander konsistent sind. Die Ungleichungen (6) und (7) fordern, dass die durch y definierte Reihenfolge eindeutig und zyklensfrei ist. Es ist aus dem wohluntersuchten *Linear Ordering Problem* bekannt, dass diese Ungleichungen ausreichen um eine eindeutige Anordnung zu garantieren. Wir bezeichnen alle Belegungsvektoren (\bar{x}, \bar{y}) die diese vier Ungleichungstypen erfüllen als *LO-zulässig*.

Schließlich bleibt noch, Ungleichungen zu definieren, die garantieren, dass unsere Lösung tatsächlich eine gültige Kreuzungskonfiguration liefert, d.h. dass der Realisierbarkeitstest bei einer finalen Lösung des ILPs tatsächlich positiv ausfällt. Kuratowski zeigte schon 1930 [Kur30], dass nicht-planare Graphen G' genau dadurch charakterisiert werden können, dass sie mindestens eine *Kuratowski-Unterteilung* (engl. *-Subdivision*) enthalten: Dies ist ein Teilgraphen $U \subseteq G'$ der dadurch entsteht, dass man einzelne Kanten eines K_5 (vollständiger Graph auf 5 Knoten) oder $K_{3,3}$ (vollständiger bipartiter Graph mit 3 Knoten pro Partitionsmenge) durch Ketten von Kanten ersetzt.

Intuitiv müssen wir also in unserem ILP für jede Kuratowski-Unterteilung K in G fordern, dass die Kanten von K sich untereinander mindestens einmal kreuzen. Darüber hinaus reicht es nicht aus, nur die Kuratowski-Unterteilungen in G zu betrachten, sondern alle solchen Teilgraphen die durch eine beliebige *Teilplanarisierung* G' (d.h. Festsetzen einiger Kreuzungen, und Einfügen von entsprechenden Hilfsknoten) entstehen können. Für eine formale Definition der notwendigen Konzepte und Mengen siehe [Chi08]. An dieser Stelle möge es reichen, dass wir uns das grobe Konzept eines *Kreuzungsschattens*—

abhängig von einer ganzzahligen LO-zulässigen Lösung (\bar{x}, \bar{y}) , und einer Kuratowski-Unterteilung K in der daraus entstehenden Teilplanarisierung—vorstellen. Er wird durch das Paar $(\mathcal{X}, \mathcal{Y})$ spezifiziert und gibt dabei die minimalen Mengen der x - und y -Variablen an, die auf 1 gesetzt sein müssen, damit K existiert. Für Kuratowski-Unterteilungen im Originalgraph gilt daher $(\mathcal{X}, \mathcal{Y}) = (\emptyset, \emptyset)$. Wir definieren *Kuratowski-Ungleichungen* als:

$$\sum_{\{e,f\} \in \mathcal{CP}(K)} x_{\{e,f\}} \geq 1 - \sum_{a \in \mathcal{X}} (1 - x_a) - \sum_{b \in \mathcal{Y}} (1 - y_b). \quad (8)$$

Wenn wir die beiden Summen auf der rechten Seite für einen Moment ignorieren, garantiert (8), dass sich mindestens ein Kantenpaar in K kreuzt (wobei $\mathcal{CP}(K)$ genau die dafür infrage kommenden Kantenpaare angibt). Die beiden eben ignorierten Summen „schalten“ die Ungleichung ab, sollten die für die Existenz von K notwendigen Variablen nicht gesetzt sein: die rechte Seite würde dann nicht-positiv, und die Ungleichung damit immer erfüllt. Offensichtlich benötigt unser ILP eine exponentielle Anzahl solcher Kuratowski-Ungleichungen, und das Lösen des Programms ist daher nicht trivial.

4 Lösungsstrategien

ILPs werden i.d.R. durch *Branch-and-Bound* Verfahren gelöst, bei denen sukzessive untere Schranken durch *LP-Relaxierungen* (das ILP ohne Ganzzahligkeitsbedingungen) berechnet werden, und Teilprobleme z.B. durch Fixieren der binären Variablen auf 0 und 1 erzeugt werden. Siehe z.B. [Wol98] für eine Einführung.

Unabhängig davon ob wir SECM oder OEMC betrachten, müssen wir mit exponentiell vielen Kuratowski-Ungleichungen umgehen. Wir lösen das Problem daher mittels einem *Branch-and-Cut* Verfahren, d.h. wir erweitern obiges Verfahren dahingehend, dass wir in jedem Teilproblem nicht alle Ungleichungen betrachten, sondern nur eine ausgewählte Teilmenge dieser. Nach dem Berechnen einer LP-Relaxierung müssen wir die fraktionale Lösung untersuchen, ob nicht-betrachtete Ungleichungen durch sie verletzt werden. Wenn ja, werden diese (bzw. einige dieser) dem Modell hinzugefügt und eine neue LP-Relaxierung berechnet. Dieses Vorgehen wird als *Separierung* bezeichnet.

Weiterhin stellt sich heraus, dass die Anzahl der Variablen, obwohl polynomiell, in der Praxis zu groß wird, um das ILP effizient berechnen zu können. Daher führen wir auch *Spaltengenerierungsschemata* ein, d.h. anstatt nur Ungleichungen dynamisch dem Modell hinzuzufügen, starten wir auch nur mit einer Teilmenge der Variablen und fügen die restlichen nur bei Bedarf hinzu.

Separierung. Wir starten ohne jegliche Ungleichungen des Typs (7) und (8). In einem ersten Schritt lassen sich (7) einfach nach Bedarf generieren, indem man sie enumerativ testet. Das Separieren der Kuratowski-Ungleichungen gestaltet sich weitaus komplexer, da es eine offene Frage darstellt, ob Ungleichungen solchen Typs exakt in polynomieller Zeit separiert werden können. Daher verwenden wir ein heuristisches Verfahren. Dieses Verfahren ist so gearbetet, dass es ausschließlich verletzte Ungleichungen findet; im Falle von

ganzzahligen Lösungen ist garantiert, dass es solche findet sofern sie existieren. Einzig bei fraktionalen Lösungen kann es sein, dass keine verletzte Ungleichung identifiziert wird, obwohl nicht alle erfüllt sind. Auf die Gültigkeit und Optimalität des gesamten Prozesses hat dies jedoch keinen Einfluss, da wir ggf. nur früher einen Branch durchführen als unbedingt nötig wäre.

Klassische heuristische Separierungen arbeiten zumeist auf Lösungen, die durch Runden der fraktionalen LP-Lösungen entstehen. Dies wäre in unserem Fall aber nicht zulässig, da wir LO-zulässige Lösungen zum Separieren benötigen. Daher runden wir zunächst nur die x -Variablen, und setzen dann die $y_{e,\cdot}$ -Variablen für jede Kante e . Sei F_e die Menge der Kanten f mit $\bar{x}_{\{e,f\}} = 1$. Wir definieren einen vollständigen bigerichteten Graphen K_e mit einem Knoten für jedes Element aus F_e . Danach setzen wir das Gewicht der Kante (f, g) in K_e auf den fraktionale Lösungswert $\bar{y}_{e,f,g}$, für alle $f, g \in F_e$. Durch (heuristisches) Lösen eines Linear Ordering Problems auf diesem gewichteten Hilfsgraphen können wir ganzzahlige LO-zulässige Werte für die y -Variablen ableiten. Wir nennen die entstehende Lösung eine *Ganzzahlige Interpretation* der fraktionalen Lösung, und generieren aus ihr eine Teilplanarisierung G' des Originalgraphen, indem wir die Kreuzungen durch Hilfsknoten ersetzen. Danach suchen wir Kuratowski-Unterteilungen in G' : Moderne Planaritätstests liefern einen solchen Zeugen der Nichtplanarität in linearer Zeit. Für jede identifizierte Kuratowski-Unterteilung lässt sich nun der Kreuzungsschatten berechnen und die daraus resultierende Ungleichung auf Verletzung überprüfen.

Spaltengenerierung. Für SECM betrachteten wir zunächst ein „klassisches“ Vorgehen mittels *reduzierter Kosten*. Dieses benutzt im Wesentlichen abstrakte algebraische Überlegungen basierend auf [DW60]. Danach entwickelten wir ein spezifisches *kombinatorisches* Spaltengenerierungsschema, das auf unser Problem speziell zugeschnitten ist, und die Entscheidung bzgl. der Notwendigkeit von zusätzlichen Variablen auf Basis von kombinatorischen und graphentheoretischen Beobachtungen trifft. Letzteres dominiert ersteres sehr klar. Darüber hinaus konnten wir die Ideen von letzterem auch auf die neuere OEM Formulation übertragen, für die wir den in Folge skizzierten kombinatorischen Spaltengenerierungsansatz entwickelten.

Wir starten ausschließlich mit den x -Variablen, da wir davon ausgehen können, dass nur wenige Kanten tatsächlich in mehrere Kreuzungen involviert sein werden. Es lässt sich zeigen, dass y -Variablen erst notwendig werden, wenn eine Ganzzahlige Interpretation mehrere Kreuzungen (z.B. mit den Kanten f und g) auf einer Kante e spezifiziert. Insbesondere ist es dann ausreichend, die Variablen $y_{e,f,g}$ und $y_{e,g,f}$ (zusammen mit den notwendigen Ungleichungen aus (4)–(7)) zu betrachten. Eine Besonderheit ist dabei, dass die neuen Variablen—anders als in klassischen Spaltengenerierungsschemata—die duale Schranke niemals senken. Daher liefert jede LP-Relaxierung unabhängig von den bisher generierten Variablen stets eine gültige untere Schranke.

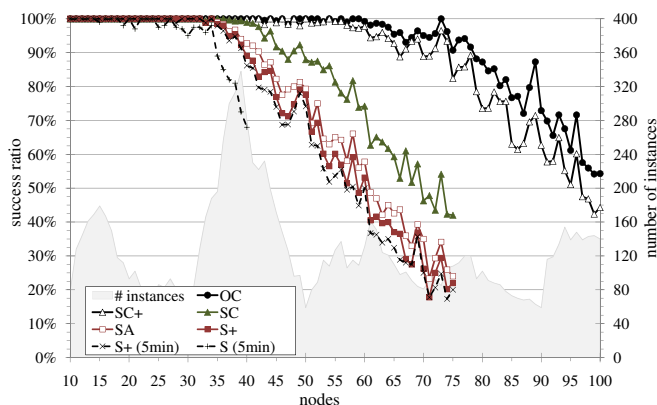


Abbildung 1: Erfolgsrate (*success ratio*) in Abhängigkeit der Knotenanzahl (*nodes*) für die verschiedenen in der Dissertation entwickelten exakten Algorithmen und ihrer Entwicklungsstufen. Die beiden besten Kurven zeigen die aktuellsten Varianten für SECM (*SC+*) und OECM (*OC*).

5 Praxis und Erweiterungen

Um den obig skizzierten Algorithmus in der Praxis anzuwenden, benötigt man noch einige zusätzliche Module: Einen der wichtigsten Aspekte dabei stellt die Vorverarbeitung (*Pre-processing*) der Instanz dar. Wir entwickelten dafür einen linearen Algorithmus um den *nicht-planaren Kern* (*non-planar core*) eines Graphen zu berechnen. Dieser entsteht durch rekursives Schrumpfen von 3-Zusammenhangskomponenten in gewichtete Kanten; unser ILP kann in der Zielfunktion einfach erweitert werden, um mit Kantengewichten umzugehen. Diese Vorverarbeitung generiert Graphen die auf den einschlägigen Testinstanzen (s.u.) nur rund halb so groß sind wie die ursprüngliche Eingabe.

Um die Separierung zu beschleunigen, entwickelten wir eine analysetechnisch aufwendige Erweiterung des Planaritätstests von Boyer und Myrvold [BM04], die es uns ermöglicht statt nur einer einzigen, gleich mehrere unterschiedliche Kuratowski-Unterteilungen in optimaler Zeit (d.h. linear in Ein- und Ausgabe) zu finden. Dies erlaubt uns, den Separationsprozess in der Praxis drastisch zu beschleunigen. Die Nutzung der derzeit stärksten bekannten Kreuzungszahlheuristik aus [GM04] erlaubt uns darüber hinaus das frühzeitige Abschneiden von unnötigen Teilproblemen.

In Kombination mit diesen Ingredienzien können wir die nun geschilderte Lösungsgüte erreichen. Als Testset dient uns dazu die weitverbreitete *Rome Graph Library* [DGL⁺97], die rund 11.500 (dünne) Graphen mit 10–100 Knoten enthält. Diese basieren auf Realinstanzen aus dem Bereich des Software-Engineering, und sind daher für Graphenzeichenprobleme interessant. Abbildung 1 zeigt die Erfolgsrate in Abhängigkeit der Graphengröße, d.h. wieviel Prozent der Graphen einer bestimmten Größe innerhalb einer halben Stunde auf einem Standard-PC¹ beweisbar optimal gelöst werden. Es werden 93.3% aller Graphen—and sogar mehr als die Hälfte der größten Graphen—optimal gelöst. Genauere Auswertungen zeigen, dass wir (bis auf 6 Instanzen) alle Graphen mit einer Kreuzungszahl kleiner-gleich 20 lösen.

¹AMD Opteron 2.4 GHz, 32bit, 2GB RAM pro Prozess

Erweiterungen. Unser Preprocessing eignet sich auch für die Berechnung der Skewness, Thickness, Coarseness und des Genus eines Graphen. In der Dissertation wird darüber hinaus gezeigt, wie sich sowohl ILP Formulierungen als auch heuristische Verfahren auf verwandte Kreuzungszahlmaße anwenden lassen: Dabei sei z.B. die *simultane* Kreuzungszahl erwähnt, wie sie beim konsistenten gleichzeitigen Zeichnen von mehreren Graphen auftritt. Zu dieser Problemstellung liefern wir zusätzlich weiterführende Komplexitätsresultate und Schranken. Wir betrachten u.A. auch die *Hypergraphen*-Kreuzungszahl. Dabei decken wir einen neuen Zusammenhang mit der *Minor-monotonen* Kreuzungszahl auf und zeigen wie diese Konzepte für das Zeichnen von elektrischen Schaltplänen genutzt werden können. Unsere Experimente zeigen dabei eine Einsparung von durchschnittlich 42–71% gegenüber den bisher besten bekannten Ergebnissen auf den einschlägigen Benchmarksets.

Unsere ILPs scheinen darüber hinaus vielversprechend für die Untersuchung interessanter Graphenklassen—z.B. vollständige Graphen, verallgemeinerte Petersen Graphen, etc.—zu sein. Um solche komplexeren Strukturen in der Praxis schnell genug berechnen zu können, müssen wir i.d.R. weitere Ungleichungsklassen einführen, die z.B. minimale Kreuzungszahlen von komplexeren Teilstrukturen (im Vergleich zu simplen Kuratowski-Unterteilungen) fordern, bzw. die Symmetrie der Eingabegraphen brechen um das mehrmalige Berechnen von äquivalenten Teilproblemen zu verhindern.

Die Dissertation enthält auch eine Studie über die polyedrischen Eigenschaften der im ILP so kritischen Kuratowski-Ungleichungen. Durch die auch in der Praxis enorme Anzahl der benötigten Ungleichungen könnte man vermuten, dass es sich dabei um mathematisch schwache Ungleichungen handelt. Jedoch können wir zeigen, dass selbst einfache Kuratowski-Ungleichungen in vollständigen (und vollständigen bipartiten) Graphen tatsächlich Facetten des Kreuzungszahl-Polytops definieren, und daher in jeder gültigen Formulierung enthalten sein müssen.

6 Zusammenfassung und Ausblick

Durch die skizzierten neuen Algorithmen ist es nun zum ersten Mal möglich, die Kreuzungszahlen vieler Graphen von praktischem Graphenzeicheninteresse tatsächlich exakt zu berechnen. Selbst im unerfolgreichen Fall erhalten wir zusätzlich zu einer gültigen Lösung auch eine spezifische Gütegarantie in Form einer unteren Schranke. Diese Ergebnisse lassen sich auch auf verwandte Konzepte (z.B. die Kreuzungszahl von elektrischen Schaltnetzen, die durch Hypergraphen spezifiziert werden) erweitern.

Die Entwicklung wurde vor allem auch durch die Prinzipien des *Algorithm Engineerings* geleitet, d.h. wir befanden uns in einem steten Kreislauf aus Design und Analyse von Algorithmen, sowie deren Implementierung und Evaluation. Die Analyse der dabei auftretenden Bottlenecks führte immer wieder zu neuen algorithmischen Überlegungen, wie z.B. Spaltengenerierungen, beschleunigte Identifikation von Kuratowski-Unterteilungen, etc. Für die weitere Forschung auf diesem Gebiet liegen vor allem drei Schritte nahe: Zum einen könnten verbesserte Heuristiken (z.B. auf Basis von Knoten-Einfüge-Routinen

[CGMW09]) durch stärkere obere Schranken die Performanz des exakten Verfahrens steigern. Zum anderen wäre auch die Identifikation von weiteren, verstärkenden Ungleichungsklassen, sowie die Betrachtung spezieller Graphenklasse (siehe Abschnitt 5) von großem Interesse.

Literatur

- [BCE⁺08] C. Buchheim, M. Chimani, D. Ebner, C. Gutwenger, M. Jünger, G. W. Klau, P. Mutzel und R. Weiskircher. A branch-and-cut approach to the crossing number problem. *Discrete Optimization, Special Issue in Memory of George B. Dantzig*, 5(2):373–388, 2008.
- [BL84] S. N. Bhatt und F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28:300–343, 1984.
- [BM04] J. M. Boyer und W. J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.
- [CG07] M. Chimani und C. Gutwenger. Algorithms for the hypergraph and the minor crossing number problems. In *Proc. ISAAC '07*, Jgg. 4835 of LNCS, Seiten 184–195. Springer, 2007.
- [CG08] M. Chimani und C. Gutwenger. Non-planar core reduction of graphs. *Discrete Mathematics*, 2008. Im Druck. Eine Vorversion erschien in *Proc. GD '05*, LNCS 3843, Seiten 223–234.
- [CGM09] M. Chimani, C. Gutwenger und P. Mutzel. Experiments on exact crossing minimization using column generation. *ACM Journal of Experimental Algorithms*, 2009. Im Druck. Eine Vorversion erschien in *Proc. WEA '06*, LNCS 4007, Seiten 303–315.
- [CGMW09] M. Chimani, C. Gutwenger, P. Mutzel und C. Wolf. Inserting a vertex into a planar graph. In *Proc. SODA '09*, Seiten 375–383. ACM-SIAM, 2009.
- [Chi08] M. Chimani. *Computing crossing numbers*. Dissertation, TU Dortmund, 2008. <http://hdl.handle.net/2003/25955>.
- [CHM09] M. Chimani, P. Hliněný und P. Mutzel. Approximating the crossing number of apex graphs. submitted, 2009.
- [CJS08] M. Chimani, M. Jünger und M. Schulz. Crossing minimization meets simultaneous drawing. In *Proc. PacificVis '08*, Seiten 33–40, 2008.
- [CM09] S. Cabello und B. Mohar. Crossing and weighted crossing number of near planar graphs. In *Proc. GD '08*, Jgg. 5417 of LNCS, Seiten 38–49. Springer, 2009.
- [CMB08] M. Chimani, P. Mutzel und I. Bomze. A new approach to exact crossing minimization. In *Proc. ESA '08*, Jgg. 5193 of LNCS, Seiten 284–296. Springer, 2008.
- [CMS08] M. Chimani, P. Mutzel und J. M. Schmidt. Efficient extraction of multiple Kuratowski subdivisions. In *Proc. GD '07*, Jgg. 4875 of LNCS, Seiten 159–170. Springer, 2008.
- [DGL⁺97] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari und F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry: Theory and Applications*, 7(5–6):303–325, 1997.

- [DW60] G. B. Dantzig und P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.
- [EGS00] G. Even, S. Guha und B. Schieber. Improved approximations of crossings in graph drawing. In *Proc. STOC '00*, Seiten 296–305, 2000.
- [GHLS07] I. Gitler, P. Hliněný, J. Leanos und G. Salazar. The crossing number of a projective graph is quadratic in the face-width. *Electronic Notes in Discrete Mathematics*, 29:219–223, 2007.
- [GJ83] M. R. Garey und D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316, 1983.
- [GM04] C. Gutwenger und P. Mutzel. An experimental study of crossing minimization heuristics. In *Proc. GD '03*, Jgg. 2912 of *LNCS*, Seiten 13–24. Springer, 2004.
- [Gro01] M. Grohe. Computing crossing numbers in quadratic time. In *Proc. STOC '01*, 2001.
- [Guy69] R. K. Guy. The decline and fall of Zarankiewicz’s theorem. In *Proof Techniques in Graph Theory, Proc. 2nd Ann Arbor Graph Theory Conference '68*, Seiten 63–69. Academic Press, 1969.
- [HS07] P. Hliněný und G. Salazar. Approximating the crossing number of toroidal graphs. In *Proc. ISAAC '07*, Jgg. 4835 of *LNCS*, Seiten 148–159. Springer, 2007.
- [KR07] K. Kawarabayashi und B. Reed. Computing crossing number in linear time. In *Proc. STOC '07*, Seiten 382–380, 2007.
- [Kra91] J. Kratochvíl. String graphs II: Recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52:67–78, 1991.
- [Kur30] C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.
- [Pur97] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proc. GD '97*, Jgg. 1353 of *LNCS*, Seiten 248–261. Springer, 1997.
- [Tur77] P. Turán. A note of welcome. *Journal of Graph Theory*, 1:7–9, 1977.
- [Vrt08] I. Vrt’o. Crossing numbers of graphs: A bibliography. <ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf>, 2008.
- [Wol98] L. A. Wolsey. *Integer programming*. Discrete Mathematics and Optimization. Wiley-Interscience, 1998.
- [Zar53] K. Zarankiewicz. The solution of a certain problem on graphs of P. Turán. *Bulletin de l’Academie Polonaise des sciences, Cl. III*, 1:167–168, 1953.



Markus Chimani wurde am 8. Januar 1980 in Wien, Österreich, geboren. Er erhielt sein Diplom 2004 mit Auszeichnung und unter Mindeststudiendauer an der Technischen Universität Wien, und begann nach einem halbjährigen Forschungsaufenthalt in Boston, USA, sein Promotionsstudium 2005 an der Universität Dortmund. Während seiner Promotionszeit beschäftigte er sich außer mit den in diesem Artikel vorgestellten Themen unter anderem auch mit anderen Fragestellungen des Graphenzeichnens sowie dem Themengebiet des topologischen Netzwerkdesigns.