Programmierpraktikum Diskrete Optimierung
Steiner-Baum-Probleme

Stephan Held
held@or.uni-bonn.de

Sommersemester 2022

1 Problemformulierung

Das Minimum Steiner-Baum-Problem (SMT) ist wie folgt definiert. Zu einer gegebenen Menge S von Terminalen wird ein längenminimaler Baum gesucht, der alle Terminale verbindet. Im Rahmen des Programmierpraktikums sind wir vor allem an Steinerbäumen in der Ebene ($S \subset \mathbb{R}^2, |S| < \infty$) mit Manhattan-Distanzen (ℓ_1-Norm), wie sie im VLSI-Design vorherrschend sind, oder Steinerbäumen in gewichteten Graphen (G, c) und ($S \subseteq V(G)$) interessiert.

Beide Probleme sind NP-schwer, wobei das Steiner-Baum-Problem in $(\mathbb{R}^2, \ell_p)$ ($p \geq 1$) beliebig genau approximiert werden kann, während dies für Steinerbäume in Graphen nicht geht, sofern P $\neq$ NP. Es gibt jedoch Approximationsalgorithmen mit konstanter Approximationsgüte. Z.B. übersteigt die Länge eines minimalen Spannbaumes einen minimalen Steiner-Baum höchstens um den Faktor 1.5 in der ℓ_1-Metrik, bzw. um den Faktor 2 in Graphen.

Algorithmen für Graphen können auch auf geometrische Instanzen angewandt werden. Für ℓ_1-Instanzen existiert immer ein minimaler Steiner-Baum im sogenannten Hanan-Gitter, welches dadurch entsteht, dass durch jedes Terminal eine horizontale und vertikale Linie gelegt wird und an den Kreuzungspunkten zweier Linien ein Knoten eingefügt wird.

Eine Erweiterung bilden sogenannte Cost-Distance-Steinerbäume, bei denen die Summe aus Gesamtlänge und gewichteten Weg-Längen im Baum minimiert werden soll. Diese spiegeln wider, dass über die Bäume Signale transportiert werden, die nicht zu langsam sein sollen.

2 Eingabe-Daten

Das Format für die Steiner-Baum-Instanzen ist das STP Data Format: `http://steinlib.zib.de/format.php`, welches zunächst für Graphen definiert ist, aber auch bei geometrischen Instanzen angewandt werden kann. Für geometrische Instanzen ist die Coordinates-Section maßgeblich. Die Graph-Section legt bei geometrischen Instanzen nur die Anzahl der Terminale fest.

Kleine übersichtliche Testinstanzen finden sich auf den Webseiten zum Praktikum `http://www.or.uni-bonn.de/lectures/ss22/praktikum_ss22.html` sowie auf den Seiten zur 11th DIMACS-Challenge `https://dimacs11.zib.de/`

Programmiersprache

Die Implementierungen müssen in C++ ausgeführt werden und unter Linux mit dem g++ oder clang kompilierbar sein, insbesondere müssen sie dem C++20-Standard genügen. Außer den Standard-Bibliotheken der STL dürfen keine Hilfsbibliotheken verwendet werden, es sei denn, es wird in der Aufgabenstellung ausdrücklich erwähnt oder es wird mit dem Betreuer abgesprochen.

Der Code muss verständlich programmiert und kommentiert werden. Selbstverständlich müssen die Programme fehlerfrei funktionieren. Insbesondere werden sie mit den Programmen valgrind (`http://valgrind.org`), ubsan/tsan sowie dem statischen Code-Checker von clang `https://clang-analyzer.llvm.org` überprüft. Von diesen Programmen gefundene Fehler führen zu Abzügen.

Einführungsaufgabe

Als Teil der Einführungsaufgabe sollte zunächst eine Einleseroutine für Steiner-Bauminstanzen implementiert werden. Soweit nicht explizit in der speziellen Aufgabenstellung angegeben, sollte anschließend als obere Schranke ein minimaler Spannbaum berechnet werden, wobei dessen halbe Länge als untere Schranke dient (2/3-Länge in der Manhattan-Metrik).

Spätester Abgabetermin für die Einführungsaufgabe ist der 24.04.2022, 24 Uhr.
Es wird jedoch empfohlen, auch die Implementierung der Gesamtaufgabe schon während der Semesterferien weitestgehend abzuschließen, da in der Vorlesungszeit häufig wenig Zeit bleibt.

3 Aufgaben

3.1 Geometrische Algorithmen für die ℓ_1-Metrik

Aufgabe 1: Der FLUTE-Algorithmus ermöglicht die schnelle Bestimmung optimaler Bäume auf kleinen Instanzen per Table-Lookup. Er soll implementiert werden und Tabellen für bis zu acht Terminalen berechnen. Größere Terminalmengen sollten zunächst in kleinere Mengen partitioniert und dann heuristisch zusammengefügt werden. Eine Variante sollte darin bestehen, iterativ einen minimalen Spannbaum dadurch zu verbessern, dass Teilbäume mit bis zu 8 Knoten optimal verbunden werden.

Chris Chu and Yiu-Chung Wong. FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. In IEEE Transactions on Computer-Aided Design, vol. 27, no. 1, pages 70-83, January 2008.

Aufgabe 2: Die schnellsten optimalen Algorithmen basieren auf der Beobachtung, dass minimale Steinerbäume immer in sogenannte Full Steiner Trees (FST) zerlegt werden können. Eine hinreichende Obermenge von FSTs, die einen minimalen Steiner-Baum überdeckt, kann in der Praxis effizient enummeriert werden. Anschließend wird zum optimalen Lösen eine optimale Teilmenge an FSTs bestimmt, die alle Terminale überspannt. Dieses entspricht dem NP-schweren *Minimum Spanning Tree Problem in Hypergraphen*, das mittels ganzzahliger Optimierung bestimmt werden könnte. In dieser Aufgabe soll das Spannbaumproblem aber mit Hilfe eines Greedy-Algorithmus approximativ gelöst werden.

Martin Zachariasen: Rectilinear full Steiner tree generation. Networks 33(2): 125-143 (1999)

Baudis, G., Gröpl, C., Hougardy, S., Nierhoff, T. and Prömel, H.-J. : Approximating Minimum Spanning Sets in Hypergraphs and Polymatroids. Technical Report (2000).

Aufgabe 2b: Anstelle des Greedy-Algorithmus soll ein ganzzahliges Programm zum optimalen Lösen der minimalen Spannbaum-Instanzen in Hypergraphen aufgestellt und gelöst werden. Hierzu soll das Problem als ganzzahliges lineares Program forumliert und mit Hilfe eines existierenden Solvers, z.B. CPLEX oder Gurobi, gelöst werden.

3.1.1 Schnelle 1.5-Approximationen

Bei den folgenden 2 Aufgaben geht es darum, eine 1.5-Approximation innerhalb einer Laufzeit von $O(|S| \log |S|)$ zu finden, was eine Verwendung des Hanan-Gitters der Größe $O(|S|^2)$ ausschließt. Stattdessen soll eine Delaunay-Triangulierung genutzt werden, die linear viele Kanten hat, falls die Terminale in allgemeiner Lage sind, d.h. kein Terminalpaar liegt auf einer Geraden mit der Steigung ± 1. Letzteres kann durch leichte Perturbation erreicht werden.

Aufgabe 3: Berechnen Sie einen approximativen Steiner-Baum, dessen Länge durch die Länge eines minimalen Spannbaums beschränkt ist, mit Hilfe einer Delaunay-Triangulierung und des MST-H Algorithmus von Liu et al.:

Chih-Hung Liu, Shih-Yi Yuan, Sy-Yen Kuo, and Szu-Chi Wang. 2009. High-performance obstacle-avoiding rectilinear steiner tree construction. ACM Trans. Des. Autom. Electron. Syst. 14, 3, Article 45 (June 2009).

Anschließend soll der Baum nachoptimiert werden, indem L-Strukturen umgeklappt werden, oder Liniensegmente verschoben werden, wenn dies die Länge verkürzt.

Aufgabe 4: Wie Aufgabe 3, aber unter Verwendung des theoretisch schnelleren Algorithmus von Mehlhorn zum Bestimmen eines Steiner-Baums in der Delaunay-Triangulierung.

Kurt Mehlhorn. 1988. A faster approximation algorithm for the Steiner problem in graphs. Inf. Process. Lett. 27, 3

3.2 Algorithmen in Graphen

Aufgabe 5: Der iterative 1-Steiner-Algorithmus setzt iterativ ein oder mehrere Steinerpunkte ein und berechnet einen minimalen Spannbaum auf der modifizierten Terminalmenge. Dadurch transformiert er einen minimalen Spannbaum sukzessive in einen meist viel kürzeren Steiner-Baum. Die effizienteste Variante hat kubische Laufzeit auf ℓ_1-Instanzen.

Bei der Implementierung ist es wichtig, Spannbäume auf den leicht wechselnden Knotenmengen schnell inkrementell zu berechnen. Durch heuristisches Ausdünnen der Steinerpunkkandidatenmenge soll die Laufzeit per Parameter beschränkt werden können, um größere Instanzen zu lösen.

Closing the gap: near-optimal Steiner trees in polynomial time. Jeff Griffith, Gabriel Robins, Jeffrey S. Salowe, and Tongtong Zhang. IEEE Trans. on CAD of Integrated Circuits and Systems 13(11):1351–1365 (1994)

Anschließend soll die Lösung durch lokale Suche verbessert werden: *Eduardo Uchoa, Renato Fonseca F. Werneck: Fast Local Search for Steiner Trees in Graphs. Proceedings of the Twelfth Workshop on Algorithm Engineering and Experiments, ALENEX 2010, Austin, Texas, USA, 2010*

Aufgabe 6: Wie in Aufgabe 5 soll hier lokale Suche verwendet werden, allerdings ausgehend von einem durch Mehlhorns Algorithmus bestimmten 2-optimalen Baum in Graphen.

Kurt Mehlhorn. 1988. A faster approximation algorithm for the Steiner problem in graphs. Inf. Process. Lett. 27, 3.

Eduardo Uchoa, Renato Fonseca F. Werneck: Fast Local Search for Steiner Trees in Graphs. Proceedings of the Twelfth Workshop on Algorithm Engineering and Experiments, ALENEX 2010, Austin, Texas, USA, 2010

Aufgabe 7: Die in der Praxis schnellsten Algorithmen Steinerbäume

Stefan Hougardy, Jannik Silvanus, Jens Vygen: Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm, Mathematical Programming Computation, 9(2), 135-202, 2017

Eine wesentliche Beschleunigung durch goal-oriented search with 1-tree lower bound.

Aufgabe 8: Implementieren Sie einen exakten Algorithmus basierend auf ganzzahliger Optimierung nach. Insbesondere soll das Augenmerk auf der initialen Reduktion sein, das IP kann z.B. mit einer gängigen Fluss-Formulierung, z.B. Bidirectional Cut, aufgestellt und gelöst werden.

Rehfeld, D. and Koch, T. (2021), Implications, conflicts, and reductions for Steiner trees, Mathematical Programming, 2021, https://doi.org/10.1007/s10107-021-01757-5.

3.3 (Uniforme) Cost-Distance-Steinerbäume

Beim (uniformen) Cost-Distance-Steinerbaum-Problem gibt es eine spezielle Wurzel $r \in S$ und Längengewichte λ_s für alle $s \in S \setminus \{r\}$. Gesucht ist ein Baum Y, der die Summe

$$c(E(Y)) + \sum_{s \in S \setminus \{r\}} \lambda_s c(E(Y_{[r,s]}))$$

minimiert, wobei $Y_{[r,s]}$ der eindeutige r-s-Weg in Y ist. Wir beschränken uns hier auf Instanzen in der Manhattan-Ebene $(\mathbb{R}^2, \ell_1)$.

Als untere Schranke dient hier $\frac{2}{3}MST + \sum_{s \in S \setminus \{r\}} \lambda_s ||r - s||_1$, die im Rahmen der Einführungsaufgabe bestimmt werden soll, wobei MST die Länge eines minimalen Spannbaums ist.

Im STP-Format werden die Wurzel r und Längengewichte $(\lambda_s)_{s \in S \setminus \{r\}}$ in der Terminals-Section angegeben. Instanzen finden Sie hier:

`https://www.or.uni-bonn.de/~held/praktikum/steiner22/cost-dist-instances.tar`.

Aufgabe 9: Held, S. and Khazraei, A.: "An Improved Approximation Algorithm for the Uniform Cost-Distance Steiner Tree Problem", WAOA, 2020. Startend von einer Steinerbaum-Approximation wird diese in Cluster beschränkten Längengewichts aufgeteilt, die mit Hilfe eines Kürzesten-Wege-Baums an die Wurzel angeschlossen werden. Der Kürzeste-Wege-Baum sollte nicht länger als $2 \log |S|$ mal die Länge eines minimalen Steinerbaums haben, dies kann durch iteratives Matching erreicht werden (beschrieben in Rotter, D. and Held, S.: "Shallow-Light Steiner Arborescences with Vertex Delays", IPCO 2013). In der Praxis bietet es sich an, den Algorithmus mit verschiedenen μ-Werten laufen zu lassen und das beste Ergebnis zu nehmen. Am Ende kann das Postprocessing aus G. Chen, P. Tu, and E. F. Y. Young: "SALT: Provably Good Routing Topology by a Novel Steiner Shallow-Light Tree Algorithm", ICCAD 2017, TCAD 2019, benutzt werden, um insb. kreuzende Kanten zu vermeiden.

Aufgabe 10: Rotter, D. and Held, S.: "Shallow-Light Steiner Arborescences with Vertex Delays", IPCO 2013.

Dieses Paper findet einen Tradeoff zwischen der Gesamtlänge und der Einhaltung von Delay-Schranken. Das im Paper vorkommende Vertex-Delay nehmen wir hier als $b = 0$ an. Durch geeignete Wahl des Tradeoff-Parameters ϵ wird auch ein konstanter Appriximationsfaktor für das uniforme Cost-Distance-Steinerbaum erreicht (beschrieben wie in Held, S. and Khazraei, A.: "An Improved Approximation Algorithm for the Uniform Cost-Distance Steiner Tree Problem", WAOA, 2020). In der Praxis bietet es sich an, den Algorithmus mit verschiedenen ϵ-Werten laufen zu lassen und das beste Ergebnis zu nehmen. Am Ende kann das Postprocessing aus G. Chen, P. Tu, and E. F. Y. Young: "SALT: Provably Good Routing Topology by a Novel Steiner Shallow-Light Tree Algorithm", ICCAD 2017, TCAD 2019, benutzt werden, um insb. kreuzende Kanten zu vermeiden.

Aufgabe 11: Meyerson, Munagala, Plotkin: Cost-Distance: Two-Metric Network Design, SIAM Journal Compouting, 2008, 1648–1659.

Dieser Ansatz hätte auch für das nicht-uniforme Cost-Distance-Steinerbaumproblem eine Approximationsgüte von $O(\log |S|)$. Die Lösung wird durch iteratives Matching bestimmt. Um dieses zu berechnen kann/soll BlossomV (`https://pub.ist.ac.at/~vnk/software/blossom5-v2.05.src.tar.gz`) benutzt werden. Am Ende kann das Postprocessing aus G. Chen, P. Tu, and E. F. Y. Young: "SALT: Provably Good Routing Topology by a Novel Steiner Shallow-Light Tree Algorithm", ICCAD 2017, TCAD 2019, benutzt werden, um insb. kreuzende Kanten zu vermeiden.

Sämtliche Paper finden Sie auch unter `https://uni-bonn.sciebo.de/s/Ynzt0Mwr0nvihfo`.

Spätester Abgabetermin für das Programmierpraktikum ist der 10.07.2022, 24 Uhr.

Am Ende des Semesters muus jeder Teilnehmer seine Implementierung im Rahmen eines Block-Seminars vorstellen. Hierbei sollen der Algorithmus, die interessantesten Code-Fragmente sowie experimentelle Ergebnisse vorgestellt werden.

Voraussichtlicher Termin hierfür ist der Nachmittag des 15.07.2022.