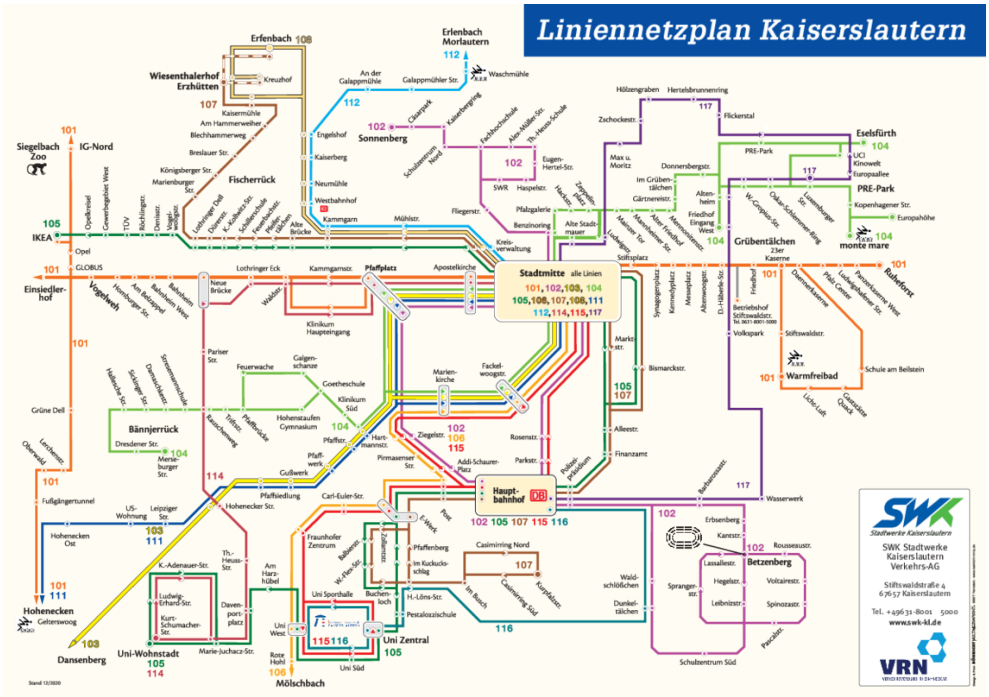
Algorithmische Bewertung von ÖPNV-Netzen



Name: Raphael Gaedtke

Fachgebiet: Mathematik/Informatik

Projektbetreuer: S. Kirch-Rink

Schule: Hohenstaufen-Gymnasium Kaiserslautern

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

Beinahe jede Stadt hat ein System von Buslinien und Bushaltestellen, mit dem Fahrgäste kostengünstig und ohne eigenes Fahrzeug auch größere Strecken innerhalb der Stadtgrenzen überwinden können - den Öffentlichen Personennahverkehr (ÖPNV).
Beim Betrachten eines Linienplans fallen zwischen verschiedenen Städten aber große Unterschiede auf: Bei einigen schneiden sich alle Buslinien im Stadtzentrum, bei anderen gibt es Linien, die vom Zentrum strahlenförmig ausgehenden Linien tangential schneiden und wieder andere Städte haben eine Hauptverkehrsachse, von der alle anderen Linien an verschiedenen Stellen abzweigen.
Diese Unterschiede sind durch die Größe, die Lage oder das allgemeine Straßennetz einer Stadt bestimmt, es stellt sich aber auch die Frage, ob es allgemeingültige Vor- und Nachteile unterschiedlicher Netztypen aus Sicht der Fahrgäste oder der Netzbetreiber gibt.
Um diese Frage zu beantworten, muss es zunächst eine Möglichkeit geben, ein ÖPNV-Netz in einem bestimmten Straßennetz objektiv zu bewerten.
Deswegen habe ich in dieser Arbeit Kriterien entwickelt, die für solche Linienpläne rechnersich ermittelt werden können und die angeben, wie schnell sich die Fahrgäste in der Stadt bewegen können oder wie teuer diese Infrastruktur für ein Betreiberunternehmen ist. Außerdem habe ich in der Folge Algorithmen, die diese Kriterien für eine gegebene Stadt mit möglichst geringer Laufzeit ermitteln, gefunden und implementiert.
Mithilfe dieser Implementierung können dann Versuchsreihen mit unterschiedlichen Netztypen in randomisiert generierten Städten durchgeführt werden, um so zu beobachten, welchen Einfluss die Wahl eines bestimmten Netztyps unter den untersuchten Kriterien hat. Auf diese Weise sollen die Vor- und Nachteile dieser Typen allgemein betrachtet werden können.

1.2 Theoretische Grundlagen

1.2.1 Graphen

Definition 1. Ein Graph $G = (V, E)$ ist ein Tupel aus einer Menge V von Knoten und einer Multimenge E von Kanten. Dabei ist $E \subseteq (V \times V)$, jede Kante $e \in E$ des Graphen ist also ein Tupel zweier nicht notwendigerweise verschiedener Knoten a und b. Die Kante e verbindet dann die Knoten a und b, wobei die Kanten geordnete oder ungeordnete Tupel sein können.
Sind $a, b \in V$ zwei Knoten des Graphen mit $(a, b) \in E$, die also durch eine Kante des Graphen verbunden sind, so heißen a und b adjazent zueinander und jeder der beiden Knoten inzident zur Kante $e = (a, b)$.[1]

Soll ein Graph grafisch dargestellt werden, können die Knoten dafür als Kreise oder Punkte in der euklidischen Ebene repräsentiert werden. Um eine Kante darzustellen, werden die zu dieser Kante inzidenten Knoten durch Linien miteinander verbunden.
Bei dieser Art der Darstellung ist die genaue Lage, Größe oder Form der Knoten und Kanten im Raum unerheblich, wichtig ist nur die Anzahl der Knoten und die Art und Anzahl der Verbindungen zwischen ihnen.

Definition 2. Es sei $G = (V, E)$ ein Graph. Ein Pfad oder Weg mit k Knoten zwischen den beiden Knoten v_1 und v_k ist dann ein Graph $P = (V_p, E_p)$, wobei $V_p = \{v_1, v_2, v_3, ..., v_k\}$ und $E_p = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), ..., (v_{k-1}, v_k)\}$ sowie $E_p \subseteq E$ und $V_p \subseteq V$ gilt.

[1] Definitionen 1, 2, 4-7: https://hpi.de/friedrich/teaching/units/graphentheorie.html

Ein solcher Pfad kann auch eindeutig als eine Abfolge $(v_1, v_2, v_3, ..., v_k)$ von k paarweise verschiedenen Knoten des Graphen G angegeben werden. Kommen Knoten mehrmals vor, wird stattdessen von einem Kantenzug gesprochen.

Graphen können nach unterschiedlichen Eigenschaften klassifiziert werden:

Definition 3. Hat ein Graph $G = (V, E)$ eine Kante, die einen Knoten des Graphen mit sich selbst verbindet, so wird diese Kante als Schlinge bezeichnet. Mehrfachkanten liegen vor, wenn zwei oder mehr Kanten inzident zu denselben beiden Knoten sind. Ein einfacher Graph ist ein Graph ohne Schlingen oder Mehrfachkanten.[2]

Definition 4. Ein Graph $G = (V, E)$ ist zusammenhängend, wenn zwischen jedem Paar von Knoten in V ein Pfad existiert.

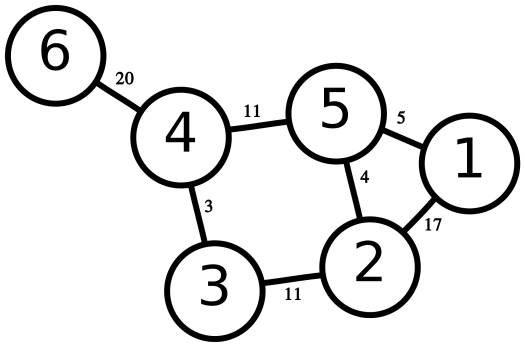
Definition 5. Ein Graph $G = (V, E)$ heißt gerichtet, wenn die Knotentupel in E, die die Kanten des Graphen repräsentieren, geordnet sind. Die Kanten haben dann eine Richtung. Ist $(a, b) \in E$, so geht eine Kante von Knoten a aus zu Knoten b.
Um einen gerichteten Graphen grafisch zu darzustellen, werden die Kanten durch Linien mit Pfeilspitzen repräsentiert.

Definition 6. Den Kanten eines Graphen $G = (V, E)$ können Gewichte zugeordnet werden. Dies geschieht durch die Definition einer Kostenfunktion $c : E \to \mathbb{R}$, die jeder Kante des Graphen eine reelle Zahl zuordnet. Existiert für einen Graphen eine solche Kostenfunktion, so ist dieser Graph gewichtet.

In einem Graphen mit gewichteten Kanten können dann auch Pfaden und Kantenzügen Gewichten zugewiesen werden:

Definition 7. Es sei $G = (E, V)$ ein zusammenhängender und gewichteter Graph und $a, b \in V$. Dann sei $s_G : V \times V \to \mathbb{R}$ die Funktion, die jedem Paar (a, b) von Knoten die minimale Summe der Kantenkosten eines Pfades in G von a nach b zuordnet.
Ein Pfad zwischen a und b mit den minimalen Kosten $s_G(a, b)$ heiße auch „kürzester Weg“ von a nach b. Im ungerichteten Graphen gilt dabei insbesondere $\forall a, b \in V : s_G(a, b) = s_G(b, a)$.

Beispiel 1. In der Abbildung wird ein gewichteter, ungerichteter, einfacher und zusammenhängender Graph mit 6 Knoten und 7 Kanten gezeigt.
Der kürzeste Weg vom Knoten 1 zum Knoten 6 verläuft über die Zwischenknoten 5 und 4 und hat die Kosten $5 + 11 + 20 = 36$.



[2] Definition 3: https://www.uni-siegen.de/fb6/analysis/overhagen/vorlesungsbeschreibungen/skripte/graphentheorie1.pdf, S. 8

1.2.2 Verwendete Graphenalgorithmen

1.2.2.1 Der Dijkstra-Algorithmus

Der Dijkstra-Algorithmus nimmt als Eingabe einen gewichteten, zusammenhängenden, aber nicht notwendigerweise ungerichteten Graphen $G = (V, E)$ mit nichtnegativen Kantengewichten und einen Startknoten $s \in V$ entgegen und liefert für jeden Knoten $a \in V$ den Wert der Funkion $s_G(s, a)$ und damit die Länge des kürzesten Weges von s nach a zurück.
Dafür wird für jeden Knoten u die aktuell kürzeste bekannte Weglänge vom Knoten s in der Variablen $dist[u]$ mit dem Startwert ∞ gespeichert.
Außerdem wird eine Prioritätswarteschlange verwendet, die Tupel der Form (d, u) mit $d \in \mathbb{R}, u \in V$ speichert und aufsteigend nach dem Wert von d sortiert ist. Ein solches Tupel (d, u) in der Warteschlange besagt, dass ein Weg zu u mit der Länge d bekannt, aber noch nicht auf $dist[u]$ übertragen ist.
Zu Beginn enthält diese Warteschlange lediglich das Tupel $(0, s)$, da der Startknoten s immer mit einer Distanz von 0 erreicht werden kann. In jedem folgenden Schritt wird dann der erste Tupel (d, u) aus der Prioritätswarteschlange entfernt, wobei der erste Tupel immer der mit dem kleinsten Wert von d ist. Ist für diesen Tupel dann $d < dist[u]$, wird $dist[u]$ zu d aktualisiert und für alle zu u adjazenten Knoten ein eventuell verbesserter Tupel in die Prioritätswarteschlange geschoben.
Der Algorithmus terminiert, wenn die Prioritätswarteschlange leer ist - dann stehen für $\forall u \in V$ auch die korrekten Werte von $s_G(s, u)$ in der Datenstruktur $dist$. Es ergibt sich eine Laufzeit in $O((\mid V \mid + \mid E \mid) \log \mid V \mid)$, weil jeder Knoten und jede Kante einmal betrachtet und in logarithmischer Zeit in die Prioritätswarteschlange geschoben wird.
Der Dijkstra-Algorithmus aktualisiert also immer die Distanz des Knotens, der die aktuell kürzeste, nicht berücksichtigte Distanz zu s hat. Diese Vorgehensweise führt zu korrekten Ergebnissen, weil alle Kantengewichte nach Voraussetzung positiv sind und ein aktualisierter Knoten von später erreichten (d.h. weiter entfernten) Knoten deshalb nicht mit kürzerer Distanz erreicht werden kann.[3]

1.2.2.2 Der Floyd-Warshall-Algorithmus

Der Floyd-Warshall Algorithmus nimmt als Eingabe einen gewichteten, zusammenhängenden Graphen $G = (V, E)$ entgegen und liefert für jedes Knotenpaar $(a, b) \in V \times V$ den Wert von $s_G(a, b)$ und damit die Länge eines jeden kürzesten Pfades zwischen zwei Knoten von G zurück.
Dafür operiert der Algorithmus auf der Ajazenzmatrix des Graphen. Diese enthält zu Beginn als Eintrag mit den Indizes (i, j) das Gewicht der Kante zwischen den Knoten mit den Indizes i und j, ∞, wenn keine Kante zwischen diesen beiden Knoten existiert und 0 bei allen Einträgen (i, i). Diese Matrix wird modifiziert und enthält am Ende als Eintrag mit den Indizes (i, k) den Wert von $s_G(i, k)$.
Im Algorithmus werden alle Knoten durchgegangen. Für jeden Knoten j wird über alle Knotenpaare (i, k) im Graphen iteriert, wobei $dist[i][k]$ auf $(dist[i][j] + dist[j][k])$ gesetzt wird, wenn dadurch eine Verbesserung des Werts von $dist[i][k]$ erzielt wird.
Vor dem ersten Schritt sind in der Matrix also nur Pfade mit genau zwei Knoten berücksichtigt, nach dem ersten alle Pfade mit dem ersten Knoten als „Zwischenknoten“, nach dem zweiten alle Pfade mit den ersten beiden als „Zwischenknoten“ usw.
Allgemein enthält die Matrix nach j Schritten alle kürzesten Pfade, wenn die ersten j Knoten als „Zwischenknoten“ verwendet werden dürfen. Nach $\mid V \mid$ Schritten sind alle kürzesten Wege enthalten. Mit „Zwischenknoten“ eines Pfades sind dabei Knoten gemeint, die Teil des Pfades, aber weder sein Anfang noch sein Ende sind.
Da in jedem der $\mid V \mid$ Schritte alle $\mid V \mid^2$ Paare von Knoten durchgegangen werden müssen, hat dieser Algorithmus eine Laufzeit in $O(\mid V \mid^3)$.[4]

[3] Halim, Halim, Effendy 2020, S. 227-231
[4] Halim, Halim, Effendy 2020, S. 242-244

1.2.2.3 Der Algorithmus von Kruskal

Definition 8. Ein Baum ist ein zusammenhängender Graph mit $v \in \mathbb{N}_{>0}$ Kanten und $(v-1)$ Kanten.[5]

Definition 9. Ein Spannbaum ist ein Teilgraph eines zusammenhängenden Graphen $G = (V, E)$, der alle Knoten in V enthält und ein Baum und somit zusammenhängend ist. Ist G gewichtet, so ist der minimale Spannbaum der Spannbaum mit der geringsten Summe von Kantengewichten.[6]

Kruskal's Algorithmus nimmt die Kantenliste eines zusammenhängenden, gewichteten Graphen $G = (V, E)$ entgegen und liefert die Kantenliste seines minimalen Spannbaums zurück.
Dazu wird die Kantenliste zu Beginn aufsteigend nach Kantengewicht sortiert, sodass die Kanten in der Reihenfolge ihrer Kantengewichte durchgegangen werden können.
Für jede Kante wird dann überprüft, ob die sie bildenden Knoten schon durch einen Pfad verbunden sind. Genau dann, wenn dies nicht der Fall ist, wird die Kante in den minimalen Spannbaum eingefügt. Dieser Algorithmus konstruiert einen zusammenhängenden Teilgraphen mit der Kantenmenge von V mit $(\mid V \mid -1)$ Kanten und somit einen Spannbaum.
Würde eine der gewählten Kanten nicht wirklich zum minimalen Spannbaum gehören, so müsste sie im Algorithmus von Kruskal übersprungen werden, obwohl sie zwei zu diesem Zeitpunkt noch nicht verbundenen Kanten verbindet. An ihrer statt kann aber nur eine Kante mit einem höheren Gewicht eingesetzt werden, was der Minimalität des Ergebnisses widerspricht. Also produziert der Algorithmus von Kruskal korrekte Ergebnisse.
Das Sortieren der Kanten erfolgt in $O(\mid E \mid \log \mid E \mid)$, das Durchgehen der Kanten in $O(\mid E \mid))$ und das Überprüfen, ob zwei der Knoten schon verbunden sind, in faktisch konstanter Zeit. Diese kann durch das Verwenden der sogenannten „Union Find"-Datenstruktur erzielt werden.[7]
Diese unterstützt nach der Initialisierung (nach der jeder Knoten in einer eigenen Menge ist) zwei Operationen, die jeweils in faktisch konstanter Zeit ablaufen: Sie gibt für einen Knoten einen Repräsentanten seiner Menge zurück (der für jeden Knoten der Menge gleich ist) oder vereint zwei Mengen. Durch das Vergleichen der Repräsentanten zweier Mengen kann entschieden werden, ob die Mengen gleich sind.
Im Algorithmus von Kruskal werden jeweils alle zusammenhängenden Knotenmengen als eine Menge gespeichert und zwei dieser Mengen werden nach dem Hinzufügen einer sie verbindenden Kante vereint.
Die Laufzeit des Algorithmus insgesamt ist also $O(\mid E \mid \log \mid E \mid)$.[8]

Beispiel 2. Die folgende Abbildung zeigt einen zusammenhängenden, gewichteten Graphen mit seinem minimalen Spannbaum:

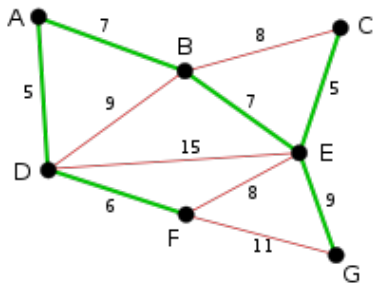


Abbildung 1.1: Die zum Spannbaum gehörenden Kanten sind grün, alle anderen sind rot eingefärbt.

[5] Halim, Halim, Effendy 2020, S. 255
[6] Skiena 2020, S. 244
[7] Skiena 2020, S. 458
[8] Skiena 2020, S. 248-253

2 Modellierung von ÖPNV-Netzen

Um Eigenschaften von ÖPNV-Netzen objektiv messbar zu machen, müssen diese durch eine mathematische Struktur dargestellt werden. Hierzu wird die in Kapitel 1.2.1 beschriebene Struktur eines Graphen verwendet.
Im Folgenden wird das ÖPNV-Netz der Einfachheit halber ausschließlich als Netz von Buslinien mit Haltestellen bezeichnet, es sollen aber auch andere Verkehrsmittel wie z.B. Straßenbahnen eingeschlossen sein.

2.1 Modellierung als Graph

2.1.1 Darstellung eines ÖPNV-Netzes

Ein ÖPNV-Netz kann beschrieben werden als eine Menge von Haltestellen und eine Menge von Verbindungen zwischen je zwei Haltenstellen. Diese Struktur ist repräsentierbar durch einen Graphen $O = (S, C)$, bei dem die Knoten die Haltestellen und die Kanten die Verbindung zwischen diesen Haltestellen darstellen. Die Menge S sei hier die Menge der Knoten, also der Haltestellen und die Menge C die Menge der Kanten, also der Verbindungen zwischen diesen Haltestellen.
In einem ÖPNV-Netz ist es unsinnnig, Busse einzusetzen, die zweimal hintereinander an derselben Haltestelle halten. Deswegen darf angenommen werden, dass der Graph O keine Schlingen enthält.
Auf einem Netzplan können mehrere Linien vorkommen, die eine Verbindung parallel nutzen. Für einen Fahrgast ist es aber unerheblich, welches Fahrzeug zu welcher Linie gehört, weswegen in O auch keine Mehrfachkanten vorkommen sollen. Insgesamt ist O also ein einfacher Graph, wenn Umsteigezeiten zwischen den Linien in der Modellierung der Kantengewichte berücksichtigt werden.
Wäre es für ein bestimmtes Paar von Haltestellen nicht möglich, mit Bussen im ÖPNV-Netz von einer Haltestelle zu der anderen zu kommen, so wäre der Graph O nicht zusammenhängend. Dann könnten die Zusammenhangskomponenten in O aber auch einzeln betrachtet werden. Daher wird auch angenommen, dass O ein zusammenhängender Graph ist und es somit möglich ist, durch die alleinige Verwendung von öffentlichen Verkehrsmitteln von jeder Haltestelle zu jeder anderen zu gelangen.
Schlussendlich ist es für Netzbetreiber auch üblich, Linien in beide Richtungen zu bewirtschaften. O soll daher ein ungerichteter Graph sein. Es wird also angenommen, dass es für die Fahrtzeit keinen Unterschied macht, in welche Richtung ein Fahrgast eine Linie benutzt.

Um ein ÖPNV-Netz objektiv beurteilen zu können, muss auch das zugrunde liegende Straßennetz der jeweiligen Stadt berücksichtigt werden, da die bestmögliche Fahrtzeit einzelner Verbindungen von diesem abhängig ist. Ein Staßennetz lässt sich ebenfalls durch einen Graphen $G = (I, R)$ darstellen, wobei I die Menge der Kreuzungen und R die Menge der Verbindungen zwischen Kreuzungen repräsentieren soll.
Analog zum ÖPNV-Netz kann davon ausgegangen werden, dass der Graph G zusammenhängend ist. Die Verbindungen des zusammenhängenden ÖPNV-Netz müssen nämlich allesamt in G repräsentierte Straßen benutzen, während Kreuzungen, die von keiner Haltestelle aus erreicht werden können, irrelevant sind. Dazu wird auch angenommen, dass alle Verkehrsmittel des ÖPNV - wie auch zum Beispiel Straßenbahnen - nur Verbindungen des Straßennetzes nutzen.
Dabei können Straßen, die einen Rundweg von einer Kreuzung zu sich selbst darstellen und Straßen, die den selben Verlauf haben wie eine kürzere Straße, unberücksichtigt bleiben, wenn vorausgesetzt wird, dass sich jede Haltestelle in der Nähe einer Kreuzung befindet und es für die Fahrzeit unerheblich ist, mit welcher Straße diese Kreuzung erreicht wird. Es ist dann $S \subseteq I$. G sei also ohne Schlingen oder Mehrfachkanten und somit ebenfalls ein einfacher Graph.

In Straßennetzen sind allerdings gerichtete Straßen in Form von Einbahnstraßen durchaus üblich. Trotzdem soll auch G ungerichtet sein. Andernfalls könnte auch O nicht mehr ungerichtet sein, da die Fahrtzeit einer Verbindung im ÖPNV-Netz sonst auch von der Richtung dieser Strecken abhinge.

2.1.2 Kantengewichte

Für die Graphen O und G müssen außerdem Kostenfunktionen $c_O : C \to \mathbb{R}$ und $c_G : R \to \mathbb{R}$, die jeder Kante eine reelle Zahl zuordnen, definiert werden. Diese Zahl beschreibt, wie viel Zeit benötigt wird, um die entsprechende Verbindung oder den entsprechenden Straßenabschnitt zu bewältigen.
Für das im Graphen G dargestellte Straßennetz soll jede Kante dabei einer reellen Zahl zugeordnet werden, die angibt, wie viele Minuten es durchschnittlich dauert, die dargestellte Straße mit einem Fahrzeug mit der Geschwindigkeit eines Busses zu bewältigen.
Diese Durchschnittszeit bezieht - im Unterschied zur reinen Angabe der Streckenlänge - auch das Verkehrsaufkommen auf den jeweiligen Straßen und die Höhe der Geschwindigkeitsbegrenzung mit ein. Deswegen kann auch nicht davon ausgegangen werden, dass im Graphen G die Dreiecksungleichung gilt. Diese entspricht der Aussage

$$\forall a, b, c \in G : c_G(a, c) \leq c_G(a, b) + c_G(b, c) \tag{2.1}$$

die in graphentheoretischen Problemstellungen mit Weglängen häufig als Vereinfachung genutzt werden kann. In realen Straßennetzen ist es nämlich möglich, dass die Fahrt über die direkte Verbindung zweier Kreuzungen aufgrund starken Verkehrsaufkommens länger dauert als der „Umweg“ über eine dritte.
Die Kantenkosten im Graphen O können nicht auf diese Weise bestimmt werden, da die Zeit, in der eine Strecke zurückgelegt werden kann, vom Startzeitpunkt und den Fahrplänen abhängt. Möchte ein Fahrgast eine Route mit mehreren Haltestellen befahren, beruht die Fahrtdauer auch auf eventuellen Umsteigezeiten.
Diese Informationen sollen in einer einzigen reellen Zahl zusammengefasst werden, die dann die Kosten einer Kante im Graphen O angibt. Zu diesem Zweck wird die Hilfsfunktion ω definiert:

Definition 10. Es sei $e \in C$ eine Kante im Graphen O, die die Verbindung zweier Haltestellen in einem ÖPNV-Netz repräsentiert. Dann sei $\omega : C \to \mathbb{R}$ die Funktion, die jeder Kante e eine reelle Zahl zuordnet, die angibt, wie viele Minuten durchschnittlich zwischen zwei aufeinanderfolgenden Abfahrtszeiten eines Busses auf der dargestellten Verindung liegen (Hier und im Folgenden wird dafür eine Richtung der Kante willkürlich ausgewählt und angenommen, dass die Wahl dieser Richtung unerheblich ist).

Möchte ein Fahrgast eine Verbindung (a, b) im ÖPNV-Netz O nutzen, so benötigt er für die Fahrt allein $s_G(a, b)$ Minuten. Kommt er allerdings zu einem zufälligen Zeitpunkt an der Ausgangshaltestelle an, muss er noch durchschnittlich $\frac{\omega(a,b)}{2}$ Minuten auf seinen Bus warten (Jede Wartezeit zwischen 0 und $\omega(a, b)$ hat die gleiche Wahrscheinlichkeit, womit der Erwartungswert für diese Zeit $\frac{\omega(a,b)}{2}$ ist). Insgesamt ergibt sich für das Benutzen einer einzelnen Verbindung eine Fahrtzeit von $s(a, b) + \frac{\omega(a,b)}{2}$ als die Summe der echten Fahrtzeit und der Wartezeit vor der Fahrt.
Meistens benutzen Fahrgäste aber Routen, die über mehrere Haltestellen fahren, was zu übermäßig hohen Kantenkosten durch die Addition von $\frac{\omega(a,b)}{2}$ für jeden Abschnitt führt. Um die Kantenkosten trotzdem unabhängig vom Kantenzug zu halten, werden diese Werte noch mit einem konstanten Faktor $u < 1$ multipliziert und somit verkleinert. Für die Kostenfunktion c_O ergibt sich folgende Definition:

Definition 11. In einem gegebenen, durch einen Graphen $O = (S, C)$ repräsentierten ÖPNV-Netz sei u eine Konstante mit $u < 1$, die niedrig ist, wenn die Routen der Fahrgäste tendenziell mehr Haltestellen durchlaufen (Dieser Faktor hängt allerdings auch vom Verhalten der Fahrgäste ab, und lässt sich deswegen nicht aus O ermitteln. Stattdessen wird u vom Anwender vorgegeben). Dann ist

$$c_O : C \to \mathbb{R}, (a, b) \mapsto s_G(a, b) + \frac{u \cdot \omega(a, b)}{2}. \tag{2.2}$$

2.2 Mögliche Kennzahlen von ÖPNV-Netzen

Die mathematische Modellierung eines ÖPNV-Netzes eröffnet die Möglichkeit, Eigenschaften dieses Netzes in Zahlen zu fassen. Prinzipiell gibt es unendlich viele Optionen für die Defintion solcher zahlengestützer Kriterien und jeder Anwender kann diese beliebig variieren.
Im Folgenden werden drei verschiedene Kriterien vorgeschlagen, die die Interessen verschiedener Parteien (also die der Fahrgäste und die der Nahverkehrsunternehmen) möglichst breit abbilden sollen. Diese Kriterien können genutzt werden, um aus empirischen Untersuchungen einzelner ÖPNV-Netze unterschiedlicher Typen allgemeine Aussagen über die Qualitäten verschiedener Netztypen abzuleiten. Dabei sollen die Lage und Anzahl der Haltestellen als vorgegeben und nur die Verbindungen zwischen den Haltestellen als variabel betrachtet werden.

1. Für Fahrgäste ist es von Interesse, Strecken im ÖPNV-Netz möglichst schnell zurückzulegen. Wie schnell dies für eine jeweilige Strecke möglich ist, hängt aber auch vom zugrundeliegenden Straßennetz ab, weswegen es sinnvoll ist, das Verhältnis zwischen dem Zeitaufwand einer Strecke im ÖPNV- bzw. Straßennetz zu betrachten.
   Dieses Verhältnis kann je nach gewählter Strecke variieren, weswegen für die Bewertung des gesamten Netzes der Durchschnitt über alle Strecken zwischen je zwei Haltestellen gebildet werden soll. Aber auch extreme Werte dieses Verhältnisses sollen berücksichtigt werden, weswegen hier bewusst der das arithmetische Mittel und nicht der Median gewählt wird.

   Definition 12. Es sei $O = (S, C)$ ein ein ÖPNV-Netz repräsentierender Graph und $G = (I, R)$ der Graph, der das dazugehörige Straßennetz modelliert. Das Verhältnis zwischen der kürzesten Strecke im ÖPNV-Netz und der kürzesten Strecke im Straßennetz im durchschnittlichen Fall werde mit SNV_O bezeichnet. Es gilt

   $$SNV_O = \frac{\sum_{(a,b)\in S\times S} \frac{s_O(a,b)}{s_G(a,b)}}{\mid S \times S \mid}. \tag{2.3}$$

   SNV_O berechne sich also als die Summe des Verhältnisses der Kosten einer Strecke im ÖPNV-Netz zu den Kosten derselben Strecke im Straßennetz geteilt durch die Anzahl dieser Strecken.Ein ÖPNV-Netz kann als „besser“ gelten, wenn dieses Verhältnis geringer ist.

2. Für Unternehmen, die ein ÖPNV-Netz betreiben, sind vor allem dessen Kosten interessant, die wiederum vor allem durch die Bereitstellung der einzelnen Busse anfallen. Aus einem gegebenen Linienplan folgt allerdings nicht direkt die Anzahl der aufzuwendenden Busse, da beispielsweise auch Leerfahrten anfallen können. Der Bedarf an Bussen sollte trotzdem in etwa proportional zur Anzahl an Fahrten sein, die sich aus dem Fahrplan direkt ergeben. Damit lässt sich dieser Bedarf direkt aus O abschätzen.
   Dazu wird die Kostenfunktion ω betrachtet, die für jede Verbindung zwischen zwei Haltestellen angibt, wie groß die Abstände zwischen der Abfahrt zweier Bussen (in einer Richtung) ist. Der Kehrwert dieser Größe ergibt die Anzahl von Bussen pro Zeiteinheit für die jeweilige Strecke.
   Werden die Kehrwerte aller Kantenkosten in O aufsummiert, ergibt sich deswegen ein Wert, der niedriger ist, wenn laut Fahrplan weniger Busse fahren.

   Definition 13. Für einen gegebenen Graphen $O = (S, C)$ soll ein Wert für die Anzahl der aufzuwendenden Busse im ÖPNV-Netz mit BTA_O bezeichnet werden. Diese Zahl wird durch

   $$BTA_O = \sum_{(a,b)\in C} \frac{1}{\omega(a,b)} \tag{2.4}$$

   berechnet. Auch hier stehen kleinere Werte für ein „besseres“ ÖPNV-Netz.

3. Das Kriterium SNV_O gibt das Verhältnis von Fahrtzeit im ÖPNV-Netz und im Straßennetz im Durchschnitt über alle Strecken an. Für einen einzelnen Fahrgast ist dieses Verhältnis aber nur in einigen wenigen Fällen interessant und für einige insbesondere auch im schlechtesten. Es ist deswegen im Interesse der Fahrgäste, das Verhältnis aus SNV_O im schlechtesten Fall zu minimieren, weswegen es hier ebenfalls als Kriterium vorgeschlagen wird.

   Definition 14. Es sei $O = (S, C)$ ein ein ÖPNV-Netz repräsentierender Graph und $G = (I, R)$ der Graph, der das dazugehörige Straßennetz modelliert. Das Verhältnis zwischen der kürzesten Strecke im ÖPNV-Netz und der kürzesten Strecke im Straßennetz im schlechtesten Fall werde mit VSM_O bezeichnet.

$$VSM_O = \max_{(a,b)\in S\times S} \frac{s_O(a,b)}{s_G(a,b)}. \tag{2.5}$$

2.3 Algorithmische Bestimmung der Kennzahlen

Um die in Kapitel 2.2 beschriebenen Kriterien für ein ÖPNV-Netz mithilfe eines Computers berechnen zu können, muss für die Berechnung einer jeden dieser Kennzahlen ein Algorithmus angegeben werden. Die Eingabe dieser Algorithmen seien dabei immer die beiden Graphen $O = (S, C)$ und $G = (I, R)$ zusammen mit den Kostenfunktionen ω bzw. c_G und c_O. Die Ausgabe besteht dann aus der jeweiligen Kennzahl SNV_O, BTA_O oder VSM_O. Diese können folgendermaßen berechnet werden:

1. Für SNV_O wird über alle Knotenpaare (a, b) in $S \times S$ iteriert und der Wert $\frac{s_O(a,b)}{s_G(a,b)}$ einer Summe hinzugefügt. Insgesamt ergeben sich so $\mid S \times S \mid$ Schritte, in denen aber auch die Werte der Funktionen s_O und s_G ausgewertet werden müssen.
   Um dies in akzeptabler Laufzeit umzusetzen, werden die in Kapitel 1.2.2 beschriebenen Graphenalgorithmen eingesetzt, um vor dem Durchgehen aller Knotenpaare Vorberechnungen durchzuführen, nach denen s_O und s_G in konstanter Laufzeit ausgewertet werden können.
   Von der Funktion S_O wird jeder Funktionswert mindestens einmal abgefragt - die Länge eines jeden kürzesten Pfades muss bekannt sein. Diese Längen können mithilfe des Algorithmus von Floyd-Warshall in einer Laufzeit in $O(\mid S \mid^3)$ berechnet werden. In einem durschnittlichen ÖPNV-Netz sollte für die Anzahl der Stationen $\mid S \mid \leq 200$ gelten. Dann ist nämlich $\mid S \mid^3 \leq 8 \cdot 10^6$, während ein durchschnittlicher Computer eine Taktfrequenz von etwa 1 GHz hat und die Berechnung somit auch mit großem konstanten Faktor innerhalb einer Sekunde ausführen kann.
   Für s_G ist diese Berechnung ungeeignet, weil $\mid I \mid$ als die Anzahl der Kreuzungen einer Stadt viel zu groß für eine akzeptable Laufzeit sein kann. Es werden allerdings auch nicht alle kürzesten Pfade benötigt, sondern nur die zwischen jedem Paar von Bushaltestellen. Die Vorberechnung lässt sich deshalb erreichen durch die Anwendung des Dijkstra-Algorithmus auf G mit jedem Knoten einer Bushaltestelle als Startknoten. Die Laufzeit für diesen Schritt ist dann $O(\mid S \mid \cdot \mid I \mid \log \mid I \mid)$.
   Mit allen Vorberechnungen ist die Laufzeit des Algorithmus insgesamt also $O(\mid S \mid^3 + \mid S \mid \cdot \mid I \mid \log \mid I \mid + \mid S \times S \mid)$, was für $\mid I \mid \leq 10^4$ und $\mid S \mid \leq 200$ auf einem durchschnittlichen Computer akzeptable Laufzeiten liefert.

2. Für die Bestimmung von BTA_O muss über die Kanten von O iteriert werden. Für jede dieser Kanten $(a, b) \in C$ kann der Wert $\frac{1}{\omega(a,b)}$ in konstanter Zeit ausgewertet werden, was zu einer Laufzeit von $O(\mid C \mid)$ führt.

3. Die Berechnung von VSM_O gleicht stark der von SNV_O. Es wird ebenfalls eine Vorberechnung aller Werte von s_G und s_O benötigt und es müssen ebenfalls alle Paare von Haltestellen durchgegangen werden.
   Der Unterschied liegt in der Tatsache, dass die Addition von Quotienten $\frac{s_O(a,b}{s_G(a,b)}$ ersetzt wird durch die Abfrage, ob der aktuelle Koeffizient größer ist als der bisher größte gefundene.

Da beide Operationen konstante Laufzeit benötigen, bleibt die Laufzeit des Verfahrens gleich (wobei ein Programm, das sowohl SNV_O als auch VSM_O berechnet, die Vorberechnungen nur einmal durchführen muss).

2.4 Implementierung

Die in Kapitel 2.3 beschriebene algorithmische Bestimmung der Kennzahlen habe ich in C++ umgesetzt. Mithilfe des dabei entstandenen Programms können beispielhafte ÖPNV-Netze bewertet und aus Experimenten mit verschiedenen ÖPNV-Netzen allgemeine Aussagen abgeleitet werden.

2.4.1 Eingabe und interne Speicherung

Die Eingabe durch den Nutzer muss die Graphen $O = (S, C)$ und $G = (I, R)$ sowie die zur Bestimmung der Kostenfunktionen c_O und c_G notwendigen Informationen beinhalten.
Da die genaue Lage der Kreuzungen und Haltestellen für die Berechnung der beschriebenen Kennzahlen unerheblich ist, sollen diese auch gar nicht angegeben und die Haltestellen und Kreuzungen durch je einen Index identifiziert werden. Der Nutzer muss dann noch die Verbindungen zwischen diesen Orten beschreiben.
Eine jede solche Verbindung wird eindeutig definiert durch die Indizes der Endpunkte, der Kostenfunktion c_G für G und der Funktion ω für O. Durch die zusätzliche Angabe des Faktors u liegen genug Informationen vor, um $c_O(a, b)$ nach Gleichung 2.2 für alle Tupel $(a, b) \in S$ zu berechnen.
Die Graphen selbst werden als Adjazenzliste und O zusätzlich als Adjazenzmatrix in von der C++-Standardbibliothek bereitgestellten Containern gespeichert.

2.4.2 Berechnung der Kennzahlen

Die Zahl BTA_O lässt sich berechnen, indem mithilfe einer `for`-Schleife über alle Kanten iteriert und für jede Kante der entsprechende vom Nutzer vorgegebene Wert von $\frac{1}{\omega}$ zu einer Ausgabevariable addiert wird.
Für die Berechnung von SNV_O und VSM_O müssen die Werte von $s_O(a, b)$ und $s_G(a, b)$ für $a, b \in S$ vorberechnet und zwischengespeichert werden. Gespeichert werden diese in zweidimensionalen Containern von `double`-Werten und berechnet durch eine Umsetzung des Floyd-Warshall-Algorithmus bzw. eines mehrmaligen Aufruf des Dijkstra-Algorithmus.
Der Dijkstra-Algorithmas wird von jedem Knoten in O aus einmal ausgeführt, was alle Werte von $s_G(a, b)$ ergibt. Aus diesen können mithilfe direkt die Kantengewichte der Buslinien berechnet und in die Adjazenzmatrix von O eingetragen werden. Ausführen des Floyd-Warshall-Algorithmus auf dieser Matrix ergibt alle Werte von $s_O(a, b)$.
Liegen all diese Funtionswerte vor, können SNV_O und VSM_O durch zwei ineinanderverschachtelte Schleifen, die über alle Paare von Knoten iterieren, berechnet werden.

3 Beispiele

Ziel der Arbeit ist es, Methoden zu entwicklen, mit denen einzelne ÖPNV-Netze bewertet werden können, um daraus allgemeine Aussagen abzuleiten.
Deswegen wurden mithilfe eines weiteren, selbstgeschriebene Programms randomisiert fiktive Städte mit ca. 10^4 Kreuzungen und 150 Haltestellen erzeugt und in diesen Städten Buslinien für verschiedene Netztypen generiert. Auf diesem Weg lässt sich schnell eine große Menge von Beispieldaten generieren, mit denen dann Experimente durchgeführt werden können.

3.1 Beispiele verschiedener Netztypen

Es werden beispielhaft drei verschiedenen Netztypen untersucht:

1. Ein in der Realität häufig vorkommender Netztyp ist der des radialen Netzes mit einem Stadtzentrum, von dem alle Linien sternförmig ausgehen.

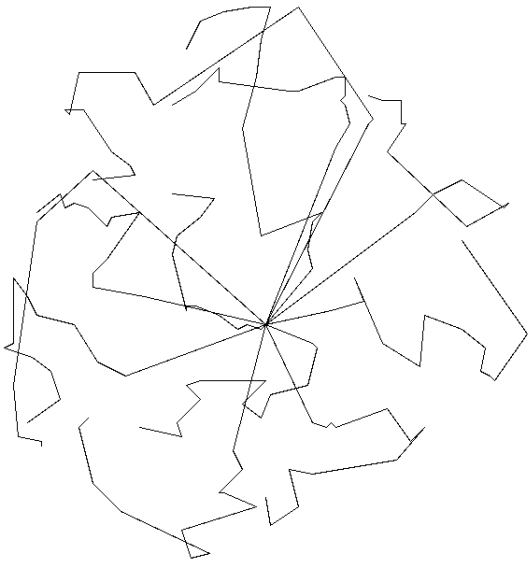


Abbildung 3.1: Visualisierung eines radialen Netzes, bei dem die Linien durch Striche dargestellt werden.

   Anders als bei einem Linienplan werden die Bushaltestellen in der Abbildung räumlich so platziert, wie sie sich auf der Straßenkarte der fiktiven Stadt befinden. Deswegen können sich die Buslinien auch schneiden.

2. Ein radiales Netz kann durch tangentiale Linien erweitert werden, die die Linien des radialen Netzes miteinander verbinden. Dieses soll im folgenden auch „Spinnennetz“ genannt werden.

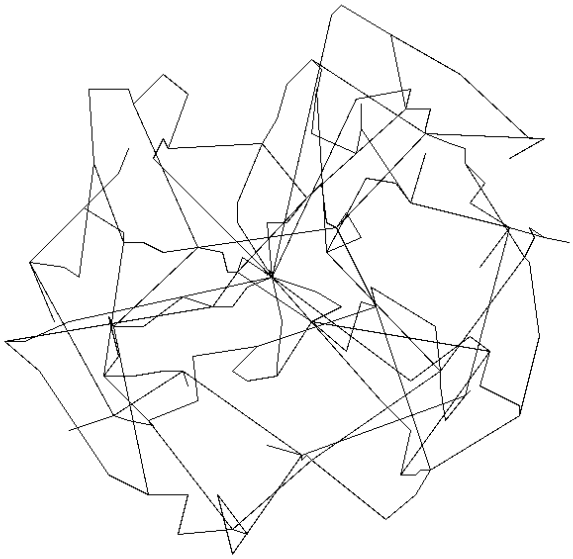


Abbildung 3.2: Visualisierung eines Spinnenetzes

3. Es wird auch das ÖPNV-Netz untersucht, dass die Haltestellen mit einer möglichst geringen Gesamtlänge der Busverbindungen (womit mit „Gesamtlänge“ die Summe der Kantenkosten gemeint ist) verbindet. Ein solcher Graph soll im Folgenden auch „MST“ (für Minimum Spanning Tree) genannt werden.

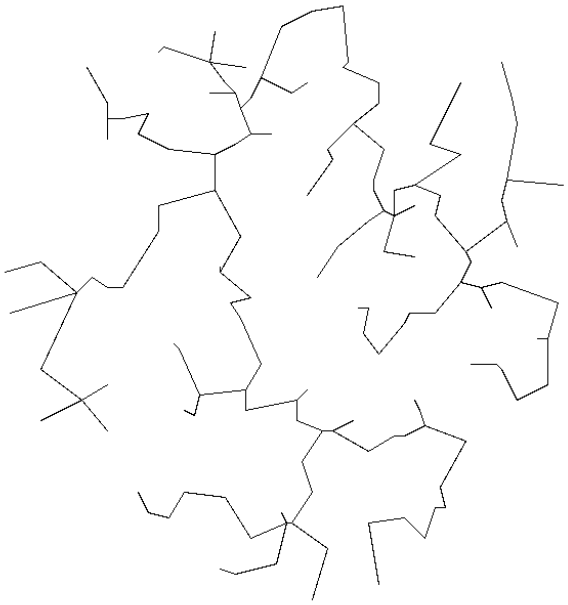


Abbildung 3.3: Visualisierung eines MSTs

3.2 Randomisierte Erzeugung der Städte und Netze

Um sicherzustellen, dass das Straßennetz wirklich zusammenhängend ist, wird es in der Implementierung zunächst so behandelt, als lägen die Kreuzungen alle in einer Ebene auf Punkten mit ganzzahligen Koordinaten und als seien sie mit allen vier Nachbarkreuzungen mit einem Abstand von 1 -sofern vorhanden- verbunden.
In dieser Darstellung ist das Straßennetz eine Ansammlung von Quadraten, deren Eckpunkte von Kreuzungen gebildet werden und deren Kanten die Straßen darstellen. Ausgehend von einem Anfangsquadrat aus vier Kreuzungen können so an zufälligen, mit bisherigen Kreuzungen verbundenen Stellen neue Kreuzungen eingefügt werden, bis die gewünschte Anzahl von Kreuzungen (hier 10^4) nicht mehr unterschritten wird.
Dies führt mehrheitlich zu Städten, die grob kreisförmig sind und in denen Kreuzungen zumeist von vier Richtungen aus erreichbar sind - was durchaus realistisch ist. Die geographische Lage der Kreuzungen zueinander ist dabei unerheblich, die Kantengewichte des Graphen (und damit der Zeitaufwand einzelner Straßen) wird per Zufallsgenerator zugewiesen.
Anschließend werden die (hier 150) Bushaltestellen zufällig auf die Kreuzungen verteilt. Da Bushaltestellen in der Realität selten allzu nah beieinanderliegen, wird nicht zugelassen, dass Bushaltestellen über weniger als vier Kreuzungen von anderen Haltestellen aus erreicht werden können.
Diese Schritte sind bei allen Versuchen gleich, lediglich die Verbindungen der Bushaltestellen untereinander unterscheiden die verschiedenen Netztypen.

Zur Erzeugung von radialen Netzen wird zunächst eine Haltestelle als Zentrum so gewählt, dass diese in der Verteilung der Kreuzungen auf Punkte in der Ebene mit ganzzahligen Koordinaten möglichst mittig in der Stadt liegt. Es werden dann immer die Endpunkte der von diesem Zentrum strahlenförmig ausgehenden Linien verwaltet und in jedem Schritt eine noch nicht angebundene Haltestelle mit einem solchen Endpunkt (oder dem Zentrum) verbunden. Der Endpunkt und die neue Haltestelle werden dabei mit minimalem Abstand gewählt.
In der Praxis entstehen so aber häufig Netze, in denen fast alle Haltestellen auf einer Linie liegen, was nicht dem strahlenförmigen Typus entspricht. Daher wird für die Anzahl der Bushaltestellen in einer einzelnen Linien eine konstante Obergrenze verwaltet (Experimente legen unter den gegebenen Umständen den Wert 12 nahe).

Zur Erzeugung von Spinnennetzen wird zunächst ein radiales Netz erstellt und dieses dann um tangentiale Linien erweitert. Dafür sollen jeweils die dritten, sechsten, neunten und zwölften Haltestellen einer Linie (bei Zählung vom Zentrum aus) zu einem Ring verbunden werden.
Für einen solchen Ring werden zunächst die in diesen einbezogenen Haltestellen und die Kosten aller Verbindungen zwischen zwei solchen Haltestellen ermittelt. Der Ring selbst entspricht dann dem minimalem Spannbaum unter dem Teilgraphen von O, der von diesen Haltestellen gebildet wird und wird durch den Algorithmus von Kruskal berechnet (der Ring ist dann nicht wirklich abgeschlossen - es fehlt eine einzelne Kante).

Ein MST ist ein minimaler Spannbaum des gesamten Graphen O und kann direkt mit dem Algorithmus von Kruskal berechnet werden.

3.3 Ergebnisse und Interpretation der Beispiele

Für jeden oben vorgestellten Netztyp wurden je zehn Städte generiert und getestet. Für radiale Straßennetze ergaben sich dabei folgende Werte:

Versuch Nr.	SNV_O	BTA_O	VSM_O
1	4,325	3,725	125,659
2	3,565	3,725	109,819
3	3,777	3,725	165,467
4	3,945	3,725	209,691
5	3,501	3,725	78,942
6	3,910	3,725	60,540
7	4,120	3,725	505,000
8	3,952	3,725	58,368
9	3,671	3,725	57,275
10	4,396	3,725	99,207
Durchschnitt	3,916	3,725	146,997

Ergebnisse für Spinnennetze:

Versuch Nr.	SNV_O	BTA_O	VSM_O
1	2,240	4,775	28,368
2	2,454	4,75	55,200
3	2,736	4,75	70,667
4	2,309	4,775	69,275
5	2,503	4,75	101,100
6	2,215	4,775	58,667
7	2,692	4,775	37,067
8	2,464	4,75	91,469
9	2,318	4,75	183,500
10	2,564	4,775	124,677
Durchschnitt	2,450	4,763	81,999

Ergebnisse für MSTs:

Versuch Nr.	SNV_O	BTA_O	VSM_O
1	3,856	3,725	32,121
2	4,064	3,725	28,067
3	3,995	3,725	25,009
4	3,793	3,725	29,649
5	3,119	3,725	20,128
6	3,798	3,725	33,530
7	4,251	3,725	32,152
8	4,109	3,725	30,870
9	4,165	3,725	30,488
10	3,457	3,725	23,181
Durchschnitt	3,861	3,725	28,520

Diese Versuchsergebnisse lassen sich folgendermaßen interpretieren:
Die Werte für SNV_O liegen bei radialen Netzen mit durchschnittlich 3,916 am höchsten und bei Spinnenetzen mit 2,450 am niedrigsten. MSTs befinden sich mit etwa 3,861 näher bei radialen Netzen, womit Spinnenetze von den drei untersuchten Netztypen die beste durchschnittliche Fahrzeit für die Fahrgäste bieten.
Hierbei fällt auf, dass die Ergebnisse von MSTs bei diesem Kriterium stark zwischen 3,119 und 4,215 schwanken, während die Werte für Spinnennetze zwischen 2,215 und 2,692 und für radiale zwischen 3,501 und 4,325 liegen. SNV_O ist bei MSTs also stärker von der zugrunde liegenden Stadt abhängig, als dies

bei den anderen beiden Netztypen der Fall ist.
Begründen lassen sich diese Beobachtungen dadurch, dass Fährgäste, die von einer Haltestelle zu einer anderen fahren wollen, immer eine Route über den Schnittpunkt aller Linien wählen müssen (insofern die beiden Haltestellen nicht auf derselben Linie liegen), was zu längeren Wegen führt.
In Spinnennetzen können diese Wege durch die tangentialen Linien abgekürzt werden, während Routen in MSTs ähnlich wie in radialen Netzen eindeutig gewählt werden müssen und so zu weniger direkten Wegen führen. Trotzdem sind MSTs „besser“, weil die Wege nicht alle über ein künstlich gewähltes Zentrum führen müssen.
Das Kriterium BTA_O nimmt innerhalb eines Netztyps sehr ähnliche oder auch immer gleiche Werte an. Diese liegen für MSTs und radiale Netze hier immer bei 3,725 und bei Spinnenetzen etwas höher bei 4,763.
Die Gleichheit dieser Werte ist zunächst mit dem Verfahren, mit dem die Städte und die Liniennetze generiert wurden, erklärbar: Dieses gibt für die Frequenz der Busse einer Linie immer denselben Wert an. In der Realität folgt diese aus den Fahrten der Busse inklusive Leerfahrten und der Wichtigkeit verschiedener Strecken, die hier nicht modelliert wurden. Daher wird hier ein Standardwert verwendet, womit das Kriterium BTA_O besser ist, wenn es im ÖPNV-Netz weniger Verbindungen gibt.
Radiale Netze und MSTs haben dabei immer den geringstmöglichen Wert, da sie n Haltestellen mit der Mindestanzahl von $(n-1)$ Linien verbinden. Spinnennetze müssen für Busunternehmen teurer sein, da sie als radiales Netz mit hinzugefügten tangentialen Linien dargestellt werden können.
Stark unterschiedliche Zahlen ergeben sich für das Kriterium VSM_O, bei dem der Durchschnitt für radiale Netze mit 146,997 am höchsten, für MSTs mit 28,520 am niedrigsten und für Spinnennetze bei 81,999 liegt.
Bei diesem Kriterium schwanken die Werte für radiale Netze am stärksten, und zwar zwischen 60,540 und 505,000, während MSTs am stabilsten zwischen 20,128 und 33,530 sind.
Die schlechten Werte von radialen Netzen erklären sich dadurch, dass in radialen Netze zwei Haltestellen vorkommen können, die im Straßennetz nah beieinander, aber auf unterschiedlichen Buslinien und weit entfernt vom Zentrum liegen können. Dann ist der Quotient zwischen der Fahrtzeit der weiten Strecke über das Zentrum und der kurzen Strecke im Straßennetz extrem hoch.
Aus diesen Zahlen lässt sich ableiten, dass für Fahrgäste radiale Netze in den beiden für sie relevanten Kriterien VSN_O und VSM_O am schlechtesten sind, während Spinnennetze zu einer besseren Durchschnitts- und MSTs zu einer besseren Maximalfahrzeit führen. Für Busunternehmen sind radiale Netze und MSTs gleich billig und günstiger als Spinnennetz - demzufolge wären MSTs eine bessere Wahl als die häufig vorkommenden radialen Netze, da sie für Netzbetreiber nicht teurer und für Fahrgäste besser sind.

Die Gültigkeit der Versuchsergebnisse ist allerdings durch die Modellierung und den Generator der Eingaben stark eingschränkt:
So werden etwa bei der in Kapitel 2 beschriebenen Modellierung Eigenheiten einzelner Städte wie beispielsweise Einbahnstraßen nicht berücksichtigt. Einen besonders großen Einfluss hat auch die Tatsache, dass Umstiegszeiten, Fahrpläne und Leerfahrten in einen einzelnen Funktionswert einer Kostenfunktion zusammengefasst werden. Dadurch wird nicht berücksichtigt, ob für eine Fahrt ein Linienwechsel gemacht werden muss oder nicht.
Die unterschiedlichen Frequenzen von Bussen werden auch beim Generieren der Eingabedaten nicht berücksichtigt. Außerdem ist die Konstruktion von radialen Netzen oder Spinnennetzen auf einem gegebenen Stadtplan nicht eindeutig und die Eigenschaften eventuell auch von der Größe der Stadt abhängig, die hier nicht berücksichtigt wurde.
Die hier erzielten Ergebnisse müssen also unter Vorbehalt gestellt werden.

4 Fazit und Ausblick

In der vorliegenden Arbeit habe ich ÖPNV-Netze modelliert, zahlengestütze Kriterien zur Beurteilung solcher Netze vorgeschlagen und Algorithmen zur Berechnung dieser Kriterien gefunden. Außerdem habe ich diese Algorithmen implementiert und auf beispielhafte, von mir randomisiert generierte ÖPNV-Netze angewandt, um über die Vor- und Nachteile bestimmter Netztypen zu entscheiden.

In Zukunft könnte die Modellierung noch stark verbessert werden, beispielsweise durch Einbezug von Umstiegszeiten, unterschiedlichen Fahrplänen zu unterschiedlichen Uhrzeiten oder das Berücksichtigen von Leerfahrten und Dienstplänen der Busfahrer. Gerade die Umstiegszeiten könnten besser dargestellt werden durch ein Modellierung, die auch die Linienführung und damit den Zusammenhang zwischen einzelnen Busverbindungen berücksichtigt. Hierfür müsste natürlich auch die Berechnung der vorgeschlagenen Kriterien angepasst werden.
Außerdem gibt es unbegrenzte Möglichkeiten für weitere, auf einer besseren Modellierung beruhende Kriterien. Diese könnten zum Beispiel Aussagen über die Länge der Umstiegszeiten oder die Auslastung einzelner Linien treffen oder die Kosten eines Netzes für die Betreiber realistischer ermitteln.
Ähnliche Verbesserungen sind für das Generieren beispielhafter Eingaben möglich, die ja auf derselben Modellierung aufbauen. Außerdem könnten die Frequenzen von Bussen auf einzelnen Verbindungen bewusster angepasst werden, um zentraler gelegene Linien öfter nutzbar zu machen und dies finanziell durch eine Kostensenkung auf abgelegeneren Linien auszugleichen. Mit Eingaben unterschiedlicher Größe, Experimenten von größerem Umfang und Daten aus realen Städten ließen sich dann deutlich belastbarere Aussagen treffen.
Das Problem lässt sich auch insofern abwandeln, als dass die Orte der Haltestellen bisher Teil der Eingabe sind. In einer Erweiterung könnten Fußwege im Straßennetz berücksichtigt und die Positionen der Haltestellen danach ausgerichtet werden. Anstatt die für die Fahrt zwischen zwei Haltestellen nötige Zeit würde dann nach dem Zeitaufwand für die Strecke zwischen zwei beliebigen Kreuzungen einer Stadt gefragt werden.
Für alle diese Erweiterungen wären aber mehr Daten über das Straßennetz einer Stadt nötig, also beispielsweise Gehzeiten im Straßennetz oder die Lage von häufig besuchten Punkten in der Stadt.

Es stellt sich aber auch die Frage, inwiefern es möglich ist, aus einer gegebenen Prioritätsliste von Kriterien und einer gegebene Stadt direkt auf ein möglichst gutes ÖPNV-Netz zu schließen, ohne einen Umweg über langwierige Experimente zu gehen, deren Ergebnisse einzeln interpretiert werden müssen und die eingeschränkt sind durch die Netztypen, die sich Menschen vor den Experimenten ausdenken. Und selbst wenn ein optimaler Netztyp bekannt wäre, müsste dieser ja noch an eine einzelne Stadt angepasst werden (so muss für ein radiales Netz ja beispielsweise ein Zentrum gewählt werden).
Hier könnte es eine Möglichkeit sein, die Kriterien (mit unterschiedlichen Gewichtungen als Angabe von Prioritäten) als Bewertungsfunktion zu benutzen, die die Qualität eines ÖPNV-Netzes in einer einzelnen Zahl ausdrückt. Die Ermittlung eines optimalen Netzes wäre mithilfe dieser Funktion ein Optimierungsproblem, dass vermutlich zu komplex ist, um exakt gelöst zu werden.
Für Optimierungsprobleme dieser Art existieren aber Verfahren für Näherungslösungen, die auch hier angewandt werden könnten - etwa evolutionäre Algorithmen oder das Simulated-Annealing-Verfahren.

5 Quellen

- Halim, Steven; Halim, Felix; Effendy, Suhendry: Competitive Programming 4: The Lower Bound of Programming Contests in the 2020s: Book 1 Chapter 1-4: Handbook for IOI and ICPC Contestants, and for Programming Interviews. o.O, 2020
- Kerzig, Dennis: Graphentheorie.
  https://hpi.de/friedrich/teaching/units/graphentheorie.html
  zuletzt abgerufen am 22.05.2022
- Overhagen, Theo: Graphentheorie: Mathematik: Universität Siegen.
  https://www.uni-siegen.de/fb6/analysis/overhagen/
  vorlesungsbeschreibungen/skripte/graphentheorie1.pdf
  zuletzt abgerufen am 22.05.2022
- Skiena, Steven S.: THE Algorithm Design MANUAL, Third Edition. Cham ZG, 2020

5.1 Abbildungsverzeichnis

- Titelbild: https://www.swk-kl.de/produkte-services/busverkehr/busfahrplan
  /buslinien-plan, zuletzt abgerufen am 13.01.2022
- Seite 5, Abbildung 1.1: https://upload.wikimedia.org/wikipedia/commons/thumb/8/87/
  Kruskal_Algorithm_6.svg/200px-Kruskal_Algorithm_6.svg.png
- Seite 11, Abbildung 3.1: eigene Darstellung
- Seite 12, Abbildung 3.2: eigene Darstellung
- Seite 12, Abbildung 3.3: eigene Darstellung

Sämtliche Ein-, Ausgabe- und Quelltextdateien können unter
https://github.com/mousebeaver/JugendForschtAlgorithmischeBewertungVonOPNVNetzen
abgerufen werden.

5.2 Unterstützung

Die vorliegende Arbeit basiert auf einer Facharbeit im Fach Mathematik, die im Schuljahr 2022/23 am Hohenstaufen-Gymnasium in Kaiserslautern erarbeitet und von Frau Kirch-Rink betreut wurde.