Ergebnisse der Berechnungen

Modelle

Wir betrachten zur erst das klassische Modell zum Frequenzzuordnungsproblem (engl. Channel Assigment Problem, CAP). Dieses Problem ist dem $L(d_1, d_2, \ldots, d_r)$-Labeling-Problem der Graphentheorie äquivalent.
Gegeben seien ein zusammenhängender Graph $G = (V,\ E)$ und eine Folge $d_1 \geq d_2 \geq \cdots \geq d_r$ der ganzen Zahlen, wobei r kleine gleich maximales Durchmesser des Graphen G ist. Wir bezeichnen mit $dist(i, j)$ die Distanz zwischen den Knoten $i, j \in V$. Gesucht wird eine minimale Zahl λ, sodass Graph G eine $L(d_1, d_2, \ldots, d_r)$-Labeling c mit $\lambda = \max\{c(i), i \in V\}$ hat.
Das von uns betrachtete Mixed-Integer-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 reelwertigen Variablen $c_i, i \in \mathbb{R}$, übertragen. Dafür formulieren wir ein erweitertes Model:

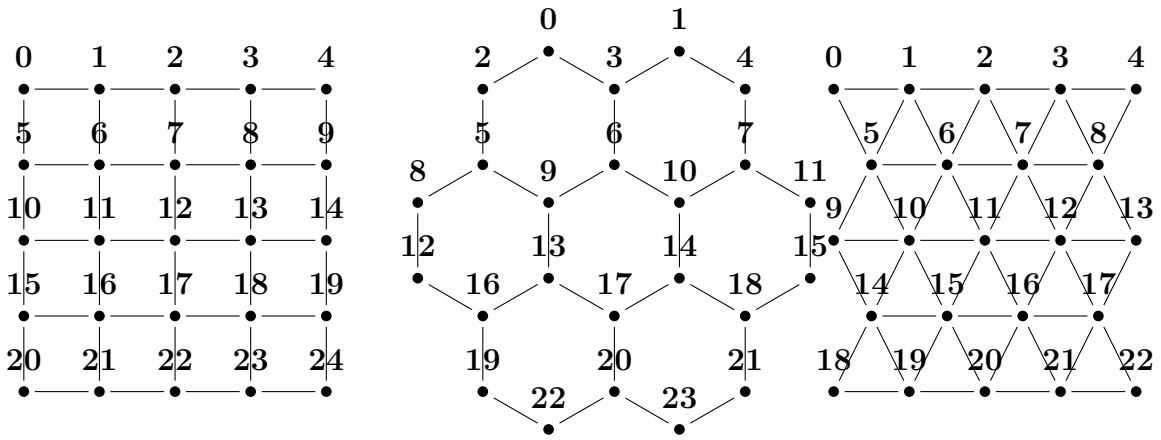
$$\min_{\lambda \in \mathbb{R}} \lambda \tag{8}$$

$$\begin{aligned}
\textbf{s.t. } & c_i \leq \lambda, && \forall i \in V && (9)\\
& c_i \geq 0, && \forall i \in V && (10)\\
& c_j - c_i + M z_{ij} \geq f(dist(i, j)), && \forall i, j \in V && (11)\\
& c_i - c_j + M(1 - z_{ij}) \geq f(dist(i, j)) && \forall i, j \in V && (12)\\
& z_{ij} \in \{0, 1\} && \forall i, j \in V && (13)\\
& c_i \in \mathbb{R}, && \forall i \in V && (14)
\end{aligned}$$

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.

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 betrachtet (insgesamt 24 Knoten).



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

Abbildung 1: In Betracht genommene Gridgraphen

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 reelwertiger 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 Table 3, 4, 5, haben wir die Treppenfunktionen für die entsprechende klassische Formulierung wie folgt definiert:

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

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

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

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

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

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

Die Ergebnisse der Berechnungen mit den Treppenfunktionen sind gleich den aus der klassischen Formulierung, brauchen aber mehr Zeit, um fertig zu werden (vgl. Table 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	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 reelwertiger 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 hexagonalem 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 Models zu verbessern. Wir beschreiben weiter verschiede 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. 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	3.42 sec		6651.87 sec	1838.73 sec
Triangular	9.78029	9.78029		21.9068	21.9068
	8.88 sec	57.01 sec		59064.01 sec	103898.60 sec
Square	8.63494	8.63494		19.9067	19.9067
	7.25 sec	33.05 sec		21222.8 sec	27793.8 sec

Tabelle 6: Der Vergleich der Laufzeit des Modells und des Modells mit den zusaetzlichen 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-and-Bound nicht.

- Beschränkung von λ

  Ein anderer Ansatz ist zu versuchen, gute obere Schranke für λ zu finden. Dafür haben wir den Algorithmus der Binäresuche 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 Anfangsschranke für λ wurde das Ergebnis der entsprechenden klassischen Formulierung genommen (λ_Z), und als obere $2\lambda_Z$ 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 um die Hälfte der Lückenlenge, oder obere nach unten 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 $26.77 + 21.59$ sec	6.41699 3.42 sec		16.6243 $257637 + 139576$ sec	16.6243 1838.73 sec
Triangular	9.78029 $89.25 + 55.26$ sec	9.78029 57.01 sec		21.9068	21.9068 103898.60 sec
Square	8.63494 $358.97 + 253.17$ sec	8.63494 33.05 sec		19.9067	19.9067 27793.8 sec

Tabelle 7: *Branch-and-Bound* Ansatz für die Bestimmung oberer und unterer Schranke 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 Optimierungszeit.

Man kann sehen, dass dieser Verfahren im Gegensatz zu den Erwartungen keine Zeitverbesserung geleistet hat. Es liegt laut (REFERENZ) daran, dass die Aufgabe die Zulässigkeit vom CAP zu prüfen, mindestens $NP-$schwer ist.

Außerdem, nach dem wir die zulässige Menge verkleinert haben, ist es schwer im Suchbaum einige Zweige abzuschneiden (Bounding) und die Suche nach dem Optimum dauert länger als davor.

Eine andere Möglichkeit λ abzuschätzen ist die Untersuchung von den Teilgraphen des ursprünglichen Graphen G. Z.B. die Lösung λ' des Problems für den Teilgraphen G', die das Minimum der Summe $\sum_{i \in V(G')} c(i)$ gewährleistet, liefert die untere Schranke für die Lösung λ des ursprünglichen Graphen G.

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.

Für die Teilgraphen löst man separat das Problem der Minimierung der Summe der Labels, die die Ungleichungen 9-14 erfüllen. Für den ganzen Gridgraphen fügt man für alle Knoten, die die bestimmten Teilgraphenstruktur bilden, zusätzliche Ungleichungen hinzu.

Die Zusammenfassung der besten Ergebnisse fuer verschiedene Grridgraphen und Teilgraphen findet man in den Tabellen 8, 9, 10.

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

Hexagonal Lattice	$d(x) = 3 - x$	$d(x) = 4 - x$
Der ganze Graph	6.41699 3.42 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 8: Untersuchung der Teilgraphen, Fall der Gridgraphen aus der Sechsecken.

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	11.90sec	42412sec

Tabelle 9: 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 10: Untersuchung der Teilgraphen, Fall der Gridgraphen aus der Vierecken.

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.

Literatur

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