Ergebnisse der Berechnungen

Allgemeine Begriffe

Gegeben seien ein Graph $G = (V,\ E)$ und eine positive ganze Zahl k. Die k-Färbung des Graphen G ist eine Funktion $c : V \to \{1, 2, \dots k\}$ aus der Menge der Knoten des Graphen G in die Menge von ganzen Zahlen, die kleiner gleich k sind (siehe Harris et al., 2008).
Die Färbung heißt zulässig, wenn für alle Paare $u, v \in V$ adjanzenter Knoten $c(u) \neq c(v)$ gilt.
Die $L(d_1, d_2, \dots, d_r)$-Labeling, wobei r kleine gleich maximales Durchmesser des Graphen G ist, ist eine Färbung mit der Eigenschaft, dass die Labels eines Paars von Knoten mit der Abstand i sich mindestens um d_i unterscheiden. Mit anderen Worten, es soll gelten $|c(u) - c(v)| \geq d_i \ \ \forall u, v \in V : dist(u, v) = i, 1 \leq i \leq r$ (vgl. Bertossi et al., 2003).
Wir bezeichnen mit $dist(u, v)$ die Distanz zwischen den Knoten $u, v \in V$, d.h. den kürzest Weg zwischen Knoten u und v.

Modelle

Wir betrachten zur erst das klassische Modell zum Frequenzzuordnungsproblem (engl. Channel Assigment Problem, CAP). Dieses Problem ist dem $L(d_1, d_2, \dots, d_r)$-Labeling-Problem der Graphentheorie äquivalent.
Gegeben seien ein zusammenhängender Graph $G = (V,\ E)$ und eine Folge $d_1 \geq d_2 \geq \dots \geq d_r$ der ganzen Zahlen, wobei r kleine gleich maximales Durchmesser des Graphen G ist. Gesucht wird eine minimale Zahl λ, sodass Graph G eine $L(d_1, d_2, \dots, d_r)$-Labeling $c : V \to \mathbb{Z}$ mit $\lambda = \max\{c(u), u \in V\}$ hat.
Wir nummerieren alle Knoten des Graphen G durch und verwenden weiter die Schreibweise c_i anstatt $c(i)$ für den i-ten Knoten des Graphen G.
Das von uns betrachtete ganzzahliges Optimierungsproblem sieht folgendermaßen aus:

$$\min_{\lambda \in \mathbb{Z}} \lambda \tag{1}$$

$$\begin{aligned}
\textbf{s.t. } & c_i \leq \lambda, && \forall i \in V && (2)\\
& c_i \geq 0, && \forall i \in V && (3)\\
& c_j - c_i + M z_{ij} \geq d_{dist(i,j)}, && \forall i, j \in V && (4)\\
& c_i - c_j + M(1 - z_{ij}) \geq d_{dist(i,j)} && \forall i, j \in V && (5)\\
& z_{ij} \in \{0, 1\} && \forall i, j \in V && (6)\\
& c_i \in \mathbb{Z}, && \forall i \in V && (7)
\end{aligned}$$

Die zusätzliche Variable z_{ij} ist gleich 1, falls $c_i \geq c_j$ gilt, und 0 sonst. Daraus folgt, dass die Bedingungen 4 und 5 mit dem hinreichend großen M der Bedingung $|c_i - c_j| \geq d_{dist(i,j)}$ entsprechen.
Wir möchten nun dieses Problem auf den mehr allgemeineren Fall der reellwertigen Variablen $c_i \in \mathbb{R}$, übertragen. Dafür formulieren wir ein erweitertes Modell:

$$\min_{\lambda \in \mathbb{R}} \lambda \tag{8}$$

$$\textbf{s.t. } c_i \leq \lambda, \qquad \forall i \in V \tag{9}$$

$$c_i \geq 0, \qquad \forall i \in V \tag{10}$$

$$c_j - c_i + M z_{ij} \geq f(dist(i,j)), \qquad \forall i,j \in V \tag{11}$$

$$c_i - c_j + M(1 - z_{ij}) \geq f(dist(i,j)) \qquad \forall i,j \in V \tag{12}$$

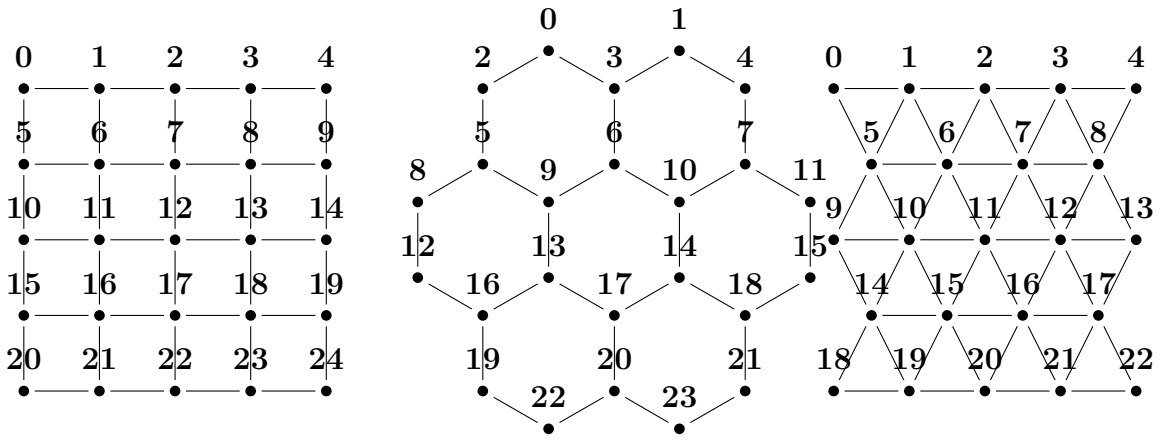
$$z_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{13}$$

$$c_i \in \mathbb{R}, \qquad \forall i \in V \tag{14}$$

Hier unter der Distanz $dist(i,j)$ wird eine euklidische Distanz zwischen den Knoten i,j gemeint und Funktion f ist eine Abbildung $\mathbb{R} \to \mathbb{R}$.
Wir möchten die zwei formulierte Modelle vergleichen. Dafür betrachten wir im klassischen Fall die $L(2,1)$ und $L(3,2,1)$-Labelings. Dies entspricht der Wahl der linearen Funktion $f(x) = d - x$ mit den Konstanten $d = 3$ und $d = 4$ im zweiten Fall.
Die weiter im Text aufgeführten Ergebnisse der Berechnungen sind mit Hilfe von IBM® ILOG® CPLEX® Optimization Studio C++ Bibliothek von IBM (Version 12.51) erhalten. Zur Lösung von gemischt-ganzzahligen Optimierungsproblemen wird in dieser Bibliothek *Branch&Cut*-Algorithmus angewendet [1].

Gridgraphen

Der Vergleich wird für spezielle Graphen durchgeführt, nämlich Gridgraphen. Dabei werden die Graphen mit den Zellen als Dreiecke (insgesamt 23 Knoten), Vierecke (insgesamt 25 Knoten) und Sechsecke (insgesamt 24 Knoten) betrachtet (siehe Abb.1).



a) Square Lattice b) Hexagonal Lattice c) Triangular Lattice

Abbildung 1: In Betracht genommene Gridgraphen

[1] http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/index.jsp?topic=%2Filog.odms.cplex.help%2Frefcppcplex%2Fhtml%2Fbranch.html

Für diese Graphen lassen sich die Werte der $L(2,1)$ und $L(3,2,1)$-Labelings einfach berechnen. Das Modell mit den reellen Zahlen ist schwerer zu lösen und braucht deswegen deutlich mehr Zeit (siehe Tabelle 1).

Lattice	$f(x) = 3 - x$	$L(2,1)$		$f(x) = 4 - x$	$L(3,2,1)$
Hexagonal	6.41699	5		16.6243	9
	1.69 sec	0.08 sec		1838.73 sec	0.29 sec
Triangular	9.78029	8		21.9068	18
	10.96 sec	0.34 sec		103883 sec	167.29 sec
Square	8.63494	6		19.9067	11
	4.92 sec	0.2 sec		27793.8 sec	0.68 sec

Tabelle 1: Ergebnisse für $L(2,1)$, $L(3,2,1)$ im klassischen Fall und Funktion $f(x) = 3-x$, $f(x) = 4 - x$ im Fall der reellwertigen Labeling

0.1 Treppenfunktion

Alternativ zu der linear absteigender Funktion kann für f eine Treppenfunktion gewählt werden. Anhand der Zusammenhang zwischen den Graphdistanzen und euklidischen Distanzen, siehe Tabelle 3, 4, 5, haben wir die Treppenfunktionen für die entsprechende klassische Formulierung wie folgt definiert:

$$f_{hexagonal,L(2,1)}(x) = \begin{cases} 2 & \text{wenn } x \leq 1 \\ 1 & \text{wenn } x \leq \sqrt{3} \end{cases}$$

$$f_{hexagonal,L(3,2,1)}(x) = \begin{cases} 3 & \text{wenn } x \leq 1 \\ 2 & \text{wenn } x \leq \sqrt{3} \\ 1 & \text{wenn } x \leq \sqrt{7} \end{cases}$$

$$f_{triangular,L(2,1)}(x) = \begin{cases} 2 & \text{wenn } x \leq 1 \\ 1 & \text{wenn } x \leq 2 \end{cases}$$

$$f_{triangular,L(3,2,1)}(x) = \begin{cases} 3 & \text{wenn } x \leq 1 \\ 2 & \text{wenn } x \leq 2 \\ 1 & \text{wenn } x \leq 3 \end{cases}$$

$$f_{square,L(2,1)}(x) = \begin{cases} 2 & \text{wenn } x \leq 1 \\ 1 & \text{wenn } x \leq 2 \end{cases}$$

$$f_{square,L(3,2,1)}(x) = \begin{cases} 3 & \text{wenn } x \leq 1 \\ 2 & \text{wenn } x \leq 2 \\ 1 & \text{wenn } x \leq 3 \text{ und } x \neq \sqrt{8} \end{cases}$$

Die Ergebnisse der Berechnungen mit den Treppenfunktionen sind gleich den aus der klassischen Formulierung, brauchen aber mehr Zeit, um fertig zu werden (vgl. Tabelle 2).

Lattice	$f(x)$	$L(2,1)$		f(x)	$L(3,2,1)$
Hexagonal	5	5		9	9
	0.61 sec	0.08 sec		1.59 sec	0.29 sec
Triangular	8	8		18	18
	7.94 sec	0.34 sec		1170.4 sec	167.29 sec
Square	6	6		11	11
	6.51 sec	0.2 sec		68.91 sec	0.68 sec

Tabelle 2: Ergebnisse für $L(2,1)$ und $L(3,2,1)$ im klassischen Fall und Treppenfunktionen im Fall der reellwertigen Labeling

Graphdistanz	euklidische Distanz
1	1
2	$\sqrt{3}$
3	2,$\sqrt{7}$
4	3, $\sqrt{12}$
5	$3\sqrt{13}$, 4, $\sqrt{19}$
6	$\sqrt{21}$, $\sqrt{27}$
7	5, $\sqrt{28}$

Tabelle 3: Graph- und euklidische Distanzen in einem hexagonalen Gridgraphen mit 24 Knoten.

Graphdistanz	euklidische Distanz
1	1
2	$\sqrt{3}$, 2
3	$\sqrt{7}$, 3
4	$\sqrt{12}$, $\sqrt{13}$, 4
5	$\sqrt{19}$, $\sqrt{21}$
6	$\sqrt{28}$

Tabelle 4: Graph- und euklidische Distanzen in einem Gridgraphen aus Dreiecken mit 23 Knoten.

Graphdistanz	euklidische Distanz
1	1
2	$\sqrt{2}$, 2
3	$\sqrt{5}$, 3
4	$\sqrt{8}$, $\sqrt{10}$, 4
5	$\sqrt{13}$, $\sqrt{17}$
6	$\sqrt{18}$, $\sqrt{20}$
7	5
8	$\sqrt{32}$

Tabelle 5: Graph- und euklidische Distanzen in einem Gridgraphen aus Vierecken mit 25 Knoten.

0.2 Verbessern der Laufzeit

Wie wir gesehen haben, ist das Modell mit den reellen Zahlen viel langsamer als es mit den ganzen Zahlen. Da aber das erste nah an der Praxis liegt, ist es von Interesse, die Laufzeit des Modells zu verbessern. Wir beschreiben weiter verschiedene Ansätze, die diesem Ziel dienen sollten.

- Beschränkung der Konstanten M
  Um die Nichtkonvexität des Problems wegen der Ungleichung $|c_i - c_j| \geq d_{dist(i,j)}$ zu beheben, haben wir die Umschreibung mit dem "großen M"benutzt (Ungleichungen 11, 12). Die hohen Werte vom M vergrößern die zulässige Menge vom Problem 3 und verlangsamen somit die Optimierung.

  Aus der klassischen Formulierung folgt, dass es genügt, dass $M \geq d_1 + \lambda$ ist (siehe Halasz, 2013). Durch die Abschätzungen für λ können wir somit auch die Konstante M abschätzen ($M >= nd_1$, wobei n Anzahl der Knoten im Graphen ist (siehe Halasz, 2013)). Alternative können wir M durch den Lösung λ_Z des entsprechenden klassischen Problem beschränken, z.B. durch $M \leq 3\lambda_Z$. Die Ergebnisse der Berechnung mit dem so beschränkten Konstanten M sind in der Tabelle 6 zusammengefasst.

Lattice	$d(x) = 3 - x$(new)	$d(x) = 3 - x$(old)		$d(x) = 4 - x$(new)	$d(x) = 4 - x$(old)
Hexagonal	6.41699	6.41699		16.6243	16.6243
	1.82 sec	1.69 sec		6651.87 sec	1838.73 sec
Triangular	9.78029	9.78029		21.9068	21.9068
	8.88 sec	10.96 sec		59064.01 sec	103898.60 sec
Square	8.63494	8.63494		19.9067	19.9067
	7.25 sec	4.92 sec		21222.8 sec	27793.8 sec

Tabelle 6: Der Vergleich der Laufzeit des Modells und des Modells mit den zusätzlichen Beschränkungen auf Konstanten M: für $d(x) = 3 - x$ ist $M \leq 2\lambda_Z$, für $d(x) = 4 - x$ ist $M \leq 3\lambda_Z$, wobei λ_Z das Ergebnis des entsprechenden klassischen Problems ist, also $L(2,1)$ und $L(3,2,1)$.

  Wie können sehen, dass bis auf einen Lauf ist das Modell schneller geworden. Die Erhöhung der Laufzeit im Fall des Graphen aus den Sechsecken kann daran liegen, dass die hinzugefügte Ungleichungen nicht hinreichend sind. Sie schaffen zusätzliche Schwierigkeiten für das Programm, da sie erfüllt werden müssen, helfen aber beim *Branch&Bound* nicht.

- Beschränkung von λ

  Ein anderer Ansatz ist zu versuchen, gute obere Schranke für λ zu finden. Dafür haben wir den Algorithmus der binären Suche verwendet.

  Die Idee ist, solange die obere und untere Schranken vom λ zusammen zu ziehen, bis die Lücke zwischen ihnen klein genug ist, und erst danach die Optimierung mit

den strengeren Schranken zu starten. Als untere Schranke (LB) für λ wurde das Ergebnis der entsprechenden klassischen Formulierung genommen (λ_Z), und als obere Schranke (UB) die Werte $2\lambda_Z$ für die Funktion $d(x) = 3 - x$ und $3\lambda_Z$ für die Funktion $d(x) = 4 - x$. In jeder Iteration wird versucht entweder untere Schranke nach oben oder obere Schranke nach unten um die Hälfte der Lückengröße $UB - LB$ zu verschieben. Dabei wird jedes mal ein Zulässigkeitsproblem gelöst.

Die Ergebnisse dieses Ansatzes für die zulässige Lückengröße 0.5 sind in der Tabelle 7 aufgeführt.

Lattice	$d(x) = 3 - x$(new)	$d(x) = 3 - x$(old)		$d(x) = 4 - x$(new)	$d(x) = 4 - x$(old)
Hexagonal	6.41699 $5.18 + 21.59 =$ 26.77 sec	6.41699 1.69 sec		16.6243 $118061 + 139576 =$ 257637 sec	16.6243 1838.73 sec
Triangular	9.78029 $105.8 + 253.17 =$ 358.97 sec	9.78029 10.96 sec		21.9068	21.9068 103898.60 sec
Square	8.63494 $33.99 + 55.26 =$ sec 89.25 sec	8.63494 4.92 sec		19.9067	19.9067 sec 27793.8 sec

Tabelle 7: Ansatz der *binären Suche* für die Bestimmung oberer und unterer Schranken für λ. Die Lückengröße ist 0.5. Der erste Summand in der Summe steht für die Zeit, die die Suche nach den besseren Schranken braucht, und der zweite Summand steht für die eigentliche Optimierung.

Man kann sehen, dass dieser Verfahren im Gegensatz zu den Erwartungen keine Zeitverbesserung geleistet hat.

Man kann es dadurch erklären, dass nach dem wir die zulässige Menge des Problems verkleinert haben, ist es dem Cutting Algorithmus schwer die nicht optimale Zweige im Suchbaum abzuschneiden und er enumeriert alle Möglichkeiten.

- Zusätzliche Ungleichungen

  Um das Lösungsprozess zu verbessern, kann man in das vorhandene Modell zusätzliche Ungleichungen hinzufügen, die die zusätzliche Abhängigkeiten zwischen den Variablen ausdrucken.

  Um solche Ungleichungen zu gewinnen, haben wir in unserer Untersuchung die Teilgraphen des ursprünglichen Graphen G beobachtet. Dabei löst man für die Teilgraphen separat das Problem der Minimierung der Summe der Labels, die die Ungleichungen 9-14 erfüllen.

  Sei λ' das Ergebnis eines solchen Problems für einen Teilgraphen G'. Für den ganzen Gridgraphen fügt man dann für alle Knoten, die den Teilgraphen G' bilden, zusätzliche Ungleichungen in Form $\sum_{i \in V(G')} c(i) \geq \lambda'$ hinzu.

Die Form und die Größe der Teilgraphen G' kann variiert werden. Wir haben hauptsächlich die Bauelemente der Gridgraphen (entspricht Dreiecke, Vierecke und Sechsecke) in Betracht benommen, u.a.: Dreiecke mit der Seitenlänge 1, 2, 3, Sechsecke mit der Seitenlänge 1, 2 , Trapez, das die Hälfte eines Sechseckes darstellt, Vierecke der Größe 1, 2, 3 und kleine viereckige Gridgraphen mit 4, 9 Knoten (siehe Tabelle 8, wo für jeden untersuchten Teilgraphen optimale Werte der Summe $\sum_{i \in V(G')} c(i)$ eingeführt sind).

Teilgraph	$d(x) = 3 - x$	$d(x) = 4 - x$
Dreieck mit der Seitenlänge 1	$\sum c(i) = 6$	$\sum c(i) = 9$
Dreieck mit der Seitenlänge 2	$\sum c(i) = 3$	$\sum c(i) = 6$
Dreieck mit der Seitenlänge 3	$\sum c(i) = 0$	$\sum c(i) = 3$
Viereck 1×1 (4 Konten)	$\sum c(i) = 10.3431$	$\sum c(i) = 16.3431$
Viereck 2×2 (4 Konten)	$\sum c(i) = 2.68629$	$\sum c(i) = 8.68629$
Viereck 3×3 (4 Konten)	$\sum c(i) = 0$	$\sum c(i) = 2$
Viereck 2×2 (9 Konten)	$\sum c(i) = 29.1922$	$\sum c(i) = 59.4033$
Viereck 1×2 (6 Konten)	$\sum c(i) = 16.2273$	$\sum c(i) = 30.283$
Trapez mit der Seitenlänge 1	$\sum c(i) = 7.0718$	$\sum c(i) = 13.0718$
Sechseck mit der Seitenlänge 1	$\sum c(i) = 16.6077$	$\sum c(i) = 31.6077$
Sechseck mit der Seitenlänge 2	$\sum c(i) = 3$	$\sum c(i) = 10.8231$

Tabelle 8: Teilgraphen mit entsprechenden optimalen Werten

Die Zusammenfassung der besten Ergebnisse für verschiedene Gridgraphen und Teilgraphen findet man in den Tabellen 9, 10, 11.

Hexagonal Lattice	$d(x) = 3 - x$	$d(x) = 4 - x$
Der ganze Graph	6.41699 1.69 sec	16.6243 1838.73 sec
Teilgraphen		
Sechsecke mit der Seitenlänge 1	1.37 sec	1588.7 sec
Sechsecke mit der Seitenlänge 1, 2	1.75 sec	1641.71 sec
Trapez	1.77 sec	1529.24 sec

Tabelle 9: Untersuchung der Teilgraphen, Fall der Gridgraphen aus der Sechsecken.

Bei der Gridgraphen, die aus der Sechsecken bestehen, liefert der beschriebene Ansatz keine Verbesserung der Laufzeit.

Bei der Gridgraphen aus der Dreiecken kann man eine Verbesserung der Laufzeit beobachten. Aber immerhin braucht die Berechnung viel Zeit und mit der Einführung der zusätzlichen Nebenbedingungen auch mehr Speicherplatz.

Und im dritten Fall des Gitters aus den Vierecken hat der Ansatz wiederum keine deutliche Verbesserung der Laufzeit gebracht.

Triangular Lattice	$d(x) = 3 - x$	$d(x) = 4 - x$
Der ganze Graph	9.78029 10.96 sec	21.9068 103883 sec
Teilgraphen		
Dreiecke mit der Seitenlänge 1	12.59 sec	46933 sec
Dreiecke mit der Seitenlänge 1,2,3	10.02 sec	92228.1 sec
Sechsecke mit der Seitenlänge 1	11.90 sec	42412 sec

Tabelle 10: Untersuchung der Teilgraphen, Fall der Gridgraphen aus der Dreiecken.

Square Lattice	$d(x) = 3 - x$	$d(x) = 4 - x$
Der ganze Graph	8.63494 4.92 sec	19.9067 27793.8 sec
Teilgraphen		
Vierecke mit der Seitenlänge 1	12.76 sec	37945.53 sec
Vierecke mit der Seitenlänge 1,2,3 ohne Überlappen	7.11 sec	19619.38 sec
Vierecke 1×2	6.22 sec	28612.78 sec

Tabelle 11: Untersuchung der Teilgraphen, Fall der Gridgraphen aus der Vierecken.

Literatur

Alan A. Bertossi, Cristina M. Pinotti, and Richard B. Tan. Channel assignment with separation for interference avoidance in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):222–235, 2003.

V. Halasz. Symmary. Universität Heidelberg Arbeitsgruppe Discrete and Combinatorial Optimization, 2013.

John M. Harris, Jeffry L. Hirst, and Michael J. Mossinghoff. *Combinatorics and Graph Theory*. Springer, 2th edition, 2008.