Bachelorarbeit

Simultane Pfadzeichnungen auf dem Gitter

Timur Sultanov

Abgabedatum:	19. Mai 2025
Betreuende:	Jun.-Prof. Dr. Philipp Kindermann
	Prof. Dr. Stefan Näher

Universität Trier
Fachbereich IV - Informatik
Algorithmik

Zusammenfassung

bliblablub

Inhaltsverzeichnis

1 Einleitung **4**
1.1 Motivation und Relevanz des Themas 4
1.2 Aufbau und Zielsetzung der Arbeit 4

2 Theoretische Grundlagen **5**
2.1 Grundlagen der Graphentheorie 5
2.1.1 Graph Drawing: Methoden und Modelle 6
2.1.2 Simultaneous Embedding 8
2.2 Einführung in Linear Programming 8
2.2.1 Definitionen und Prinzipien der LP 8
2.2.2 Lösungsverfahren für LP 8
2.3 Stand der Forschung und verwandte Arbeiten 8

3 Methodik und Modellierung **9**
3.1 Formulierung des Problems als ILP 9
3.2 Implementierung des ILP-Modells 11

4 Experimente und Evaluation **12**
4.1 Versuchsaufbau und Testfälle 12
4.2 Auswertung der Ergebnisse 12
4.3 Diskussion der Ergebnisse 12

5 Ausblick und zukünftige Arbeiten **13**

Literaturverzeichnis **14**

1 Einleitung

1.1 Motivation und Relevanz des Themas

Die effiziente Darstellung von mehreren Graphen auf einer gemeinsamen Fläche stellt eine zentrale Herausforderung im Bereich der Graphvisualisierung dar. Besonders im Kontext des sogenannten *Simultaneous Embedding* gewinnen solche Probleme zunehmend an Bedeutung, da sie in den verschiedensten Anwendungsgebieten wie der Schaltkreisoptimierung oder auch Netzwerkanalyse eine Rolle spielen. In vielen praktischen Szenarien ist es notwendig, mehrere zusammenhängende Strukturen gleichzeitig und ohne visuelle Konflikte abzubilden, um deren Beziehungen und Interaktionen besser nachvollziehbar zu machen.
Durch die gleichzeitige Einbettung mehrerer Graphen auf demselben Gitter können überlappende Informationen effizient dargestellt werden, was die Lesbarkeit und Interpretierbarkeit komplexer Datensätze verbessert. Insbesondere mit dem zunehmenden Bedarf an leistungsfähigen Visualisierungs- und Optimierungstechniken in der Industrie und Forschung wird die Entwicklung effizienter Algorithmen und Modelle für das Simultaneous Embedding immer relevanter

1.2 Aufbau und Zielsetzung der Arbeit

Diese Arbeit beschäftigt sich mit der Lösung des Simultaneous Embedding-Problems durch den Einsatz von linearen Optimierungsmethoden. Ziel ist es, eine modellgestützte Herangehensweise zu entwickeln, die es ermöglicht, zwei oder mehrere Pfade gleichzeitig platzsparend und konfliktfrei auf einem Gitter mit einer orthogonalen Zeichnung darzustellen. Wir verwenden ein lineares-ganzzahliges Optimierungsmodell, bei welchem das Problem mit Hilfe von Variablen, Nebenbedingungen und einer Zielfunktion definiert wird und das Ergebnis als Konfiguration dieser Variablen dargestellt wird. Aus dieser Konfiguration kann dann eine entsprechende orthogonale Zeichnung abgeleitet werden.
Zu Beginn der Arbeit werden die wesentlichen Begriffe aus der Graphentheorie und der Optimierung erläutert, um ein gemeinsames Verständnis für das behandelte Problem zu schaffen. Es folgt die formale Formulierung des Simultaneous Embedding als linearesganzzahliges Optimierungsproblem. Anschließend erfolgt die technische Umsetzung des Modells in Python unter der Verwendung von Gurobi. Darauf aufbauend werden Experimente zur Laufzeitanalyse und Skalierbarkeit durchgeführt, um die Praxistauglichkeit des Ansatzes zu überprüfen. Die Arbeit schließt mit einer kritischen Reflexion der Ergebnisse und der Ableitung von Potenzialen für weiterführende Forschung ab.

2 Theoretische Grundlagen

Die theoretischen Grundlagen dieser Arbeit bilden das Fundament für das Verständnis der angewandten Methoden und Modelle im Bereich der Grapheneinbettung und der Linearen Programmierung . Zunächst werden die grundlegenden Definitionen der Graphentheorie vorgestellt, gefolgt von einer Einführung in gängige Methoden und Modelle des Graph Drawings. Danach werden die Besonderheiten und Herausforderungen des Simultaneous Embedding-Problems erläutert. Anschließend werden die grundlegenden Konzepte und Lösungsverfahren der Linear Programming vorgestellt und der Stand der Forschung und verwandte Arbeiten präsentiert.

2.1 Grundlagen der Graphentheorie

Wir beginnen mit den grundlegenden Definitionen der Graphentheorie, welche die Basis für die in dieser Arbeit behandelten Probleme darstellen. Wir folgen hierbei der Notation aus dem Buch *Graph Theory* von Diestel [Die17]. Ein endlicher Graph $G = (V, E)$ besteht aus einer endlichen Menge von Knoten V und einer endlichen Menge von Kanten $E \subseteq V \times V$. Ein *Multigraph* ist ein spezieller Graph, bei dem zusätzlich ein Mapping M existiert mit $M : E \to V \cup [V]^2$, der jeder Kante ein oder zwei Endknoten zuordnet. Diese Graphen können dadurch die gleiche Kante mehrmals enthalten. Wir sagen, dass eine Kante $e = (u, v)$ *inzident* ist zu u und v. Die zwei Knoten betitelt man auch als die *Endknoten* von e und als *adjazent* zueinander. Der *Grad eines Knoten* ist die Anzahl an Kanten, die zum Knoten inzident sind.
Ein *Pfad* von G ist ein nicht-leerer Graph $P = (V_1, E_1)$ mit $V_1 = \{x_0, x_1, \ldots, x_k\} \subseteq V$ und $E_1 = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\} \subseteq E$, bei dem alle Knoten disjunkt sind. Ein Pfad, der alle Knoten von G enthält, bezeichnet man als *Hamiltonpfad*.
Ein Graph ist *planar*, wenn er so in eine Ebene gezeichnet werden kann, dass seine Kanten sich nur an ihren Endknoten treffen. Eine solche Zeichnung nennt man eine *planare Einbettung*.
Grundlegend sind Graphen abstrakte mathematische Objekte. Konkret sind Graphen dafür geeignet, Relationen zwischen Objekten zu beschreiben. Dabei entsprechen die Objekte Knoten und die Relationen zwischen den Objekten entsprechen Kanten. Stehen eine Menge von Objekten in verschiedenen Arten in Relation, dann resultiert das in parallele Kanten, wodurch es sich um Multigraphen handelt. Durch diese Eigenschaft besitzen Graphen nicht nur in der Theorie Relevanz in Bereichen wie Spracherkennung, Compilerbau, Bioinformatik und Datenmodellierung, sondern haben auch etliche Anwendungen im realen Leben. Seien es Straßenpläne, Produktempfehlungen bei zum Beispiel Amazon oder auch das World Wide Web, Graphen sind wichtige Bestandteile des alltäglichen Lebens[KW01].

2.1.1 Graph Drawing: Methoden und Modelle

Das Graph Drawing befasst sich mit der visuellen Darstellung von Graphen im zweidimensionalen oder dreidimensionalen Raum. Ein Problem bei Graph Drawing lautet: Wie soll ein Graph gezeichnet werden, sodass die Informationen bestmöglich dargestellt werden? Wie schon festgestellt wurde, haben Graphen viele unterschiedliche Anwendungsbereiche. Aus diesen resultieren auch viele verschiedene Anforderungen an die visuelle Darstellung dieser Graphen.
Es gibt zumindest einige allgemeine Konzepte und Techniken, die eine große Anzahl an Anwendungsbereichen abdecken. Ein gutes Layout zu erstellen, ist somit oft damit verbunden, ein oder mehrere dieser Kriterien zu optimieren. Im Buch *Drawing Graphs: Methods and Models* [KW01] werden einige ästhetische Kriterien benannt. Es werden nun die Kriterien aufgezählt, die im Kontext der simultanen Einbettung wichtig sind:

- **Kreuzungsminimierung:** Wenn zu viele Kreuzungen von Kanten existieren, dann fällt es dem menschlichen Auge schwerer zu sehen, welche Knoten durch die Kanten verbunden sind. Ist eine Zeichnung ohne Kreuzung möglich, dann sind diese oft zu bevorzugen. Die Minimierung von Kreuzungen ist auch ein wichtiges technisches Kriterium in Gebieten wie Schaltplänen, um die Anzahl an Schichten zu reduzieren.

- **Biegungsminimierung:** Dieses Kriterium ist besonders bei orthogonalen Zeichnungen von Bedeutung, da das menschliche Auge Kanten mit wenigen Biegungen leichter verfolgen kann. Auch in VLSI-Schaltungen stellen Biegungen in Leitungen potenzielle Schwachstellen dar und können zu Problemen wie Signalreflexionen oder Verzögerungen führen.

- **Flächenminimierung:** Das minimieren der genutzten Fläche sorgt für ein ansprechenderes Bild durch das Füllen des Platzes mit homogener Dichte. Insbesondere in Städten ist es wichtig, die Verkehrs- und Infrastrukturelemente so zu platzieren, dass sie wenig Fläche verbrauchen, um mehr Raum für Bebauung oder Grünflächen zu lassen.

- **Kantenlängenminimierung:** Die Kantenlängenminimierung verbessert die Lesbarkeit, da nahe beieinanderliegende Knoten Zusammenhänge schneller erkennbar machen. Besonders in der Verkehrsnetzplanung werden Kanten möglichst kurz gehalten, um Straßen effizient zu platzieren und die Übersichtlichkeit der Karte zu erhöhen.

Nachdem nun grundlegende ästhetische Kriterien für Graphenzeichnung erläutert wurden, soll im Folgenden auf verschiedene spezielle Zeichnungsmodelle eingegangen werden. Diese Modelle beschreiben konkrete Konventionen und Regeln, nach denen Graphen unter bestimmten geometrischen und technischen Beschränkungen visualisiert werden.
Je nach Anwendungsgebiet kommen unterschiedliche Darstellungsformen zum Einsatz, um spezifische Anforderungen - wie etwa eine klare Strukturierung, Platzersparnis oder

technische Umsetzbarkeit – zu erfüllen. Zu den bekanntesten und am häufigsten verwendeten Modellen zählen orthogonale Zeichnungen, Straight-Line-Zeichnungen und Rectangle Drawings, aber auch weitere Varianten, die je nach Problemstellung bevorzugt werden. Im Folgenden werden diese speziellen Zeichnungsmodelle vorgestellt und hinsichtlich ihrer jeweiligen Stärken, Schwächen und typischen Anwendungsfelder diskutiert[Tam14].

Ein **Straight-Line Drawing** eines Graphen $G = (V, E)$ ist eine Einbettung von G in die Ebene, bei der jeder Knoten $v \in V$ als eindeutiger Punkt im $\mathbb{R}^2$ dargestellt wird und jede Kante $e = (u, v) \in E$ durch ein gerades Liniensegment zwischen den Punkten u und v repräsentiert wird. Alle Kanten sind dabei gerade und enthalten keine Biegungen. Eine zentrale Eigenschaft dieser Darstellungsform ist, dass Straight-Line Drawings grundsätzlich ohne Einschränkung für alle Graphen möglich sind. Jedoch sind kreuzungsfreie Straight-Line Drawings nur dann möglich, wenn der Graph planar ist. Nach dem Satz von Fáry besitzt jeder planare Graph eine solche kreuzungsfreie Straight-Line-Darstellung [BKR15].
Straight-Line Drawings werden insbesondere in der Visualisierung und Analyse von Netzwerken eingesetzt, da sie eine einfache und visuell klare Darstellung bieten. In Anwendungen wie sozialen Netzwerken, Netzplänen oder Software-Architekturdiagrammen sorgt die Reduktion von Biegungen und die direkte Verbindung der Knoten für eine hohe Lesbarkeit und eine schnelle Erfassbarkeit der graphischen Struktur.

Eine **orthogonale Zeichnung** eines Graphen $G = (V, E)$ ist eine Einbettung von G in die Ebene, bei der jeder Knoten $v \in V$ als Punkt im $\mathbb{R}^2$ dargestellt wird und jede Kante $e = (u, v) \in E$ ausschließlich aus horizontalen und vertikalen Liniensegmenten besteht. Kanten können dabei rechtwinklige Biegungen enthalten.
Orthogonale Zeichnungen finden vor allem in technischen Anwendungsfeldern wie der Schaltplanerstellung, im VLSI-Design und bei der Visualisierung von Prozess- und Flussdiagrammen breite Verwendung. Durch die Einschränkung auf horizontale und vertikale Segmente wird eine rasterbasierte Struktur gefördert, die insbesondere bei der Herstellung von Leiterplatten und Chips von Vorteil ist.
Durch die Einschränkung auf rechtwinklige Biegungen, können keine Graphen eingebettet werden, die Knoten mit einem Grad größer als vier enthalten. Ein zentrales Optimierungskriterium in orthogonalen Zeichnungen ist die Minimierung der Anzahl an Biegungen je Kante, um die Lesbarkeit zu verbessern.

Ein **Rectangle Drawing** ist eine spezielle Form der orthogonalen Zeichnung, die nur für planare Graphen definiert ist. In einem Rectangle Drawing werden alle Knoten als nicht-überlappende, rechteckige Bereiche (Boxen) dargestellt. Jede Kante $e = (u, v) \in E$ wird dabei als eine horizontale oder vertikale Linie gezeichnet, die zwei benachbarte Rechtecke entlang ihrer gemeinsamen Kanten verbindet.
Rectangle Drawings werden häufig verwendet, wenn eine klare Flächenaufteilung und Strukturierung notwendig ist, wie etwa bei UML-Diagrammen, Floorplan-Layouts in der VLSI-Entwicklung oder der Visualisierung hierarchischer Daten. Da die Rechtecke einander nicht überlappen und der gesamte Graph die Zeichenfläche vollständig und

ohne Lücken füllt, ist dieses Modell besonders platzsparend.
Rectangle Drawings sind besonders dann von Vorteil, wenn sowohl die Übersichtlichkeit als auch eine kompakte Flächennutzung zentrale Anforderungen sind.

2.1.2 Simultaneous Embedding

2.2 Einführung in Linear Programming

2.2.1 Definitionen und Prinzipien der LP

2.2.2 Lösungsverfahren für LP

2.3 Stand der Forschung und verwandte Arbeiten

3 Methodik und Modellierung

3.1 Formulierung des Problems als ILP

Es sei $G = (V, E)$ der Multigraph der in ein Gitter $D = (P, F)$ eingebettet werden soll, wobei $V = \{v_1.v_2, \ldots, v_n\}$ die Knoten und $E = \{e_1, e_2, \ldots, e_n\}$ die entsprechenden Kanten darstellen. Das Gitter ist hierbei ein Tupel aus Gitterpunkten $P = \{(x, y) \mid (x, y) \in \mathbb{Z}^{\not\in}\}$ und Gitterkanten $F = \{(p, q) \mid p, q \in P\}$. Die Kanten kann man nun in zwei Mengen partitionieren, denen man dann zwei verschiedenen Hamiltonpfaden zuordnet. Ziel ist es nun, diese zwei Pfade so auf einem Gitter zu zeichnen, sodass diese für sich gesehen planar auf diesem Gitter eingebettet sind. Dabei soll dies möglichst optimiert bezüglich des genutzten Platzes auf einem Gitter $D = (P, F)$ vollzogen werden. Es ist also erlaubt, dass sich die Pfade gegenseitig kreuzen können.
Um für diese Aufgabenstellung ein entsprechendes ganzzahliges lineares Optimierungsproblem zu formulieren, werden zuerst Variablen definiert, sodass danach eine Zielfunktion und Nebenbedingungen geschaffen werden können.
Jeden Knoten $v \in V$ muss einem Gitterpunkt $p \in D$ zugeordnet werden, auf welchem der Knoten platziert wird. Hierfür definieren wir für alle Knoten $v \in V$ und jeden Gitterpunkt $p \in D$ die Variable $\sigma(v, p)$. Gilt nun, dass der Knoten v auf dem Gitterpunkt p liegt, dann $\sigma(v, p) = 1$, ansonsten $\sigma(v, p) = 0$.
Für jede Kante $e \in E$ und jede Gitterkante $(p, q) \in F$ benötigen wir zudem noch eine Variable $\delta(e, p, q)$, die analog zu $\sigma(v, p)$ den Wert Eins annimmt, wenn die Kante e die Gitterkante (p, q) belegt, ansonsten den Wert Null.
Wir möchten nun die Anzahl an gebrauchten Gitterkanten minimieren. Hierfür ist folgende Zielfuntkion $F(x) = z$ zu minimieren mit

$$z = \sum_{e \in E} \sum_{(p,q) \in F} \delta(e, p, q)$$

Damit die Pfade korrekt eingebettet werden können, sind einige Nebenbedingungen notwendig. Jeder Knoten $v \in V$muss auf genau einem Gitterpunkt $p \in P$ liegen, wobei jeder Gitterpunkt nicht von mehreren Knoten belegt werden darf. Damit dies gesichert ist, sind zwei Nebenbedingungen notwendig. Für Ersteres muss folgende Bedingung gelten:

$$\forall v \in V \sum_{p \in P} \sigma(v, p) = 1$$

Damit gilt, dass jeder Gitterpunkt von höchstens einem Knoten belegt wird, muss gelten:

$$\forall p \in P \sum_{v \in V} \sigma(v, p) \leq 1$$

Nun müssen wir sicherstellen, dass auch die Kanten der Pfade korrekt eingebettet werden. Kanten dürfen nicht abrupt stoppen und an einer komplett anderen Stelle weiterlaufen. Zudem dürfen sie nicht über platzierte Knoten laufen. Auch ist es wichtig, dass der Pfad sich nicht selbst kreuzen darf. Um all dies zu verhindern, sind einige Nebenbedingungen notwendig.
Damit Kanten kontinuierlich sind beobachten wir, wie sich eine Kante $e \in E$ mit $e = (u, v)$ verhält. Seien nun $\sigma(u, p) = 1 \wedge \sigma(v, p) = 1$ für beliebige $p, q \in D, p \neq q$. Man sieht, dass die Kante e den Gitterpunkt p genau einmal ausgehend verlässt und eine Gitterkante zu einem anderen Gitterpunkt dafür benutzt. Da Pfade kreisfrei sind, gibt es keine eingehende benutzte Gitterkante und nicht noch mehr ausgehend benutzten Gitterkanten. Analog gilt dies für den Endknoten v dieser Kante. Für alle anderen Gitterpunkte $r \in D, r \neq p, q$ gilt nun, dass die Kante e diese möglicherweise passiert. In diesem Fall benutzt e eine eingehende Gitterkante und eine entsprechende ausgehende Gitterkante. Passiert e einen Gitterpunkt nicht, dann gibt es weder eingehende noch ausgehende benutzte Gitterkanten an dem Punkt. In jedem Fall kann man schließen, dass die Summe an eingehenden Gitterkanten und ausgehenden Gitterkanten an einem Gitterpunkt, die die Kante e benutzt, identisch sein müssen außer an den Positionen der Endknoten u, v, bei der die Differenz gleich eins ist, damit es keine abrupten Kantenabbrüche existieren und zudem gesichert ist, dass die Kante bei u anfängt und v endet. Als Nebenbedingung formuliert:

$$\forall e = (v, u) \in E, v, u \in V, \forall p \in D : \sum_{(p,q) \in F} \delta(e, p, q) - \sum_{(q,p) \in F} \delta(e, q, p) = \sigma(v, p) - \sigma(u, p)$$

Aufbauend darauf formulieren wir nun eine Bedingung, die verhindert, dass sich eine Kante e selbst schneidet. Hierfür müssen wir sicherstellen, dass die Kante an jedem Gitterpunkt höchstens einmal ein- bzw. austretet:

$$\forall e \in E, \forall p \in D \sum_{(p,q) \in F} \delta(e, p, q) \leq 1 \wedge \sum_{(q,p) \in F} \delta(e, q, p) \leq 1$$

Uns fehlt nun noch, dass eine Kante keinen Gitterpunkt passieren darf, an dem schon ein Knoten gelegt wurde, der nicht zu der Kante inzident ist. Definieren wir dafür die Summe aller eingehenden und ausgehenden genutzten Gitterkanten am Gitterpunkt p von einer Kante e:

$$flow\_sum = \sum_{(p,q) \in F} \delta(e, q, p) + \sum_{(q,p) \in F} \delta(e, q, p)$$

Für jede Kante e an jedem Gitterpunkt p, an dem ein nicht inzidenter Knoten w liegt, muss gelten, dass e diese Stelle nicht passieren darf. Liegt an der Stelle kein Knoten, dann darf die Kante e diesen Punkt passieren : $flow\_sum \leq 2 \cdot (1 - \sigma(w, p))$.
Abschließend fehlt uns nur noch eine Nebenbedingung, die verhindert, dass es innerhalb eines Pfades zu Überschneidungen kommt. Sei nun W die Menge an Pfaden. Für alle $P_i = (V_i, E_i) \in W$ mit $V_i \subseteq V$ und $E_i \subseteq E$ und für jeden Gitterpunkt $p \in D$ definieren

wir die aggregierte Anzahl an korrespondierenden Gitterkanten an p, die von den Kanten $e_i \in E_i$ genutzt werden:

$$aggregated\_flow = \sum_{e \in p_i} (\sum_{(p,q) \in F} \delta(e,p,q) + \sum_{(q,p) \in F} \delta(e,q,p))$$

Nun gibt es zwei Szenarien für jeden Pfad. Entweder wird der Gitterpunkt p gar nicht passiert, wodurch gilt $aggregated\_flow = 0$ oder der Punkt wird passiert, dann darf es höchstens eine eingehende und höchstens eine ausgehende Kante geben an dem Punkt, damit es keine Überschneidungen gibt, wodurch gilt $aggregated\_flow \leq 2$.

3.2 Implementierung des ILP-Modells

4 Experimente und Evaluation

4.1 Versuchsaufbau und Testfälle

Insgesamt werden also $|V| \times |P| + |E| \times |F|$ Variablen benötigt.

4.2 Auswertung der Ergebnisse

4.3 Diskussion der Ergebnisse

5 Ausblick und zukünftige Arbeiten

[Die17][Tam14] [DB99] [KW01]

Literaturverzeichnis

[BKR15] Thomas Bläsius, Stephen G. Kobourov und Ignaz Rutter: Simultaneous Embedding of Planar Graphs, 2015. `https://arxiv.org/abs/1204.5853`.

[DB99] Giuseppe Di Battista: *Graph drawing algorithms for the visualization of graphs*. An Alan R. Apt book. Prentice Hall, 1999.

[Die17] Reinhard Diestel: *Graph theory*. Graduate texts in mathematics 173. Springer, 5th edition Auflage, 2017.

[KW01] Michael Kaufmann und Dorothea Wagner: Drawing Graphs Methods and Models, 2001.

[Tam14] Roberto Tamassia: *Handbook of graph drawing and visualization*. Discrete mathematics and its applications. CRC Press, 2014.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Trier, den 19. Mai 2025

. . . . . .
Timur Sultanov