technische universität
dortmund

Bachelorarbeit

Optimierungsalgorithmen zur Orientierung von planaren Graphen

Simon Bülhoff
Oktober

Gutachter:
Prof. Dr. Kevin Buchin
M.Sc. Antonia Kalb

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl 11
Arbeitsgruppe Algorithm Engineering
http://ls11-www.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Zielsetzung

Das Thema der Bachelorarbeit sind Optimierungsalgorithmen zur Orientierung von planaren Graphen. Genauer werden planare Spanngraphen betrachtet. Spanngraphen besitzen für jedes Knotenpaar im Graphen einen Pfad von einem Knoten zum Anderen. Dabei sind t-Spanngraphen solche, für die diese Pfade maximal um Faktor t länger sind als im vollständigen Graphen. Der Faktor t wird Dilationsfaktor genannt und es soll Ziel dieser Arbeit sein, die Kanten gegebener Spanngraphen derart zu orientieren, dass der Faktor t kleinstmöglich ist.
Orientieren ist eine Art des Richtens von Graphkanten, die keine Rückkanten zulässt. Dabei Planarität zu gewährleisten, bedeutet, dass sich außerdem keine Kante schneidet.

Anwendung finden gerichtete t-Spanngraphen unter Anderem in der Analyse von Straßennetzen [1], der Roboterbewegungsplanung [9] sowie dem Designen von
Netzwerkinfrastrukturen[6]. Obwohl die Konstruktion von geometrischen t-Spanngraphen bereits weitläufig erforscht wurde [12], ist die Erforschung von orientierten Spanngraphen noch nicht so weit fortgeschritten.
Vor diesem Hintergrund ist die Optimierung von Orientierungsalgorithmen für planare t-Spanngraphen umso interessanter. Es konnte gezeigt werden, dass es für einen ungerichteten Graph NP-Schwer ist zu entscheiden, ob es eine Orientierung gibt deren Dilation unterhalb eines gewissen Grenzwertes t ist [5].
Daher ist es nicht Ziel dieser Arbeit nach einem Algorithmus zu suchen, der für jeden gegebenen Graphen die minimale Dilation findet.
Stattdessen liegt der Fokus darauf, Algorithmen zu finden, für die bestimmte Garantien in Bezug auf die Minimierung der maximalen oriented Dilation (ödil")gegeben werden können.

1.2 Aufbau der Arbeit

Die Arbeit ist wie folgt aufgebaut:Anch der Einleitung im ersten Kapitel, werden im zweiten Kapitel, werden die für das Verständnis der Arbeit notwendigen Begriffe und graphentheoretischen Grundlagen erläutert. In diesem Rahmen wird auch die Ausgangssituation genau umschrieben.
Vor diesem Hintergrund wird dann auf die Lösungsmöglichkeiten auf Basis der theoretischen Voraussetzungen genauer eingegangen. Im dritten Kapitel werden die Algorithmen entwickelt. Dabei wird für jeden Algorithmus die theoretische Grundlage und die Methodik des Algorithmus erklärt.
Im vierten Kapitel wird die Pipeline vorgestellt, welche im Rahmen dieser Arbeit entwickelt wurde und mit deren Hilfe die Algorithmen geprüft werden.
Speziell wird dabei auf die Hilfsfunktionen eingegangen, die für das Anwenden der Algorithmen unabdingbar sind.
Im Anschluss folgen die Ergebnisse und ihre Auswertung, unterteilt nach den zu Grunde liegenden Graphkonstruktionsansätzen. Die in dieser Arbeit genutzten Graphkonstruktionsansätze sind: Delaunay Triangulierung, Greedy Triangulierung und eine selbst entworfene Strategie zur Konstruktion von Graphen mit Flächen, die mehr als drei Kanten haben(Polygons Methode). Die Delaunay und die Greedy Triangulierung sind weit verbreitete Methoden zur Generierung von planaren Graphen. Die Polygons Methode dient dazu, einige Anwendungsfelder, wie beispielsweise Straßennetze, realitätsnäher zu simulieren. Nach Abschluss der Auswertung wird im fünften Kapitel ein Ausblick auf mögliche Weiterentwicklungen gegeben.

Kapitel 3

Orientierungsalgorithmen

3.1 Bruteforce

Der intuitivste Ansatz zur Orientierung der Spanngraphen ist die Bruteforce Methode. Diese Methode wird bei vielen Problemen in der Informatik beispielhaft herangezogen, um die grundsätzliche Herangehensweise der weiteren Algorithmen zu illustrieren. Zudem verdeutlicht der Bruteforce Ansatz, weswegen es überhaupt komplexerer Algorithmen bedarf: In den meisten Fällen sind entweder die Laufzeit und/oder der Speicherbedarf des Bruteforceansatzes nicht akzeptabel.
In diesem Abschnitt wird zunächst der Bruteforceansatz vorgestellt und die Implementierung mittels Pseudocodes verdeutlicht. Im Anschluss wird die Laufzeit analysiert.

1. Der Algorithmus erhält den Graphen $G = (V, E)$ als Input.

2. Um die Lösung zu finden, werden alle möglichen Iterationen von Kantenorientierungen geprüft. Es gibt dabei $2^{|E|}$ verschiedene Iterationen. Für jede Iteration wird, sofern sie einen stark zusammenhängenden Graphen ergibt, die oriented Dilation berechnet. Alle Iterationen, für die der Graph nicht stark zusammenhängend ist, werden vorzeitig abgebrochen.

3. Nach Prüfung sämtlicher Iterationen, wird die kleinste so berechnete, oriented Dilation *minodil* zurückgegeben. Dieses Ergebnis ist für den gegebenen Graphen das Bestmögliche.

Der Bruteforceansatz ist, wie eingangs bereits angedeutet, nicht effizient. Das liegt an der Anzahl an Kantenorientierungen, die der Algorithmus prüfen muss. Von einem Graphen $G = (V, E)$ mit $|V| = 8$ ausgehend, kann auf Basis von 2.1.1 mit bis zu $3*8-6 = 18 Kanten$ gerechnet werden.
Das bedeutet, dass $2^{18} = 262144$ verschiedene Orientierungen geprüft werden müssen. Dieses Vorgehen ist für größere Graphen unbrauchbar.

Eingabe: ungerichteter Graph $G = (V, E)$
Ausgabe: $\min\{odil(\overrightarrow{G}) \mid \overrightarrow{G}$ ist eine Orientierung von G$\}$

```
1: all_orientations ← list // Liste aller 2^|E| verschiedenen Bitstrings
2: min_odil ← ∞
3: for G⃗ ∈ all_orientations do
4:    for idx, direction IN enumerate(orientation) do
5:       Richte die Kanten entsprechend der Orientation
6:    end for
7:    if nicht zusammenhängend then
8:       continue // Beendet die Schleife, wenn der Graph nicht stark zusammenhängt
9:    end if
10:   sp_matrix ← |V| × |V|- Matrix, sodass P_ij =Länge des kürzesten Pfades von i nach
      j in G⃗
11:   odil ← calculate_odil(pointset, sp_matrix)
12:   max_odil ← berechne odil von G⃗
13:   min_odil ← min (min_odil, max_odil)
14: end for
15: return min_odil
```

Algorithmus 3.1: Orientierung mit optimaler Dilation mittels BruteForce

Trotzdem ist der Bruteforceansatz für kleine Knotenmengen ein gutes Werkzeug um die Güte eines Optimierungsverfahrens mit Hinblick auf die errechnete *max-odil* zu bewerten.

3.2 ILP für konsistent orientierte Flächen

Ein Ansatz, der für das Orientieren des Graphen nahe liegt, ist das konsistente Orientieren der Flächen des Graphen.
Diese Herangehensweise hat den Vorteil, dass innerhalb jeder konsistent orientierten Fläche $f \in F$ für jeden Knoten $v \in V$ ein Pfad P zu jedem anderen Knoten der Fläche existiert. Das trifft noch keine Aussage über die erwartbare Güte dieser Lösung. Es gibt nach heutigem Stand keinen Beweis für eine theoretisch erwartbare Güte. Es konnte allerdings gezeigt werden, dass für konvexe Knotenmengen das konsistente Orientieren der Flächen für die gierige Triangulierung einen $O(n)$-Spanngraphen ergibt [5].

3.2.1 Naiver Ansatz

Im Folgenden wird ein ILP betrachtet, mit dessen Hilfe die Anzahl der konsistent orientierten Flächen maximiert wird. Gegeben ist ein Graph $G = (V, E)$ mit Flächen F. Außerdem die Variablen $p_i, a_i, b_i, CW_e, CCW_e \in 0, 1$ sowie $E_f \in \mathbb{R}^+$ als Summe der Kanten einer Fläche $f \in F$
Dabei zeigen a_i und b_i an, ob eine Fläche $f_i \in F$ konsistent in CW oder CCW orientiert ist. Die Variable p_i speichert, ob eine Fläche $f_i \in F$ konsistent orientiert ist.
Die Variablen CW_e und CCW_e speichern für eine Kante $e \in E$ die Orientierungsrichtung. Es werden zwei Variablen zum Speichern der Richtung genutzt, da so die spätere Identifikation unorientierter Kanten leichter ist.
Die Zielfunktion, mit der das ILP arbeitet lautet wie folgt:

Zielfunktion des ILPs

maximiere $\sum_{i=1}^{n} p_i$

Die Berechnung von p_i ist folgendermaßen definiert:

Berechnung von p_i für eine Fläche f_i

$p_i = a_i + b_i$

Die konsistente Orientierung einer Fläche muss nun mit ILP-Bedingungen ausgedrückt werden. Zu diesem Zweck werden die Kanten E_f jede Fläche $f \in F$ im Graphen, mit Ausnahme der konvexen Hülle, in CW- und CCW Kanten zerlegt.
Handelt es sich bei der Kante e um eine CW Kante, wird ihr $CW_e = 1$ und ihr $CCW_e = 0$ gesetzt. Handelt es sich um eine CCW Kante, wird ihr $CCW_e = 1$ und ihr $CW_e = 0$ gesetzt.
Für das ILP ergeben sich damit folgende Bedingungen:

Konsistenzbedingungen für die Orientierung

1. $a_i \leq \frac{\sum \mathrm{CW}}{E_f}$ — also muss a_i kleiner oder gleich der Summe der CW Kanten der Fläche geteilt durch die gesamte Kantenanzahl und damit kleiner gleich eins sein. Dabei kann a_i nur eins sein, wenn alle Kanten der Fläche in CW sind.
2. $a_i \geq \frac{\sum \mathrm{CW} - (E_f - 1)}{E_f}$ — also muss a_i größer oder gleich der Summe der CW Kanten, abzüglich der gesamten Kantenanzahl minus eins, geteilt durch die gesamte Kantenanzahl sein. Somit kann a_i nicht gleich null sein, wenn alle Kanten in CW sind.
3. $b_i \leq \frac{\sum \mathrm{CCW}}{E_f}$ — analog zu (1)

4. $b_i \geq \frac{\sum \text{CCW} - (E_f - 1)}{E_f}$ analog zu (2)

Zusätzlich muss beachtet werden, dass beim Betrachten verschiedener Flächen Kanten, die bereits für andere Flächen orientiert wurden, nicht vom ILP-Solver opportun umorientiert werden. Daher bedarf es noch der folgenden Bedingung:

Exklusivität der Kantenorientierung

$CW_e + CCW_e \leq 1$

Dies bedeutet, dass für eine Kante e nur CW_e oder CCW_e auf eins gesetzt werden kann.

3.2.2 Ansatz mit Graphfärbung

Der naive Ansatz funktioniert zwar, ist aber schon für kleine Knotenmengen (|V| = 20) nicht besonders performant. Zum einen die bereits erwähnte NP-Vollständigkeit von ILPs. Zum Anderen versucht der naive Ansatz des ILPs bisher sämtliche Flächen im Graphen konsistent zu orientieren. Allerdings sind nicht alle Graphen konsistent orientierbar. Dies hat zur Folge, dass das ILP versucht, die Zielfunktion auf einen Wert zu bringen, der nicht erreicht werden kann. Um das ILP performanter zu machen, kann die Zielfunktion entsprechend eingeschränkt werden. Die Variabel y ist definiert als $y = |F| - |F|_{\text{unorientable}}$. $F_{\text{unorientable}}$ ist die Anzahl der Flächen die nicht konsistent orientiert werden können.

Einschränkung der Zielfunktion

$\sum_{f=1}^{n} (a_i + b_i) \leq y$

Ziel ist es nun, diese nicht konsistent orientierbaren Flächen zu identifizieren. Zu diesem Zweck wird die bereits vorgestellte Knotenfärbung des dualen Graphen genutzt. Um dennoch eine bestmögliche 2-Färbung zu erhalten, wird eine 4-Färbung des dualen Graphen ausgeführt und mittels eines 4-Färbbarkeits ILPs die Nutzung der dritten und vierten Farbe minimiert.
Es muss noch die Variable $x_{i,c} \in 0, 1$ mit $c \in 1, 2, 3, 4$ eingeführt werden. $x_{i,c}$ zeigt an, ob einem Knoten $v_i \in V$ die Farbe c zugewiesen wurde.
Die Zielfunktion dieses ILPs lautet:

Zielfunktion

Min $\sum_{i \in V} x_{i,3} + x_{i,4}$

Die Bedingung, dass Knoten einfarbig sind, muss zusätzlich noch sichergestellt werden:

Knoten dürfen maximal eine Farbe haben

$$x_{i,1} + x_{i,2} + x_{i,3} + x_{i,4} = 1$$

Damit es sich um eine gültige Färbung des Graphen handelt, dürfen benachbarte Knoten nicht gleichfarbig sein:

Benachbarte Knoten müssen verschiedene Farben haben

$$x_{i,c} + x_{j,c} \leq 1 \quad \forall (i,j) \in E, \quad \forall c \in \{1,2,3,4\}$$

Das so konstruierte ILP liefert eine Färbung wie in Abb. 2.2. Das Ergebnis des ILPs für die 4-Färbung ist dann die minimale Anzahl Knoten dritter und vierter Farbe für einen gegebenen dualen Graphen.
Dies ist gleichzeitig die Anzahl der nicht konsistent orientierbaren Flächen im eigentlichen Graphen. Das Ergebnis des 4-Färbungs ILPs ist also $F_{\text{unorientable}}$ und dient damit als Input für das verbesserte ILP für konsistent orientierte Flächen.
Beide Versionen des ILPs finden die gleiche Anzahl konsistent orientierter Flächen.

3.2.1 Theorem. *Für Triangulierungen liefert das Orientieren mit dem ILP immer einen stark zusammenhängenden Graphen.*

Gegeben eine ungerichtete Triangulierung $T = (V, E)$, sei $\overrightarrow{T} = (V, \overrightarrow{E})$ die vom ILP orientierte Triangulierung. Es ist zu zeigen, dass $\overrightarrow{T}$ stark zusammenhängend ist. Sei p, q ein beliebiges Knotenpaar in V. Wir zeigen, dass es in $\overrightarrow{T}$ einen Pfad von p nach q gibt. Da T zusammenhängend ist, existiert ein Pfad von p nach q in T. Sei (u, v) eine beliebige Kante auf diesem Pfad. Ohne Beschränkung der Allgemeinheit ist die Kante $\overrightarrow{(u,v)}$ in $\overrightarrow{T}$. Es ist zu zeigen, dass in $\overrightarrow{T}$ ein Pfad von v nach u existiert. Es muss eine Fallunterscheidung über die Flächen, zu denen (u, v) benachbart ist, gemacht werden:

1. Falls die Kante (u, v) zu einem konsistenten Dreieck gehört, dann kann durch das Traversieren dieser Fläche von v aus u erreicht werden.

2. Für den Fall, dass die Kante (u, v) zu keinem konsistenten Dreieck gehört, ist die Fläche f_{uvw} zu betrachten. Die Kanten (v, w) und (w, u) gehören zu konsistent orientierten Flächen. Wäre dem nicht so, würde das ILP die Fläche f_{uvw} konsistent orientieren. Daher ist es entweder durch Traversieren der Kante (v, w) oder durch das Traversieren der konsistent orientierten Fläche zu der (v, w) gehört möglich, von v aus w zu erreichen. Für die Kante (w, u) gilt das gleiche Vorgehen. Somit ist es insgesamt möglich von v aus u zu erreichen.

Die Abbildung Abb. 3.1 zeigt alle möglichen Fälle, falls $\overrightarrow{(u,v)}$ nicht Teil einer konsistent orientierten Fläche ist.

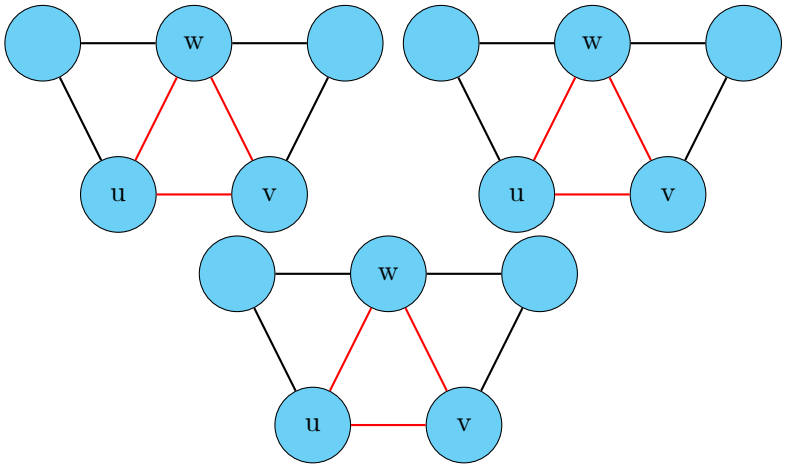

Abbildung 3.1: Wenn die Fläche f_{uvw} nicht konsistent ist, existiert immer ein Pfad von v nach u.

3.2.2 Theorem. *Für Graphen mit Flächen mit mehr als* 3 *Kanten, liefert das Orientieren mit dem ILP nicht immer einen stark zusammenhängenden Graphen.*

Wir zeigen die Aussage mit einem Gegenbeispiel. Der in Abb. 3.2 gezeigt Graph müsste,

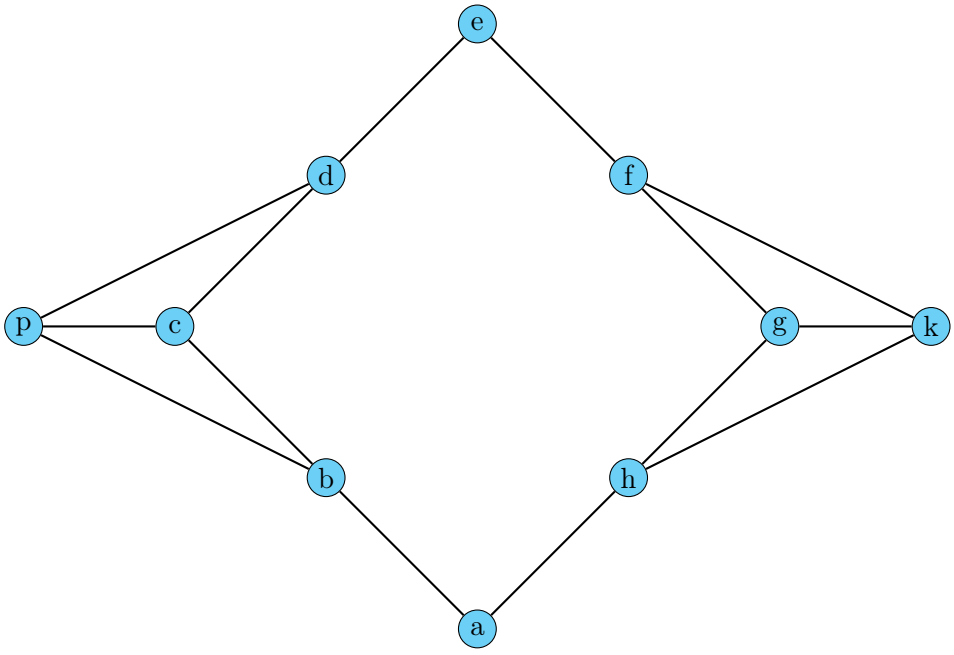

Abbildung 3.2: Der unorientierte Ausgangsgraph

um stark zusammenhängend zu sein, wie in Abb. 3.3 orientiert werden.

In dem in Abb. 3.3 gezeigten Graphen, sind zwar alle Knoten erreichbar, aber es sind zwei Flächen F_{cdp} und F_{ghk} nicht konsistent orientiert. Das ILP wird daher zu folgendem Ergebnis kommen:

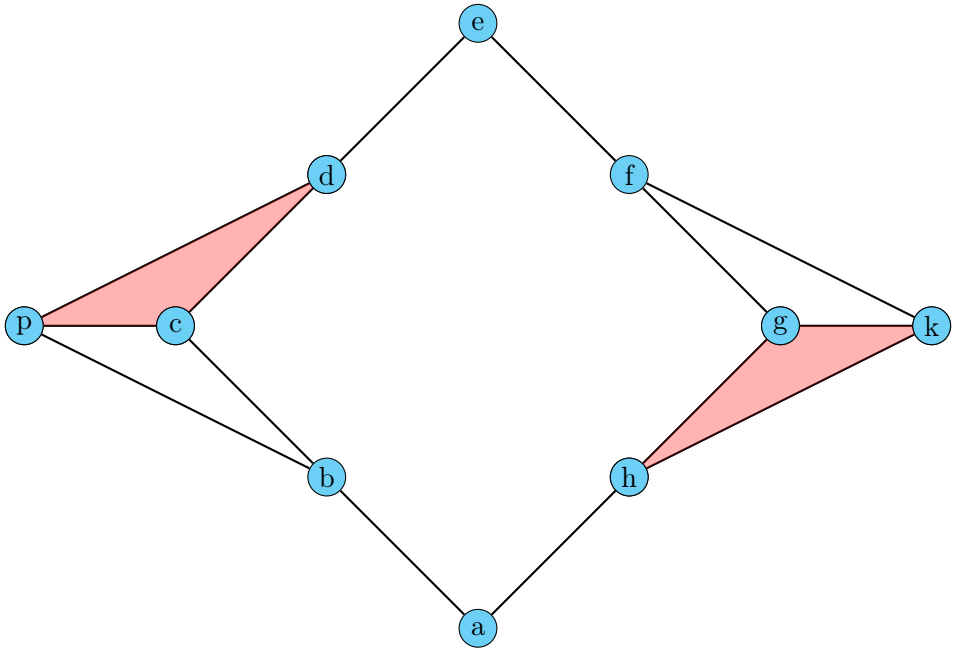

Abbildung 3.3: Ein Graph mit zwei nicht konsistent orientierten Flächen

In Abb. 3.4 ist nur noch eine Fläche nicht konsistent orientiert. Allerdings ist der Graph durch die Orientierung der Kanten $\overrightarrow{(b,a)}$, $\overrightarrow{(h,a)}$, $\overrightarrow{(d,e)}$ und $\overrightarrow{(f,e)}$ nicht mehr stark zusammenhängend. So gibt es keinen Pfad von c nach g.

In dem Gegenbeispiel ist es möglich, durch das Umorientieren von $\overrightarrow{(b,a)}$ zu $\overrightarrow{(a,b)}$ und $\overrightarrow{(f,e)}$ zu $\overrightarrow{(e,f)}$, den starken Zusammenhang herzustellen. Allerdings ist das ILP nicht in der Lage diese Verbesserung zu detektieren und eine derartige Umorientierung vorzunehmen.

Bruteforce Erweiterung

Um Graphen wie in Abb. 3.4 erfolgreich orientieren zu können, muss die Funktionalität des ILPs erweitert werden. Dabei wird ausgenutzt, dass die Orientierung aller Kanten, die nicht Teil einer konsistent orientierten Fläche sind, frei gewählt werden kann.
Nach Lösen des ILPs werden sämtliche Kanten, die nicht an konsistent orientierten Flächen liegen, durch ungerichtete Kanten ersetzt. Für diese ungerichteten Kanten werden in einem nächsten Schritt per Bruteforce alle möglichen Orientierungen getestet. Dabei wird für jede Permutation geprüft, ob der resultierende Graph stark zusammenhängend ist. Ist der starke Zusammenhang gegeben, werden die Orientierungen dieser Permutation für die Kanten übernommen und der Algorithmus terminiert.
Im Falle des Graphen in Abb. 3.4 würden die Kanten $\overrightarrow{(b,a)}$,$\overrightarrow{(d,e)}$,$\overrightarrow{(f,e)}$ und $\overrightarrow{(h,a)}$ durch die Kanten $(a,b),(c,d),(e,f) und (a,h)$ ersetzt werden.
Im zweiten Schritt würde der Algorithmus durch Bruteforcen dann mögliche Orientierun-

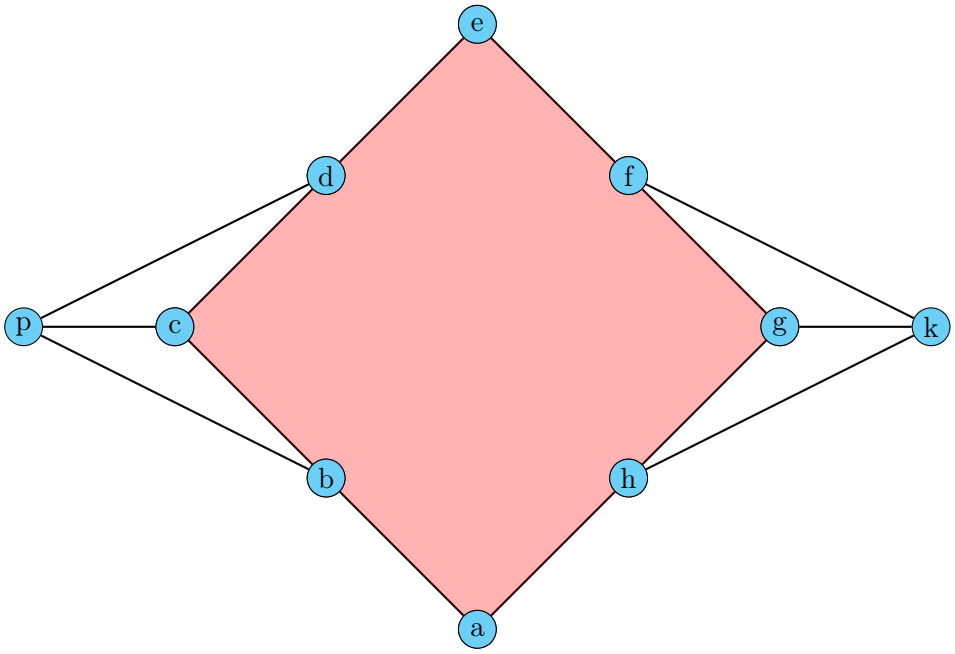

Abbildung 3.4: Der Graph aus 3.3, optimiert durch das ILP

gen für diese Kanten testen und beispielsweise $\overrightarrow{(a,b)}$,$\overrightarrow{(d,e)}$,$\overrightarrow{(e,f)}$ und $\overrightarrow{(h,a)}$ als gültiges Ergebnis ausgeben.

3.3 ILP für Kanten an orientierten Flächen

Ein Algorithmus der sich sowohl von der Herangehensweise als auch von der Implementierung an das ILP für konsistent orientierte Flächen anlehnt, ist das ILP für Kanten an orientierten Flächen. Die Grundüberlegung dieses Algorithmus ist es, dass anstatt jede Fläche zu orientieren, nun versucht wird sicherzustellen, dass jede Kante Teil mindestens einer orientierten Fläche ist.
In Abb. 2.5 ist dies beispielsweise für sämtliche Kanten bis auf die Kante $\overrightarrow{(C,A)}$ gegeben. Obwohl die Kante $\overrightarrow{(A,D)}$ Teil der nicht konsistent orientierten Fläche f_{ACD} ist, ist sie auch Teil der konsistent orientierten Fläche f_{ABD} und zählt damit als Kante an einer konsistent orientierten Fläche.

Eine wichtige Eigenschaft dieser Heuristik ist, dass für einen Graphen $G = (V, E)$ mit Flächen F alle Flächen $f_{\text{außen}} \in F$ deren Kanten $e_{\text{außen}} \in E$ Teil der konvexen Hülle sind mit hoher Priorität konsistent orientiert werden. Das ist eine Folge daraus, dass die $e_{\text{außen}}$ Kanten nur Teil von jeweils einer Fläche sind und diese daher konsistent orientiert sein muss, wenn die Kanten die Bedingung, an einer konsistent orientierten Fläche zu liegen, erfüllen sollen.

Der Aufbau des ILPs für Kanten an orientierten Flächen ist dem Aufbau des ILPs für konsistent orientierte Flächen auch sehr ähnlich. Die Bedingungen und Variablen sind komplett analog zu denen für das ILP für konsistent orientierte Flächen
Es gelten für diesen ILP die gleiche Bedingungen für die Konsistenz der Orientierung wie in 1, 2, 3, 4. Genau wie die Exklusivitätsbedingung 3.2.1. Auch die Berechnung von p_i für eine Fläche f_i erfolgt analog zu 3.2.1. Zusätzlich gibt es noch die Variable $o_e \in 0, 1$ die anzeigt, ob eine Kante e Teil einer konsistent orientierten Fläche ist.
Das ILP hat folgende Zielfunktion:

Zielfunktion

Max $\sum_{e \in E} o_e$

Bedingung für die Kanten o_e von f_i

$o_e \geq p_i \quad \forall e \in f_i$ und $f_i \in F$

Diese Bedingung stellt sicher, dass alle Kanten der Fläche f_i als an einer konsistent orientierten Fläche liegend betrachtet werden, wenn f_i konsistent orientiert ist.

Bedingung für das Setzen von o_e

$o_e \leq \sum_{i \in \text{Flächen, die } e \text{ enthalten}} f_i$

Mit dieser Bedingung wird sichergestellt, dass o_e nur auf 1 gesetzt werden kann, wenn es Teil einer konsistent orientierten Fläche f_i ist.
Es ist noch die Frage zu beantworten, ob der so orientierte Graph stark zusammenhängend ist. Es gilt für Triangulationen genau die gleiche Argumentation wie in 3.2.2.
Als Gegenbeispiel für den starken Zusammenhang von Polygonen mit mehr als drei Kanten, kann der K_4 aus Abb. 2.5 um sechs Außenkanten erweitert werden, was einen Graphen wie in Abb. 3.5 ergibt. Für diesen Graphen gilt, wie auch für den K_4, dass mindestens eine der Flächen nicht konsistent orientierbar ist. Dabei ist es für das ILP unerheblich, welche der Flächen nicht konsistent orientiert wird: Es läuft immer auf drei Kanten hinaus, die nicht an einer konsistent orientierten Fläche liegen.
In Abb. 3.5 ist das die Fläche f_{abcde}. Dabei tritt, wie auch schon für das ILP für konsistent orientierte Flächen, das Problem auf, dass es keine Heuristik gibt, die das Orientieren der Kanten (c, d), (d, e) und (b, e) betrachtet. In diesem Beispiel sorgt die Orientierung der Kanten $\overrightarrow{(d, e)}$ und $\overrightarrow{(b, e)}$ dafür, dass es für den Knoten e keine ausgehende Kante und damit keinen Pfad von e zu einem anderen Knoten gibt. Es ist dabei leicht zu sehen, dass das Umorientieren von $\overrightarrow{(b, e)}$ nach $\overrightarrow{(e, b)}$ den Graphen stark zusammenhängend machen würde. Dabei lässt sich das ILP genauso wie das ILP für konsistent orientierte Flächen mit einer

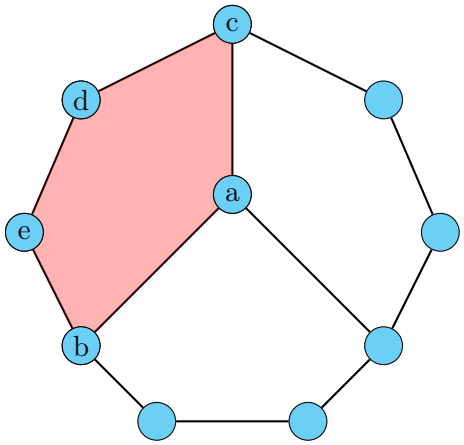

Abbildung 3.5: Ein abgewandelter K_4 für den das ILP niemals einen starken Zusammenhang garantieren kann

Bruteforce Methode für nicht an konsistent orientierten Flächen liegenden Kanten erweitern.

3.4 Gierige Orientierung

Bisher wurden Orientierungsalgorithmen genutzt, die ungeachtet der eigentlichen Graphgeometrie orientieren. So ist es bei Auftreten eines K_4-Subgraphen, wie in Abb. 2.5, egal welchen Umfang die einzelnen Flächen haben: Die nicht konsistent orientierte Fläche wird zufällig gewählt.
Für die Berechnung der odil eines Graphen ist der Umfang der Flächen durchaus von Interesse.
Es lässt sich intuitiv die schlussfolgern, dass es für die Minimierung der odil von Interesse ist, kleine Flächen vor großen Flächen konsistent zu orientieren.
Darauf aufbauend wird nun die Herangehensweise für den gierigen Orientierungsalgorithmus entworfen:

1. Zunächst werden die Flächen des Graphen anhand ihres Umfangs aufsteigend sortiert.
2. Der Algorithmus prüft nun, beginnend mit der Fläche mit dem geringsten Umfang, ob diese noch konsistent orientierbar ist. Hierzu wird eine Tiefensuche entlang der Flächenkanten ausgeführt und geprüft, ob es noch möglich ist, mit Flächenkanten und ihren Knoten einen Kreis zu beschreiben.
3. Ist eine konsistente Orientierung möglich, wird die Fläche konsistent, eventuell unter Berücksichtigung der bereits orientierten Kanten, orientiert. Ist keine konsistente

Orientierung möglich, wird diese Fläche nicht weiter behandelt und mit der nächstgrößeren Fläche weitergemacht.

4. Wurden alle Flächen betrachtet werden alle übriggebliebenen Kanten, also Kanten von Flächen, die nicht mehr konsistent orientierbar waren und die nicht durch andere konsistent orientierbare Flächen orientiert wurden, zufällig orientiert.

Für Triangulationen gilt bezüglich der Frage nach dem starken Zusammenhang des gierig orientierten Graphen die gleiche Argumentation wie in 3.2.2.
Für Flächen mit mehr als drei Kanten kann das Gegenbeispiel aus Abb. 3.4 erneut genutzt werden. Angenommen der Umfang der Fläche $f_{abcdefgh}$ ist größer als der Umfang der vier anderen Flächen. So würde der gierige Algorithmus diese Fläche zuletzt zu orientieren versuchen.
Da in diesem Fall eine konsistente Orientierung nicht mehr möglich ist, würde der Algorithmus die Kanten (A, B),(A, H),(D, E) und (E, F) laut Verfahren zufällig orientieren, was zum Auftreten des gezeigten, nicht stark zusammenhängenden Graphen, führen kann. Auch in diesem Fall kann durch eine Erweiterung wie für das ILP für konsistent orientierte Flächen der starke Zusammenhang sichergestellt werden.

Eingabe: ungerichteter Graph $G = (V, E, F)$
Ausgabe: orientierter Graph $\overrightarrow{G} = (V, \overrightarrow{E}, F)$

```
1: sortiere die Liste alle Flächen in aufsteigender Reihenfolge nach Umfang
2: for f ∈ F do
3:    if ∀e ∈ f_i noch nicht orientiert then
4:       f_i wird zufällig in CW oder CCW orientiert
5:    else
6:       if f_i ist noch konsistent orientierbar then
7:          f_i wird entsprechend der bereits orientierten Kanten in CW oder in CCW
            orientiert.
8:       end if
9:    end if
10: end for
11: orientiere alle verbleibenden Kanten zufällig
12: return G→
```

Algorithmus 3.2: Gierige Orientierung der Kanten eines planaren Graphen

3.4.1 Theorem. *Die Laufzeit des gierigen Algorithmus liegt in* $O(N * log(N))$

Das Berechnen der Flächenumfänge für jede Fläche $f \in F$ benötigt jeweils $|v_f|$ Berechnungen, wobei $|v_f|$ die Anzahl der Knoten adjazent zu der Fläche f sind. Über alle Flächen

ergibt sich damit O(|F| * |V|) Zeit.
Das Sortieren der berechneten Flächenumfänge nach ihrer Größe, benötigt $O(|F|*log(|F|))$ Zeit.
Die Tiefensuche zur Kreisdetektion für eine Fläche $f \in F$ hat $|v_f|$ Schritte. Sie wird im worst case für jede Fläche, außer der Ersten, in beide Richtungen ausgeführt. Außerdem wird auch hier jede Kante, die keine Außenkante ist, zweimal betrachtet. Damit ergibt sich eine Gesamtlaufzeit von $O(|F| * |V|)$ Zeit.
Das zufällige Orientieren der verbleibenden Kanten prüft für alle Kanten, ob sie orientiert sind und führt, falls dem nicht so ist, eine Orientierung per Zufall aus. Daraus ergibt sich eine Laufzeit von $O(|E|)$ Zeit.
Zusammengenommen ist zu erkennen, dass die Gesamtlaufzeit von $O(|F| * log(|F|))$ dominiert wird. Daher ist die Worst Case Laufzeit in $O(N * log(N))$ Zeit, eine definitive Verbesserung gegenüber der Worst Case Laufzeit der ILPs.

3.4.1 Flächen-ILP mit Geometrie Bias

Aus dem die Geometrie beachtenden, gierigen Ansatz ergibt sich zusätzlich noch eine weitere mögliche Verbesserung für das ILP für konsistent orientierte Flächen:
In dem die Zielfunktion abgewandelt wird, ist es für das ILP auch möglich, die Geometrie des Graphen zu berücksichtigen:

Zielfunktion mit Geometrie Bias

maximiere $\quad \sum_{i=1}^{n} p_i \cdot \frac{1}{|E_i| \cdot x}$

p_i ist wie im ursprünglichen ILP eine binäre Variable, die anzeigt, ob eine Fläche f_i konsistent orientiert ist oder nicht. $|E_i|$ ist dabei die gesamte Länge der Kanten von f_i. Die Variable x ist ein Bias Faktor der beliebig angepasst werden kann, um den Einfluss der Kantenlängen auf das Ergebnis zu verstärken oder abzuschwächen.
Da in dieser neuen Zielfunktion p_i durch die gesamte Länge der Kanten geteilt wird, ist der Einfluss von großen Flächen auf das Ergebnis niedriger als der Einfluss von kleinen Flächen. Somit wird das konsistente Orientieren von kleinen Flächen priorisiert.
Die konkreten Effekte dieser Abänderung werden im nächsten Kapitel genauer betrachtet.

Literaturverzeichnis

[1] Aronov, B. ; Buchin, K. ; Buchin, M. ; Jansen, B. ; Jong, T. D. ; van Kreveld, M. ; Löffler, M. ; Luo, J. ; Speckmann, B. ; Silveira, R. I.: Connect the Dot: Computing Feed-Links for Network Extension. In: *J. Spatial Information Science* 3 (2011), S. 3–31. http://dx.doi.org/10.5311/JOSIS.2011.3.47. – DOI 10.5311/JOSIS.2011.3.47

[2] Berg, M. de ; Cheong, O. ; Kreveld, M. van ; Overmars, M. : *Computational Geometry: Algorithms and Applications*. 3rd. Springer, 2008. – XII, 386 S. http://dx.doi.org/10.1007/978-3-540-77974-2. – ISBN 978–3–540–77974–2

[3] Bondy, J. A. ; Murty, U. S. R.: *Graph Theory with Applications*. Macmillan, 1976. http://dx.doi.org/10.1137/1021086. – ISBN 9780333177914

[4] Buchin, K. ; Gudmundsson, J. ; Kalb, A. ; Popov, A. ; Rehs, C. ; Renssen, A. van ; Wong, S. : Oriented Spanners. In: *31st Annual European Symposium on Algorithms (ESA 2023)*, 2024. http://dx.doi.org/10.4230/LIPIcs.ESA.2023.26, S. 1–25

[5] Buchin, K. ; Kalb, A. ; Rehs, C. ; ; Schulz, A. : Oriented dilation of undirected graphs. In: *EuroCG2024 - 40th European Workshop on Computational Geometry*, 2024, S. 1–8

[6] Burkhart, M. ; Rickenbach, P. V. ; Wattenhofer, R. ; Zollinger, A. : Does Topology Control Reduce Interference? In: *Proc. 5th ACM Internat. Sympos. Mobile Ad Hoc Networking and Computing*, 2004. http://dx.doi.org/10.1145/989459.989462, S. 9–19

[7] Conforti, M. ; Cornuéjols, G. ; Zambelli, G. : *Integer Programming*. Springer, 2020. http://dx.doi.org/10.1007/978-3-319-11008-0. – ISBN 9783319110073, 9783319110080

[8] Diestel, R. : *Graduate texts in mathematics*. Bd. 173: *Graphentheorie*. Springer, 2017. – xviii + 428 S. http://dx.doi.org/10.1007/978-3-662-53622-3. – ISBN 9783662536216

[9] Dobson, A. ; Bekris, K. E.: Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning. In: *Internat. J. Robotics Research* 33 (2014), Nr. 1, S. 18–47. http://dx.doi.org/10.1177/0278364913498292. – DOI 10.1177/0278364913498292

[10] Klein, R. ; Kutz, M. : Computing Geometric Minimum-Dilation Graphs Is NP-Hard. In: *Proc. Graph Drawing (GD 2006), Lecture Notes in Computer Science* Bd. 4372, Springer, 2007. http://dx.doi.org/10.1007/978-3-540-70904-6_20, S. 196–207

[11] Lewis, R. M. R.: *Guide to Graph Colouring: Algorithms and Applications*. 2. Springer Cham, 2021 (Texts in Computer Science). http://dx.doi.org/https://doi.org/10.1007/978-3-030-81054-2. – ISBN 978–3–030–81054–2

[12] Narasimhan, G. ; Smid, M. : *Geometric Spanner Networks*. Cambridge University Press, 2007. http://dx.doi.org/10.1017/CBO9780511546884

[13] Preparata, F. P. ; Shamos, M. I.: *Computational Geometry: An Introduction*. Springer, 1985 (Monographs in Computer Science). http://dx.doi.org/10.1007/978-1-4612-1098-6. – ISBN 978–0–387–96131–6

[14] Robbins, H. E.: A theorem on graphs, with an application to a problem on traffic control. In: *American Mathematical Monthly* 46 (1939), Nr. 5, S. 281–283. http://dx.doi.org/10.2307/2303897. – DOI 10.2307/2303897

[15] Robertson, N. ; Sanders, D. ; Seymour, P. ; Thomas, R. : The four-colour theorem. In: *Journal of Combinatorial Theory. Series B* 70 (1997), Mai, S. 2–44. http://dx.doi.org/10.1006/jctb.1997.1750. – DOI 10.1006/jctb.1997.1750. – ISSN 0095–8956

[16] Volkmann, L. : *Fundamente der Graphentheorie*. Springer, 1996. – 313 S. http://dx.doi.org/10.1007/978-3-7091-9449-2. – ISBN 3–211–82774–9

[17] Wilson, R. : *Introduction to Graph Theory*. Longman, 2010. – 75–76 S. – ISBN 9780273728894