Karlsruhe Institute of Technology

Shortest-Path Cover auf eingeschränkten Graphklassen

Bachelorarbeit
von

Jörg D. Weisbarth

An der Fakultät für Informatik
Institut für Theoretische Informatik

Erstgutachter:	Prof. Dr. Dorothea Wagner
Zweitgutachter:	Prof. Dr. Peter Sanders
Betreuende Mitarbeiter:	Moritz Baum, M.Sc.
	Dipl.-Inform. Julian Dibbelt
	Dr. Ignaz Rutter

Bearbeitungszeit: 13. Januar 2012 – 11. Mai 2012

Urheberschaftserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die hier angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 11. Mai 2012

Danksagung

Ich möchte mich herzlich bei Prof. Dr. Dorothea Wagner dafür bedanken, dass sie mir diese Bachelorarbeit ermöglicht hat. Außerdem möchte ich mich bei meinen Betreuern Moritz Baum, Julian Dibbelt und Ignaz Rutter für ihre ebenso zahl-, wie hilfreichen Korrekturvorschläge, Diskussionen über Ideen und Anregungen, sowie für die Zeit, die sie investiert haben, bedanken. Das hat mir bedeutend dabei geholfen durch diese Arbeit zu lernen, wie man an wissenschaftliche Themen und Forschungen herangeht.

Zusammenfassung

Abraham et al. [AFGW10] führen Shortest-Path-Cover (SPCs) mit Dichtegarantien ein, das sind Knotenteilmengen eines Graphen, die alle kürzesten Pfade ab einer bestimmten Länge abdecken. Damit konnten Algorithmen zur Suche nach kürzesten Wegen auf Straßennetzwerken wie REACH, Contraction Hierarchies und Transit Nodes verbessert oder zumindest bessere Zeitgarantien angegeben werden. Der Grad der Verbesserung ist abhängig von der Dichte eines SPCs. Je weniger dicht dessen Knoten verstreut sind, desto größer fallen die Verbesserungen für die obigen Algorithmen aus. Abraham et al. führen die Highway Dimension als Struktureigenschaft für Graphen ein. Von ihr wird angenommen, dass sie auf realen Straßennetzen niedrig ausfällt. Diese Eigenschaft verwenden sie bei der Polynomialzeitberechnung von SPCs der Dichte $O(h \log n)$ für Graphen der Highway Dimension h.

In dieser Arbeit werden SPCs auf eingeschränkten Graphklassen behandelt. Für Wege und Bäume werden minimale SPCs in linearer Zeit und für Kreise in $O(n \log n)$ berechnet. Diese genügen automatisch auch der Dichte h, d.h., dass in der Umgebung eines jeden Knotens maximal h Knoten aus einem SPC vorhanden sind. Für außenplanare Graphen mit der Outerpath-Eigenschaft wird eine 2-Approximation angegeben und für allgemeine außenplanare Graphen eine 3-Approximation. Die Berechnung minimaler SPCs ist auf gerichteten Graphen mit nicht eindeutigen kürzesten Wegen APX-schwer. Eine Approximation kann nicht besser als $(1 - o(1)) \ln n$ sein. Selbst auf ungerichteten Graphen mit eindeutigen kürzesten Wegen kann eine solche nicht besser als Faktor $1,36$ sein. Für planare, ungerichtete Graphen mit eindeutigen kürzesten Wegen bleibt die NP-Vollständigkeit erhalten, jedoch ist es nicht klar, ob das auch für die APX-schwere gilt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

1. Einführung

Die Routenplanung ist eines der bekanntesten Beispiele für Alltagsanwendungen mit einem weit erforschtem Hintergrund in der Theoretischen Informatik. Edsger Dijkstra präsentierte in den späten 50er Jahren einen nach ihm benannten Algorithmus, der einen kürzesten Weg zwischen zwei Orten mit fast linear wachsendem Zeitaufwand findet [Dij59]. Da kontinentale Straßennetze über eine enorme Größe verfügen, wird seither versucht die Berechnung deutlich zu beschleunigen. Spielraum dazu ist genug vorhanden, denn der Dijkstra-Algorithmus in seiner Urform sucht große Teile des Graphen ab, die nicht interessant für das Ergebnis der Anfrage sind. Einige der entwickelten Methoden basieren auf Vorberechnungen. Die nachfolgenden Anfragen nach kürzesten Wegen können anschließend effizienter beantwortet werden. Zu diesen Algorithmen gehören beispielsweise Arc Flags [Lau04], Transit Nodes [BFM+07] und Contraction Hierarchies [GSSD08]. Ein Überblick bieten Delling et al. [DSSW09].

Für einen partitionierten Graphen sind *Arc-Flags* Zusatzinformationen für jede einzelne Kante, die beschreiben, ob diese auf einem kürzesten Weg liegt, dessen Endpunkt in der Partition des Zielknotens liegt. Suchanfragen werden dann mit einer Variante des Dijkstra-Algorithmus beantwortet, wobei nur Kanten berücksichtigt werden müssen, für die das entsprechende Flag gesetzt ist. In der Contraction Hierarchies Vorberechnungsphase werden in den Graphen *Shortcuts* eingefügt. Das sind spezielle Kanten zwischen zwei Knoten, die im Ausgangsgraph nicht vorhanden sind und einen kürzesten Weg zwischen diesen Knoten repräsentieren. Zur Erzeugung dieser Shortcuts wird der Graph nach und nach kontrahiert, indem immer ein einzelner Knoten ausgelöscht wird. Je zwei seiner Nachbarn werden anschließend genau dann durch einen Shortcut miteinander verbunden, wenn zuvor der einzige kürzeste Weg über den entfernten Knoten führte. Die Reihenfolge, in der die Knoten ausgelöscht werden, definiert eine strenge Totalordnung. Die Suche nach einem kürzesten Weg wird auch hier im Wesentlichen mit einem modifizierten bidirektionalen Dijkstra-Algorithmus auf dem mit allen Shortcuts angereichertem Ausgangsgraphen beantwortet. Vom Start- und vom Zielknoten aus werden lediglich Kanten zu höherrangigen Knoten betrachtet. Sobald beide Suchbereiche nicht weiter ausgedehnt werden können, liegt ein Knoten im Schnitt der Suchbereiche, der die Summe der Entfernungen von den jeweiligen Ausgangspunkten minimiert, auf einem kürzesten Weg.

Die *Transit Nodes* in einem Graphen sind ausgewählte Knoten, die ihre kürzesten Wege untereinander via Distanztabellen kennen. Zusätzlich ist jeder Knoten mit den Transit Nodes

in seiner „Umgebung“ über Shortcuts vernetzt. Das Verfahren basiert auf der Beobachtung, dass man in Straßennetzen, um von einem lokalen Punkt im Netz aus gesehen in die „weite Ferne“ zu gelangen, einen Knoten aus einer verhältnismäßig kleinen Menge sogenannter *Acces Nodes* – in Abhängigkeit vom Startknoten – passieren muss. Die Menge aller Acces Nodes aller Knoten ist die Menge der Transit Nodes. Der Algorithmus beantwortet Suchanfragen mittels Dijkstra oder anderer Kürzeste-Wege-Algorithmen, falls Start- und Ziel nahe beieinander liegen und verwendet andernfalls die Distanztabellen zwischen den Transit Nodes.

Die oben genannten Algorithmen funktionieren insbesondere auf Straßennetzwerken erstaunlich gut. Das veranlasste Abraham et al. [AFGW10] nach Gründen dafür zu suchen. Die obige Beobachtung, die bereits bei der Entwicklung des Transit Node Verfahrens zum Tragen kam, inspirierte die Autoren zu den Definitionen der *Highway Dimension* und des *Shortest Path Covers*. Erstere garantiert für einen Graph mit Highway Dimension h, dass es um jeden Knoten herum, in einer Kugel mit einem gewissen Radius, h Knoten gibt, sodass jeder kürzeste Pfad ab einer bestimmten Länge, der vollständig in der Kugel verläuft, wenigstens einen dieser h Knoten enthalten muss. Die zweite Definition beschreibt eine Knotenteilmenge eines Graphen, die für jeden kürzesten Weg ab einer Mindestlänge wenigstens einen Knoten enthält und „gut“ über den Graphen verteilt ist. Sie konnten nachweisen, dass viele Algorithmen auf Graphen mit geringer Highway Dimension beweisbaren Effizienzkriterien genügen. So kann mittels SPCs beispielsweise eine geschickte strikte Ordnung der Knoten auf einem Graphen für Contraction Hierarchies gefunden werden. Auch lässt sich damit für den Transit Node Algorithmus eine gute Auswahl an Transit Nodes generieren.

In der Ausgangsarbeit wird ein Greedy-Algorithmus angegeben, mit dessen Hilfe ein SPC der Dichte $O(h \log n)$ berechnet werden kann, wobei h die Highway Dimension und n die Anzahl der Knoten des Graphen ist. In [ADF+11] wird dies mit Hilfe von Erkenntnissen aus dem Bereich des maschinellen Lernens auf $O(h \log h)$ verbessert, unter der zusätzlichen Annahme von eindeutigen kürzesten Wegen. Dazu wird die Verwandtschaft des Problems ein minimales Hitting Set zu finden mit dem Problem ein minimales SPC zu finden ausgenutzt, denn eine Instanz des letzteren kann immer in eine Instanz des ersteren mit Vapnik–Chervonenkis-Dimension (VC-Dimension) 2 transformiert werden (solange kürzeste Wege eindeutig sind). Für diese sind effiziente Algorithmen [BG95] bereits bekannt. Diese verbessern den Speicherbedarf und die Zeitgarantien für die von Abraham et al. [AFGW10] angegebenen, SPC-basierten Varianten der dort besprochenen Algorithmen.

In dieser Arbeit werden die Highway Dimension und die Berechnung von kleinst möglichen Shortest Path Covern in Polynomialzeit für bestimmte Graphklassen untersucht. In Graphen mit der Highway Dimension h existiert immer – und unabhängig vom oben erwähnten Parameter r – ein SPC, sodass sich von jedem Knoten aus, in der Kugel vom Radius $2r$ um ihn herum, höchstens h Knoten aus diesem SPC befinden. Alle kleinst möglichen SPCs halten dieses Dichtekriterium ein ([ADF+11], Theorem 4). Daher ist das Problem minimale Shortest-Path-Cover zu finden so interessant und wird hier untersucht.

Resultate

Es ist schwierig Graphmodifikationen vorzunehmen, die ein minimales SPC unverändert lassen. Jedoch macht die Unterteilung einer Kante – bei Erhaltung des Gesamtgewichts – ein minimales SPC auf keinen Fall kleiner. Weiter wird durch Reduktion vom bekannten Vertex-Cover-Problem bewiesen, dass die Berechnung minimaler SPCs für ungerichtete Graphen – auch mit eindeutigen kürzesten Wegen – im Allgemeinen APX-schwer ist. Die

NP-Vollständigkeit bleibt erhalten, falls man zusätzlich annimmt, der Graph sei planar. Ob das für die APX-schwere gilt, bleibt unklar. Eine Reduktion vom Hitting Set-Problem zeigt, dass für beliebige gerichtete Graphen, mit nicht notwendiger Weise eindeutigen kürzesten Wegen, eine Approximation nicht besser als $(1 - o(1)) \ln n$ sein kann. Die Berechnung minimaler SPCs funktioniert für Wege und Bäume in Linearzeit und für Kreise allgemein in $O(n \log n)$, wobei bei den meisten Kreisen ein Linearzeitalgorithmus zum Ziel führt. Für außenplanare Graphen wird eine 3-Approximation vorgestellt und für solche, die zusätzlich die Outerpath-Eigenschaft haben, eine 2-Approximation. Die Highway Dimension beträgt für Wege und Kreise höchstens 8 und ist bereits für Bäume beliebig hoch.

Die Arbeit ist wie folgt strukturiert:

Kapitel 2 enthält grundlegende Definitionen für die gesamte Arbeit und erste einfache und allgemeine Aussagen über SPCs.

Kapitel 3 betrachtet die Komplexität des Problems, minimale SPCs zu berechnen, durch Reduktionen von den bekannten Problemen der Berechnung minimaler Hitting Sets und Vertex Covers.

Kapitel 4 enthält exakte und effiziente Algorithmen zur Berechnung minimaler SPCs in Wegen, Bäumen und Kreisen. Außerdem wird die Highway Dimension der genannten Graphklassen betrachtet.

Kapitel 5 enthält einen Algorithmus zur Berechnung von 2-Approximationen für außenplanare Graphen mit der Outerpath-Eigenschaft, sowie einen Algorithmus zur 3-Approximation für allgemeine außenplanare Graphen.

Kapitel 6 Fasst kurz diese Arbeit zusammen und bietet einen Ausblick auf weitere mögliche Forschungen, die darauf aufbauen können.

2. Grundlagen

Im ersten Abschnitt dieses Kapitels werden viele der Definitionen und Konventionen eingeführt, auf die im weiteren Verlauf des Dokuments zurückgegriffen wird. Im anschließenden Abschnitt werden einige grundlegende Aussagen über SPCs behandelt.

2.1 Definitionen

Die meisten der hier verwendeten Begriffe und Notationen sind aus anderen Arbeiten zu Problemen in der Graphentheorie bekannt. Für eine endliche Menge M wird ihre Kardinalität mit $|M|$ angegeben und ihre Potenzmenge mit 2^M. Eine Partition einer Menge M ist eine Aufteilung von M in paarweise disjunkte Teilmengen. Ein *Graph* $G = (V, E)$ besteht aus einer Menge V von *Knoten* und einer Relation $E \subseteq V \times V$, deren Elemente *Kanten* genannt werden. Er sei mit einer Gewichtsfunktion $|\cdot| : E \to \mathbb{R}_{\geq 0}$ ausgestattet, die jeder Kante ein *Kantengewicht* zuordnet. Ein *Teil-* oder *Subgraph* $H = (V', E')$ von G ist ein Graph mit $V' \subseteq V$ und $E' \subseteq E$. Eine beliebige Teilmenge V' von V *induziert* einen Subgraphen H von G durch $H := (V', E')$ mit $E' := \{(v_1, v_2) \in E : v_1, v_2 \in V'\}$.

Ein *Weg* in einem Graphen ist eine geordnete Sequenz $(v_1, v_2, \ldots, v_t)$ von Knoten aus V, sodass jeweils zwei in der Sequenz benachbarte Knoten im Graphen durch eine Kante verbunden sind. Diese Sequenz kann man auch als Menge auffassen. Entsprechend sind zwei Wege w_1 und w_2 (paarweise) *disjunkt*, falls der Schnitt der mit ihnen assoziierten Knotenmengen leer ist. Ein Weg w heißt *abgedeckt* bezüglich einer Teilmenge $C \subseteq V$, falls C mindestens einen der Knoten aus w enthält. Das Gewicht $|w|$ eines Wegs w ist die Summe seiner Kantengewichte. Ein Weg $w = (v_s, \ldots, v_t)$ ist genau dann ein *kürzester Weg*, wenn es keinen weiteren Weg zwischen v_s und v_t gibt, dessen Gewicht kleiner ist. Sofern es einen eindeutigen kürzesten Weg zwischen zwei Knoten v_s und v_t gibt, wird er mit $P(v_s, v_t)$ notiert. Der *Abstand* eines Knotens v_s zu einem Knoten v_t ist definiert als die Länge eines kürzesten Wegs von v_s nach v_t, falls dieser existiert. Zu einem Knoten u ist die *Kugel* $B(u, r)$ der durch die Knotenmenge aller Knoten mit Abstand r von u, induzierte Subgraph. Der *Radius* eines Graphen G ist die kleinste Zahl r, für die ein Knoten u in G existiert, sodass $B(u, r)$ G vollständig enthält.

Ein Graph lässt sich durch *Separatoren* wie folgt zerlegen. Eine Teilmenge $S \subset V$ ist ein Separator, wenn der durch $V \setminus S$ induzierte Subgraph von G *unzusammenhängend* ist, d.h.

es gibt zwei Knoten in diesem Graphen, die durch keinen Weg verbunden sind. Mit $G \setminus H$ ist der durch $V \setminus V'$ induzierte Subgraph gemeint.

Die nachfolgenden Definitionen sind der Beobachtung geschuldet, dass in Straßennetzen ein kürzester Pfad aus einer kompakten Lokalität in die „weite Ferne“ oftmals zwangsläufig über eine sehr kleine Anzahl an „Zutrittsknoten“ führt [AFGW10].

Definition 1 (Shortest Path Cover (SPC))**.** *Gegeben ein Graph G ist eine Menge $C \subset V$ genau dann ein r-SPC von G, wenn für je zwei Knoten mit mindestens dem Abstand r ein kürzester Pfad zwischen diesen Knoten durch C abgedeckt wird.*

Sei $H = (V', E')$ ein Teilgraph eines Graphen G, dann ist eine Menge $C' \subset V'$ ein lokales *r-SPC, wenn für je zwei Knoten in H, zwischen denen ein vollständig in H verlaufender, kürzester Pfad mindestens der Länge r existiert, ein ebensolcher durch einen Knoten aus C' abgedeckt wird. Zur deutlicheren Unterscheidung zwischen dem SPC eines Teilgraphen und dem eines ganzen Graphen, kann letzterer optional als* global *bezeichnet werden.*

Ein minimales oder kleinstes (globales/lokales) r-SPC von G ist ein (globales/lokales) r-SPC minimaler Kardinalität.

Es wäre ebenfalls eine ähnliche Definition möglich, in der gefordert wird, dass zwischen je zwei Knoten *alle* kürzesten Pfade, die die Mindestlänge r erreichen oder überschreiten, abgedeckt sein müssen. Falls alle kürzesten Wege eindeutig sind, fallen die beiden Definitionen zusammen. Für diese Arbeit wurde die Version in Definition 1 gewählt, da sie einerseits stark genug für die Beweise in der Arbeit von Abraham et al. [AFGW10] ist, andererseits zusätzlichen Spielraum in Graphen mit nicht eindeutigen kürzesten Wegen ermöglicht. Beispielsweise kann ein minimales SPC hier kleiner ausfallen als mit der anderen Definitionsmöglichkeit. An dieser Stelle taucht schon eine Problemstellung auf, der hier einen Namen gegeben werden soll.

Problem 1 (MIN-SPC)**.** *Das Problem zu gegebenen Graphen G und Parameter r ein minimales r-SPC zu berechnen, wird mit MIN-SPC bezeichnet.*

Die Definition eines lokalen SPCs wurde in der oben zitierten Arbeit noch nicht eingeführt. Sie wird hier verwendet, um die nachfolgende Definition anders zu formulieren, wie von Abraham et al.

Definition 2 (Highway Dimension (HD))**.** *Ein kantengewichteter Graph $G = (V, E)$ hat genau dann die Highway Dimension h, wenn h die kleinste natürliche Zahl ist, für die gilt:*

Zu jedem Parameter $r \in \mathbb{R}_{\geq 0}$ und jedem Knoten $u \in V$ gibt es in der Kugel $B(u, 4r)$ ein lokales r-SPC mit höchstens h Knoten.

Falls alle kürzesten Wege eindeutig sind, ist diese Definition äquivalent zu der von Abraham et al. Von realen Straßennetzen nimmt man an, dass sie sich durch eine geringe HD auszeichnen. Abraham et al. haben dazu ein Modell erstellt, das versucht der Art nahe zu kommen, wie in der realen Welt Straßennetze gebaut werden. Es wurde für Graphen, die nach diesem Modell erstellt werden, bewiesen, dass sie über eine niedrige HD verfügen.

Es sollen nun noch einige weitere, häufig benötigte Definitionen folgen.

Definition 3 (Bedürftiger Weg)**.** *Ein Weg w ist genau dann bedürftig (bezüglich dem Parameter r), wenn er ein kürzester Weg ist und mindestens die Länge r hat.*

Es zeichnen sich also genau diejenigen Wege durch diese Eigenschaft aus, die in Graphen mit eindeutigen kürzesten Wegen von jedem r-SPC abgedeckt werden müssen.

Definition 4 (r-Minimalität)**.** *Ein r-bedürftiger Weg $w = (v_s, v_{s'}, \ldots, v_{t'}, v_t)$ ist genau dann r-minimal, wenn seine um die beiden Endknoten v_s, bzw. v_t gekürzten Teilwege $(v_{s'}, \ldots, v_{t'}, v_t)$, bzw. $(v_s, v_{s'}, \ldots, v_{t'})$ jeweils die Länge r unterschreiten.*

Das Problem 1 hat eine hohe Verwandtschaft zu einem anderen Problem, auf das es leicht reduziert werden kann. Dafür wird zunächst die folgende Definition benötigt.

Definition 5 (Hitting Set [GJ79])**.** *Gegeben eine Grundmenge M und ein System $S = \{T_1, \ldots, T_n\}$ von Teilmengen von M. Eine Teilmenge H von M ist ein Hitting Set von S, wenn H mit jeder der Mengen in S mindestens ein Element gemeinsam hat.*

Eine mögliche Problemstellung ist, gegeben eine Menge S von Teilmengen, ein Hitting Set für S zu finden. Eine triviale Lösung dafür findet sich schnell, indem man einfach alle Elemente in das Hitting Set aufnimmt. Anders sieht es mit dem zugehörigen Optimierungsproblem aus.

Problem 2 (Hitting Set Optimierungsproblem (MIN-HS) [Kar72])**.** *Sei M eine Grundmenge, sowie S eine Menge von Teilmengen von M. Das Problem ein kleinstmögliches Hitting Set von S zu finden wird hier mit MIN-HS bezeichnet.*

Eng verwandt damit sind die beiden folgenden Definitionen.

Definition 6 (Set Cover)**.** *Gegeben sei eine Grundmenge M und n Teilmengen $T_1, \ldots, T_n$. Ein Set Cover ist eine Teilmenge C von $\{T_1, \ldots, T_n\}$, sodass ihre Vereinigung M ist, d.h. $\bigcup_{T \in C} = M$.*

Problem 3 (Set Cover Optimierungsproblem (MIN-SC) [Kar72])**.** *Sei M eine Grundmenge, sowie S eine Menge von Teilmengen von M. Das Problem ein kleinstmögliches Set Cover von M zu finden wird hier mit MIN-SC bezeichnet.*

Tatsächlich handelt es sich bei MIN-HS und MIN-SC um dasselbe Problem, nur unter einem andern Blickwinkel. Beispielsweise lässt sich eine Instanz von MIN-HS in eine Instanz von MIN-SC transformieren, indem jedes Element als Menge betrachtet wird, dessen Elemente die Mengen sind, in denen es zuvor enthalten war.

Da hier Probleme vor allem mit Bezug zur Routenplanung untersucht werden sollen, dürfen einige Annahmen über Graphen gemacht werden, die auf reale Straßennetze oft zutreffen. Es werden nur endliche und einfache (d.h. es existiert maximal eine Kante zwischen je zwei Knoten) Graphen betrachtet. Die Einfachheit wird angenommen, weil man sich nur für kürzeste Wege interessiert und es daher genügt zwischen zwei Knoten nur die günstigste Kante zu betrachten. Alle Kantengewichte sind aus $\mathbb{R}_{\geq 0}$, insbesondere hat keine Kante ein negatives Gewicht. Sofern nicht anders angegeben, sind alle Kanten ungerichtet und es gibt zwischen je zwei Knoten maximal einen kürzesten Weg.

2.2 Erste Folgerungen

In diesem Abschnitt werden einige Aussagen über SPCs in allgemeinen Graphen gemacht. Zuvor werden für das nachfolgende Lemma noch zwei Begriffe benötigt.

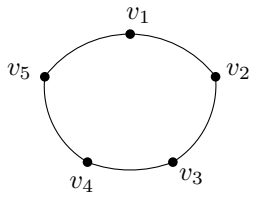


Abbildung 2.1: Der komplette Kreis.

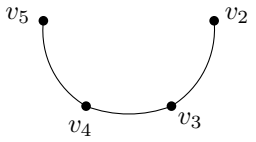


Abbildung 2.2: Ausschnitt aus einem Kreis. Alle Kanten haben Gewicht 1.

Definition 7 (Zeuge und Zeugenmenge)**.** *Gegeben sei ein Graph G und ein positives, reelles r. Eine Zeugenmenge ist eine Menge von paarweise disjunkten, r-bedürftigen Wegen. Ein Element einer Zeugenmenge heißt Zeuge.*

Lemma 8. *Sei $G = (V, E)$ ein Graph und $C \subseteq V$ ein zugehöriges r-SPC.*

(a) Für einen r-bedürftigen Weg $w = (v_s, \ldots, v_t)$ in G muss mindestens einer der Knoten aus $\{v_s, \ldots, v_t\}$ in C enthalten sein.

(b) Sei $Z = \{w_1, \ldots, w_n\}$ eine Zeugenmenge, dann enthält C mindestens n Elemente.

(c) Eine Teilmenge $C' \subset V$ ist genau dann ein r-SPC von G, falls es alle r-minimalen Wege in G abdeckt.

Beweis. Aussage a geht – aufgrund der Einschränkung, dass kürzeste Wege eindeutig sind – unmittelbar aus der Definition von r-bedürftigen Wegen hervor.

Zu Aussage b: Aus jeder der n als Knotenteilmengen interpretierten Zeugen muss jeweils mindestens ein Element auch in C enthalten sein. Da die Teilmengen paarweise disjunkt sind, muss C nach Aussage a aus jeder dieser Mengen jeweils mindestens ein Element enthalten.

Zu Aussage c: Es ist klar, dass ein r-SPC alle r-minimalen Wege abdecken muss. Es bleibt zu zeigen, dass damit auch jeder r-bedürftige Weg abgedeckt ist. Sei w ein r-bedürftiger, aber nicht r-minimaler Weg. Dann enthält w einen r-minimalen Teilweg w', der nach Voraussetzung abgedeckt ist. Somit ist auch w abgedeckt. □

Eine traditionelle Herangehensweise bei vielen Problemen ist es, diese in kleinere, gleichartige Teilprobleme zu zerlegen um daraus eine Gesamtlösung zusammenzusetzen. Wenn es darum geht ein minimales r-SPC zu finden, so ist es im Allgemeinen nicht möglich den Graphen in kleinere Instanzen zu zerteilen, um diese dann separat zu lösen. Ein Grund ist, dass erst durch das Zusammenlegen zweier Teilgraphen r-bedürftige Wege entstehen könnten. Ein anderer, dass ein Teilgraph eines Graphen nicht zwingend dieselben kürzesten Wege enthält. Das wird schon bei einer der einfachsten Graphklasse relevant, in der die Wege zwischen zwei Knoten nicht eindeutig sind – den Kreisen. An dieser Stelle wird dazu ein Beispielgraph für den Parameter $r = 3$ betrachtet. Die Abbildung 2.1 zeigt einen Graph und die Abbildung 2.2 einen Teilgraph desselben. Es handelt sich um einen Kreis mit den Knoten 1 bis 5. Jede Kante hat das Gewicht 1. In der Abbildung 2.2 ist (v_2, v_3, v_4, v_5) offensichtlich ein bedürftiger Weg und ein minimales 3-SPC hat demnach mindestens die Größe 1. Betrachtet man jedoch den Gesamtgraphen, so ist derselbe Weg kein kürzester Weg mehr und muss somit nicht mehr abgedeckt werden. Tatsächlich ist das minimale 3-SPC des Graphen leer. Alle kürzesten Pfade haben maximal die Länge 2.

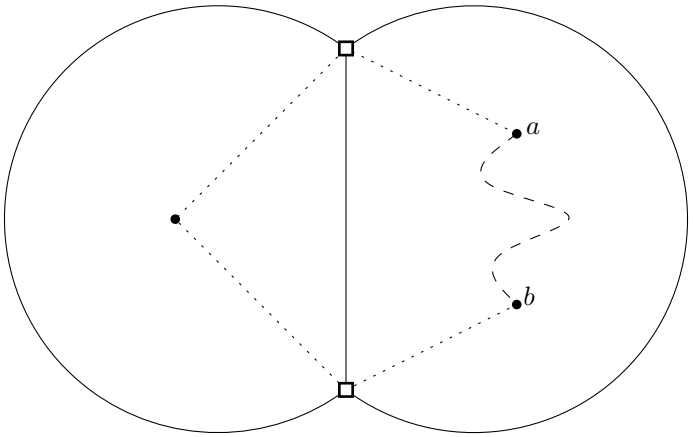


Abbildung 2.3: Beispiel für einen Weg, der in ungünstiger Weise über zwei Teilgraphen hinweg geht

Das eigentliche Problem hierbei sind kürzeste Pfade, deren Endknoten zwar beide in einem bestimmten Teilgraphen liegen, die aber diesen zwischendurch verlassen. Abbildung 2.3 illustriert diese Situation. Hier liegen die Punkte a und b im selben Teilgraph H. Die viereckigen Punkte sind alle Punkte in H, die zu einem Knoten in $G \setminus H$ inzident sind. Der kürzeste Pfad zwischen a und b – hier punktiert dargestellt – geht über die viereckigen Knoten aus H heraus und wieder hinein. Daher muss ein längerer, nicht-saturierter Pfad in H zwischen den beiden Knoten – hier gestrichelt dargestellt – nicht von einem r-SPC des Gesamtgraphen G abgedeckt werden, wohl aber von einem des Teilgraphen H. Diese Betrachtung motiviert das nachfolgende Kriterium.

Definition 9 (Zerlegbarkeitskriterium)**.** *Sei G ein Graph und H ein Teilgraph von G, dann genügt H dem Zerlegbarkeitskriterium, falls für jedes paar zweier Knoten v_s und v_t aus H der kürzeste Pfad $P(v_s, v_t)$ zwischen ihnen in G nur aus Kanten aus H besteht.*

Bemerkung 10. *(a) Sind zwei Teilgraphen G_1 und G_2 eines Graphen G nur durch eine Kante e verbunden, so genügen automatisch beide dem Zerlegbarkeitskriterium.*

(b) Erfüllt ein Teilgraph H eines Graphen das Zerlegbarkeitskriterium, so ist ein lokales r-SPC von H identisch mit einem globalen von H.

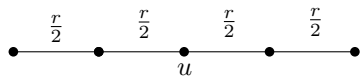
(c) Sei $H = (V', E')$ ein Teilgraph eines Graphen G, der das Zerlegbarkeitskriterium erfüllt und sei weiterhin C ein r-SPC von G, dann ist auch $C \cap V'$ ein r-SPC von H.

Beweis. Zu Aussage a: Ein Weg, der einen der beiden Teilgraphen verlässt und wieder betritt, kann nie ein eindeutiger Kürzester sein, da er zweimal die Kante e enthält und somit auch einen nicht-negativen Kreis.

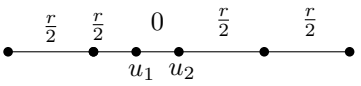
Aussage b ergibt sich unmittelbar daraus, dass der kürzeste Weg zwischen zwei Knoten aus H – gemäß dem Zerlegbarkeitskriterium – vollständig in H verläuft. Somit muss ein lokales SPC C von H diesen auch abdecken. Damit ist C auch ein globales SPC von H.

Es wird nun Aussage c bewiesen: Sei w ein beliebiger r-bedürftiger Weg zwischen zwei Knoten in H, dann muss C mindestens einen Knoten v aus w enthalten. Da w komplett in H verläuft, muss v in H sein und somit auch in $C \cap V'$. □

Mit Hilfe der Definition 9 lässt sich nun doch eine gewisse Zerlegbarkeit erzielen.



(a) Ungesplitteter Knoten u



(b) u gesplittet in u_1 und u_2

Lemma 11 (Zerlegungslemma)**.** *Sei $G = (V, E)$ ein* beliebiger *Graph, S ein Separator und $G'_1 = (V'_1, E'_1)$, beziehungsweise $G'_2 = (V'_2, E'_2)$ die durch S voneinander getrennten Teilgraphen. Sei $\{S_1, S_2\}$ eine Partitionierung von S, sodass sowohl der durch $G'_1 \cup S_1$ induzierte Subgraph G_1 von G, als auch der durch $G'_2 \cup S_2$ induzierte Subgraph G_2 von G jeweils zusammenhängend sind, dann gilt:*

(a) Sei C_1 ein r-SPC von G_1, C_2 ein r-SPC von G_2 und der Separator S in deren Vereinigung $C_1 \cup C_2$ enthalten, dann ist $C := C_1 \cup C_2$ ein r-SPC von G.

(b) Zusätzlich genüge sowohl G_1 als auch G_2 dem Zerlegbarkeitskriterium und C_1 und C_2 seien minimale r-SPCs, dann ist C ein minimales r-SPC von G.

Beweis. Zu a: C ist ein r-SPC. Sei w ein bedürftiger Weg. Falls w komplett in einem der beiden Teilgraphen G_1 oder G_2 verläuft, so muss er entweder durch C_1 oder durch C_2 abgedeckt sein. Falls w sich über beide Teilgraphen erstreckt, enthält er mindestens einen Knoten aus dem Separator S und wird damit ebenfalls durch C abgedeckt.

Zu b: Es muss die Minimalität von C nachgewiesen werden. Betrachte dazu ein minimales r-SPC C' von G und deren Partitionselement $C'_1 := C' \cap G_1$. Da sowohl C'_1 als auch C_1 beide r-SPCs von G_1 sind und C_1 als minimal gewählt war, muss gelten $|C_1| \leq |C'_1|$. Mit der gleichen Argumentation gilt: $|C_2| \leq |C' \cap G_2|$, also insgesamt: $|C'| = |C' \cap G_1| + |C' \cap G_2| \geq |C_1| + |C_2| = |C|$.

Ohne das Zerlegbarkeitskriterium könnte man hier nicht behaupten, dass C'_1 ein r-SPC von G_1 sei. Abbildung 2.3 zeigt ein Gegenbeispiel. Analoges gilt für C'_2. □

Die vielen starken Bedingungen, die für Fall b gelten müssen, lassen schon vermuten, dass in vielen Fällen eine Zerlegung nicht existiert oder nicht effizient gefunden werden kann. Bei einem nicht-zusammenhängenden Graphen genügt es, für jede einzelne Zusammenhangskomponente ein minimales r-SPC zu suchen und diese dann zu vereinigen. Der Separator ist die leere Menge. *Daher werden im Folgenden nur zusammenhängende Graphen betrachtet.*

Graphmodifikationen

Statt Graphen zu zerlegen, kann man auch auf die Idee kommen Modifikationen an diesen zu untersuchen, die ein minimales r-SPC oder dessen Größe nur auf kontrollierte Weise verändern oder sogar unverändert lassen. Eine Idee wäre beispielsweise das Zerlegen eines Knotens in zwei Knoten, die durch eine Kante mit Gewicht 0 verbunden sind und zusammen dieselben Nachbarschaftsbeziehungen haben, wie der Knoten, den sie ersetzen. In Abbildung 2.4a stellt die Menge $\{u\}$ ein minimales r-SPC dar. Möchte man den Knoten jedoch splitten, wie das in Abbildung 2.4b geschehen ist, benötigt ein minimales SPC mindestens zwei Knoten.

Auf ähnliche Weise sieht man ein, dass das Zusammenziehen mehrerer Kanten oder das Ersetzen einer Kante durch einen Kantenzug von mehreren Kanten die Größe eines minimalen SPCs verändern kann. Dieses wird dadurch jedoch nicht kleiner.

Satz 12 (Ersetzen von Kanten durch Kantenzüge)**.** *Gegeben ein Graph G. Sei $e := (v_s, v_t)$ eine Kante mit dem Gewicht w. Modifiziert man G zu G' dahingehend, dass e durch einen Kantenzug $(v_s, v_1, \ldots, v_n, v_t)$ ersetzt wird, wobei $\{v_1, \ldots, v_n\}$ neu hinzugefügte Knoten sind, und das Gesamtgewicht der neuen Kanten exakt gleich w ist, dann ist ein minimales r-SPC des modifizierten Graphen mindestens genauso groß, wie eines des ursprünglichen Graphen G.*

Beweis. Aus einer Lösung C' für G' kann man eine Lösung C für G erstellen, die höchstens genau so groß ist. Sei $B := \{w_1, \ldots, w_n\}, n \in \mathbb{N}$ die Menge aller r-bedürftigen Wege in G. Der Weg $w_i, i \in \{1, \ldots, n\}$ hat in G' eine (ebenso r-bedürftige) Entsprechung w_i'. Es gilt:

$$w_i' = \begin{cases} (v_a, \ldots, v_s, v_1, \ldots, v_n, v_t, \ldots, v_b) & \text{, falls } w_i = (v_a, \ldots, v_s, v_t, \ldots, v_b) \\ w_i & \text{, sonst} \end{cases}$$

Falls keines der $v_i, i \in \{1, \ldots, n\}$ in C' enthalten ist, so ist $C := C'$ eine gleich große Lösung für G. Andernfalls ist $C := C' \setminus \{v_1, \ldots, v_n\} \cup \{v_s\}$ eine höchstens gleich große Lösung für G, denn alle der $w_i', i \in \{1, \ldots, n\}$, die einen der Knoten $v_1, \ldots, v_n$ enthalten, müssen auch v_s enthalten. □

3. Komplexität

In diesem Kapitel sollen einige Erkenntnisse im Hinblick auf die Schwierigkeit des MIN-SPC-Problems gewonnen werden.

Satz 13. *MIN-SPC ist NP-schwer.*

Beweis. Vom Vertex-Cover-Problem ist die NP-Vollständigkeit bereits bekannt [Kar72] und es lässt sich auf MIN-SPC reduzieren. Einem Graphen G, für den ein minimales Vertex-Cover gefunden werden soll, wird mit einer Gewichtsfunktion für die Kanten ausgestattet, wonach jede Kante das Gewicht r hat ($r \geq 0$, beliebig). Das funktioniert in Polynomialzeit. Die r-SPCs dieses Graphen sind genau alle Teilmengen von V, in denen sich zu allen Kanten mindestens je ein inzidenter Knoten befindet. Somit ist ein r-SPC in diesem Graphen auch ein Vertex-Cover im Ausgangsgraphen und MIN-SPC ein NP-schweres Problem. □

Bemerkung 14. *Diese Reduktion funktioniert auch, wenn man die Eindeutigkeit der kürzesten Wege fordert.*

Beweis. Dazu werden die Kanten so mit ganzen Zahlen größer als r gewichtet, dass zwei verschiedene Wege auch immer verschiedene Längen aufweisen und jede Kante ein kürzester Weg ist. Sei nun r gleich 1. Weiterhin betrage zu jeder Kante $e_i, i \in \{1, \ldots, |E|\}$, das Gewicht $|e_i|$ von e_i exakt $1 + 2^{-i}$. Damit erreicht keine Kante das Gewicht 2 und jede Kante ist immer noch ein kürzester Weg. Liest man die Gewichte von Wegen in Binärdarstellung, so kann man anhand der Nachkommastellen die darin enthaltenen Kanten erkennen. Insbesondere gibt es keine zwei verschiedene Wege gleichen Gewichts. □

Das zugehörige Entscheidungsproblem, ob ein Graph G ein SPC der Größe k besitzt, liegt in NP, da sich die Korrektheit einer Lösung leicht in polynomieller Zeit nachprüfen lässt. Mit dem vorangegangenen Satz übertragen sich einige Eingenschaften des Vertex-Cover-Problems [PY88]. Darunter die APX-Vollständigkeit, die besagt, dass es keine beliebig gute Approximation der optimalen Lösung gibt, wenn P ungleich NP vorausgesetzt wird.

Korollar 15. *MIN-SPC ist APX-vollständig. Vielmehr kann eine Approximation nicht besser als der Faktor* $1,36$ *sein [DS05]. Für planare Graphen bleibt die NP-Vollständigkeit erhalten.*

Ob auch die APX-Vollständigkeit auf planaren Graphen erhalten bleibt, ist noch unklar, da dies für Vertex-Cover nicht gilt [Bak94].

Im Folgenden soll die Verwandtschaft zwischen den beiden Problemen MIN-SPC und MIN-HS erklärt werden, welche bereits in [ADF+11] untersucht und verwertet wird. Beide Probleme lassen sich auf das jeweils andere in Polynomialzeit reduzieren.

Satz 16. *MIN-SPC ist mit polynomiellen Zeitaufwand auf MIN-HS reduzierbar. Die Lösungsgröße bleibt dabei erhalten.*

Beweis. Aus einer Instanz des MIN-SPC-Problems lässt sich eine Instanz des MIN-HS-Problems generieren, sodass eine Lösung für diese exakt einer Lösung gleicher Größe für die Ausgangsinstanz entspricht. Da umgekehrt aber auch jedes SPC einer Instanz des MIN-SPC-Problems auch einem gleich großen Hitting Set für die korrespondierende MIN-HS-Instanz entspricht, überträgt sich die Eigenschaft der Minimalität zwischen den Lösungen.

Sei $G := (V, E)$ ein beliebiger Graph. Für G lässt sich – beispielsweise über einen All-Pair-Shortest-Path-Algorithmus – die Menge B aller r-bedürftigen Wege (hier jeweils als Menge aufgefasst) in G berechnen. Sei C nun ein Hitting Set von B, dann enthält C aus jeder der Mengen in B mindestens ein Element und ist somit aufgrund der Konstruktion von B auch ein r-SPC von G. Umgekehrt enthält ein r-SPC aufgrund der Definition von B aus jeder Menge in B mindestens ein Element und ist somit auch ein Hitting Set von B. □

Für die umgekehrte Richtung werden hier – um die Lösungsgröße nicht aufzublähen – gerichtete Kanten und nicht eindeutige kürzeste Wege erlaubt.

Satz 17. *MIN-HS ist in polynomieller Zeit auf MIN-SPC reduzierbar. Die Lösungsgröße bleibt dabei erhalten.*

Beweis. Im Folgenden wird eine Instanz von MIN-HS in eine Instanz von MIN-SPC transformiert. Jedem Element aus der MIN-HS-Instanz entspricht genau ein Knoten aus der MIN-SPC-Instanz, sodass eine Lösung der Größe k der MIN-HS-Instanz auch automatisch einer Lösung der Größe k der korrespondierenden MIN-SPC-Instanz entspricht und umgekehrt.

Für eine Grundmenge M seien die n Teilmengen $T_1, \ldots, T_n$ gegeben. Der Parameter r sei auf 2 festgelegt, sowie alle Kantengewichte in dem nun zu konstruierenden Graphen G auf 1. Die Idee ist es, dass Knoten, die für Mengen stehen, im Graphen mit Knoten, die für die einzelnen Elemente der Grundmenge stehen, so verbunden werden, dass entsprechende r-minimale Wege erzeugt werden, die abgedeckt werden müssen. Ein Element soll genau dann in einer Lösung des MIN-HS-Problems enthalten sein, wenn dessen Knoten in einer Lösung des MIN-SPC-Problems des entsprechenden Teilgraphen ist.

Der Graph ist aus drei Schichten aufgebaut. In der ersten Schicht ist für jede der einzelnen Teilmengen $T_i, i \in \{1, \ldots, n\}$ ein ihr zugeordneter *Mengenknoten* enthalten. In Schicht 2 ist für jedes Element aus M ein ihm zugeordneter *Elementknoten* enthalten. Ein Mengenknoten hat genau dann eine gerichtete Kante zu einem Elementknoten, wenn das zugehörige Element in der zugehörigen Menge ist. Weiterhin hat jeder Elementknoten eine gerichtete Kante zu jedem der $|M| + 1$ *Dummy-Knoten* aus Schicht 3. Diese sorgen dafür, dass für jede Menge, die mindestens ein Element enthält, r-bedürftige Wege entstehen, die von einem minimalen SPC abgedeckt werden müssen. Solche r-bedürftigen Wege enthalten aus jeder Schicht jeweils genau einen Knoten und enden in den Knoten aus Schicht 3. Falls eine

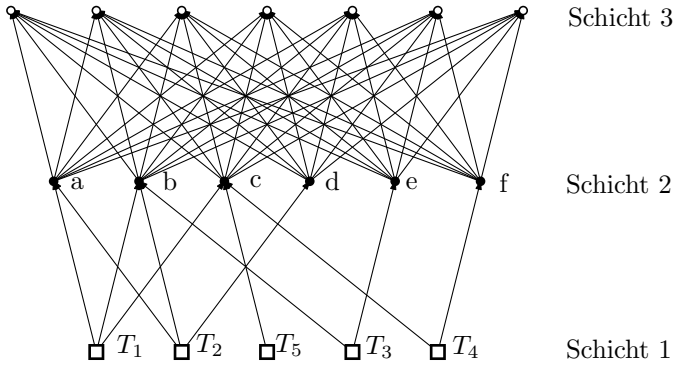


Abbildung 3.1: Beispiel für den einer Instanz des MIN-HS-Problems zugeordneten Graphen

Menge mehrere Elemente enthält, so genügt es – aus der Sicht des entsprechenden Mengenknotens – lediglich einen der benachbarten Elementknoten in ein SPC aufzunehmen, da es nach Definition 1 völlig ausreichend ist, zwischen zwei Knoten lediglich einen kürzesten, r-bedürftigen Weg abzudecken. Abbildung 3.1 zeigt ein Graph für die Probleminstanz $T_1 = \{a, b, c\}$, $T_2 = \{a, b, d\}$, $T_3 = \{b, e\}$, $T_4 = \{c, f\}$, $T_5 = \{c\}$ und $M = \{a, b, c, d, e, f\}$. Dieser illustriert beispielhaft die hier angewandte Konstruktion.

Ein minimales r-SPC, entspricht nur dann eine Lösung der Ausgangsinstanz des MIN-HS-Problems, wenn ausschließlich Elementknoten aus Schicht 2 in die Lösung aufgenommen wurden. Es gibt keinen Grund, weshalb sich ein Algorithmus für MIN-SPC daran halten sollte. Allerdings lässt sich eine Lösung, die auch Knoten aus Schicht 1 oder Schicht 3 enthält leicht in eine (gleich große) Lösung transformieren, die ausschließlich Knoten aus Schicht 2 enthält.

Sei C ein SPC der konstruierten Instanz. Angenommen C würde einen Knoten v_3 aus Schicht 3 enthalten, der als einziger einen Weg $w = (v_1, v_2, v_3)$ abdeckt, der wiederum der einzige abgedeckte, r-bedürftige Weg zwischen v_1 und v_3 ist. (Die Indizes geben die Schicht des Knotens an.) Diesen Weg muss es geben, denn andernfalls wäre das SPC nicht minimal. Daraus folgt jedoch, dass alle Wege der Form (v_1, v_2, v_j), v_j in Schicht 3 einzeln abgedeckt sein müssen, d.h., dass jeder der $|M| + 1$ Knoten aus Schicht 3 in dem SPC enthalten ist. Aber nun gibt es ein kleineres SPC, nämlich das, das einfach alle Elementknoten enthält. Nun soll angenommen werden, dass ein SPC Mengenknoten v_1 aus Schicht 1 enthält. In diesem Fall kann man das SPC zu einem höchstens gleich großem SPC transformieren, indem man einfach für jeden Mengenknoten, der aufgenommen ist, einen benachbarten Elementknoten aufnimmt. Dieser deckt wieder einen der kürzesten Wege zwischen den Knoten aus Schicht 3 und v_1 ab.

Ein minimales Hitting Set der Ausgangsinstanz setzt sich nun exakt aus den Elementen zusammen, deren Knoten in der obigen Graphkonstruktion in ein minimales r-SPC aufgenommen wurde, das lediglich Knoten aus Schicht 2 enthält. □

Da eine Lösung für das MIN-SPC-Problems des hier vorgestellten Graphen exakt genau so groß ist, wie eine Lösung der korrespondierenden MIN-HS-Instanz, übertragen sich einige Eigenschaften, die über das MIN-HS-Problem und über das verwandte MIN-SC-Problem bekannt sind [Fei98].

Korollar 18. *Das MIN-SPC Problem ist auf gerichteten Graphen, mit nicht eindeutigen kürzesten Wegen, nicht nur APX-schwer, sondern kann nicht besser als mit dem Faktor* $(1 - o(1)) \ln n$ *approximiert werden, sofern* P *ungleich* NP *ist.*

4. MIN-SPC und HD-Berechnung in einfachen Graphklassen

In diesem Kapitel werden Algorithmen für Wege, Bäume und Kreise vorgestellt, die minimale SPCs berechnen. Zu jeder der genannten Graphklassen wird die Highway Dimension betrachtet.

4.1 Wege

Ein Weg der Größe $n \in \mathbb{N}$ ist ein Graph $G = (V, E)$, der aus den Knoten $v_1, \ldots, v_n \in V$ und den Kanten $(v_1, v_2), \ldots, (v_{n-1}, v_n)$ besteht. Das Problem MIN-SPC lässt sich hier effizient lösen, indem eine Zerlegung wie in Lemma 11 gefunden wird.

Lemma 19 (Zerlegung in Weg-Graphen)**.** *Sei v_a ($a \in \mathbb{N}$) der erste Knoten, für den $|(v_1, \ldots, v_a)| \geq r$ ist. (Also $a := \min(\{t \in \{1, \ldots, n\} : |(v_1, \ldots, v_t)| \geq r\})$) Dann ist*

(a) $C_1 := \{v_a\}$ ein minimales r-SPC des durch $v_1, \ldots, v_a$ induzierten Subgraphen G_1 und

(b) $S = C_1$ ein Separator zwischen $G_1 \setminus \{v_a\}$ und $G_2 := G \setminus G_1$.

(c) G_1 und G_2 genügen jeweils dem Zerlegbarkeitskriterium nach Definition 9.

Beweis. Zu a: C_1 ist ein r-SPC. Angenommen es gibt einen nicht abgedeckten, bedürftigen Weg $P(v_i, v_j)$ in G_1, dann ist v_a nicht auf diesem Weg. Weil alle Kantengewichte jedoch nicht-negativ sind, gilt in jedem Fall $|P(v_1, v_{a-1})| \geq |P(v_i, v_j)| \geq r$. Das ist ein Widerspruch zu der Wahl von a.

C_1 ist minimal. Sei C_1' ein beliebiges anderes, minimales r-SPC von G_1, dann enthält C_1' mindestens einen der Knoten $v_1, \ldots, v_a$. (Andernfalls würde C_1' nicht den bedürftigen Weg $P(v_1, v_a)$ abdecken und wäre somit kein r-SPC.) Also enthält C_1' mindestens so viele Knoten wie C_1.

Zu b: Das ist trivial, da alle Wege von G_2 nach G_1 über den Knoten v_a führen.

Zu c: G_1 und G_2 sind jeweils zusammenhängend und alle Wege sind eindeutig, somit liegen auch die kürzesten Wege zwischen zwei Knoten dieser Teilgraphen vollständig in diesen. □

Diese Erkenntnis kann dazu genutzt werden, das Problem MIN-SPC auf Wegen in Linearzeit zu lösen.

Satz 20 (MIN-SPC in Wegen)**.** *Das Problem MIN-SPC ist auf Wegen mit linearem Zeitaufwand lösbar.*

Beweis. Zunächst wird ein Algorithmus angegeben, der dieses Problem optimal löst. Anschließend wird seine Laufzeit diskutiert.

Man sieht leicht ein, dass der durch $v_{a+1}, \ldots, v_n$ induzierte Subgraph G_2 aus dem vorangegangenen Lemma 19 wieder ein Weg-Graph ist, allerdings ein kleinerer als der Ausgangsgraph. Man kann nun rekursiv dieses kleinere Problem gleicher Art lösen, bis das Gesamtgewicht aller Kanten in einem verbleibenden Weg unter r gesunken ist, denn in dem Fall ist die leere Menge ein minimales SPC. Die Vereinigung der Teillösungen ist nach Lemma 11 (Zerlegungslemma) ein minimales SPC. Der im Folgenden beschriebene Algorithmus 1 realisiert dieses Verfahren.

Algorithmus 1: Weg-SPC(G, r)

Daten : Graph, $\mathbb{R}_{\geq 0}$
Ergebnis : 2^V
wenn *Gesamtweglänge* $< r$ **dann**
  zurück $\emptyset$;
Ende
sonst
  $a := \min(\{t \in \{1, \ldots, n\} : |(v_1, \ldots, v_t)| \geq r\})$;
  zurück $\{v_a\} \cup$ Weg-SPC$(G \setminus \{v_1, \ldots, v_a\}, r)$
Ende

Der Algorithmus geht den Weg einmal von vorne nach hinten durch und hat daher linearen Zeitaufwand. □

Nun folgt eine Aussage über die HD von Weg-Graphen.

Satz 21 (HD in Wegen)**.** *Ist G ein Weg, so beträgt seine HD höchstens* 8*. Diese Abschätzung ist scharf.*

Beweis. Zunächst wird gezeigt, dass die Highway Dimension kleiner oder gleich 8 ist. Im Anschluss wird – um die Schärfe nachzuweisen – ein Beispiel für einen Weg gegeben, dessen Highway Dimension genau 8 ist.

Eine obere Schranke für die HD von Weg-Graphen kann man finden, indem man – für beliebiges positives, reelles r – eine obere Schranke für ein lokales r-SPC einer Kugel vom Radius $4r$ findet. Siehe dazu die Definition der HD. Sei nun H eine solche Kugel. Sie ist selbst wieder ein Weg-Graph. Es ist zu zeigen, dass ein lokales r-SPC in H existiert, das höchstens acht Elemente enthält. Da in Weg-Graphen alle Wege eindeutig sind, erfüllt H das Zerlegbarkeitskriterium und nach Bemerkung 10 ist jedes lokale r-SPC in H damit auch

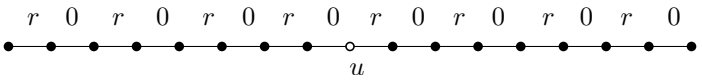


Abbildung 4.1: Weg mit HD 8

ein (globales) r-SPC von H. Die erste Aussage ist also bewiesen, wenn der Algorithmus 1 für einen beliebigen Weg-Graphen G von höchstens dem Radius $4r$ ein r-SPC höchstens der Größe 8 findet. Sei l die Länge des Weg-Graphen G. Sie beträgt – Aufgrund des Radius von G – höchstens $8r$. Falls l nicht kleiner als r ist, nimmt der obige Algorithmus einen Knoten in ein r-SPC des Restwegs auf, der höchstens die Länge $l - r$ hat. Dies kann maximal $l/r \leq 8r/r = 8$ Mal geschehen, womit die erste Aussage bewiesen ist.

Um die Schärfe nachzuweisen, wird der Weg-Graph aus Abbildung 4.1 angegeben, dessen HD genau 8 ist. Die $4r$-Kugel um den Knoten u ist exakt der gesamte Graph. Acht der Kanten haben mindestens das Gewicht r. Es muss also jede dieser Kanten zu mindestens einem Knoten in einem r-SPC inzident sein. Da diese jedoch paarweise knotendisjunkt sind, muss ein r-SPC nach Lemma 8b mindestens acht Knoten enthalten. □

4.2 Bäume

Ein *Baum* T ist ein Graph, in dem es zwischen je zwei Knoten genau einen Weg gibt. Dabei ist einer der Knoten als *Wurzel* ausgezeichnet. Welcher das ist, spielt im Nachfolgenden keine Rolle. Die *Blätter* sind die Knoten, die nicht die Wurzel sind und genau einen Nachbarn haben. Jeder Knoten v, der kein Blatt ist, ist ein Separator und gleichzeitig Wurzel eines zusammenhängenden *Unterbaums* $T(v)$.

Ziel ist es im Folgenden, ein minimales r-SPC für einen beliebigen Baum zu finden. Dazu wird zunächst das Lemma 11 (Zerlegungslemma) angewandt um dieses im Anschluss daran zur Konstruktion eines Algorithmus zu nutzen.

Lemma 22 (Zerlegung in Bäumen)**.** *Sei G ein Baum und v aus G ein so gewählter Knoten, dass $T(v)$ einen r-bedürftigen Weg enthält, aber kein echter Teilbaum von $T(v)$ dies tut. Weiterhin sei C' ein minimales r-SPC von $G \setminus T(v)$, dann ist $C := C' \cup \{v\}$ ein minimales r-SPC von G.*

Beweis. Es sind die Voraussetzungen vom Zerlegungslemma 11 nachzuweisen. Demnach muss $\{v\}$ ein minimales r-SPC von $G_1 := T(v)$ sein und gleichzeitig ein Separator von $G_1 \setminus \{v\}$ und $G_2 := G \setminus G_1$. Außerdem müssen sowohl G_1, als auch G_2 das Zerlegbarkeitskriterium erfüllen.

Die Menge $\{v\}$ ist ein minimales r-SPC von G_1. Sei w ein beliebiger Weg mit mindestens der Länge r in G_1. Dann muss v ein Teil des Wegs sein. Andernfalls gäbe es einen Knoten noch tieferen Levels, in dessen Unterbaum ein r-bedürftiger Pfad enthalten wäre. Also ist w abgedeckt.

Die Menge $\{v\}$ ist ein minimales r-SPC in G_1. Nach der Wahl von v gibt es in G_1 mindestens einen r-bedürftigen Weg w. Somit bildet $\{w\}$ eine Zeugenmenge und jedes minimale r-SPC in G_1 hat mindestens die Größe 1.

G_1 und G_2 erfüllen aufgrund der Eindeutigkeit der Wege und weil beide Teilgraphen zusammenhängend sind, das Zerlegbarkeitskriterium. □

Satz 23. *Das Problem MIN-SPC lässt sich auf Bäumen mit linearem Zeitaufwand lösen.*

Beweis. Zuerst wird ein Algorithmus angegeben, der ein minimales SPC findet. Anschließend wird über die Laufzeit diskutiert.

Die Idee des Algorithmus ist es, rekursiv für einen Baum einen Knoten v zu finden, der die Voraussetzung aus dem vorangegangenen Lemma 22 erfüllt und diesen mit einem minimalen r-SPC des Restgraphen $G \setminus T(v)$ zu vereinen. Wie findet man für einen Knoten v, heraus, ob er diese Voraussetzung erfüllt? Eine notwendige Bedingung ist, dass er auf einem r-bedürftigen Weg liegt. Dafür gibt es zwei Möglichkeiten.

(a) Es gibt ein Blatt b_l in $T(v)$, mit: $|(b_l, \ldots, v)| \geq r$.

(b) Es gibt zwei Blätter b_o, b_p in $T(v)$, mit: $|(b_o, \ldots, v, \ldots, b_p)| \geq r$.

Ob einer der beiden Fälle eintritt kann dann bestimmt werden, wenn v für jeden seiner k Nachfolger v_i den längsten Weg w_i in $T(v_i)$ kennt, der v_i als Endknoten hat. Die Länge des längsten Wegs w in $T(v)$ mit v als Endknoten ist dann das Maximum über alle $|w_i|+|(v_i, v)|$. Zu beachten ist dabei, dass man so ein $|w|$ auf diese Weise rekursiv berechnen kann. Analog kann die Länge w' des längsten Wegs in $T(v)$ bestimmt werden, der v als Endknoten hat und zu w kantendisjunkt ist. Ist $|w|$ größer oder gleich r, dann tritt Fall a ein. Ist $|w| + |w'|$ größer oder gleich r, so Fall b.

Die zweite notwendige Voraussetzung für einen solchen Knoten v kann sichergestellt werden, indem man sich von den einzelnen Blätter beginnend hocharbeitet und den ersten Knoten nimmt, für den einer der beiden oben genannten Fälle eintritt. In der nachfolgenden Pseudocode-Implementierung (Algorithmus 2) simulieren die Knoten v, für die das erstmals zutrifft, dass sie in $T(v)$ einen r-bedürftigen Weg beherbergen, über ihre Rückgabe an den sie aufrufenden Vaterknoten ein Abschneiden von $T(v)$ vom Restgraph. Die Implementierung wird mit dem Wurzelknoten aufgerufen. Die zurückgegebene Menge ist ein minimales r-SPC.

Die Laufzeit des Algorithmus liegt in $O(n)$. Jeder Knoten wird einmal aufgerufen führt für jeden seiner Nachbarn eine konstante Anzahl an Rechnungen durch. Damit wird jede Kante im Graphen höchstens zwei Mal besucht. Die Laufzeit ergibt sich dann daraus, dass die Anzahl der Kanten ebenfalls in $O(n)$ liegt. □

Damit ist das Problem MIN-SPC für Bäume gelöst und es folgen Aussagen über die HD eines Baums.

Satz 24. *Zu jeder beliebigen natürlichen Zahl h gibt es einen Baum, der die HD h hat – oder anders gesagt – es gibt keine feste obere Schränke bezüglich der HD von Bäumen.*

Beweis. Betrachtet man die Definition 2 der HD, kann man die Aussage beweisen, indem man für jede natürliche Zahl h und jede reelle, positive Zahl r einen Baum G angibt, in dem eine Kugel von höchstens dem Radius $4r$ existiert, deren lokales r-SPC mindestens h Knoten enthält.

Der Baum wird folgendermaßen konstruiert. Es gibt eine Wurzel s mit h Nachfolgern $v_1, \ldots, v_h$. Jeder dieser Nachfolger v_i hat jeweils einen weiteren Nachfolger v_i'. Alle Kantengewichte sind auf r festgelegt. Die Abbildung 4.2 illustriert den Graphen. Die Menge

Algorithmus 2: Baum-SPC-rek(G, r, v)

Daten : Graph, $\mathbb{R}_{\geq 0}$, Knoten in G
Ergebnis : $2^V \times \mathbb{R}_{\geq 0}$
// Vorausgesetzte Funktionen:

anc: $V \to V$ bildet einen Knoten auf seinem Vater im Baum ab.

des: $V \to 2^V$ bildet einen Knoten auf die Menge seiner Nachfolger ab.

wenn v *ist Blatt in* G **dann**
  zurück $(\emptyset,\ |(v, \mathrm{anc}(v))|)$;
Ende
sonst
  $\{v_1, \ldots, v_k\} := \mathrm{des}(v)$;
  für $i = 1 \ldots k$ **tue**
    $(s_i, w_i) :=$ Baum-SPC-rek(G, r, v_i);
  Ende
  $m_1 := \max(\{w_1, \ldots, w_k\})$;
  $m_2 := \begin{cases} 0 & \text{falls } k = 1 \\ \max(\{w_1, \ldots, w_k\} \setminus \{m_1\}) & \text{sonst} \end{cases}$;
  wenn $m_1 + m_2 \geq r$ **dann** v befindet sich auf einem bedürftigen, noch nicht abgedeckten Weg mit mindestens der Länge r
    zurück $(\{v\} \cup \bigcup_{i \in \{1,\ldots,k\}} s_i,\ 0)$;
    // v wird in das SPC mit aufgenommen. Die 0 entspricht dem Abschneiden des Baums $T(v)$ vom Restgraphen.
  Ende
  sonst
    zurück $(\bigcup_{i \in \{1,\ldots,k\}} s_i,\ |(v, \mathrm{anc}(v))| + m_1)$;
    // Das zweite Element im Tupel ist nun wieder ein längster Pfad von einem Blatt in $T(\mathrm{anc}(v))$ zum Knoten $\mathrm{anc}(v)$
  Ende
Ende

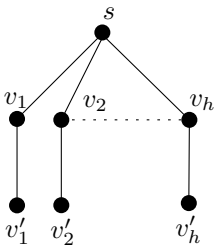


Abbildung 4.2: Baum mit beliebigen Ausgangsgraden

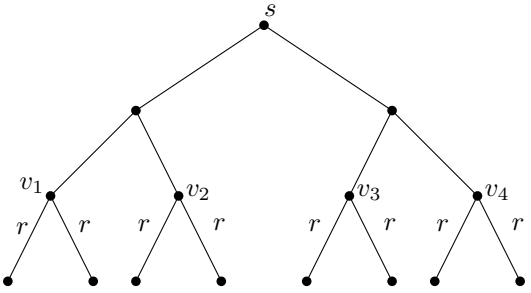


Abbildung 4.3: Baum mit beliebig kleinen, nicht-negativen Kantengewichten

$\{(v_1, v'_1), \ldots, (v_h, v'_h)$ ist eine Zeugenmenge der Größe h. Gemäß Lemma 8b hat ein minimales r-SPC dieses Graphen mindestens die Größe h und nach der Definition der HD gilt das Gleiche für sie. Der Radius des Baums ist sogar kleiner als $4r$. Jeder Knoten kommt über höchstens 2 Kanten mit dem Gesamtgewicht $2r$ zur Wurzel s. Daher hat der zwischen zwei Knoten längst mögliche Weg höchstens die Länge $4r$. □

Im Beweis zum vorangegangenen Satz wurde ausgenutzt, dass ein Knoten über beliebig viele Nachbarn verfügen kann. Der nächste Satz zeigt jedoch, dass dies jedoch für seine Aussage nicht zwingen notwendig ist.

Satz 25. *Sei k eine beliebige natürlich Zahl, die größer oder gleich 3 ist. Zu jeder beliebigen natürlichen Zahl h gibt es einen Baum mit der HD h, in dem kein Knoten über mehr als k Nachbarn verfügt und keine Kante über das Gewicht 0.*

Beweis. Das Vorgehen ist analog zum letzten Satz, nur, dass für gegebene Parameter r und h ein Baum angegeben wird, in dem ein minimales r-SPC mindestens die Größe h hat und in dem kein Knoten über mehr als k Nachbarn verfügt.

Ohne Beschränkung der Allgemeinheit wird angenommen, dass k 3 ist und h größer gleich 2 und eine Zweierpotenz ist. Bei dem konstruierten Graphen handelt es sich um einen vollständigen Binärbaum mit $2h$ Blättern. Die Tiefe des Baums ist $k := \log_2(h) + 1$. Alle Kanten, die zu einem der Blätter inzident sind, haben dabei das Gewicht r, alle andern Kanten jeweils ein beliebiges, positives Gewicht kleiner oder gleich $r/(k-1)$. Die Abbildung 4.3 zeigt eine Instanz des hier beschriebenen Graphen für den Parameter $h = 4$. Seien nun $v_1, \ldots, v_h$ die Knoten auf der vorletzten Ebene, also diejenigen, die zu einem Blatt inzident sind. Zu jedem dieser v_i sei v'_i einer der beiden benachbarter Blattknoten, beliebig gewählt. Dann ist $\{(v_1, v'_1), \ldots, (v_h, v'_h)$ eine Zeugenmenge der Größe h. Folglich ist h die Mindestgröße eines r-SPCs und damit auch die Mindestgröße der HD.

Es bleibt zu zeigen, dass der Graph einen Radius kleiner oder gleich $4r$ hat. Von einem beliebigen der Knoten $v_1, \ldots, v_h$ aus enthält der Weg zur Wurzel s $k-1$ Kanten. Er ist daher

höchstens $(k-1) \cdot r/(k-1) = r$) lang. Dadurch ist der Weg von einem beliebigen Blatt zur Wurzel höchstens $r + r = 2r$ lang und der Weg zwischen zwei beliebigen Knoten höchstens $4r$. Insbesondere ist dadurch der Radius kleiner als $4r$ (nämlich höchstens $2r$). □

Eine feste, obere Schranke für die HD von Bäumen existiert also nicht, selbst wenn man den maximalen Knotengrad auf 3 einschränkt und Kanten mit Gewicht 0 verbietet. Eine Abschätzung nach oben in Abhängigkeit von n ist leicht möglich. Im schlimmsten Fall muss für jede Kante ein zu ihr inzidenter Knoten in einem minimalen r-SPC S enthalten sein. Da Bäume höchstens $n-1$ Kanten enthalten, beträgt die Größe von S höchstens $\lceil (n-1)/2 \rceil$. Graphen, die lediglich aus einer Kante mit dem Gewicht r bestehen, erreichen diese Obergrenze.

4.3 Kreise

Ein Kreis-Graph $G = (V, E)$ der Größe $n \in \mathbb{N}$ bestehe aus den Knoten $v_0, \dots v_{n-1} \in V$ und den Kanten $(v_0, v_1), \dots, (v_{n-2}, v_{n-1}), (v_{n-1}, v_0)$, wobei die Indizes alle modulo n zu betrachten sind. Zwischen zwei verschiedenen Knoten gibt es genau zwei kantendisjunkte, einfache Wege. Sei w ein Weg zwischen den beiden Knoten v_s und v_t, dann enthält der zweite, dazu kantendisjunkte Weg w^c, im Folgenden *Komplementweg* genannt, exakt alle Kanten des Graphen, die nicht in w vorkommen. Dementsprechend kann sein Gewicht leicht berechnet werden, falls die Summe aller Kantengewichte $W := \sum_{e \in E}$ bekannt ist. Es beträgt $W - |w|$. Somit ist in konstanter Zeit überprüfbar, ob ein Weg, dessen Länge man kennt, ein Kürzester ist. Dies ist er genau dann, wenn sein Gewicht kleiner oder gleich $W/2$ ist. Interessiert man sich für ein minimales r-SPC kann daher davon ausgegangen werden, dass r kleiner oder gleich $W/2$ ist. Andernfalls gibt es zwischen allen Knoten einen Weg, der kürzer als r ist und das minimale SPC ist leer.

Im Folgenden ist es das Ziel in einem Graphen eine Knotenmenge $\{v_x\}$ unter der Annahme, dass v_x in irgend einem minimalen r-SPC enthalten ist, zu einem minimalen r-SPC zu ergänzen.

Satz 26. *Gegeben sei ein Kreis G und ein Knoten v_x in G, von dem die Mitgliedschaft in einem minimalen r-SPC bekannt ist, dann kann die Menge $\{v_x\}$ mit linearem Zeitaufwand zu einem r-SPC ergänzt werden.*

Beweis. Es wird ein Algorithmus angegeben, für den für jeden weiteren Knoten, den er zusätzlich zu v_x zu seiner Lösung hinzufügt, ein Zeuge aus einer Zeugenmenge existiert, der v_x nicht enthält. Die Minimalität kann dann mit Lemma 8b garantiert werden, denn nach Voraussetzung existiert ein minimales r-SPC C', das v_x enthält und C' muss ebenfalls von jeden Zeugen ein Knoten enthalten. Nach der Angabe des Algorithmus wird die Laufzeit beschrieben.

Der Algorithmus arbeitet sich – ohne Beschränkung der Allgemeinheit – im Uhrzeigersinn vor. Er sucht sich dabei immer r-bedürftige Wege $(v_s, \dots, v_t)$, die als Zeugen einer Zeugenmenge fungieren und nimmt für jeden Zeugen einen weiteren Knoten auf. Sei C das zu konstruierende r-SPC. Es werden folgende Invarianten benötigt:

(a) Kein Knoten aus der Menge $\{v_x, \dots, v_{s-1}\}$ (auf dem Kreis im Uhrzeigersinn gesehen) ist Teil eines durch C noch nicht abgedeckten Wegs.

(b) Für jeden von v_x verschiedenen Knoten in C gibt es in einer Zeugenmenge Z einen Zeugen, der v_x nicht enthält.

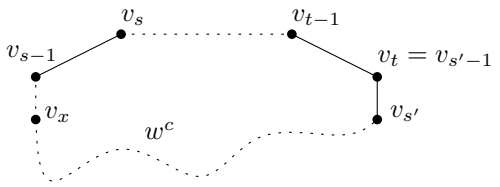


Abbildung 4.4: Illustration zum Satz 26.

Initial wird das durch die Belegungen $C \leftarrow \{v_x\}$, $Z \leftarrow \emptyset$ und $s \leftarrow v_{x+1}$ erfüllt.

Angenommen es gelten die Invarianten a und b. Sei v_t so gewählt, dass $w = (v_s, \ldots, v_t)$ mindestens die Länge r hat, der Teilweg $(v_s, \ldots, v_{t-1})$ jedoch nicht, dann wird zwischen zwei Fällen unterschieden. Entweder ist w ein kürzester Weg zwischen v_s und v_t oder nicht. Es wird im Folgenden der erste Fall betrachtet. Die Invariante b kann wieder wieder erhalten werden, indem v_t in C aufgenommen wird und w in Z aufgenommen wird. Dass Z wieder eine Zeugenmenge ist, ergibt sich daraus, dass w zwischen den Knoten v_s und v_t liegt, alle andern Zeugen aus Z jedoch zwischen v_x und v_{s-1} nach Voraussetzung Also ist w (paarweise) disjunkt zu allen andern Wegen aus Z und enthält auch nicht v_x. Sei $s' := t+1$, so erhält das die Invariante a für s' (anstelle von s). Es ist für einen beliebigen Knoten v zwischen v_s und $v_{s'-1}$ zu zeigen, dass er nicht Teil eines nicht abgedeckten, r-bedürftigen Wegs ist. Ein solcher wäre – nach Voraussetzung – abgedeckt, falls einer der Knoten zwischen v_x und v_{s-1} darin enthalten wäre. Ebenso wäre er abgedeckt, falls v_t enthalten wäre. Folglich können nur Knoten aus $\{v_s, \ldots, v_{t-1}\}$ darin enthalten sein. Abbildung 4.4 illustriert diese Situation. Aufgrund der Wahl von t ist der Weg $(v_s, \ldots, v_{t-1})$, der sogar alle diese Knoten enthält, kürzer als r und somit kein r-bedürftiger Weg. Offensichtlich gilt das für jeden weiteren Weg, der lediglich aus diesen Knoten beseht.

Im zweiten Fall wird lediglich $s' := s + 1$ das „neue“ s. Es wird kein weiterer Knoten zu C hinzugefügt. Damit ist der Erhalt der Invariante b trivial. Auch die Invariante a bleibt erhalten. In diesem Fall muss man sich nur davon überzeugen, dass v_s nicht Teil eines nicht abgedeckten, r-bedürftigen Wegs w_b ist. Angenommen so ein Weg w_b existiert. Der Knoten v_s kann auf keinen Fall ein mittlerer Knoten in w_b sein, denn dieser müsste dann auch v_{s-1} enthalten – ein Widerspruch zur Invariante. Also kann w_b höchstens noch von v_s aus in Richtung des Uhrzeigersinns verlaufen. Jedoch kann keiner der Knoten aus $\{v_t, \ldots, v_x\}$ ein Endknoten von w_b sein, denn $(v_s, \ldots, v_t)$ ist kein kürzester Weg und damit gilt das auch für alle andern Wege, die zwischen v_s und einem der Knoten aus $\{v_t, \ldots, v_x\}$ im Uhrzeigersinn verlaufen. Der Komplementweg muss jeweils kürzer sein. Folglich kann w_b nur noch Knoten aus $\{v_s, \ldots, v_{t-1}\}$ enthalten. Analog zum ersten Fall oben ist w_b damit zwangsweise kürzer als r und somit nicht r-bedürftig.

Das ist schon fast die komplette Beschreibung des Algorithmus. Er beginnt mit der oben genannten Initialisierung von C und s, wählt ein t – nach obigen Voraussetzungen – und erhält im Anschluss, je nach Fall, die Invarianten a und b. Siehe dazu Algorithmus 3. Sobald der Weg im Uhrzeigersinn zwischen v_s und v_x kürzer als r ist, kann abgebrochen werden und die Invarianten sind für den gesamten Graph erfüllt. Die Menge C ist damit ein minimales r-SPC. Da s in jedem der Fälle um mindestens den Wert 1 erhöht wird, ist der Algorithmus nach endlich vielen Schritten fertig. Genauer ist der Zeitaufwand linear. Das Gesamtgewicht aller Kanten kann in Linearzeit berechnet werden. Die Werte s und t werden jeweils höchstens n mal erhöht. Die Länge des betrachteten Wegs kann in einer Variablen d gespeichert werden, die bei jeder Modifikation von s oder t angepasst wird. □

Algorithmus 3: Kreis-SPC-Erg(G, r, v_x)

Daten : Graph, $\mathbb{R}_{\geq 0}$, Zwei Knoten aus G
Ergebnis : 2^V
// Vorbereitungen
$W := \sum_{e \in E} |e|$; // Gesamtgewicht aller Kanten
$C \leftarrow \{v_x\}$;
$s \leftarrow x + 1$; // Index des Startknotens des betrachteten Segments
$t \leftarrow s$; // Index des Endknotens des betrachteten Segments
$d \leftarrow 0$; // Länge des Wegs $(v_s, ..., v_t)$
solange $v_t \neq v_x$ **tue**
  $d \leftarrow d + |(v_t, v_{t+1})|$;
  $t \leftarrow t + 1$;
  solange $d \geq r$ **tue**
    wenn $d \leq W - d$ **dann** Der Weg zwischen v_s und v_t ist ein kürzester Weg
      $C \leftarrow C \cup \{v_t\}$;
      $s \leftarrow t + 1$;
      $t \leftarrow t + 1$;
      $d \leftarrow 0$;
    Ende
    sonst Der Komplementweg ist kürzer
      $d \leftarrow d - |(v_s, v_{s+1})|$;
      $s \leftarrow s + 1$;
    Ende
  Ende
Ende
zurück C

Der vorangegangene Satz funktioniert ausschließlich, falls von einem Knoten die Mitgliedschaft in einem minimalen r-SPC bekannt ist. Daher stellt sich die Frage, ob auf diese Voraussetzung verzichtet werden kann, ohne Effizienzeinbußen bei der Berechnung eines minimalen r-SPC hinnehmen zu müssen. Für die nachfolgenden Betrachtungen werden zwei Einschränkungen für Kreise eingeführt. Die erste ist, dass es keine Kante gibt, deren Gewicht größer oder gleich r ist. Gäbe es eine solche Kante und sie wäre kein kürzester Weg zwischen den beiden Knoten, die sie verbindet, so kann sie aus dem Kreis entfernt werden, ohne Auswirkungen auf Lösungen des MIN-SPC Problems. Es gäbe keinen r-bedürftiger Weg, der sie enthält. Der resultierende Graph wäre ein Weg und für solche ist mit dem besprochenen Algorithmus 1 ein effizientes Verfahren bekannt. Wäre sie ein kürzester Weg, so könnte der vorangegangene Algorithmus 3 jeweils einmal mit ihren beiden Endknoten aufgerufen werden, da einer von beiden dann Teil einer optimalen Lösung sein muss. Das Gesamtresultat wäre das kleinere der beiden erhaltenen r-SPCs.

Die zweite Einschränkung betrifft das Auftreten von kürzesten Komplementwegen, wie es bei der Wahl der betrachteten Wege im vorangegangenen Algorithmus geschehen ist. Diesen speziellen Komplementwegen soll nun ein Name gegeben werden.

Definition 27 (Lange Komplementwege)**.** *Gegeben ein Kreis G und ein Weg $w = (v_s, \ldots, v_t)$ heißt w^c langer Komplementweg, falls $|w|$ mindestens die Länge r hat, $(v_s, \ldots, v_{t-1})$ jedoch nicht. (Wege im Uhrzeigersinn gesehen.)*

Sie können – unter der oben begründeten Annahme, dass jede Kante ein Gewicht kleiner als r hat – für viele Kreise ausgeschlossen werden. Das ist der Gegenstand des folgenden Lemmas 28.

Lemma 28. *Sei G ein Kreis, das Gewicht jeder Kante kleiner als r, W das Gesamtgewicht aller Kanten und r kleiner oder gleich $W/4$, dann sind lange Komplementwege keine kürzesten Wege.*

Beweis. Sei nun w^c ein solcher langer Komplementweg zu einem Weg $w = (v_s, \ldots, v_t)$. Das bedeutet gemäß der vorangegangenen Definition, dass $w' = (v_s, \ldots, v_{t-1})$ kürzer als r ist, w jedoch nicht. Der Beweis erfolgt durch Widerspruch.

Angenommen w^c sei ein kürzester Weg, dann trifft dies nicht für w zu. Es soll gezeigt werden, dass dann r größer als $W/4$ sein muss. Da w kein kürzester Weg ist, ist sein Gewicht größer als $W/2$. Mit $|w'| < r$ und $|(v_{t-1}, v_t)| < r$ folgt unmittelbar $2r > r + |(v_{t-1}, v_t)| > |w'| + |(v_{t-1}, v_t)| = |w| > W/2$ und daraus durch Umformung $r > W/4$. □

Generell kann diese Einschränkung nicht gemacht werden. Viel mehr kann man an dem Beispielgraph 4.5 sehen, dass es sogar mehrfach geschehen kann, dass ein langer Komplementweg ein kürzester Weg ist. Das Gesamtgewicht W aller Kanten beträgt exakt 100. Der Parameter r sei mit 40 festgelegt. Von w_1 (im Uhrzeigersinn) ausgehend ist der erste Weg, der die Länge r überschreitet der Weg $w_1 \cdot k_1 \cdot w_2 \cdot k_2$ mit einer Gesamtlänge von 59. Sein Komplementweg ist kürzer. Von w_2 aus ist der erste Weg, der die Länge r überschreitet der Weg $w_2 \cdot k_2 \cdot w_3 \cdot k_3$ mit einer Gesamtlänge von 51. Auch hier ist der Komplementweg kürzer.

Es wurden für den nachfolgenden Teil sowohl Kanten mit einem Gewicht größer oder gleich r, als auch das Auftreten von langen Komplementwegen, die kürzeste Wege sind, ausgeschlossen. Diese Einschränkungen sind stark genug, um für einen Kreis in Linearzeit ein

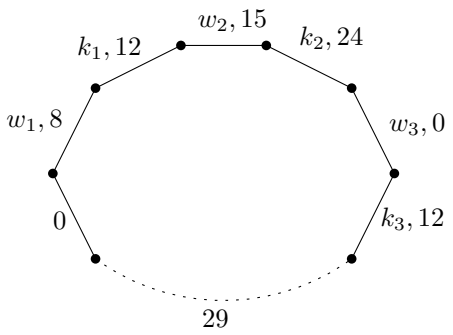


Abbildung 4.5: Zwei kürzere Komplementwege im Kreis

r-SPC zu berechnen. Um im Beweis zu dieser Aussage tatsächlich den linearen Zeitaufwand nachweisen zu können, wird das nachfolgende Lemma benötigt. Es ermöglicht es in bestimmten Situationen, im direkten Vergleich zwischen zwei Knotenteilmengen, die beide zu möglichst kleinen SPCs erweitert werden sollen, eine der beiden zu verwerfen, weil eine optimale Lösung mit der anderen nicht schlechter ist.

Lemma 29 (Konkurrierende r-SPC)**.** *Gegeben sei ein Kreis G und zwei Knotenteilmengen $C_1 = \{v_1, v_2\}$ und $C_2 = \{u_1, u_2\}$, sodass es keinen in G r-bedürftigen Weg gibt, der Knoten aus $M_1 := \{v_1, \ldots, v_2\}$ enthält und nicht durch C_1 abgedeckt ist. Analoges soll für $M_2 := \{u_1, \ldots, u_2\}$ mit C_2 gelten. Sei zusätzlich M_1 eine Teilmenge von M_2, dann ist eine kleinstmögliche Ergänzung von M_1 zu einem r-SPC von G nicht größer, als eine kleinstmögliche Ergänzung von M_2 zu einem r-SPC.*

Beweis. Seien $\tilde{C_1}$ und $\tilde{C_2}$ kleinstmögliche Ergänzungen von C_1, bzw. C_2 zu r-SPCs. Es muss gezeigt werden, zeigen, dass $|C_1 \cup \tilde{C_1}|$ kleiner oder gleich $|C_2 \cup \tilde{C_2}|$ ist. Da C_1 genauso groß wie C_2 ist ist das Äquivalent mit der Aussage, dass $\tilde{C_1}$ nicht mehr Elemente als $\tilde{C_2}$ enthält. Dies kann man nachweisen, indem man zeigt, dass auch $\tilde{C_2}$ eine Ergänzung von C_1 zu einem r-SPC des Gesamtgraphen ist. Sei nun also w ein r-minimaler Weg in G. Es wird zwischen drei Fällen unterschieden.

(a) Der Weg w enthält nur Knoten aus M_1. In diesem Fall wird w durch einen Knoten aus C_1 abgedeckt.

(b) Der Weg w verläuft vollständig in $G \setminus M_1$. Da dieser Graph ein Teilgraph von $G \setminus M_2$ ist, muss er durch einen Knoten aus $\tilde{C_2}$ abgedeckt sein.

(c) Der Weg w enthält sowohl Knoten aus M_1, als auch Knoten aus $G \setminus M_1$. Folglich muss er einen der beiden Knoten aus C_1 enthalten und ist damit abgedeckt.

Also wäre auch $\tilde{C_2}$ eine Ergänzung von C_1 zu einem r-SPC und die Aussage ist bewiesen. □

Satz 30. *Sei G ein Kreis und r eine reelle, positive Zahl. Falls jede Kante in G ein geringeres Gewicht als r aufweist und es keinen langen Komplementweg gibt, der ein kürzester Weg ist, so kann MIN-SPC auf G in linearer Zeit gelöst werden.*

Beweis. Die Idee ist es, den Algorithmus 3 zu verschlanken und für alle möglichen Startknoten v_1 bis v_n zu parallelisieren, sodass ein minimales r-SPC in Linearzeit berechnet werden kann.

Der neue Algorithmus 4 arbeitet mit derselben Strategie, wie der Algorithmus 3 aus dem Satz 26. Es werden also simultan für alle v_x aus $\{v_1, \ldots, v_n\}$ minimale Ergänzungen C_x zu SPCs bestimmt und ein Kleinstes davon als Lösung ausgegeben. Dabei werden wieder Wege der Form $(v_s, \ldots, v_t)$ betrachtet, nur immer n davon gleichzeitig – je einer für jedes x aus $\{1, \ldots, n\}$.

Offenbar ist es nicht für jedes x einen Zähler d_x zu führen. Es müssten in jedem Schritt des Algorithmus alle n Zähler aktualisiert werden, was die Laufzeit beeinträchtigen würde. Daher ist es sinnvoller für jedes der r-SPCs die Differenz Δ_x zu einem globalen Wert d. Somit ergibt sich als Schleifeninvariante, dass für alle x $d + \Delta_x =$ die Länge der seit dem letzten Knoten in C_x nicht abgedeckten Strecke ist. Diese kann durchaus negativ sein, falls der Algorithmus den ersten Knoten v_x in C_x noch nicht erreicht hat. Zu Beginn ist das für fast alle x der Fall. Die Indizes x aus $\{1, \ldots, n\}$ sollen absteigend nach dem jeweils aktuellen Wert von $d + \Delta_x$ in einer Warteschlange „queue“ sortiert sein.

Bemerkung 31. *Nach dem Vergrößern des Zählers d um die nächste Kante gibt es folgende Szenarien:*

1. *Für das SPC mit der längsten, nicht abgedeckten Strecke seit dessen zuletzt hinzugefügtem Knoten gilt, dass diese die Länge r nicht überschreitet. In diesem Fall gibt es nichts zu tun, denn bei allen anderen SPCs besteht dadurch ebenfalls kein Handlungsbedarf.*

2. *Nur für das SPC mit der längsten, bisher nicht abgedeckten Strecke (seit dessen zuletzt hinzugefügtem Mitglied) gilt, dass sie die Länge r überschreitet. Damit besteht nur bei diesem Handlungsbedarf. Der Zielknoten der zuletzt berücksichtigten Kante wird zu ihm hinzugefügt und seine Δ-Differenz so angepasst, dass die Schleifeninvariante wieder erfüllt wird.*

3. *Es besteht bei den den ersten k Elementen in „queue“ zugehörigen SPCs Handlungsbedarf. Nach Lemma 29 genügt es allein das SPC von den ersten k zu betrachten, dessen zuletzt aufgenommener Knoten am weitesten zurückliegt.*

□

Im Algorithmus 4 ist das Nichtauftreten von kürzesten Komplementwegen deshalb wichtig, weil andernfalls die Warteschlange nicht so leicht sortiert gehalten werden kann. Im schlimmsten Fall müsste man sie bei jedem Auftreten einmal vollständig durchlaufen. Jedoch lässt sich das – auf Kosten der Effizienz – beherrschen.

Satz 32. *Das Problem MIN-SPC lässt sich für Kreise in $O(n \log n)$ lösen.*

Beweis. Man kann im Algorithmus 4 des vorangegangenen Satzes die Warteschlange durch eine Prioritätenliste ersetzen, die den Wert von Schlüsseln in $O(\log n)$ sowohl verringern, als auch erhöhen kann. □

Damit ist das Problem MIN-SPC für Kreise relativ effizient gelöst.

Im Folgenden geht es um die HD eines Kreises. Hier gilt der nachfolgende Satz:

Satz 33 (HD in Kreisen)**.** *Die HD eines Kreises ist höchstens* 8*. Diese Abschätzung ist scharf.*

Algorithmus 4: Kreis-SPC-KKK(G, r)

Daten : Graph, $\mathbb{R}_{\geq 0}$
Ergebnis : 2^V
$W := \sum_{e \in E} |e|$; `// Gesamtgewicht aller Kanten`
wenn $r > W/2$ **dann** Alle kürzesten Wege sind kürzer als r
  zurück $\emptyset$;
Ende
sonst
  `// Vorbereitungen`
  $d \leftarrow 0$;
  queue $\leftarrow [1, \ldots, n]$; `// Absteigend nach` Δ_t `sortierte Warteschlange (siehe oben), mit allen Indizes, für deren Knoten noch das SPC berechnet wird.`
  $i \leftarrow 1$; `// Knotennummer des aktuellen Zielknotens, modulo` n `gesehen.`
  für alle $t \in \{1, \ldots, n\}$ **tue**
    $\text{spc}_t \leftarrow \{v_t\}$;
    $\Delta_t \leftarrow -|(v_n, v_1, \ldots, v_t)|$;
    $s_t \leftarrow 0$; `// Größe von` spc_t
  Ende
  `// Vorbereitung Ende`
  solange *queue ist nicht leer* **tue**
    wenn $v_i \in$ *queue* $\wedge$ v_i *wurde zweimal besucht* **dann**
      queue.del(i) ; `// Kreis für diesen Knoten geschlossen. Sein SPC wird nicht weiter bearbeitet.`
    Ende
    $d \leftarrow d + |(v_{i-1}, v_i)|$; `// Betrachte zuletzt betretene Kante` (v_{i-1}, v_i)
    $k \leftarrow$ queue.first;
    wenn $d + \Delta_k \geq r$ **dann** Wir sind jetzt in Bemerkung 31.2 oder 31.3
      $\text{spc}_k \leftarrow \text{spc}_k \cup \{v_i\}$;
      $s_t \leftarrow s_t + 1$;
      $\Delta_k \leftarrow -d - |(v_i, v_{i+1})|$;
      queue.popFront;
      queue.pushBack(k);
      solange $d + \Delta_{\textit{queue.first}} \geq r$ **tue** Betrachte überflüssige SPCs nicht mehr. Siehe dazu Bemerkung 31.3. Das spc_k ist aufgrund der Sortierung von queue das SPC, dessen letzter Knoten am weitesten zurückliegt. Durch das Aussortieren an dieser Stelle bleibt die lineare Laufzeitschranke erhalten.
        $s_{\text{queue.first}} \leftarrow \infty$;
        queue.popFront;
      Ende
    Ende
    $i \leftarrow i + 1$;
  Ende
  zurück SPC mit kleinsten Größenzähler s;
Ende

Beweis. Bei der Betrachtung einer Kugel $B(u, 4r)$ eines beliebigen Punktes u zu einem beliebigen Parameter r gibt es zwei Fälle.

(a) Mindestens eine Kante des Gesamtgraphen befindet sich nicht in der Kugel. Dann ist die Kugel ein Weg-Graph und wir wissen, dass ein lokales r-SPC hier maximal acht Knoten enthalten kann.

(b) Alle Kanten des Graphen befinden sich in der Kugel. Daraus folgt unmittelbar $8r \geq \sum_{e \in E} |e|$.) Auch hier gilt wieder: HD ≤ 8.

Aufgrund des Falls a ist somit klar, dass ein Kreis die HD 8 erreichen kann, denn für Wege wurde das bereits im Satz 21 nachgewiesen.

Es soll hier noch für Fall b die obere Schranke nachgewiesen werden. Das funktioniert so ähnlich, wie beim Nachweis dieser Schranke für Wege. Sei v_x ein Knoten, sodass (v_x, v_{x+1}) ein positives Kantengewicht hat und C ein zu konstruierendes r-SPC, das initial genau v_x enthält. Der Algorithmus 3, mit dem Startknoten v_x aufgerufen, findet eine Lösung, die maximal acht Knoten enthält. Jeder zu Beginn noch nicht abgedeckte, r-bedürftige Weg ist Teil des Wegs $(v_{x+1}, \ldots, v_{x-1})$, dessen Gesamtlänge aufgrund der Wahl von v_x geringer als $8r$ ist. Nach der Aufnahme des nächsten Knotens v_y in seine Lösung, ist jeder noch nicht abgedeckte, r-bedürftige Weg Teil des Wegs $(v_y, \ldots, v_{x-1})$, dessen Länge geringer als $7r$ ist usw. Insgesamt nimmt der Algorithmus neben v_x höchstens sieben weitere Knoten in ein minimales r-SPC auf. □

5. Approximationen für andere Graphklassen

In diesem Abschnitt werden für allgemeinere als die drei Graphklassen des vorangegangenen Kapitels Lösungen gesucht. Dabei handelt es sich jedoch lediglich um Approximationen an Lösungen des MIN-SPC-Problems.

5.1 Außenplanare Graphen

Ein *planarer* Graph ist ein Graph, der in einer Zeichnung ohne Kantenüberschneidungen in eine zweidimensionale Ebene eingebettet werden kann. Die zugehörige Einbettung ist eine *planare Einbettung*. Eine *Facette* ist ein durch Kanten begrenztes Gebiet in einer planaren Einbettung. Der *Dualgraph* – bezüglich einer planaren Einbettung eines Graphen – ist der Graph, der für jede Facette einen Knoten enthält und Kanten zwischen zwei Knoten genau dann, wenn die entsprechenden Facetten benachbart sind, d.h. durch eine gemeinsame Kante voneinander abgegrenzt werden. Der *schwache Dualgraph* ist der Dualgraph, bei dem für die äußerste Facette kein Knoten enthalten ist. *Außenplanare* Graphen sind planare Graphen, die sich in eine zweidimensionale Ebene so einbetten lassen, dass alle Knoten auf der äußeren Facette liegen. Der einer solchen Einbettung zugehörige schwache Dualgraph ist ein Baum.

Der Einfachheit halber wird der betrachtete Graph jedoch zunächst weiter eingeschränkt.

5.1.1 Außenplanare Graphen mit einem Weg als schwachen Dualgraph

Ein solcher Graph ist außenplanar und zweifach zusammenhängend (d.h. ein Separator hat mindestens die Größe 2). Die Eigenschaft einen Weg als schwachen Dualgraph zu haben, wird als *Outerpath*-Eigenschaft bezeichnet. Im Folgenden wird ein Graph immer bezüglich einer planaren Einbettung betrachtet. Abbildung 5.1 zeigt einen Beispielgraphen. Alle Knotenpaare, die an eine gemeinsame Facette angrenzen, die nicht die Äußere ist, sind Separatoren. Als „Endfacetten“ werden im Folgenden die beiden Facetten bezeichnet, deren Dualknoten am Anfang/Ende des Dualgraphwegs liegen. Im Folgenden soll eine 2-Approximation an eine Lösung des MIN-SPC-Problems für diese Graphklasse gefunden werden.

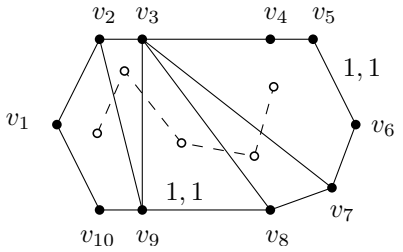


Abbildung 5.1: Beispiel für einen außenplanaren Graphen mit einem Weg als Dualgraphen, hier gestrichelt dargestellt.

Satz 34. *Für außenplanare Graphen, die die Outerpath-Eigenschaft erfüllen, kann in $O(n^4)$ eine 2-Approximation gefunden werden. Dieser Wert ist für den im Beweis besprochenen Algorithmus scharf.*

Beweis. Der Beweis geschieht in drei Schritten. Zunächst wird der Algorithmus und seine Laufzeit beschrieben, dann nachgewiesen, dass er 2-Approximationen berechnet und im Anschluss ein Beispiel gezeigt, in dem diese Abschätzung scharf ist.

Der Algorithmus beginnt mit der Berechnung von *toCover*, der Menge aller r-minimalen Wege im Graphen. Nach Lemma 8c ist eine Knotenmenge genau dann ein r-SPC des Graphen, wenn sie *toCover* abdeckt. Die Struktur der hier behandelten Graphen ermöglicht es, sich an Separatoren „entlangzuhangeln“. Dabei wird mit zwei auf der äußeren Facette benachbarten Knoten begonnen. Um den jeweils nächsten Separator zu bekommen, wird nach und nach immer je einer von beiden so durch einen unberührten Nachbarn auf der äußeren Facette ausgetauscht, dass wieder ein Separator entsteht. Jeder Knoten ist mindestens einmal Teil eines Separators. Im Verlauf des Algorithmus wird zwischen dem bereits untersuchten Teilgraphen und dem Restgraphen unterschieden. Begonnen wird an einer der beiden Endfacetten mit zwei benachbarten Knoten. Der Algorithmus arbeitet sich nun sukzessiv, entlang des Dualwegs, an Separatoren in Richtung der anderen Endfacette vor. Man betrachtet nach jedem Erweiterungsschritt den bereits untersuchten Graphen. Sobald sich ein noch nicht abgedeckter, r-minimaler Pfad im untersuchten Teil befindet, werden beide Separatorknoten in ein r-SPC mit aufgenommen. Der Algorithmus 5 geht genau so vor.

Der benötigte Zeitaufwand liegt in $O(n^4)$. Die Berechnung der Menge *toCover* kann beispielsweise mit dem Algorithmus von Floyd und Warshall durchgeführt werden, dessen Zeitaufwand in $O(n^3)$ liegt. Die äußere Schleife wird maximal n mal durchlaufen. Der Test nach einem r-minimalen, noch nicht abgedeckten Weg in ihrem Rumpf kann durch einfaches Nachschlagen in $O(n^3)$ realisiert werden. Das Entfernen aller durch das nächste aufgenommene Knotenpaar abgedeckten Wege aus *toCover* ist in Linearzeit zu bewerkstelligen.

Nach der Beschreibung des Algorithmus samt Laufzeit soll nun die Korrektheit nachgewiesen werden. Es sind dabei zwei Aussagen zu zeigen. Zum einen muss es sich bei dem Resultat um ein r-SPC handeln, zum andern darf es höchstens um den Faktor 2 von einer minimalen Lösung abweichen.

Der Algorithmus berechnet zu G ein r-SPC. Sei w ein r-minimaler Weg in G, dann sind zu einem bestimmten Zeitpunkt t während der Bearbeitung erstmals alle Knoten von w im untersuchten Teil und somit auch w selbst. Insbesondere muss einer der beiden in diesem Moment betrachteten Separatorknoten in w enthalten sein. Da beide im Endresultat enthalten sind, ist w abgedeckt.

Algorithmus 5: Outerpath-Aussenpl-SPC-2-Approx(G, r)

Daten : Graph, $\mathbb{R}_{\geq 0}$
Ergebnis : 2^V
`// Vorbereitung`
$scanned \leftarrow \emptyset$; `// Bisher untersuchter Teilgraph.`
$currentPart \leftarrow \emptyset$;
$toCover \leftarrow$ Alle minimalen, nicht-saturierten Pfade in G. ; `// Diese können z.B. über den bekannten Algorithmus von Floyd und Warshall berechnet werden.`
$spc \leftarrow \emptyset$
$a, b \leftarrow$ benachbartes Knotenpaar auf der linken Endfacette;
$currentPart \leftarrow \{a, b\}$; `// Aktuell betrachteter Teilgraph.`
`// Ende Vorbereitung`
solange $scanned \neq G$ **tue**
  wenn *Kein Pfad von toCover liegt vollständig in currentPart* **dann**
    $c \leftarrow$ noch nicht untersuchter Nachbar von a;
    wenn *c ist auf der gleichen Facette wie b* **dann**
      $a \leftarrow c$;
      $currentPart \leftarrow currentPart \cup \{a\}$;
    Ende
    sonst
      $b \leftarrow$ nächster noch nicht untersuchter Nachbar von b, der mit diesem eine gemeinsame Kante auf der äußeren Facette hat;
      $currentPart \leftarrow currentPart \cup \{b\}$;
    Ende
  Ende
  sonst
    $spc \leftarrow spc \cup \{a, b\}$;
    $toCover \leftarrow toCover \setminus$ {Alle Wege aus toCover, die von a oder von b abgedeckt werden.};
    $scanned \leftarrow scanned \cup currentPart$;
    $currentPart \leftarrow \emptyset$;
  Ende
Ende
zurück spc

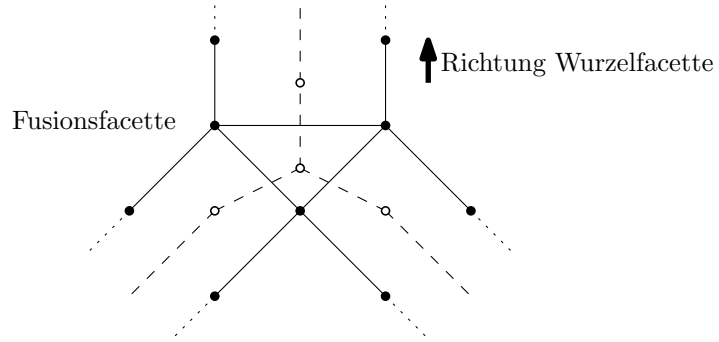


Abbildung 5.2: Zwei Dualgraphwege treffen an einer Fusionsfacette zusammen.

Es bleibt zu klären, weshalb es sich um eine 2-Approximation handelt. Sei C nun eine Ausgabe des Algorithmus, dann gib es eine Zeugenmenge Z, mit $|Z| = |C|/2$, denn für jedes Knotenpaar, das aufgenommen wird, existiert ein Weg, der sich vollständig hinter dem zuletzt aufgenommenen Separator befindet. Dies verhält sich analog zur Berechnung von minimalen SPCs auf Wegen, siehe Abschnitt 4.1. Die Behauptung folgt nun mit Lemma 8b.

Um diesen Satz vollständig zu beweisen, fehlt der Nachweis, dass der Algorithmus im Allgemeinen nichts Besseres als eine 2-Approximation findet. Dies wird durch ein Beispiel gezeigt. In Abbildung 5.1 seien alle nicht bezeichneten Kantengewichte 1, dann findet der Algorithmus für den Parameter $r = 3$ und die Startknoten v_1 und v_2 auf diesem Graphen die approximierte Lösung $\{v_3, v_8\}$. Eine minimale Lösung wäre die Menge $\{v_3\}$. □

5.1.2 Außenplanare Graphen (allgemein)

Der Algorithmus 5 soll im Folgenden auf allgemeine, außenplanare Graphen erweitert werden. Deren (schwacher) Dualgraph ist ein Baum, statt ein Weg. Der Einfachheit halber wird hier angenommen, dass die Eingabegraphen trianguliert sind. Das ist keine Einschränkung, denn jeder Graph kann so trianguliert werden, dass alle neu hinzugefügten Kanten ein unendlich hohes Gewicht und somit keinen Einfluss auf die Lösung haben. Sei nun G ein solcher triangulierter, außenplanarer Graph. Der Dualgraph T hat eine beliebig gewählte Wurzel. Die zu diesem Knoten zugehörige Facette wird als *Wurzelfacette* bezeichnet, analog wird der Begriff *Blattfacette* benutzt. Die Facetten der Dualknoten, die mehr als einen Nachfolger aufweisen, werden als *Fusionsfacetten* bezeichnet. Abbildung 5.2 skizziert eine solche. Der Dualgraph ist darin gestrichelt dargestellt. Zu einer Facette f wird der ihr im Dualgraph zugeordnete Knoten mit f^* bezeichnet, $T(f^*)$ bezeichne den Unterbaum von T mit Wurzel f^*. Der Graph $G(f)$ ist der Teilgraph von G, der von allen Knoten induziert wird, die zu einer Facette in $T(f^*)$ inzident sind.

Satz 35. *Für außenplanare Graphen, kann in $O(n^4)$ eine 3-Approximation gefunden werden.*

Beweis. Es wird zunächst ein Algorithmus beschrieben, dessen Laufzeit besprochen wird und im Anschluss wird dessen Korrektheit nachgewiesen.

Der Algorithmus geht ähnlich vor, wie der zur exakten Lösung des MIN-SPC-Problems bei Bäumen. Nach der Berechnung der Menge aller r-minimalen Wege *toCover* wird sukzessiv eine Facette f so gewählt, dass $G(f)$ einen der Wege aus *toCover* vollständig enthält, dies jedoch nicht für andere Facetten gilt, deren Dualknoten in $T(f)$ enthalten ist. Anschließend werden alle Knoten auf der Facette f in ein – zu Beginn leeres – SPC aufgenommen, alle

dadurch abgedeckten Wege aus *toCover* entfernt und auf dem Restgraphen (ohne $G(f)$) fortgefahren. Das geschieht so lange, bis *toCover* leer ist. Der Algorithmus 6 implementiert dieses Verfahren. Er wird mit der Wurzelfacette aufgerufen.

Algorithmus 6: APA-SPC-rek(f)

Daten : Facette in G
Ergebnis : $2^V \times 2^V$
$F :=$ Menge der Knoten auf der Facette;
$currentPart \leftarrow F$; `// Knoten des Teilgraphen, in dem nach nicht abgedeckten Wegen aus` *toCover* `gesucht werden muss.`
$C \leftarrow \emptyset$; `// Zu konstruierendes` r`-SPC`
wenn *f hat nur eine Nachfolgerfacette f' in G* **dann**
  $(C', currentPart') =$ APA-SPC-rek(f');
  $C \leftarrow C \cup C'$;
  $currentPart \leftarrow currentPart \cup currentPart'$;
Ende
sonst wenn *f hat zwei Nachfolgerfacetten f_1 und f_2 in G* **dann**
  $(C_1, currentPart_1) =$ APA-SPC-rek(f_1);
  $(C_2, currentPart_2) =$ APA-SPC-rek(f_2);
  $C \leftarrow C \cup C_1$;
  $C \leftarrow C \cup C_2$;
  $currentPart \leftarrow currentPart \cup currentPart_1$;
  $currentPart \leftarrow currentPart \cup currentPart_2$;
Ende
wenn *$currentPart$ enthält nicht abgedeckte Wege aus toCover* **dann**
  $C \leftarrow C \cup F$;
  $currentPart \leftarrow \emptyset$; `// Dieser und der nächste Schritt entspricht dem Abschneiden des Graphen an dieser Stelle.`
  Streiche alle durch Knoten aus F abgedeckten Wege aus *toCover* heraus;
Ende
zurück $(C, currentPart)$;

Die Laufzeit ist identisch mit der vom Algorithmus 5. Die Bestimmung für jede Facette, welche Knoten auf ihr liegen, funktioniert insgesamt in Linearzeit, ebenso die Vereinigungsoperationen der im Algorithmus vorkommenden Mengen, die für jeden Knoten durchgeführt werden.

Nach der Beschreibung des Algorithmus samt seiner Laufzeit fehlt der Nachweis der Korrektheit.

Offensichtlich ist das Ergebnis ein r-SPC. Die Argumentation ist identisch mit der im Beweis zum Lemma 22 bei Bäumen. Auch handelt es sich bei dem Ergebnis um eine 3-Approximation. Analog zum Verfahren bei außenplanaren Graphen mit der Outerpath-Eigenschaft existieren bei k aufgenommenen Knoten-Tripeln k paarweise knotendisjunkte, r-bedürftige Wege. Diese bilden zusammen eine Zeugenmenge. Die Approximationsgüte folgt daher wieder mit Lemma 8b. □

Man könnte auf die Idee kommen, den vorangegangenen Algorithmus zu einer 2-Approximation zu verbessern, denn im Fall von Nichtfusionsfacetten genügt es, lediglich die beiden Knoten der gerade hinzugefügten Facette aufzunehmen, die näher an der Wurzelfacette liegen. Bei

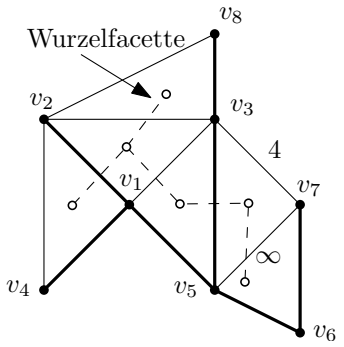


Abbildung 5.3: Schwierigkeiten bei der 2-Approximation. Der Dualgraph ist gestrichelt dargestellt und alle Kanten, die in einem 4-minimalen Weg enthalten sind, fett.

Fusionsfacetten könnten – durch geschickte Wahl – ebenfalls zwei Knoten ausreichen. Dies ist jedoch nicht so einfach möglich.

Im Graph der Abbildung 5.3 bildet das Knotentripel (v_1, v_2, v_3) die zentrale Fusionsfacette f. Alle Kantengewichte seien 1, sofern nicht anders dargestellt. Für den Parameter $r = 4$ ist die Menge $w_1 = (v_7, v_6, v_5, v_3, v_8), w_2 = (v_7, v_6, v_5, v_1, v_4), w_3 = (v_7, v_6, v_5, v_1, v_2)$ die Menge aller r-minimalen Wege. Ein minimales 4-SPC ist die Menge $\{v_5\}$. Der beschriebene Algorithmus 6 gibt (v_1, v_2, v_3) zurück.

Wollte man eine 2-Approximation berechnen, so müsste man sich für zwei der Knoten aus $\{v_1, v_2, v_3\}$ entscheiden. Weil w_2 in $G(f)$ vollständig enthalten ist und die Facette f nur an dem Knoten v_1 schneidet, so müsste v_1 einer der beiden Knoten sein. Man kann zu diese Zeitpunkt den zweiten Knoten nicht richtig bestimmen, ohne vorausschauen zu müssen. Entscheidet man sich beispielsweise für $\{v_1, v_2\}$ als Separator, dann müsste im nachfolgenden Schritt, in dem die Wurzelfacette erreicht wird, aufgrund des Wegs w_1 auch v_8 in ein 4-SPC mit aufgenommen werden, das damit insgesamt die Größe 3 hätte.

6. Ausblick

In dieser Arbeit wurde für einfache Graphklassen die Problematik der Berechnung minimaler SPCs, sowie Schranken für die Highway Dimension betrachtet. Für die Berechnung minimaler SPCs wurden einige Schwereresultate nachgewiesen Die Lösung des MIN-SPC Problems auf ungerichteten Graphen mit eindeutigen kürzesten Wegen ist APX-vollständig und falls die Graphen zusätzlich planar sind, immer noch NP-vollständig. Dies konnte durch eine Reduktion auf das bekannte Vertex-Cover-Problem nachgewiesen werden. Eine andere Reduktion auf MIN-HS führte zu dem Ergebnis, dass auf gerichteten Graphen mit nicht eindeutigen kürzesten Wegen lediglich eine $(1 - o(1)) \ln n$-Approximation in Polynomialzeit möglich ist, falls $P \neq NP$.

Es wurde untersucht, ob eine Zerlegung einer Probleminstanz in kleinere Instanzen möglich ist, um aus deren optimalen Lösungen eine optimale Lösung der Ausgangsinstanz zu berechnen. Nur unter sehr starken Voraussetzungen konnte dies erreicht werden. Alle kleineren Instanzen müssen dabei dem Zerlegbarkeitskriterium genügen und deren optimale Teillösungen müssen einen Separator zwischen ihnen enthalten. Beispielsweise für Bäume ist das ausreichend und konnte zu einem Linearzeitalgorithmus genutzt werden. Für Kreise, für die eine Zerlegung wie bei Bäumen im Allgemeinfall nicht möglich ist, wurde ein $O(n \log n)$-Algorithmus beschrieben. Viele Kreisinstanzen können jedoch auch in Linearzeit gelöst werden, sofern nicht der Parameter r starken Einschränkungen im Verhältnis zur Gesamtkreisgröße unterliegt oder einzelne Kantengewichte sehr hoch ausfallen. Außenplanare Graphen sind in Polynomialzeit 3-approximierbar und, wenn sie das Outerpath-Kriterium erfüllen, sogar 2-approximierbar.

Die Highway Dimension auf Wegen beträgt höchstens 8. Da Kreise sich lokal wie Wege verhalten, gilt dieselbe Aussage auch für diese. Bereits Bäume können jedoch eine beliebig hohe Highway Dimension aufweisen.

Das MIN-SPC Problem ist sehr schwierig zu lösen. Einige Resultate im Hinblick auf dessen Komplexität können jedoch noch näher untersucht werden. Es ist noch offen, ob es möglich ist, eine beliebig gute Approximation für ungerichtete, planare Graphen in Polynomialzeit zu berechnen. Ebenso, ob das Problem schon für speziellere Graphklassen als planare Graphen NP-vollständig ist. Bei Kreisen ist noch unklar, ob es nicht auch eine Lösung in Linearzeit für beliebige Probleminstanzen gibt. Möglich ist es auch, dass sich für planare Graphen bekannte Techniken, wie beispielsweise das Planar Separator Theorem [LT79]

oder die Baumbreite, für eine Approximation mit garantierter Güte nutzen lassen.

Literaturverzeichnis

[ADF+11] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg und Renato F. Werneck: *VC-Dimension and Shortest Path Algorithms.* In: *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP'11)*, Band 6755 der Reihe *Lecture Notes in Computer Science*, Seiten 690–699. Springer, 2011.

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg und Renato F. Werneck: *Highway Dimension, Shortest Paths, and Provably Efficient Algorithms.* In: Moses Charikar (Herausgeber): *Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'10)*, Seiten 782–793. SIAM, 2010.

[Bak94] Brenda S. Baker: *Approximation Algorithms for NP-Complete Problems on Planar Graphs.* Journal of the ACM, 1994.

[BFM+07] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders und Dominik Schultes: *In Transit to Constant Shortest-Path Queries in Road Networks.* In: *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*, Seiten 46–59. SIAM, 2007.

[BG95] Hervé Brönnimann und Michael T. Goodrich: *Almost Optimal Set Covers in Finite VC-Dimension.* Discrete and Computational Geometry, 14:463–497, 1995.

[Dij59] Edsger W. Dijkstra: *A Note on Two Problems in Connexion with Graphs.* Numerische Mathematik, 1:269–271, 1959.

[DS05] Irit Dinur und Shmuel Safra: *On the importance of being biased (1.36 hardness of approximating Vertex-Cover.* 2005.

[DSSW09] Daniel Delling, Peter Sanders, Dominik Schultes und Dorothea Wagner: *Engineering Route Planning Algorithms.* In: Jürgen Lerner, Dorothea Wagner und Katharina A. Zweig (Herausgeber): *Algorithmics of Large and Complex Networks*, Band 5515 der Reihe *Lecture Notes in Computer Science*, Seiten 117–139. Springer, 2009.

[Fei98] Uriel Feige: *A Threshold of ln n for Approximating Set Cover.* Journal of the ACM, 1998.

[Flo62] Robert W. Floyd: *Algorithm 97: Shortest path.* Communications of the ACM, 5(6):345, 1962.

[GJ79] Michael R. Garey und David S. Johnson: *Computers and Intractability. A Gui-*

de to the Theory of $\mathcal{NP}$-Completeness. W. H. Freeman and Company, 1979.

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes und Daniel Delling: *Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks.* In: Catherine C. McGeoch (Herausgeber): *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, Band 5038 der Reihe *Lecture Notes in Computer Science*, Seiten 319–333. Springer, June 2008.

[Kar72] Richard M. Karp: *Reducibility among Combinatorial Problems.* In: Raymond E. Miller und James W. Thatcher (Herausgeber): *Complexity of Computer Computations*, Seiten 85–103. Plenum Press, 1972.

[Lau04] Ulrich Lauther: *An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background.* In: *Geoinformation und Mobilität - von der Forschung zur praktischen Anwendung*, Band 22, Seiten 219–230. IfGI prints, 2004.

[LT79] Richard J. Lipton und Robert E. Tarjan: *A separator theorem for planar graphs.* SIAM Journal on Applied Mathematics, 1979.

[PY88] Christos H. Papadimitriou und Mihalis Yannakakis: *Optimization, approximation, and complexity classes.* In: *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC'88)*, Seiten 229–234. ACM Press, 1988.