Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. Dorothea Wagner

# Disjunkte kürzeste Wege in drahtlosen Sensornetzen

Diplomarbeit

Markus Maier

31. Oktober 2005

Betreuer: Prof. Dr. Dorothea Wagner

Betreuender Mitarbeiter: Dipl-Inform. Steffen Mecke

# Erklärung

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

Karlsruhe, den 31. Oktober 2005

# Kurzbeschreibung

Drahtlose Sensornetze sind relativ störungsanfällig. Deshalb kann es sinnvoll sein, Informationen über mehrere disjunkte Wege von einem Sender zu einem Empfänger zu senden, um die Redundanz zu erhöhen. Andererseits spielen Energiebetrachtungen in drahtlosen Sensornetzen eine wichtige Rolle, da für Sensoren oft keine externe Energieversorgung möglich ist. Die disjunkten Wege sollten deshalb möglichst energieeffizient sein.

In dieser Diplomarbeit werden hauptsächlich kantendisjunkte Wege betrachtet, da es für knotendisjunkte Wege auf der einen Seite schon effiziente Algorithmen gibt, sie sich aber auf der anderen Seite kaum energieeffizient finden lassen. Nach kurzen Erläuterungen zur Existenz kantendisjunkter Wege wird die Komplexität der Berechnung kantendisjunkter Wege minimaler Energie, des sogenannten $k$-EDPWMA-Problems, untersucht und gezeigt, dass es im allgemeinsten Fall NP-vollständig ist. Anschließend werden Algorithmen für Spezialfälle, insbesondere für den Fall azyklischer Energiekostengraphen beschrieben. Mit Hilfe dieses Algorithmus lässt sich eine Heuristik für allgemeine Graphen angeben, sofern die Geometrie des Netzes bekannt ist. Einige Aspekte dieser Heuristik werden experimentell untersucht. Schließlich wird das $k$-SPEDPWMA-Problem eingeführt, bei dem es um die Berechnung energieoptimaler kantendisjunkter Wege in Sensornetzen, in denen ein Knoten nur in einer Energie senden kann, geht. Es wird gezeigt, dass ein Approximationsalgorithmus, der für $k = 3$ im allgemeinen nur den Approximationsfaktor 3 liefert, in diesem Spezialfall den Approximationsfaktor 2 liefert.

# Inhaltsverzeichnis

# Danksagung

Frau Professor Dr. Dorothea Wagner danke ich sehr herzlich für das Bereitstellen dieses interessanten Themas.

Besonders bedanken möchte ich mich bei Steffen Mecke für die intensive und hilfreiche Betreuung.

Zu guter Letzt möchte ich meinen Eltern danken, die mir das Studium und damit auch diese Diplomarbeit erst ermöglicht haben.

# 1 Grundlagen: Netzwerkmodell und Wege

## 1.1 Einleitung

Durch die fortschreitende Miniaturisierung ist es möglich geworden, Sensoren mit Rechenkapazitäten und der Fähigkeit zu drahtloser Kommunikation auszustatten und sie in großen Stückzahlen günstig herzustellen. Diese Sensoren werden WINS (wireless integrated network sensors) oder *Sensorknoten* genannt [23]. Werden viele solcher Sensorknoten zu einem drahtlosen Netzwerk verbunden und mit der Fähigkeit zur Koordination versehen, um eine gemeinsame (verteilte) Aufgabe kooperativ bewältigen zu können, so spricht man von einem *drahtlosen Sensornetz*.

Für drahtlose Sensornetze gibt es eine Vielzahl möglicher Anwendungsgebiete, z. B. im Verkehr, in der produzierenden Industrie, im Gesundheitswesen oder in der Umweltbeobachtung. In [8] werden die folgenden Beispiele aufgeführt:

- Die Werkzeuge und Maschinen in einer Fabrik sind mit einem Funketikett (z. B. RFID) versehen. Sensoren, die über das Gebäude verteilt sind, können den Einsatz der Werkzeuge und Maschinen überwachen. Dadurch ist es möglich, sie zu lokalisieren, ihren Einsatz automatisch zu analyisieren usw.

- Über einer Katastrophenregion werden aus der Luft Sensorknoten abgeworfen, die sich zu einem Sensornetz organisieren. Bei einem großflächigen Waldbrand wird es dadurch z. B. möglich, die Einsatzgruppen der Feuerwehr automatisch zu den Stellen zu führen, an denen ihr Einsatz am dringendsten benötigt wird oder automatisch Evakuierungspfade zu finden.

- Die Autos in einer Großstadt werden mit Sensorknoten ausgestattet. Wenn sich zwei Autos nahe genug sind, können die Sensorknoten kommunizieren und Informationen austauschen. Dadurch ist es möglich, vor gefährlichen Straßenzuständen (z. B. Glatteis) zu warnen, oder Routen so zu planen, dass Staus vermieden werden.

Wahrscheinlich werden nicht alle der möglichen Anwendungen auch auf gesellschaftliche Akzeptanz stoßen. Die sozialen und juristischen Aspekte des Einsatzes von Sensornetzen sind jedoch nicht Thema dieser Arbeit und es bleiben sicherlich einige Gebiete, in denen der Einsatz von Sensornetzen unumstritten sinnvoll ist.

Drahtlose Netze sind wesentlich störungsanfälliger als drahtgebundene Netze. So können Verbindungen zwischen zwei Knoten (z.B. wegen Hindernissen zwischen den Knoten) oder auch ganze Knoten ausfallen (z. B. wegen Energiemangels). Falls Daten nur über

einen Weg von einem Sender zu einem Empfänger geschickt werden, führt schon der Ausfall einer Kante bzw. eines Knotens auf dem Weg zu Datenverlust. Deshalb ist es unter Umständen sinnvoll, Daten nicht nur auf einem Weg von einem Sender zu einem Empfänger zu schicken, sondern auf mehreren Wegen gleichzeitig, so dass beim Ausfall eines Weges die anderen Wege noch funktionieren und die Daten sicher ankommen. Um die Ausfallsicherheit zu gewährleisten, sollten je zwei Wege zumindest kantendisjunkt sein, d. h. keine Kanten gemeinsam benutzen. Noch sicherer sind knotendisjunkte Wege, d. h. Wege, die außer Start- und Zielknoten keine Knoten gemeinsam benutzen. In dieser Arbeit werden sowohl knoten- als auch kantendisjunkte Wege betrachtet, wobei der Schwerpunkt jedoch auf kantendisjunkten Wegen liegt.

In vielen der oben angegebenen Anwendungsgebiete für Sensornetze ist es nicht möglich, die Sensoren mit externer Energie zu versorgen. Die Sensorknoten müssen also kleine Batterien für ihre Energieversorgung benutzen. Um die Lebensdauer eines Sensornetzes zu verlängern, ist es daher wichtig, auf einen sparsamen Energieverbrauch der Knoten zu achten. Es hat sich herausgestellt, dass ein Knoten einen Großteil seiner Energie zur Kommunikation aufwendet. Deshalb ist es sinnvoll, insbesondere bei der Kommunikation einen sparsamen Energieverbrauch anzustreben.

In der Literatur gibt es viele Arbeiten, die Probleme, die bei der Kommunikation in Sensor- und Ad-hoc-Netzen auftreten, unter Energiegesichtspunkten betrachten. Dazu gehören insbesondere die Behandlung des Broadcast-Problems in [31], [3] und [18]. In [25] werden energieeffiziente Routingverfahren vorgeschlagen.

Das Problem, kantendisjunkte Wege minimaler Energie in Sensornetzen zu berechnen, wurde zum ersten Mal in [26] untersucht. In dieser Arbeit wurden die STPS- und OCND-Algorithmen zur Berechnung von knotendisjunken Wegen minimaler Energie bzw. von zwei kantendisjunkten Wegen minimaler Energie angegeben. Diese werden in Abschnitt 2.3.1 bzw. in Abschnitt 3.5.1 kurz vorgestellt.

Ein zum Problem disjunkter Wege verwandtes Problem ist das Zusammenhangsproblem, bei dem ein zusammenhängender Subgraph mit minimaler Energie gefunden werden soll, der alle Knoten enthält. In diesem Subgraphen gibt es also für jedes Knotenpaar einen Verbindungsweg. Die NP-Vollständigkeit dieses Problems wurde in [4] gezeigt.
Eine Verallgemeinerung ist das $k$-fache Zusammenhangsproblem, bei dem ein Subgraph gefunden werden soll, der alle Knoten enthält, minimale Energie hat und in dem es zu je zwei Knoten $k$ kantendisjunkte Wege gibt. Dies wiederum kann als Instanz des Minimum Energy Topologiekontrollproblems betrachtet werden, das in [19] betrachtet wird. Hier geht es darum, Subgraphen zu finden, die gewisse Topologieeigenschaften haben und minimale Energie benötigen.

## 1.2 Modellierung von Sensornetzen

In diesem Abschnitt werden zwei Modelle für Sensornetze vorgestellt: Das Modell in Abschnitt 1.2.1 ist ein sehr allgemeines Modell, in dem gerichtete Antennen, Hindernisse zwischen Knoten, usw. modelliert werden können. Dieses Modell wird den theoretischen

Teilen dieser Arbeit zugrundegelegt.

Das Modell in Abschnitt 1.2.2 kann als Spezialfall des allgemeinen Modells betrachtet werden. Man geht hier davon aus, dass alle Sensorknoten omnidirektionale Antennen haben und die Signalstärke mit der Entfernung vom Sender gleichmäßig abnimmt. Es wird in dieser Arbeit hauptsächlich für die Simulationen in Kapitel 5 verwendet.

### 1.2.1 Allgemeines Modell für Sensornetze

Ein sehr allgemeineres Modell für drahtlose Netze wird in [19] beschrieben. Ein Netz besteht dabei aus $n$ Knoten, die sowohl senden als auch empfangen können. Die maximale Sendeenergie eines Knotens $v$ sei $e_{\max}(v)$ und er kann mit jeder Energie aus dem Intervall $[0, e_{\max}(v)]$ senden. Für jedes geordnete Paar $(u, v)$ von Knoten gibt es einen Schwellwert $p(u, v)$, wobei $v$ ein Signal von $u$ nur empfangen kann, wenn $u$ mindestens mit einer Energie von $p(u, v)$ sendet. Dieser Schwellwert kann von verschiedenen Faktoren abhängen, unter anderem von der Entfernung der Knoten voneinander, von der Richtung der Antennen an Sender und Empfänger, von möglichen Hindernissen zwischen ihnen usw.

Die Topologie eines Netzes hängt wesentlich von den Sendeenergien der Knoten ab. Deshalb ist es sinnvoll, den sogenannten Energiekostengraphen zu definieren, der die Topologie eines Netzes in Abhängigkeit von den Sendeenergien der Knoten widerspiegelt:

**Definition 1.** Gegeben sei ein drahtloses Netzwerk mit $n$ Knoten $V = \{v_1, \ldots, v_n\}$, maximalen Sendeenergien $e_{\max} : V \to \mathbb{R}^+$ und Schwellwertfunktion $p : V \times V \to \mathbb{R}^+$. Der Energiekostengraph ist der gerichtete gewichtete Graph $D = (V, A)$ mit $A = \{(u, v) \mid p(u, v) < e_{\max}(u)\}$ und Gewichtsfunktion $w : A \to \mathbb{R}^+$, $w(u, v) = p(u, v)$ für alle $(u, v) \in A$.

Laut der obigen Definition kann es im Energiekostengraphen keine Mehrfachkanten zwischen zwei Knoten geben.

*Bemerkung.* In der Literatur, z. B. in [3] und [18], wird auch ein Sensornetzmodell verwendet, in dem es für jeden Knoten nur endlich viele Energiestufen gibt, mit denen er senden kann. Dies sieht auf den ersten Blick weniger mächtig aus als das hier verwendete Modell. Doch auch wenn ein Knoten in dem hier verwendeten Modell laut Definition mit jeder Energie in dem Intervall $[0, \mathcal{E}_{\max}]$ senden kann, wird er doch nur mit der Energie senden, mit der er den am weitesten entfernten Nachbarn, den er erreichen will, gerade noch erreicht. Da es nur endlich viele Nachbarn gibt, sendet ein Knoten also auch hier nur mit endlich vielen verschiedenen Energiestufen.

Den Zusammenhang zwischen Sendeenergien an den Knoten und der Topologie des Netzes stellt die folgende Definition her:

**Definition 2.** Gegeben gewichteter gerichteter Graph $D = (V, A; w)$, $s, t \in V$ und eine Funktion $e : V \setminus \{t\} \to \mathbb{R}$ mit $e(v) \in [0, \mathcal{E}_{max}(v)]$ für alle $v \in V \setminus \{t\}$.
Den Graphen $D_e = (V, A_e)$ mit $A_e = \{(i, j) \in A \mid e(i) \geq w((i, j))\}$ nennen wir den von $e$ *induzierten Subgraphen.*

### 1.2.2 Modell für Sensorknoten mit omnidirektionalen Antennen

Ein Spezialfall des allgemeinen Modells ist das folgende Modell für Netze, deren Knoten omnidirektionale Antennen besitzen, und das in den Artikeln [31] und [26] verwendet wird. Ein Sensornetz besteht dabei aus $n$ Knoten $v_1, \ldots, v_n$ mit omnidirektionalen Antennen, die in der euklidischen Ebene verteilt sind. Als Abstandsmaß $d : V \times V \to \mathbb{R}_0^+$ wird die euklidische Metrik verwendet, also $d(u,v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$, wobei $x_u, x_v$ bzw. $y_u, y_v$ die x- bzw. y-Koordinaten der Knoten $u$ und $v$ bezeichnen. Die Signalstärke nimmt mit $r^{-\alpha}$ ab, wobei $r$ die Entfernung vom Sender bezeichne und $\alpha \in [2,4]$ eine Verlustkonstante. Es gibt nun eine Verbindung von einem Knoten $u$ zu einem Knoten $v$, wenn die Signalstärke am Empfänger $v$ über einem Schwellwert $\theta$ liegt, wenn also $\mathcal{E}(u)d(u,v)^{-\alpha} \geq \theta$, wobei $\mathcal{E}(u)$ die Sendeenergie des Knotens $u$ bezeichnet. $\theta$ wird dabei so gewählt, dass eine gewünschte Bitfehlerrate erreicht wird. Bei geeigneter Normierung des Abstands kann $\theta = 1$ gewählt werden und es existiert eine Verbindung zwischen $u$ und $v$, falls $\mathcal{E}(u)d(u,v)^{\alpha}$. Es gilt also $p(u,v) = d(u,v)^{\alpha}$ und der Energiekostengraph ist der gewichtete gerichtete Graph $D = (V, A)$ mit $A = \{(u,v) \in V \times V \mid d(u,v)^{\alpha} \leq e_{\max}(u)\}$ und Gewichtsfunktion $w : A \to \mathbb{R}^+, w(u,v) = d(u,v)^{\alpha}$. Ein Beispiel für einen Energiekostengraphen in diesem Modell ist in Abbildung 1.1 dargestellt.

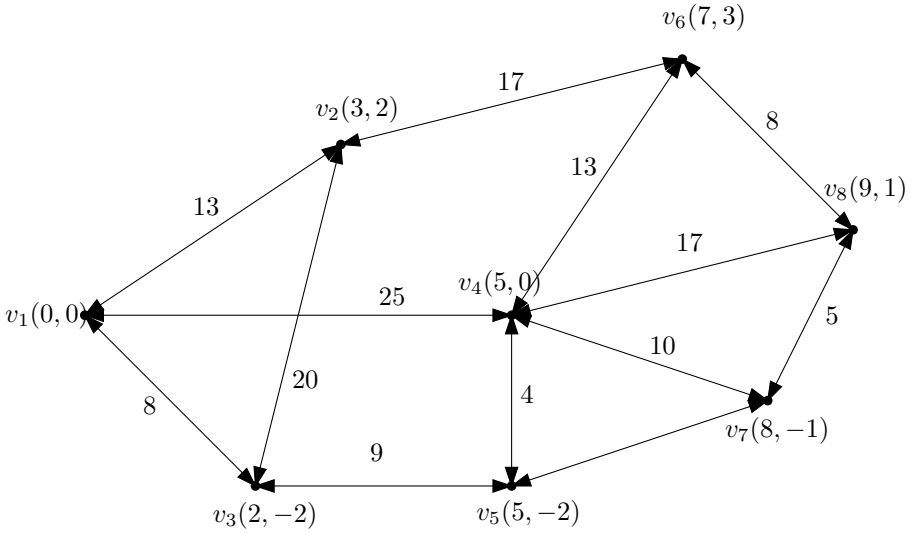



Abbildung 1.1: Energiekostengraph für ein drahtloses Netzwerk mit 8 Knoten mit omnidirektionalen Antennen in der euklidischen Ebene mit $\alpha = 2$ und $e_{\max} \equiv 25$.

Alternativ zur Angabe der maximalen Sendeenergie für jeden Knoten kann in diesem Modell auch die Angabe des maximalen Senderadius $r_{\max}(v)$ erfolgen und es gilt

$$e_{\max}(v) = r_{\max}(v)^{\alpha}.$$

Da der Radius anschaulicher ist als die Energie, wird bei den Versuchen in Kapitel 5 die Angabe des maximalen Senderadius der Angabe der maximalen Sendeenergie vorgezogen.

Während Metriken in der Ebene per definitionem die Dreiecksungleichung erfüllen, gilt dies für die Gewichte im Energiekostengraphen für $\alpha > 1$ nicht mehr. Ein einfaches Beispiel dafür zeigt Abbildung 1.2.

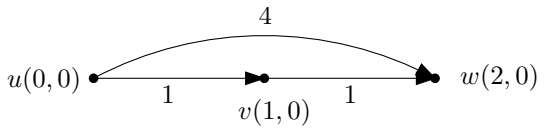
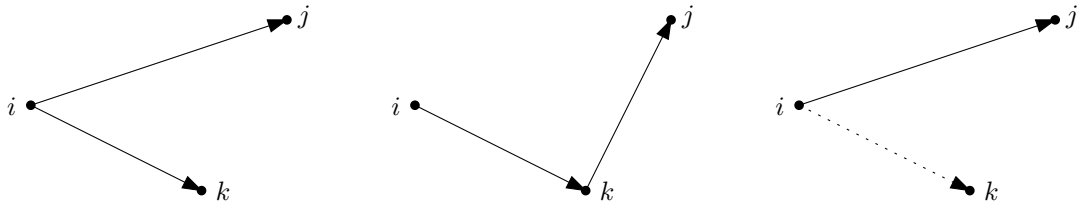



Abbildung 1.2: In Energiekostengraphen gilt die Dreiecksungleichung im allgemeinen nicht. Hier ein Beispiel mit drei Knoten und $\alpha = 2$. Es gilt $4 = w(u, w) > w(u, v) + w(v, w) = 2$, die Dreiecksungleichung ist also verletzt.

## 1.3 Der Wireless Multicast Advantage (WMA)

Wenn in einem drahtlosen Netz ein Knoten $u$ mit einer Energie $\mathcal{E}(u)$ sendet, so kann das Signal von allen Knoten $v$ mit $p(u, v) < \mathcal{E}(u)$ empfangen werden. Diese Eigenschaft kann bei Multicast- und Broadcast-Problemen ausgenutzt werden, um Energie zu sparen und wird deshalb Wireless Multicast Advantage (WMA) genannt. In Abbildung 1.3 ist das graphisch dargestellt. Soll ein Knoten $i$ eine Nachricht an zwei Knoten $j$ und $k$ senden, so muss er in drahtgebundenen Netzen entweder zuerst an den einen Knoten und dann an den anderen senden, oder ein Knoten leitet die Nachricht an den anderen weiter. In drahtlosen Netzen gibt es jedoch zusätzlich die Möglichkeit, dass $i$ an den weiter entfernten Knoten, also an den Knoten $j$ mit dem höheren Wert $p(i, j)$ sendet und der andere Knoten, hier $k$ dabei ebenfalls erreicht wird. Insgesamt führt dies zu einer Energieeinsparung, falls $p(i, j) < p(i, k) + p(k, j)$.




(a) Separates Senden an $j$ und $k$

(b) Senden von $i$ an $k$ und von $k$ an $j$

(c) Senden mit WMA

Abbildung 1.3: Ein Beispiel für den *Wireless Multicast Advantage*. Der Knoten $i$ soll eine Nachricht an die Knoten $j$ und $k$ senden. In drahtgebundenen Netzen gibt es zwei Möglichkeiten: Er sendet die Nachricht einmal an $j$ und einmal an $k$ (a) oder er sendet an $k$ und dieser sendet an $j$ (b). In drahtlosen Netzen hingegen kann er an den weiter entfernten Knoten senden und erreicht alle näheren Knoten mit (c). [26]

# 1.4 Wege

## 1.4.1 Definitionen und Notationen

Die folgenden Definitionen und Notationen sind an die Notation in [5] angelehnt.

**Definition 3.** Sei $D = (V, A)$ ein gewichteter gerichteter Graph mit Gewichtsfunktion $w : A \to \mathbb{R}$ und $s, t \in V$.
Ein *Weg* $W$ von $s$ nach $t$ (bzw. $s$-$t$-Pfad) ist eine Folge $W =< s = v_0, v_1, \ldots, v_l = t >$ von Knoten mit $(v_i, v_{i+1}) \in A$ für alle $i = 1, \ldots, l-1$.
Die Kanten von $W$ werden mit $E(W)$ bezeichnet: $E(W) = \{(v_i, v_{i+1}) \mid i = 0, \ldots l-1\}$ und die Knoten mit $V(W)$: $V(W) = \{v_0, \ldots, v_l\}$.
Das Symbol $\oplus$ bezeichne die Konkatenation von Wegen, d. h. für zwei Wege $W_1 =< u_1, \ldots, u_r >$ und $W_2 =< v_1, \ldots, v_s >$ $(r, s \in \mathbb{N})$ ist $W_1 \oplus W_2 =< u_1, \ldots, u_r, v_1, \ldots, v_s >$.
Sei $P = \{P_1, \ldots, P_k\}$ eine Menge von $k$ Wegen von $s$ nach $t$. Dann bezeichne $E(P)$ die Menge aller Kanten der Wege in $P$: $E(P) = E(P_1) \cup \ldots \cup E(P_k)$. und $V(P)$ analog die Menge aller Knoten auf Wegen in $P$: $V(P) = V(P_1) \cup \ldots \cup V(P_k)$

Das Thema dieser Arbeit sind disjunkte Wege. Grundsätzlich werden knoten- und kantendisjunkte Wege unterschieden:

**Definition 4.** Unter den Voraussetzungen wie in Definition 3 seien $k$ Wege $P_1, \ldots, P_k$ von $s$ nach $t$ gegeben.
Die Wege $P_1, \ldots, P_k$ heißen *knotendisjunkt*, falls für alle $i, j \in \{1, \ldots k\}$, $i \neq j$ gilt: $V(P_i) \cap V(P_j) = \{s, t\}$.
Die Wege $P_1, \ldots, P_k$ heißen *kantendisjunkt*, falls für alle $i, j \in \{1, \ldots k\}$, $i \neq j$ gilt: $E(P_i) \cap E(P_j) = \emptyset$.

Es ist klar, dass aus der Knotendisjunktheit von $P$ auch die Kantendisjunktheit folgt, denn wenn zwei Wege eine Kante $(u, v)$ verwenden, so besuchen beide ihre Endknoten $u$ und $v$, was im Widerspruch zur Knotendisjunktheit steht, außer wenn $u = s$ und $v = t$. Dann sind die Wege jedoch identisch. Die Umkehrung gilt nicht, wie an dem einfachen Beispiel in Abbildung 1.4 deutlich wird.

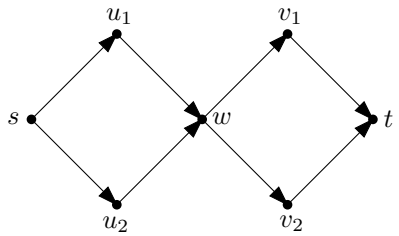



Abbildung 1.4: $P_1 =< s, u_1, w, v_1, t >$ und $P_2 =< s, u_2, w, v_2, t >$ sind kantendisjunkt aber nicht knotendisjunkt, da sie beide $w$ besuchen.

### 1.4.2 Metriken für Wege

In dieser Arbeit geht es um disjunkte kürzeste Wege. Deshalb ist es erforderlich, eine Metrik einzuführen, bezüglich derer ein Weg bzw. eine Menge von Wegen, kurz genannt werden kann. Es ist möglich, verschiedene Metriken für die Kosten einer Menge von Wegen einzuführen. In herkömmlichen drahtgebundenen Netzen, wird oft die Summe der Kantengewichte verwendet, die sogenannte Gewichtsmetrik. Eine andere, oft verwendete Metrik, ist die sogenannte Hop-Metrik, die die Anzahl der verwendeten Kanten, ohne Berücksichtigung von deren Gewichtung, angibt. In drahtlosen Netzen möchte man jedoch eine Metrik verwenden, die den WMA berücksichtigt. Deshalb wird hier die Energiemetrik verwendet.

**Definition 5.** Sei $D = (V, A; w)$ ein gewichteter gerichteter Graph mit Gewichten $w : A \to \mathbb{R}_0^+$ und $P = \{P_1, \ldots, P_n\}$ eine Menge von $k$ Wegen von $s$ nach $t$ ($s, t \in V$).
Das *Gewicht* $W(P)$ der Wege $P$ ist definiert als

$$W(P) = \sum_{e \in E(P)} w(e)$$

Die *Energie* $\mathcal{E}(P)$ der Wege $P$ ist definiert als

$$\mathcal{E}(P) = \sum_{v \in V(P)} \max_{(v,u) \in E(P)} w(v, u)$$

Die Gewichtsmetrik ist eine kantenbasierte Metrik, da das Hinzufügen einer Kante zu einer Wegemenge immer die gleichen Kosten verursacht, unabhängig davon, ob die Endknoten schon in der Pfadmenge enthalten sind. Die Energiemetrik kann dagegen als knotenbasierte Metrik betrachtet werden, da die Kosten, die das Hinzufügen einer Kante verursacht, sowohl vom Ausgangsknoten als auch von den bereits in $P$ vorhandenen Kanten abhängt. Die Kosten steigen nur, wenn das Gewicht der Kante größer ist als das Gewicht sämtlicher anderer Kanten, die von dem Knoten ausgehen und schon in $P$ sind.

Im folgenden wird die Metrik, bezüglich derer die Länge eines Weges oder einer Wegmenge gemessen wird, in der Regel explizit angegeben. Ist nur von "kürzesten Wegen" oder von "längsten Wegen" die Rede, so ist als Länge die Anzahl der Kanten gemeint (z. B. in Abschnitt 3.5.3).

In einigen Beweisen in dieser Arbeit wird sich der folgende einfache Zusammenhang als nützlich erweisen.

**Satz 1.** *Unter den Voraussetzungen von Definition 5 gilt* $\mathcal{E}(P) \leq W(P)$. *Gleichheit gilt genau dann, wenn* $k = 1$.

*Beweis.* Da die Summation über alle Kanten durch Summation über alle Knoten und anschließende Summation über alle ausgehenden Kanten ersetzt werden kann, gilt:

$$W(P) = \sum_{e \in E(P)} w(e) = \sum_{v \in V(P)} \sum_{(v,u) \in E(P)} w(v, u) \leq \sum_{v \in V(P)} \max_{(v,u) \in E(P)} w(v, u) = \mathcal{E}(P) \quad (1.1)$$

Falls $k = 1$ gibt es für jeden Knoten $v$, der auf $P$ liegt, genau eine ausgehende Kante in $E(P)$, also $\sum_{(v,u)\in E(P)} w(v,u) = \max_{(v,u)\in E(P)} w(v,u)$. Alle anderen Knoten gehen nicht in die Summation ein. Daraus folgt die Gleichheit in 1.1.
Falls $k > 1$ hat $s$ genau $k$ ausgehende Kanten zu Knoten $u_1, \ldots, u_k$. O.B.d.A. sei $(s, u_1)$ die Kante mit dem höchsten Gewicht. Da im Energiekostengraphen alle Kantengewichte positiv sind, also insbesondere $w(s, u_2) > 0$, gilt

$$\sum_{(s,u)\in E(P)} w(s,u) \geq w(s,u_1) + w(s,u_2) > w(s,u_1) = \max_{(s,u)\in E(P)} w(s,u).$$

und damit Ungleichheit in 1.1. □

# 2 Knotendisjunkte Wege

## 2.1 Einleitung

Knotendisjunkte Wege erhöhen die Ausfallsicherheit stärker als kantendisjunkte Wege, da sie sowohl bei einem Ausfall von Knoten als auch bei einem Ausfall von Kanten, d. h. Verbindungen zwischen Knoten, funktionsfähig bleiben. Da andererseits bei knotendisjunkten Wegen der einzige Knoten, der mehr als einen seiner Nachbarn erreichen muss, die Quelle $s$ ist, kann der WMA nur an der Quelle $s$ ausgenutzt werden. Unter Energiegesichtspunkten sind knotendisjunkte Wege also nicht optimal.

## 2.2 Knotendisjunkte Wege minimalen Gewichts

Auch wenn das Thema dieser Arbeit disjunkte Wege in drahtlosen Netzen sind, lohnt es sich doch, kurz den drahtgebundenen Fall zu betrachten, da Algorithmen zur Berechnung von knotendisjunkten Wegen minimalen Gewichts Bestandteil des im nächsten Abschnitts vorgestellten Algorithmus zur Berechnung knotendisjunkter Wege minimaler Energie sind.

Suurballe hat in [27] einen Algorithmus vorgestellt, der $k$ knoten- oder kantendisjunkte Wege minimalen Gewichts in einem Netzwerk berechnet. Er hat sowohl im knoten- als auch im kantendisjunkten Fall eine Komplexität in $O(kn^2)$, wobei $n$ die Anzahl der Knoten des Netzwerks ist.

Andererseits ist es in einem gerichteten Graphen schon für 2 Paare $(s_1, t_1), (s_2, t_2)$ mit $s_1 \neq s_2$ von Start- und Zielknoten NP-vollständig, zu entscheiden, ob es knotendisjunkte Wege von $s_1$ nach $t_1$ und von $s_2$ nach $t_2$ gibt [10].

## 2.3 Knotendisjunkte Wege minimaler Energie

Srinivas und Modiano geben in [26] den sog. STPS-Algorithmus (Source Transmit Power Selection) an, der $k$ knotendisjunkte Wege minimalen Gewichts in $O(kn^3)$ berechnet. Der Algorithmus nutzt aus, dass der einzige Knoten, an dem der WMA ausgenutzt werden kann, die Quelle $s$ ist.

### 2.3.1 Der STPS-Algorithmus

Sei $G = (V, E)$ ein gerichteter Graph, $w : E \to \mathbb{R}^+$ eine Gewichtsfunktion und $s$ bzw. $t$ Start- bzw. Zielknoten. $e_1, \ldots, e_l$ seien die Kanten, die den Startknoten verlassen, wobei

$w(e_1) \leq w(e_2) \leq \ldots \leq w(e_l)$.
Die folgenden Schritte werden für $i = k, k+1, \ldots, l$ wiederholt:

- Berechne den Graphen $G_i$, der aus G entsteht, indem $e_{i+1}, \ldots, e_l$ aus $G$ entfernt werden und $w(e_1) = \ldots = w(e_i) = 0$ gesetzt werden.
- Berechne in $G_i$ $k$ knotendisjunkte Wege minimalen Gewichts (z. B. mit dem Algorithmus von Suurballe). Diese Wege nennen wir $P_{i,1}, \ldots, P_{i,k}$. Setze $W_i = w(e_i) + W(P_{i,1}) + \ldots + W(P_{i,k})$.

Bestimme das $i \in \{1, \ldots, l\}$, für das $W_i$ minimal. Die dazugehörigen $k$ Wege $P_{i,1}, \ldots, P_{i,k}$ sind die knotendisjunkten Wege minimaler Energie.

# 3 Kantendisjunkte Wege

## 3.1 Einleitung

Kantendisjunkte Wege sind weniger sicher als knotendisjunkte Wege, da schon der Ausfall eines Knotens zu einem Ausfall aller Wege führen kann, falls alle Wege über diesen einen Knoten laufen. Andererseits ist der Ausfall eines Knotens in einem Sensornetz in der Regel wesentlich weniger wahrscheinlich als der Ausfall einer Verbindung zwischen zwei Knoten, d. h. einer Kante.

Es ist klar, dass sich mit dem WMA nur Energie an Knoten einsparen lässt, die von mehreren Wegen gemeinsam benutzt werden. Bei knotendisjunkten Wegen sind Energieeinsparungen also auf die Quelle beschränkt. Bei kantendisjunkten Wegen kann dagegen auch an weiteren Knoten Energie eingespart werden. Es muss also die Sicherheit gegenüber dem Ausfall von Knoten gegen die Energieeffizienz abgewogen werden.

In diesem Kapitel wird zuerst in Abschnitt 3.2 die Existenz kantendisjunkter Wege untersucht, bevor in Abschnitt 3.3 die Eigenschaften bekannter Algorithmen zur Berechnung von kantendisjunkten Wegen minimalen Gewichts beschrieben werden. Dies ist notwendig, da einige Algorithmen in späteren Abschnitten diese Algorithmen verwenden. In Abschnitt 3.4 wird das Problem, $k$ kantendisjunkte Wege minimaler Energie in einem gewichteten gerichteten Graphen zu bestimmen, formal eingeführt und seine Komplexität untersucht. In Abschnitt 3.5 werden polynomielle Algorithmen für einige Spezialfälle angegeben, darunter der Fall $k = 2$, der schon in [26] untersucht wurde, und der Fall azyklischer Graphen. Anschließend wird in Abschnitt 3.5.5 der wichtigste Approximationsalgorithmus aus [26] vorgestellt, und ein Beispiel angegeben, das zeigt, dass der dort angegebene Approximatiosfaktor scharf ist. Zum Abschluss wird in Abschnitt 3.6 eine Formulierung als ganzzahliges lineares Programm hergeleitet und die Korrektheit bewiesen. Danach wird gezeigt, dass man nicht erwarten kann, damit neue Approximationsalgorithmen herleiten zu können.

## 3.2 Existenz kantendisjunkter Wege

**Satz 2.** *Sei $D = (V, A)$ ein gerichteter Graph, $s, t \in V$ zwei Knoten und $k \in \mathbb{N}$. Es existieren genau dann $k$ kantendisjunkte Wege von $s$ nach $t$ wenn es im Netzwerk $N(V, A, s, t, 1)$ mit Quelle $s$, Senke $t$ und Kapazität 1 für alle Kanten einen $s$-$t$-Fluss mit Wert $k$ gibt.*

*Beweis.* Seien $P = \{P_1, \ldots, P_k\}$ $k$ kantendisjunkte Wege von $s$ nach $t$. Definiere eine

Abbildung $f : A \to \mathbb{R}$ durch

$$f(a) = \begin{cases} 1 & \text{falls } a \in E(P) \\ 0 & \text{sonst} \end{cases}$$

Behauptung: $f$ ist ein Fluss mit Wert $k$
Die Kapazitätsbedingung ist erfüllt, da $0 \leq f(a) \leq 1$ für alle $a \in A$. Die Flusserhaltung folgt aus der Kantendisjunktheit der Wege in $P$: Für alle Knoten $v \in V \setminus \{s, t\}$ ist die Anzahl der Wegkanten, die in $v$ hineingehen gleich der Anzahl der Wegkanten, die aus $v$ herauskommen. Aus $s$ kommen $k$ mehr Kanten heraus, als hineingehen und in $t$ gehen $k$ mehr Kanten hinein als heraus. Also gilt für einen Knoten $v \in V$

$$\begin{aligned} \sum_{(v,u)\in A} f(v,u) - \sum_{(u,v)\in A} f(u,v) &= \sum_{(v,u)\in E(P)} 1 - \sum_{(u,v)\in E(P)} 1 \\ &= \mid \{(v,u) \in E(P)\} \mid - \mid \{(u,v) \in E(P)\} \mid \\ &= \begin{cases} k & \text{falls } v = s \\ -k & \text{falls } v = t \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

und es folgt die Flusserhaltung für $f$ und der Wert $k$.
Ist umgekehrt ein Fluss $f'$ im Netzwerk $N$ gegeben, so gibt es auch einen ganzzahligen Fluss $f_k$ mit Wert $k$, da alle Kapazitäten ganzzahlig sind. Dieser kann mit der Methode von Ford und Fulkerson berechnet werden.
Aus dem Fluss $f_k$ können nun sukzessive $k$ Wege berechnet werden:

- Berechne den Graphen $D_k = (V, A')$ mit $A' = \{a \in A \mid f_k(a) = 1\}$
- Finde einen $s$-$t$-Weg $P_k$ in $D_k$. Aufgrund der Flusseigenschaft von $f_k$ muss dieser existieren.
- Setze
$$f_{k-1}(a) = \begin{cases} f_k(a) - 1 & \text{falls } a \in P_k \\ f_k(a) & \text{sonst} \end{cases}$$
- $f_{k-1}$ ist ein $s$-$t$-Fluss mit Wert $k - 1$
- Berechne $D_{k-1}$ und wiederhole die obigen Schritte usw.

Indem das obige Verfahren $k$ mal angewandt wird, können $k$ Wege konstruiert werden. Die so konstruierten Wege sind kantendisjunkt, denn gäbe es eine Kante $a \in P_i \cap P_j$ für $i \neq j$, so wäre $f(a) \geq 2$. Dies widerspricht jedoch der Kapazitätsbedingung. □

**Satz 3.** *In einem gerichteten Graphen $D = (V, A)$ gibt es genau dann $k$ kantendisjunkte Wege von $s$ nach $t$, wenn für die Kapazität jedes $s$-$t$-Schnittes $(S, V \setminus S)$ $c(S, V \setminus S) \geq k$ gilt.*

*Beweis.* Die Aussage folgt mit dem Max-Flow-Min-Cut-Theorem aus Satz 2. □

Offensichtlich lässt sich die maximale Anzahl kantendisjunkter Wege mit Hilfe eines Algorithmus zur Berechung maximaler Flüsse effizient berechnen. Die Anzahl kantendisjunkter Wege ist natürlich auch durch den Ausgangsgrad von $s$ und den Eingangsgrad von $t$ beschränkt. Da es im Energiekostengraphen keine Mehrfachkanten gibt, ist $d_{\text{out}}(s) \leq \mid V \mid -1$ und $d_{\text{in}}(t) \leq \mid V \mid -1$. Die Anzahl kantendisjunkter Wege von $s$ nach $t$ ist also auf jeden Fall durch $\mid V \mid -1$ beschränkt.

In [6] wird der folgende Greedy-Algorithmus zum Berechnen kantendisjunkter Wege vorgeschlagen: Es wird der kürzeste Weg vom Sender zum Empfänger bestimmt und die zu diesem Weg gehörenden Kanten aus dem Graphen entfernt. Nun wird wieder der kürzeste Weg gesucht usw. Dies wird solange wiederholt, bis die geünschte Anzahl von Wegen gefunden wurde oder es keinen Weg vom Sender zum Empfänger mehr gibt.

In Abbildung 3.1 ist ein Beispiel abgebildet, das zeigt, dass dieser Algorithmus nicht korrekt ist.

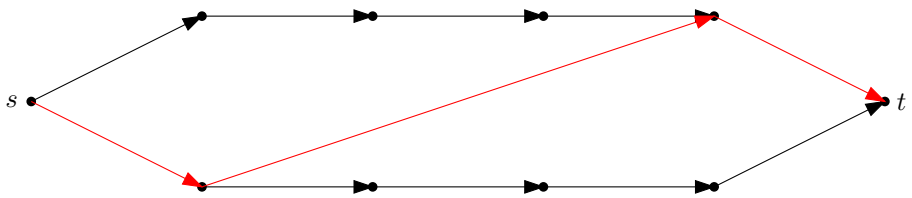



Abbildung 3.1: In diesem Graphen haben alle Kanten das gleiche Gewicht. Der Greedy-Algorithmus bestimmt im ersten Schritt den kürzesten Weg von $s$ nach $t$. Dies ist der rot eingezeichnete Weg. Im zweiten Schritt findet er dann keinen Weg mehr von $s$ nach $t$. Die maximale Anzahl kantendisjunkter Wege im Beispiel ist aber offensichtlich zwei.

### 3.2.1 Inklusionsminimale kantendisjunkte Wege

Im Allgemeinen sind $k$ kantendsjunkte Wege minimaler Energie nicht eindeutig. Wie in Abbildung 3.2 zu sehen, ist es sogar möglich, dass es zwei optimale Lösungen $P$ und $P'$ gibt, bei denen die Kanten der einen Lösung eine Teilmenge der Kanten der anderen Lösung bilden, also $E(P) \subseteq E(P')$.

**Definition 6.** Eine Menge $P$ von $k$ kantendisjunkten Wegen ist inklusionsminimal, falls für alle Mengen $P'$ von $k$ kantendisjunkten Wegen mit $\mathcal{E}(P') = \mathcal{E}(P)$ und $E(P') \subseteq E(P)$ folgt $E(P') = E(P)$.

Auch wenn in Abbildung 3.2 beide Lösungen vom Standpunkt der Energie aus betrachtet optimal sind, gibt es doch Gründe, eine inklusionsminimale Lösung zu bevorzugen:

- In Anwendungen stellt jede zusätzliche Kante ein zusätzliches Ausfallrisiko dar, so dass das Ausfallrisiko einer Wegemenge in der inklusionsminimalen Lösung minimiert wird.

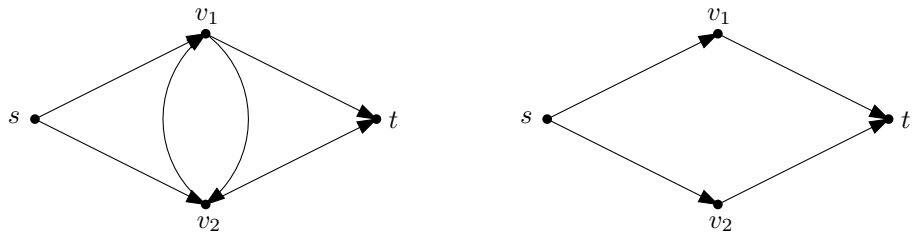



Abbildung 3.2: Falls in diesem Beispiel alle Kanten dasselbe Gewicht haben, bilden sowohl die Kanten im linken als auch die Kanten im rechten Bild 2 kantendisjunkte Wege minimaler Energie. Die Wege im rechten Bild sind inklusionsminimal.

- Beweise werden vereinfacht, in dieser Arbeit z. B. die Beweise in Kapitel 4

Wie kann zu einem gewichteten gerichteten Graph $D = (V, A)$ mit Gewichtsfunktion $w : A \to \mathbb{R}^+$ und einer Menge $P$ von $k$ kantendisjunkten Wegen eine inklusionsminimale Wegemenge $P^*$ mit $\mathcal{E}(P^*) \leq \mathcal{E}(P)$ gefunden werden?
Dieses Problem kann mittels Flussmethoden algorithmisch effizient gelöst werden.

**Definition 7.** Die $k$ Wege $P = \{P_1, \ldots, P_k\}$ induzieren Sendeenergien $e(v)$ Knoten: $e(v) = \max_{(v,u)\in E} w((v, u))$. Den von dieser Knotenbelegung induzierten Subgraphen $D_P = (V_P, A_P)$ nennen wir den *von den Wegen $P$ induzierten Subgraphen.*

*Bemerkung.* Es gilt $P \subseteq D_P$ und $\sum_{v\in V_P\setminus\{t\}} e(v) = \mathcal{E}(P)$.

**Satz 4.** *Sei $D = (V, A)$ ein gewichteter gerichteter Graph mit Gewichtsfunktion $w : A \to \mathbb{R}^+$. $P$ sei eine Menge von $k$ kantendisjunkten Wegen von einem Startknoten $s \in V$ zu einem Zielknoten $t \in V$.*
*$D_P = (V_P, A_P)$ bezeichne den von den Wegen $P$ induzierten Subgraphen von $D$ und $N = (D_P, s, t, l, u, c, k)$ ein Minimalkostenflussproblem mit unteren Kantenkapazitäten $l \equiv 0$, oberen Kantenkapazitäten $u \equiv 1$ und Kostenfunktion $c \equiv 1$ und Wert $k$.*
*Dann existiert in $N$ ein Fluss $f$ mit Wert $k$ und minimalen Kosten. Außerdem gibt es eine Menge $P^*$ von $k$ kantendisjunkten Wegen mit $E(P^*) = \{a \in A_P \mid f(a) > 0\}$ und $P^*$ ist inklusionsminimal.*

*Beweis.* Zuerst zeigen wir die Existenz des Flusses $f$ mit Wert $k$. Definiere

$$f'(e) = \begin{cases} 1 & \text{falls } e \in E(P) \\ 0 & \text{sonst} \end{cases}$$

$f'$ ist ein $s$-$t$-Fluss mit Wert $k$. Dann existiert auch ein $s$-$t$-Fluss $f$ mit Wert $k$ und minimalen Kosten. Dieser ist (nach bekannten Sätzen aus der Flusstheorie) ganzzahlig und kann mit den üblichen Methoden zum Lösen von Minimalkostenflussproblemen algorithmisch effizient berechnet werden (siehe z. B. [16]).
Aus dem ganzzahligen Fluss $f$ mit Wert $k$ kann analog zu der Konstruktion im Beweis von Satz 2 eine Menge $P^*$ von $k$ kantendisjunkten Wegen konstruiert werden, wobei

$E(P^*) = \{a \in A_P \mid f(a) > 0\}$ gilt.
Es bleibt noch die Inklusionsminimalität von $P^*$ zu zeigen. Dazu nehmen wir an, dass es $k$ kantendisjunkte Wege $\bar{P}$ mit $E(\bar{P}) \subset E(P^*)$ gibt. Der Fluss

$$\bar{f}(e) = \begin{cases} 1 & \text{falls } e \in E(\bar{P}) \\ 0 & \text{sonst} \end{cases}$$

ist ein $s$-$t$-Fluss mit Wert $k$ und $\text{cost}(\bar{f}) =\mid E(\bar{P}) \mid < \mid E(P^*) \mid = \text{cost}(f)$. Dies steht im Widerspruch dazu, dass $f$ ein Fluss mit minimalen Kosten ist. Die Wege $P^*$ sind also inklusionsminimal. □

## 3.3 Kantendisjunkte Wege minimalen Gewichts

Wie wir in Abschnitt 3.5.5 sehen werden, beruht einer der Approximationsalgorithmen für das Problem, kantendisjunkte Wege minimaler Energie zu finden, auf einer exakten Lösung des Problems, kantendisjunkte Wege minimalen Gewichts zu finden. Außerdem wird in der exakten Lösung für den Fall $k = 2$ in Abschnitt 3.5.1 ein Algorithmus zur Berechnung von Paaren kantendisjunkter Wege minimalen Gewichts von einer Quelle $s$ zu allen anderen Knoten des Graphen benötigt. Die Eigenschaften dieser beiden Algorithmen sollen hier kurz vorgestellt werden.

Zur Berechnung von kantendisjunkten Wegen minimalen Gewichts kann der Algorithmus von Suurballe, der auch zur Berechnung knotendisjunkter Wege verwendet wird, mit leichten Modifikationen benutzt werden. Er hat eine Laufzeit in $O(kn^2)$ [27].

Sollen Paare kantendisjunkter Wege minimalen Gewichts von einer Quelle $s$ zu allen anderen Knoten berechnet werden, so ist dies mit einem Algorithmus von Suurballe und Tarjan in Zeit $O(m \log_{1+m/n} n)$ und mit einem Speicherbedarf in $O(m)$ möglich [28].

## 3.4 Kantendisjunkte Wege minimaler Energie: $k$-EDPWMA

Das Problem, $k$ kantendisjunkte Wege minimaler Energie zu berechnen, wird mit $k$-EDPWMA bezeichnet. Es kann als Entscheidungs- oder Optimierungsproblem formuliert werden. Die folgende Definition formuliert $k$-EDPWMA als Entscheidungsproblem.

**Definition 8.** ($k$-EDPWMA) Gegeben gewichteter gerichteter Graph $D = (V, A)$ mit Gewichtsfunktion $w : A \to \mathbb{R}^+$, $s, t \in V$ und $B \in \mathbb{N}$.
Gibt es $k$ kantendisjunkte Wege $P = \{P_1, \ldots, P_k\}$ mit $\mathcal{E}(P) \leq B$.

Im Beweis der NP-Vollständigkeit von $k$-EDPWMA wird sich der folgende Satz nützlich erweisen:

**Satz 5.** *Unter den Voraussetzungen aus Definition 8 gilt: Es existieren genau dann $k$ kantendisjunkte Wege $P = \{P_1, \ldots, P_k\}$ von $s$ nach $t$ mit $\mathcal{E}(P) \leq B$, wenn es eine Funktion $e : V \setminus \{t\} \to \mathbb{R}$ gibt, so dass es in dem von $e$ induzierten Subgraphen $D_e$ $k$ kantendisjunkte Wege von $s$ nach $t$ gibt und $\sum_{v \in V \setminus \{t\}} e(v) \leq B$.*

*Beweis.* Seien $P = \{P_1, \ldots, P_k\}$ $k$ kantendisjunkte Wege von $s$ nach $t$ mit Energie $B$. Für die von den Wegen $P$ induzierte Knotenbelegungsfunktion $e_P$ gilt: $\sum_{v \in V_P \setminus \{t\}} e(v) = \mathcal{E}(P) \leq B$. Außerdem $E(P) \subseteq D_P$, es existieren also $k$ kantendisjunkte Wege von $s$ nach $t$.

Sei umgekehrt $e$ eine Knotenbelegungsfunktion wie im Satz gefordert und $P = \{P_1, \ldots, P_k\}$ die $k$ kantendisjunkten Wege. Es ist klar, dass $\mathcal{E}(P) \leq B$. □

Der Beweis des folgenden Satzes ist eine leichte Abwandlung eines von Steffen Mecke vorgeschlagenen Beweises:

**Satz 6.** *$k$-EDPWMA ist NP-vollständig.*

*Beweis.* Es ist klar, dass $k$-EDPWMA in NP: Wähle nichtdeterministisch für jeden Knoten $v \in V \setminus \{t\}$ eine Energie $e(v) \in [0, \mathcal{E}_{\max}(v)]$. Dann kann in polynomieller Zeit nachgeprüft werden, ob $\sum_{v \in V \setminus \{t\}} e(v) = B$ und ob es in dem von $e$ indzuierten Subgraphen $k$ kantendisjunkte Wege von $s$ nach $t$ gibt. Dies kann zum Beispiel mit Hilfe von Satz 2 und Flussmethoden erfolgen.
Der Beweis der NP-Vollständigkeit erfolgt durch Reduktion von SET COVER auf $k$-EDPWMA. Die folgende Definition des SET COVER Problems ist aus [20] entnommen.

**Definition 9.** SET COVER: Gegeben eine endliche Menge $U = \{u_1, \ldots, u_m\}$, eine Familie $F = \{S_1, \ldots, S_n\}$ von Teilmengen von $U$, also $S_i \subseteq U$ für alle $i = 1, \ldots, n$ und eine Zahl $B \in \mathbb{N}$. Gibt es $B$ Mengen aus $F$, deren Vereinigung $U$ ergibt?

Sei also nun eine Instanz von SET COVER wie in der obigen Definition gegeben. Zu dieser Instanz definieren wir einen gewichteten gerichteten Graphen $D = (V, A)$ mit Knotenmenge $V = U \cup F \cup V_1 \cup \ldots \cup V_n \cup \{s, t\}$, wobei $V_i = \{v_{i,j} \mid u_j \in S_i\}$ für $(i = 1, \ldots, n)$, Kantenmenge $A = A_1 \cup A_2 \cup A_3 \cup A_4$ mit

$$A_1 = \{(s, v_{i,j}) \mid i = 1, \ldots, n; j = 1, \ldots, \mid S_i \mid\} \tag{3.1}$$
$$A_2 = \{(v_{i,j}, S_i) \mid i = 1, \ldots, n; j = 1, \ldots, \mid S_i \mid\} \tag{3.2}$$
$$A_3 = \{(S_i, u_j) \mid u_j \in S_i; i = 1, \ldots, n; j = 1, \ldots, m\} \tag{3.3}$$
$$A_4 = \{(u_i, t) \mid i = 1, \ldots, m\}. \tag{3.4}$$

und Kantengewichten

$$w(e) = \begin{cases} 1 & \text{falls } e \in A_1 \text{ oder } e \in A_2 \text{ oder } e \in A_4 \\ Q & \text{falls } e \in A_3 \end{cases}.$$

für ein $Q \in \mathbb{N}$. Für die Zwecke dieses Beweises könnte $Q = 1$ gesetzt werden. Allerdings wird diese Konstruktion noch im Beweis zu Satz 8 benötigt und dort muss das Gewicht

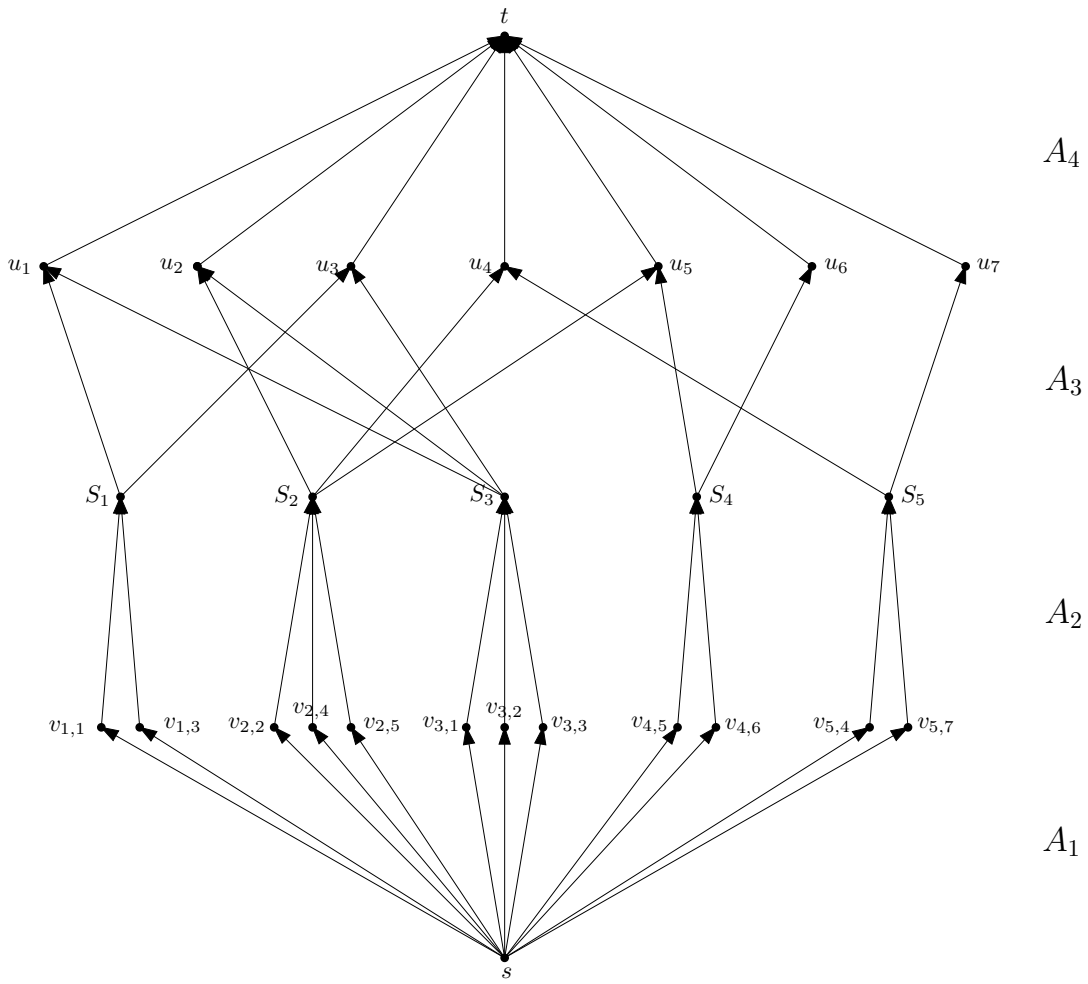



Abbildung 3.3: Reduktion des SET COVER Problems mit $U = \{u_1, \ldots, u_7\}$, $F = \{S_1, \ldots, S_5\}$, $S_1 = \{u_1, u_3\}$, $S_2 = \{u_2, u_4, u_5\}$, $S_3 = \{u_1, u_2, u_3\}$, $S_4 = \{u_5, u_6\}$, $S_5 = \{u_4, u_7\}$ auf k-EDPWMA.

$Q$ wählbar sein.
Ein Beispiel eines solchen Graphen ist in Abbildung 3.3 dargestellt.

Behauptung: Es existiert genau dann ein SET COVER mit $B$ Mengen, wenn es im Graphen $D$ eine Menge $P = \{P_1, \ldots, P_m\}$ von $m$ kantendisjunkten Wegen mit Energie $\mathcal{E}(P) \leq QB + 2m + 1$ gibt.
Vorbemerkung: Eine Menge $P'$ von $l$ kantendisjunkten Wegen von $s$ nach $t$ hat Energie $\mathcal{E}(P') = 2l + 1 + Q \mid \{S \in F \mid S \in P'\} \mid$, da der Knoten $s$ und genau $l$ Knoten in $V_1, \ldots, V_n$ bzw. $U$ mit Energie 1 senden und $\mid \{S \in F \mid S \in P'\} \mid$ mit Energie $Q$.
Wenn es ein SET COVER $S_{i_1}, \ldots, S_{i_B}$ mit $B$ Mengen gibt, definiere die Abbildung $\phi(u) = \{j \mid j \in i_1, \ldots, i_B; u \in S_j; j\text{minimal}\}$, die jedem Element der Menge $U$ die Menge der Überdeckung mit dem kleinsten Index zuweist. Für jedes $u_j$ definiere nun einen Weg $P_j =< s, v_{\phi(u_j),j}, S_{\phi(u_j)}, u_j, t >$ $(j = 1, \ldots, m)$. $P = \{P_1, \ldots, P_m\}$ sind $m$ Wege, die nach der Vorbemerkung Energie $\mathcal{E}(P) \leq QB + 2m + 1$ benötigen.

Gibt es andererseits $m$ Wege $P = P_1, \ldots, P_m$ mit Energie $\mathcal{E}(P) = QB + 2m + 1$, so besuchen diese nach der Vorbemerkung genau $B$ der Mengen aus $F$. Diese werden mit $S_{i_1}, \ldots, S_{i_B}$ bezeichnet. Aus der kantendisjunktheit von $P$ folgt, dass jedes $u \in U$ von genau einem Weg $P_u$ besucht wird. Definiere $\phi(u) = \{S \mid (S, u) \in P_u\}$. Die Abbildung $\phi$ ordnet also jedem $u \in U$ eine Menge in $S_{i_1}, \ldots, S_{i_B}$ zu, die $u$ enthält. $S_{i_1}, \ldots, S_{i_B}$ überdecken somit $U$ und es existiert ein SET COVER der Größe $B$. □

Uriel Feige konnte in [9] eine untere Schranke für die Approximierbarkeit von SET COVER zeigen, falls eine komplexitätstheoretische Behauptung, die gemeinhin als wahr angenommen wird, gilt:

**Satz 7.** *Falls es ein $\epsilon > 0$ gibt, so dass ein Polynomialzeit-Approximationsalgorithmus für SET COVER mit Approximationsfaktor $(1 - \epsilon) \ln n$ existiert, dann gilt:*

$$\text{NP} \subseteq \text{TIME}(n^{O(\log \log n)}).$$

Daraus und aus der Reduktion von SET COVER auf $k$-EDPWMA folgt der folgende Satz über die Approximierbarkeit von $k - EDPWMA$:

**Satz 8.** *Falls es ein $\epsilon > 0$ gibt, so dass ein Polynomialzeit-Approximationsalgorithmus für $k$-EDPWMA mit Approximationsfaktor $(1 - \epsilon) \ln k$ existiert, dann gilt:*

$$\text{NP} \subseteq \text{TIME}(n^{O(\log \log n)}).$$

*Beweis.* Angenommen es gibt einen Polynomialzeit-Approximationsalgorithmus für $k$-EDPWMA mit Approximationsfaktor $(1 - \epsilon) \ln k$ für ein $\epsilon > 0$.
Behauptung: Dann gibt es auch einen Polynomialzeit-Approximationsalgorithmus für SET COVER mit Approximationsfaktor $(1 - \epsilon) \ln n$.
Beweis: Sei eine Instanz $I$ von SET COVER gegeben, also eine endliche Menge $U = \{u_1, \ldots, u_m\}$ und eine Familie $F = \{S_1, \ldots, S_n\}$ von Teilmengen von $U$. Es gebe ein SET COVER der Größe $B$, also $\text{OPT}(I) = B$.
Sei $D = (V, A; w)$ der Graph, der bei der Reduktion von SET COVER auf $k$-EDPWMA im Beweis von Satz 6 beschrieben ist, wobei $Q = \frac{2((1-\epsilon)\ln m - 1)(2m+1)}{\epsilon \ln m}$. Da es ein SET COVER der Größe $B$ gibt, gibt es $m$ kantendisjunkte Wege in $D$ mit Energie $2m+1+BQ$. Der Approximationsalgorithmus berechnet also $m$ kantendisjunkte Wege $P'$ mit $\mathcal{E}(P') \leq (1 - \epsilon) \ln m(2m + 1 + BQ)$. Aus den Vorbemerkungen im Beweis von Satz 6 folgt über die Anzahl der von $P'$ benutzten Knoten aus $S_1, \ldots, S_n$:

$$\begin{aligned}
\mid V(P') \cap S_1 \cap \ldots \cap S_n \mid &= \frac{\mathcal{E}(P') - (2m+1)}{Q} \\
&\leq \frac{(1-\epsilon)\ln m(2m+1+BQ) - (2m+1)}{Q} \\
&= (1-\epsilon)\ln mB + \frac{((1-\epsilon)\ln m - 1)(2m+1)}{Q} \\
&= (1-\epsilon)\ln mB + \frac{\epsilon \ln m}{2} \leq (1-\epsilon)\ln mB + \frac{\epsilon}{2}\ln mB \\
&= (1 - \frac{\epsilon}{2})\ln mB
\end{aligned}$$

Die Mengen aus $V(P') \cap S_1 \cap \ldots \cap S_n$ bilden eine Approximation $A(I)$ für die Instanz $I$ von SET COVER. Es gilt:

$$\mid A(I) \mid = \mid V(P') \cap S_1 \cap \ldots \cap S_n \mid \leq (1 - \frac{\epsilon}{2}) \ln m \mathrm{OPT}(I)$$

Es gibt also einen Polynomialzeit-Approximationsalgorithmus für SET COVER mit Approximationsfaktor $(1 - \delta) \ln m$ für ein $\delta = \frac{\epsilon}{2} > 0$. Aus Satz 7 folgt dann NP $\subseteq$ TIME($n^{O(\log \log n)}$). □

### 3.4.1 Ein exponentieller Algorithmus für $k$-EDPWMA

Da $k$-EDPWMA NP-vollständig ist, kann es nur dann einen polynomiellen Algorithmus geben, wenn P=NP. In diesem Abschnitt soll ein exponentieller Algorithmus vorgestellt werden.

Sei also ein gewichteter gerichteter Graph $D = (V, A)$ mit Gewichtsfunktion $w : A \to \mathbb{R}^+$ sowie Start- und Zielknoten $s, t \in V$ gegeben. Wir setzen $n = |V|$ und $m = |A|$. Gesucht seien $k$ kantendisjunkte Wege von $s$ nach $t$.

Für einen Knoten $v_i$ seien $N_i = \{0, w(v, u) \mid (v, u) \in A\}$ die verschiedenen Energiestufen, mit denen er senden muss, um seine Nachbarn zu erreichen. Es ist klar, dass $\mid N_i \mid \leq n$.

Im Algorithmus 1 bezeichne $e_{\min}$ die Knotenbelegung mit der minimalen Energie und $\mathcal{E}_{\min}$ den Wert der minimalen Energie. Für jede mögliche Kombination der Sendeenergien der Knoten $e = (e_1, \ldots, e_n) \in N_1 \times \ldots \times N_n$ wird geprüft, ob die benötigte Energie kleiner ist als die Energie der besten bisher gefundenen Belegung. Falls dies der Fall ist wird der durch $e$ induzierte Subgraph $D_e$ berechnet (vgl. Definition 2). Falls es in $D_e$ einen $s$-$t$-Fluss mit Wert $k$ gibt, so gibt es $k$ kantendisjunkte Wege. $e$ ist also besser als das bisher gefundene $e_{\min}$ und folglich werden $e_{\min}$ und $\mathcal{E}_{\min}$ auf die aktuell betrachteten Werte gesetzt.

**Algorithmus 1** Exponentieller Algorithmus

```
E_min = ∞
e_min = (e_min^(1), ..., e_min^(n))
for all e = (e^(1), ..., e^(n)) ∈ N_1 × ... × N_n do
  E = Σ_{i=1}^{n} e^(i)
  if E < E_min then
    D_e ← berechneInduziertenSubgraphen(D, e)
    if es gibt s-t-Fluss mit Wert k in D_e then
      e_min = e
      E_min = Σ_{i=1}^{n} e^(i)
    end if
  end if
end for
```

Die Anzahl verschiedener Kombinationen von Sendeenergien ist

$$\mid N_1 \times \ldots \times N_n \mid = N_1 N_2 \ldots N_n \leq d_{\mathrm{out}}(v_1) \ldots d_{\mathrm{out}}(v_n) \in O(n^n)$$

Im schlimmsten Fall wird für jede davon der induzierte Subgraph berechnet und auf diesem ein Flussalgorithmus ausgeführt. Wird der Edmonds-Karp-Algorithmus verwendet, erfolgt dies in einer Zeit in $O(nm^2)$.
Insbesondere ergibt sich schon eine Laufzeit in $O(2^n)$, falls jeder Knoten nur mit 1 Energiestufe senden bzw. nicht senden kann.

## 3.5 Polynomielle Algorithmen für Spezialfälle

Da das Problem $k$-EDPWMA NP-vollständig ist, kann es nur dann polynomielle Algorithmen für das allgemeine Problem geben, wenn P=NP. Deshalb ist man an Algorithmen für Spezialfälle interessiert. Auf der einen Seite gibt es die Möglichkeit, ein festes $k_0 \in \mathbb{N}$ zu wählen. In Abschnitt 3.5.1 wird ein Algorithmus für 2-EDPWMA aus [26] beschrieben. Für $k_0 \geq 3$ ist die Komplexität von $k_0$-EDPWMA für allgemeine gerichtete Graphen unbekannt. Auf der anderen Seite gibt es die Möglichkeit, beliebiges $k$ aber spezielle Graphen zu betrachten. In Abschnitt 3.5.2 ist so ein Algorithmus für eine (sehr) spezielle Klasse von Graphen angegeben. Schließlich ist auch eine Kombination von beidem möglich, d. h. es werden Algorithmen für spezielle Graphen und feste $k_0$ gesucht. In Abschnitt 3.5.3 werden polynomielle Algorithmen für festes $k_0 \in \mathbb{N}$ und azyklische Graphen vorgestellt.

### 3.5.1 Der Fall $k = 2$

Für den Fall $k = 2$ ist in [26] der OCND-Algorithmus (Optimal Common Node Decomposition) angegeben, der für das 2-EDPWMA-Problem für einen gerichteten gewichteten Graphen $D = (V, A)$ mit $n$ Knoten in Zeit $O(n^5)$ eine Lösung berechnet.

Die Idee, die hinter dem OCND-Algorithmus steht, ist die folgende: Es wird gezeigt, dass 2 kantendisjunkte Wege in ihre Teilstücke zwischen den Knoten, die von beiden Wegen besucht werden, zerlegt werden können. Die Knoten, die von beiden Wegen besucht werden, sind aber die einzigen Knoten, an denen der WMA ausgenutzt werden kann. Zwischen diesen Knoten entsprechen kantendisjunkte Wege minimaler Energie also knotendisjunkten Wegen minimaler Energie.

- Berechne für jedes Knotenpaar $(u, v)$ zwei knotendisjunkte Wege minimaler Energie $P_{(u,v)}$ von $u$ nach $v$.

- Konstruiere den gewichteten gerichteten Hilfsgraphen $H = (V, A')$ mit

  $$A' = \{(u, v) | \text{es gibt zwei knotendisjunkte Wege von } u \text{ nach } v\}$$
  $$w(u, v) = \mathcal{E}(P_{(u,v)}).$$

- Finde in $H$ den kürzesten Weg von $s$ nach $t$. Durch Zusammensetzen der Wegstücke ergeben sich zwei kantendisjunkte Wege von $s$ nach $t$.

Falls zur Berechnung der knotendisjunkten Wege minimaler Energie zwischen je zwei Knoten der STPS-Algorithmus aus Abschnitt 2.3.1, dem der Algorithmus von Suurballe aus [27] zugrundeliegt, verwendet wird, ergibt sich eine Laufzeit in $O(n^5)$. Da wir jedoch die knotendisjunkten Wege zwischen allen Paaren von Knoten berechnen möchten, ergibt sich eine Laufzeitverbesserung, wenn stattdessen ein Algorithmus zur Berechnung zweier knotendisjunkter Wege von einem Knoten zu allen anderen Knoten in einem Graphen verwendet wird. Diesem Algorithmus kann der Algorithmus von Suurballe und Tarjan [28] zugrundegelegt werden. Dadurch verbessert sich die Gesamtlaufzeit des OCND-Algorithmus auf $O(n^4 \log n)$.

Leider wurde bisher keine Möglichkeit gefunden, die Idee des OCND-Algorithmus Algorithmus auf $k > 2$ zu übertragen.

### 3.5.2 Bäume

Für den Spezialfall, dass der Graph, der entsteht, wenn der Knoten $t$ und alle zu $t$ hinführenden oder von $t$ wegführenden Kanten entfernt werden, ein Baum ist, gibt es einen effizienten Algorithmus für $k$-EDPWMA. Da in Bäumen jeder Knoten höchstens einen Vorgänger hat, folgt für alle Knoten außer $s$ und $t$, dass sie auf höchstens einem kantendisjunkten Weg von $s$ nach $t$ liegen können. Das bedeutet aber, dass aus der Kantendisjunktheit Knotendisjunktheit folgt (umgekehrt sowieso) und es deshalb genügt, $k$ knotendisjunkte Wege zu berechnen. Dafür gibt es jedoch, wie in Abschnitt 2.3 gezeigt, effiziente Algorithmen. Andererseits ist klar, dass bei Bäumen der einzige Knoten, an dem der WMA zur Energieeinsparung ausgenutzt werden kann, die Wurzel ist. Es lohnt sich deshalb nicht, diesen Spezialfall weiter zu verfolgen.

### 3.5.3 Azyklische gerichtete Graphen

In diesem Abschnitt wird gezeigt, dass es für festes $k_0 \in \mathbb{N}$ einen Algorithmus gibt, der azyklische Instanzen von $k_0$-EDPWMA in polynomieller Zeit löst. Dazu wird zuerst gezeigt, dass jeder azyklische Graph in polynomieller Zeit in einen sogenannten eigentlichen Lagengraphen transformiert werden kann und die kantendisjunkten Wege im tranformierten Graphen wieder auf kantendisjunkte Wege im ursprünglichen Graphen abgebildet werden können, wobei die Energie sich nicht ändert. Anschließend wird der $EAS$-Algorithmus angegeben, der für festes $k_0 \in \mathbb{N}$ das $k_0$-EDPWMA auf Lagengraphen in polynomieller Zeit löst.

Die folgenden Definitionen sind an die Definitionen in [7] angelehnt.

**Definition 10.** Sei $D = (V, A)$ ein azyklischer gerichteter Graph. Eine *Lagenzuweisung* ist eine Partition von $V$ in Teilmengen $L_1, \ldots, L_h$, so dass aus $(u, v) \in A$ mit $u \in L_i$ und $v \in L_j$ folgt: $i < j$.
Die Abbildung $\pi : V \to \{1, \ldots, h\}$ ordne jedem Knoten in einem Lagengraphen seine Lage zu: $\pi(v) = i \Leftrightarrow v \in L_i$.
Das Tupel $(D, \pi)$, also einen azyklischen gerichteten Graphen versehen mit einer Lagenzuweisung nennen wir einen *Lagengraphen*.

$(D, \pi)$ ist ein *eigentlicher Lagengraph*, falls für alle Kanten $(u, v) \in E$ $\pi(v) - \pi(u) = 1$, falls also alle Kanten nur von einer Lage zur nächsten gehen.

**Satz 9.** *Jeder azyklische gerichtete Graph $D = (V, A)$ erlaubt eine Lagenzuweisung.*

*Beweis.* Jeder gerichtete azyklische Graph erlaubt eine topologische Ordnung seiner Knoten. Diese kann durch eine einfache Erweiterung der Tiefensuche in Zeit $O(n + m)$ berechnet werden [5], wobei $n$ die Anzahl der Knoten und $m$ die Anzahl der Kanten von $D$ bezeichne. Sei $V = \{v_1, \ldots, v_n\}$ und $\sigma : \{1, \ldots n\} \to \{1, \ldots n\}$ die zur topologischen Ordnung gehörende Permutation. Aus den Eigenschaften der topologischen Ordnung folgt dann, das $\sigma(i) < \sigma(j)$ für jede Kante $(v_i, v_j) \in A$ . Definiert man Lagen $L_1, \ldots, L_n$ und setzt $\pi(v_i) = L_{\sigma(i)}$ $(i = 1, \ldots, n)$, so erhält man also eine zulässige Lagenzuweisung. □

Der obige Beweis ist zwar konstruktiv und erlaubt eine einfache Berechnung einer Lagenzuweisung, doch hat diese den Nachteil, das jeder Knoten in einer eigenen Lage liegt. Aus der topologischen Sortierung folgt die Existenz von Quellen, d. h. von Knoten, die nur ausgehende aber keine eingehenden Kanten haben. In [7] wird die Longest-Path-Lagenzuweisung beschrieben, bei der die Quellen auf die Lage $L_1$ gesetzt werden und jeder andere Knoten auf die Lage, die der Länge eines längsten Weges von einer Quelle zu ihm entspricht. Ist die Länge eines längsten Weges von einer Quelle zu einem Knoten $v$ also $l$ Kanten, so wird $\pi(v) = l$ gesetzt. Diese Lagenzuweisung ist korrekt, denn für jede Kante $(u, v)$ gilt natürlich, dass die Länge eines längsten Weges von $s$ zu $v$ mindestens um 1 größer sein muss als die Länge eines längsten Weges von $s$ zu $u$ (nämlich um diese Kante). Also $\pi(u) < \pi(v)$. Auch die Longest-Path-Lagenzuweisung kann in linearer Zeit berechnet werden. Die Länge eines Weges bedeutet hier die Anzahl der Kanten auf dem Weg (nicht zu verwechseln mit Gewicht oder Energie).

**Satz 10.** *Sei $(D = (V, A), \pi)$ ein Lagengraph mit $n$ Knoten und $m$ Kanten und Gewichtsfunktion $w : A \to \mathbb{R}^+$. Dann kann in Zeit $O(mn)$ ein eigentlicher Lagengraph $(D' = (V' \subseteq V, A'), \pi')$ mit Gewichtsfunktion $w' : A' \to \mathbb{R}^+$, $O(mn)$ Knoten und $O(mn)$ Kanten konstruiert werden, so dass es für $s, t \in V$ eine bijektive Abbildung $\Phi$ von den $s$-$t$-Wegen in $D$ auf die $s$-$t$-Wege in $D'$ gibt mit $\mathcal{E}(P) = \mathcal{E}(\Phi(P))$ für alle Mengen $P$ von kantendisjunkten $s$-$t$-Wegen.*

*Beweis.* Der eigentliche Lagengraph $(D' = (V', A'), \pi')$ wird wie folgt konstruiert:

$$V' = V \cup \bigcup_{e \in A} V_e$$

$$A' = \bigcup_{e \in A} A_e$$

Für jede Kante $e = (u, v) \in A$ wird $d(e) = \pi(v) - \pi(u) - 1$ gesetzt und

$$V_e = \{v_{e,1}, \ldots, v_{e,d(e)}\}$$

$$A_e = \begin{cases} \{(u, v)\} & \text{falls } d(e) = 0 \\ \{(u, v_{e,1}), (v_{e,1}, v_{e,2}), \ldots (v_{e,d}, v)\} & \text{sonst} \end{cases}$$

definiert. Die Lagenzuweisung der Knoten wird durch

$$\pi'(v) = \begin{cases} \pi(v) & \text{falls } v \in V \\ \pi(u_1) + r & \text{falls } v = v_{e,r} \text{ für eine Kante } e = (u_1, u_2) \end{cases}$$

und die Gewichte der Kanten $e = (u, v)$ durch

$$w'(e') = \begin{cases} w(e') & \text{falls } e' \in A \\ w(e) & \text{falls } u \in V \text{ und } v = v_{e,1} \text{ für eine Kante } e \\ 0 & \textit{sonst} \end{cases}$$

Nach Konstruktion ist $(D', \pi')$ ein eigentlicher Lagengraph. Anschaulich bedeutet dies, dass für jede Kante $e = (u, v)$ in $E$, die über $d(e)$ Schichten geht, Knoten $v_{e,1}, \ldots, v_{e,d(e)}$ zu $V$ hinzugefügt und auf die Schichten $\pi(u)+1, \ldots, \pi(u)+d$ gesetzt werden. Die Kante $e$ wird entfernt und durch die Kanten $(u, v_{e,1}), (v_{e,1}, v_{e,2}), \ldots, (v_{e,d(e)}, v)$ ersetzt, wobei die erste Kante das Gewicht von $e$ erhält: $w'(u, v_{e,1}) = w(e)$ und die anderen das Gewicht 0: $w'(v_{e,i}, v_{e,i+1}) = 0$ $(i = 1, \ldots, d(e) - 1)$. Alle Kanten $(u, v) \in E$, die von einer Schicht zur nächsten gehen, bei denen also $\pi(u) = \pi(v) - 1$ werden in $E'$ übernommen und erhalten dasselbe Gewicht wie in $D$. Ein Algorithmus zur Konstruktion von $(D', \pi')$ aus $(D, \pi)$ ist Algorithmus 2. Ein Beispiel für eine solche Konstruktion ist in Abbildung 3.4 zu sehen.

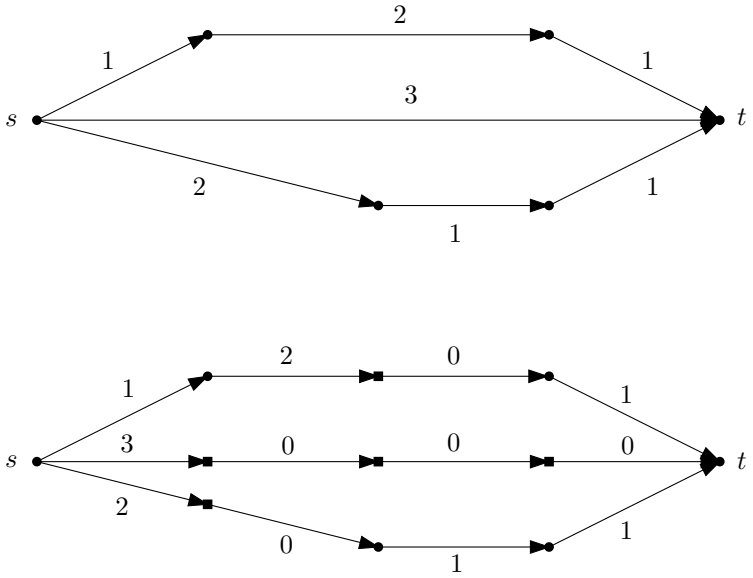



Abbildung 3.4: Konstruktion eines eigentlichen Lagengraphen aus einem Lagengraph.

Sei $\phi$ die Abbildung, die jede Kante $e = (u, v) \in A$ auf den entsprechenden Kantenzug aus $A'$ abbildet:

$$\phi(e) = \begin{cases} < u, v > & \text{falls } d(e) = 0 \\ < u, v_{e,1}, \ldots, v_{e,d(e)}, v > & \text{falls } d(e) > 0 \end{cases}$$

**Algorithmus 2** Algorithmus zur Konstruktion eines eigentlichen Lagengraphen $(D' = (V', A'), \pi')$ aus einem Lagengraphen $(D = (V, A), \pi)$

```
V' ← V
E' ← ∅
for all Knoten v ∈ V' do
  π'(v) = π(v)
end for
for all Kante e = (u, v) do
  d ← π(v) − π(u) − 1
  if d > 0 then
    for i = 0 to d do
      V' ← V' ∪ {v_{e,i}}
      π'(v_{e,i}) = π'(u) + i
    end for
    E' ← E' ∪ {(u, v_{e,1})} ∪ {(v_{e,d}, v)}
    w'(u, v_{e,1}) = w(u, v)
    w'(v_{e,d}, v) = 0
    for i = 1 to d − 1 do
      E' ← E' ∪ {(v_{e,i}, v_{e,i+1})
      w'(v_{e,i}, v_{e,i+1}) = 0
    end for
  else
    E' ← E' ∪ {e}
    w'(e) = w(e)
  end if
end for
```

und $W =< v_0, \ldots, v_l >$ ein Weg in $D$. Die Abbildung $\Phi$, die Wege in $D$ auf Wege in $D'$ abbildet, wird durch

$$\Phi(W) = \phi(v_0, v_1) \oplus \ldots \oplus \phi(v_{l-1}, v_l)$$

definiert und für eine Menge von $k$ Wegen $P = \{P_1, \ldots, P_k\}$ wird $\Phi(P) = \{\Phi(P_1), \ldots, \Phi(P_k)\}$ gesetzt. Schränkt man den Bildbereich auf Wege ein, deren Start- und Zielknoten in $V$ liegen, so ist $\Phi$ bijektiv und die Umkehrabbildung $\Phi^{-1}$ kann einfach angegeben werden:

$$\Phi^{-1}(< v_1, \ldots, v_r >) =< \psi(v_1), \ldots, \psi(v_r) >$$

wobei

$$\psi(v') = \begin{cases} v' & \text{falls } v' \in V \\ \emptyset & \text{sonst.} \end{cases}$$

der Weg in $D$ entsteht also aus dem Weg in $D'$, indem aus der Knotenfolge alle Knoten aus $V' \setminus V$ gelöscht werden. Es ist zu zeigen, dass dabei ein Weg entsteht, dass also zwei aufeinanderfolgende Knoten der entstehenden Knotenfolge durch eine Kante verbunden sind. Seien also $u, v \in V$ zwei aufeinanderfolgende Knoten in $\Phi^{-1}(< v_1, \ldots, v_r >)$. Falls

sie auch in $< v_1, \ldots, v_r >$ direkt aufeinanderfolgen ist nichts zu zeigen, denn dann ist $(u, v) \in A'$ und $u, v \in V$. Daraus folgt $(u, v) \in A$.
Sonst entstanden sie aus einer Folge $< u = v_s, v_{s+1}, \ldots, v = v_{s+i+1} >$ mit $v_{s+1}, \ldots, v_{s+i} \notin V$. Dann gehören alle diese Knoten zur selben Kante $e = (u, v)$, da nach Konstruktion jeder dieser Knoten nur eine ein- und eine ausgehende Kante hat und nur mit Knoten verbunden ist, die zur gleichen Kante gehören, oder Endknoten dieser Kante sind.
Für jede Kante $e = (u, v) \in E(P)$ gibt es eine Kante $(u, v_e) \in E(\Phi(P))$ mit $v_e = v$, falls $d_e = 0$ und $v_e = v_{e,1}$ sonst. Für alle Knoten $u \in V' \setminus V$ gilt nach Konstruktion $w'(u, v) = 0$ für alle Kanten $(u, v) \in A'$. Es folgt:

$$\begin{aligned}
\mathcal{E}(P) &= \sum_{u \in V} \max_{(u,v) \in E(P)} w(u, v) \\
&= \sum_{u \in V' \cap V} \max_{(u,v) \in E(\Phi(P))} w(u, v) + \sum_{u \in V' \setminus V} \max_{(u,v) \in E(\Phi(P))} w(u, v) \\
&= \sum_{u \in V'} \max_{(u,v) \in E(\Phi(P))} w(u, v) \\
&= \mathcal{E}(\Phi(P))
\end{aligned}$$

Zur Komplexität des Algorithmus und zur Größe von $D'$: Für jede Kante in $D$ müssen höchstens $n - 1$ neue Knoten und Kanten hinzugefügt werden. Es gilt also $\mid V' \mid \leq n + m(n-1) \in O(nm)$ und $\mid E' \mid \leq m + m(n-1) \in O(mn)$. Auch der Zeitbedarf wird im wesentlichen vom Einfügen neuer Kanten und Knoten bestimmt und der Algorithmus benötigt daher Zeit in $O(nm)$. □

Auf den ersten Blick erscheint die Abschätzung von $O(nm)$ für die Anzahl der Knoten bzw. Kanten in $D'$ sehr grob und man fragt sich, ob es keine bessere Schranke gibt. Betrachten wir dazu einen gerichteten Graphen $D = (V, A)$ mit $V = \{v_1, \ldots, v_n\}$ und $A = \{(v_i, v_j) \mid j > i\}$, jeder Knoten ist also mit allen Knoten mit größerem Index verbunden. Bei einer Zuordnung der Knoten zu Lagen muss jeder Knoten auf einer eigenen Lage liegen. Denn falls $v_i, v_j$ auf derselben Lage, kann es keine Kante von $v_i$ nach $v_j$ geben und keine Kante von $v_j$ nach $v_i$. Darauf folgt aber nach Definition von $A$, dass $i = j$. Also liegt jeder Knoten $v_i \in V$ auf Lage $L_i$. Die Kanten von $v_1$ zu $v_2, v_3, \ldots, v_n$ überspannen $0, 1, \ldots, n-2$ Lagen. Die Kanten von $v_2$ zu $v_3, v_4, \ldots v_n$ überspannen $0, 1, \ldots, n-2-1$ Lagen. Allgemein überspannen die Kanten von Knoten $v_i$ $0, 1, \ldots, n-i-1$ Lagen. Für die Anzahl $\Delta n$ der Knoten, die neu in den eigentlichen Lagengraphen $D'$ aufgenommen werden, gilt:

$$\begin{aligned}
\Delta n &= \sum_{i=1}^{n-1} \sum_{j=1}^{n-i-1} j = \frac{1}{2} \sum_{i=1}^{n-1} (n-i-1)(n-i) \\
&\geq \frac{1}{2} \sum_{i=1}^{n-1} (n-i-1)^2 = \frac{1}{2} \sum_{i=1}^{n-2} i^2 \\
&= \frac{1}{12}(n-2)(n-1)(2n-3) \in \Theta(n^3)
\end{aligned}$$

Da mit jedem neuen Knoten auch eine neue Kante hinzugenommen wird, gilt auch für die Anzahl $m'$ der Kanten in $D'$: $m' \in \Theta(n^3)$.

**Satz 11.** *Sei $D = (V, A)$ ein eigentlicher Lagengraph mit $n$ Knoten, $m$ Kanten, Gewichtsfunktion $w : A \to \mathbb{R}^+$ und Lagen $L_0, \ldots, L_h$, sowie $s, t \in V$ . Dann können in $O(m^k)$ $k$ kantendisjunkte Wege minimaler Energie von $s$ nach $t$ bestimmt werden.*

*Beweis.* O.B.d.A. wird angenommen, dass $s \in L_0$ und $t \in L_h$, denn falls $s \in L_i$ und $t \in L_j$ ist, so können die Lagen $L_0, \ldots L_{i-1}$ entfernt werden, da sie von $s$ aus nicht erreichbar sind und $L_{j+1}, \ldots, L_h$ können entfernt werden, da von ihnen aus $t$ nicht erreichbar ist. Falls dabei ein leerer Graph entsteht, gibt es keine Wege von $s$ nach $t$.

Für den Beweis wird angenommen, dass die Mengen $V$ und $E$ durch Relationen "$\leq$" vollständig geordnet sind. Dadurch ist es möglich, Kombinationen von Knoten und Kanten zu betrachten: Kombinationen von $k$ Knoten sind geordnete Knotentupel $(v_1, \ldots, v_k)$ mit $v_1 \leq v_2 \leq \ldots \leq v_k$. Analog werden Kombinationen von $k$ Kanten definiert. Für ein beliebiges $k$-Tupel $(v_1, \ldots, v_k)$ liefere $\phi(v_1, \ldots, v_k)$ das geordnete Knotentupel. $\phi$ kann durch einfaches Sortieren in Zeit $O(k \log k)$ berechnet werden.

Da $D$ ein eigentlicher Lagengraph mit $h + 1$ Lagen ist, gehen alle Kanten nur von einer Schicht zur nächsten. Wir können also eine Partitionierung der Kantenmenge $A$ in $E_1, \ldots, E_h$ durch

$$E_i = \{(u, v) \mid v \in L_i\}$$

für $i = 1, \ldots, h$ definieren. Die Kanten in $E_i$ gehen von Knoten der Lage $L_{i-1}$ zu Knoten der Lage $L_i$.

Eine Kombination von $k$ Knoten einer Lage, also ein $k$-Tupel $(v_1, \ldots, v_k)$ mit $v_1 \leq v_2 \leq \ldots \leq v_k$ steht dafür, dass ein Weg über $v_1$ verläuft, ein Weg über $v_2$, usw. Kommt ein Knoten $l$ Mal in der Kombination vor, so bedeutet dies, dass $l$ Wege über diesen Knoten verlaufen. Der Startknoten $s$ wird also durch das $k$-Tupel $(s, \ldots, s)$ und der Zielknoten $t$ durch das $k$-Tupel $(t, \ldots, t)$ repräsentiert. Für $k$ kantendisjunkte Wege kann also für jede Lage eine Kombination von $k$ Knoten angegeben werden, die jeden Knoten entsprechend der Anzahl der Wege, die über ihn verlaufen, enthält.

Sind $k$ kantendisjunkte Wege minimaler Engergie $P$ von $s$ zu einer Kombination $(v_1, \ldots, v_k)$ von Knoten aus Lage $L_i$ gegeben, so benutzen die Wege $P$ genau $k$ Kanten aus $E_i$ $(i = 1, \ldots, h)$. (Anschaulich ist klar, dass $k$ Wege mindestens $k$ Kanten von einer Schicht zur nächsten enthalten müssen. Wären es andererseits mehr als $k$ Kanten, so müsste auch mindestens eine Kante von $L_i$ nach $L_{i-1}$ dabei sein. Das ist in Lagengraphen aber nicht möglich.) Im folgenden schreiben wir $P(v_1, \ldots, v_k)$ für die Abschnitte der Wege von $s$ bis zur Kombination $P(v_1, \ldots, v_k)$.

Außerdem gilt bei eigentlichen Lagengraphen, dass Teile energieminimaler kantendisjunkter Wegemengen wieder energieminimale Wege sind: Sei $P$ eine Menge von $k$ kantendisjunkten Wegen minimaler Energie von $s$ zu einer Knotenkombination $(v_1, \ldots, v_k)$ aus $L_i$ und $(e_1 = (u_1, v_1), \ldots, e_k = (u_k, v_k))$ die Kanten aus $E_i$, die von $P$ benutzt werden. Also ist $\phi(u_1, \ldots, u_k)$ das Knotentupel zu $P$ aus Lage $L_{i-1}$. Dann sind die Teilstücke der Wege $P$ bis von $s$ bis zur Lage $L_{i-1}$ auch $k$ kantendisjunkte Wege minimalen Gewichts

zur Kombination $\phi(u_1, \ldots, u_k)$.
(Angenommen, es gäbe $k$ kantendisjunkte Wege $P'$ von $s$ nach $\phi(u_1, \ldots, u_k)$ mit

$$\mathcal{E}(P'(\phi(u_1, \ldots, u_k))) < \mathcal{E}(P(\phi(u_1, \ldots, u_k))).$$

Da $D$ ein eigentlicher Lagengraph ist, sind die Kanten der Wege $P'(\phi(u_1, \ldots, u_k))$ und $\{e_1, \ldots e_k\}$ disjunkt, da aus verschiedenen Lagen. Also könnte $P'(\phi(u_1, \ldots, u_k))$ durch Anfügen der Kanten aus $\{e_1, \ldots e_k\}$ zu einem Weg $P'(v_1, \ldots, v_k)$ erweitert werden, der geringere Energie benötigt als $P(v_1, \ldots, v_k)$. Dies steht im Widerspruch zur Minimalität von $P(v_1, \ldots, v_k)$).
Augrund dieser Eigenschaft genügt es, sich für jedes Knotentupel nur die direkten Vorgängerkanten auf den $k$ Wegen minimaler Energie zu merken. Diese implizieren eine Vorgängerkombination von Knoten usw.

Für eine Kombination $(v_1, \ldots, v_k)$ von $k$ Knoten einer Lage $L_i$ sei $\mathcal{E}_{\min}(v_1, \ldots, v_k)$ die Energie von $k$ kantendisjunkten Wegen minimaler Energie von $s$ zu den Knoten dieser Kombination und $\text{vorg}(v_1, \ldots, v_k)$ seien die $k$ Kanten aus $E_i$, die zu den Wegen minimaler Energie gehören.

Ist $(e_1 = (u_1, v_1), \ldots, e_k = (u_k, v_k))$ eine Kombination von Kanten aus $E_i$ $(i = 1, \ldots, h)$ und kennen wir $\mathcal{E}_{\min}(\phi(u_1, \ldots, u_k))$, so gilt für die Energie des minimalen Weges von $s$ zu $\phi(v_1, \ldots, v_k)$, der die Kanten $e_1, \ldots, e_k$ benutzt $(P_{e_i,\ldots,e_k}(\phi(v_1, \ldots, v_k)))$:

$$\mathcal{E}(P_{e_i,\ldots,e_k}(\phi(v_1, \ldots, v_k))) = \mathcal{E}_{\min}(\phi(u_1, \ldots, u_k)) + \text{erh}(e_1, \ldots e_k),$$

wobei

$$\text{erh}(e_1, \ldots e_k) = \sum_{u \in \{u_1,\ldots,u_k\}} \max_{(u,v) \in \{e_1,\ldots,e_k\}} w(u, v).$$

Das heißt für jeden Knoten, der in dem Tupel $(u_1, \ldots, u_k)$ vorkommt, erhöht sich die Energie um das Gewicht der maximalen Kante, die diesen Knoten als Anfangsknoten hat. $\text{erh}(e_1, \ldots e_k)$ lässt sich in Zeit polynomial in $k$ berechnen.

Sind die $k$ Wege minimaler Energie zu allen $k$-Kombinationen von Knoten aus Lage $L_{i-1}$ bekannt, so können die $k$ Wege minimaler Energie zu einer Kombination $(v_1, \ldots, v_k)$ aus Lage $L_i$ berechnet werden, indem alle Kombinationen von Kanten aus $E_i$ betrachtet werden, die zu $(v_1, \ldots, v_k)$ führen und die Energie von $k$ kantendisjunkten Wegen berechnet wird, die zu den Anfangsknoten dieser Kanten optimal sind und diese Kanten benutzen. Über alle Kantenkombinationen, die zu einer Knotenkombination führen wird minimiert.

In Algorithmus 3 wird dieses Verfahren formal beschrieben. Er berechnet für jede Knotenkombination die Energie von Wegen minimaler Energie von $s$ zu dieser Kombination, sowie die $k$ Kanten, die zu dieser Kombination führen (Vorgängerkanten). Hat man dies für alle Lagen berechnet, so ist $\mathcal{E}_{\min}(t, \ldots, t)$ die Energie von $k$ Wegen minimaler Energie von $s$ nach $t$ und die Wege können durch Backtracking gefunden werden: Angefangen mit $(t, \ldots, t)$ kann jeweils das Vorgängertupel über die Vorgängerkanten gewonnen werden. Für dieses sind auch wieder die Vorgängerkanten gespeichert usw.

Die Laufzeit des Algorithmus wird von der Anzahl der Kantenkombinationen bestimmt, die betrachtet werden müssen. Deren Anzahl ist

$$\sum_{i=1}^{h} m_i(m_i-1)\dots(m_i-k) \leq m(m-1)(m-2)\dots(m-k),$$

wobei $m_i =\mid E_i \mid$. Da für jede Kantenkombination nur Operationen ausgeführt werden, deren Laufzeit nur von $k$ abhängt und $k$ als fest betrachtet wird, liegt die Laufzeit also in $O(m^k)$.
Der Speicherbedarf von Algorithmus 3 wird durch die Anzahl der Knotenkombinationen bestimmt. Analog zu der Argumentation bei der Laufzeit liegt der Speicherbedarf in $O(n^k)$. □

**Algorithmus 3** Der Algorithmus zur Berechnung der Energie von $k$ kantendisjunkten Wegen minimaler Energie sowie der Vorgängerkanten. Die Wege können durch Backtracking gefunden werden.

**for all** $\{(v_1,\dots,v_k) \mid v_1 \leq \dots \leq v_k\}$ **do**
  $\mathcal{E}_{\min}(v_1,\dots,v_k) \leftarrow \infty$
  $\text{vorg}(v_1,\dots,v_k) \leftarrow nil$
**end for**
$\mathcal{E}_{\min}(s,\dots,s) = 0$
**for** $i = 1$ to $h$ **do**
  **for all** Kombinationen $(e_1 = (u_1, v_1), \dots, e_k = (u_k, v_k))$ von Kanten aus $E_i$ **do**
    $(u'_1,\dots,u'_k) \leftarrow \phi(u_1,\dots,u_k)$
    $(v'_1,\dots,v'_k) \leftarrow \phi(v_1,\dots,v_k)$
    $\text{erh} \leftarrow \sum_{u\in\{u'_1,\dots,u'_k\}} \max_{(u,v)\in\{e_1,\dots,e_k\}} w(u,v)$
    **if** $\mathcal{E}_{\min}(u'_1,\dots,u'_k) + erh < \mathcal{E}_{\min}(v'_1,\dots,v'_k)$ **then**
      $\mathcal{E}_{\min}(v'_1,\dots,v'_k) = \mathcal{E}_{\min}(u'_1,\dots,u'_k) + erh$
      $\text{vorg}(v'_1,\dots,v'_k) \leftarrow (e_1,\dots,e_k)$
    **end if**
  **end for**
**end for**

Das Beispiel in Abbildung 3.5 zeigt, dass die Konstruktion eines eigentlichen Lagengraphen aus einem beliebigen azyklischen Graphen vor der Anwendung des Algorithmus notwendig ist. Würden einfach in jedem Schritt alle $k$-Tupel der von den bisher erreichten Knotentupeln ausgehenden Kanten betrachtet, so würde man auf die folgende Folge von Knotenpaaren kommen: $(s,s) \to (v_1,v_2) \to (v_2,v_3) \to (v_3,v_4) \to (t,t)$. Damit hätte man eine Folge von Knotenpaaren gefunden, obwohl es keine 2 kantendisjunkten Wege von $s$ nach $t$ gibt.

### Der EAS-Algorithmus für azyklische Graphen

In diesem Abschnitt werden die obigen Teilschritte zum EAS-Algorithmus zusammengefasst, der für einen beliebigen azyklischen Graphen $D = (V, A)$ mit $n$ Knoten $m$ Kanten

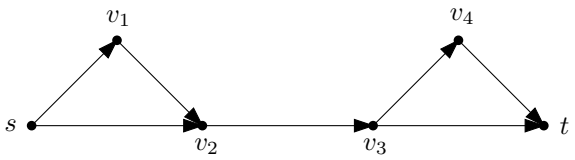



Abbildung 3.5: Der Algorithmus kann nicht ohne die vorherige Konstruktion eines eigentlichen Schichtengraphen angewandt werden.

und $s, t \in V$ sowie $k_0 \in \mathbb{N}$ $k_0$ kantendisjunkte Wege in Polynomialzeit berechnet.
Der Algorithmus führt die folgenden Schritte aus:

1. Berechnung einer Lagenzuweisung für alle Knoten (z. B. Longest Path)

2. Konstruktion eines eigentlichen Lagengraphen $D' = (V', A')$ mit $O(nm)$ Knoten und Kanten nach Satz 10 in Zeit $O(nm)$

3. Exakte Lösung in $D'$ in Zeit $O(n^{k_0}m^{k_0})$ und Rückübertragen der Lösung auf $D$ wie im Beweis von Satz 10.

Der Zeitkomplexität dieses Algorithmus wird von der exakten Lösung in $D'$ bestimmt, liegt also in $O(n^{k_0}m^{k_0})$. Auch der Speicherbedarf liegt in $O(n^{k_0}m^{k_0})$. Die Komplexität des Algorithmus ist also zu hoch, als dass er direkt in Sensornetzen berechnet werden könnte. Für große Graphen oder große $k$ kann der Algorithmus selbst auf leistungsfähigen Rechnern nicht in vernünftiger Zeit ausgeführt werden, bzw. überschreitet den maximal verfügbaren Speicher. Einige praktische Erfahrungen werden bei den Versuchen zur ESAS-Heuristik in Kapitel 5 beschrieben.

### 3.5.4 Die ESAS-Heuristik

Wie oben gezeigt kann für festes $k$ das $k$-EDPWMA-Problem in Polynomialzeit gelöst werden. Allerdings sind die Energiekostengraphen von Sensornetzen nur in den wenigsten Fällen azyklisch. In drahtlosen Sensornetzen, in denen die maximalen Sendeenergien für alle Knoten gleich sind, gibt es genau dann eine Kante im Energiekostengraph von einem Knoten $u$ zu einem Knoten $v$, wenn es auch eine Kante von $v$ zu $u$ gibt. Es entsteht also ein Zyklus. Kann der exakte Algorithmus für azyklische Graphen in diesen Fällen benutzt werden, um eine Heuristik abzuleiten, die möglichst gute Näherungslösungen liefert?
Eine Heuristik zur Berechnung von Näherunglösungen mit Hilfe des exakten Algorithmus für azyklische Graphen ist die folgende zweistufige Vorgehensweise:

- In einem ersten Schritt wird ein geeigneter azyklischer Subgraph $G'$ des Energiekostengraphen $G$ berechnet.

- Im zweiten Schritt wird der exakte Algorithmus für azyklische Graphen auf den Subgraphen $G'$ angewandt.

Wie kann ein geeigneter azyklischer Subgraph berechnet werden? Das Problem, den maximalen azyklischen Subgraphen zu berechnen, ist NP-Vollständig [11]. Außerdem sollten die Lage von Quelle und Senke in die Berechnung eingehen: Ein azyklischer Subgraph, bei dem die Quelle keine ausgehenden Kanten besitzt, ist für unser Problem nicht hilfreich.Ein weiteres Beispiel, bei dem der maximale azyklische Subgraph nicht weiterhilft, zeigt Abbildung 3.6. Es ist also sicherlich sinnvoll, die Lage von Quelle und

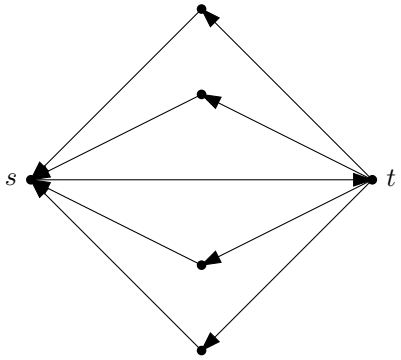



Abbildung 3.6: Der maximale azyklische Subgraph ist zur Berechnung einer Näherungslösung nicht geeignet, da er die Lage von Quelle und Senke nicht berücksichtigt. In diesem Graphen würde durch Entfernen der Kante $(s, t)$ ein azyklischer Subgraph entstehen. In diesem gäbe es aber keinen Weg mehr von $s$ nach $t$.

Senke bei der Berechnung eines azyklischen Subgraphen zu berücksichtigen. Man kann nun die Geometrie eines Sensornetzes ausnutzen, um einen azyklischen Subgraphen zu berechnen: Jeder Knoten darf nur an Knoten senden, die dem Ziel näher liegen als er selbst. Übertragen auf den Energiekostengraphen heisst das, dass Kanten $(u, v)$ entfernt werden, wenn $d(u, t) \leq d(v, t)$. Es ist sinnvoll, für die Quelle $s$ eine Ausnahme zu machen, und hier alle in $s$ hineingehenden Kanten zu entfernen, während alle von $s$ ausgehenden Kanten beibehalten werden. Es entsteht der Graph $D' = (V, A')$ mit $A = \{(s, u) \mid u \in V\} \cup \{(u, v) \mid u, v \in V \setminus \{s\}, d(v, t) < d(u, t)\}$.

**Satz 12.** *Der gerichtete Graph $D' = (V, A')$ ist azyklisch.*

*Beweis.* Es ist klar, dass es in $D'$ keinen Zyklus geben kann, der $s$ enthält, da jeder solche Zyklus eine in $s$ hineingehende Kante $(u, s), u \in V$ enthalten müsste. Es gibt in $A'$ aber keine solchen Kanten.
Angenommen es gibt einen Zyklus $u_1, \ldots u_l$ mit $u_i \in V \setminus \{s\}$ für $i = 1, \ldots, l$, so gibt es Kanten $(u_i, u_{i+1}) \in A'$ für alle $i \in \{1, \ldots, l\}$. Aus der Konstruktion von $A'$ folgt also $d(u_1, t) > d(u_2, t) > \ldots > d(u_l, t)$. Andererseits bilden die Kanten aber einen Zyklus, also auch $(u_l, u_1) \in A'$ und damit $d(u_1, t) < d(u_l, t)$. Dies ist ein Widerspruch und $D'$ ist folglich azyklisch. □

Der azyklische Subgraph $D'$ des Energiekostengraphen $D$ kann mit Hilfe von Algorithmus 4 und bei Verwendung geeigneter Datenstrukturen in Zeit $O(|A|)$ berechnet werden. Leider müssen auch bei diesem Algorithmus nicht alle Wege erhalten bleiben. Ein Bei-

**Algorithmus 4** Algorithmus zur Konstruktion eines azyklischen Subgraphen $D' = (V, A')$ aus einem Graphen $D = (V, A)$

```
for all Kanten e = (u, v) ∈ A do
  if u ≠ s then
    if v = s ∨ d(u, t) ≤ d(v, t) then
      entferne (u, v) aus A
    end if
  end if
end for
```

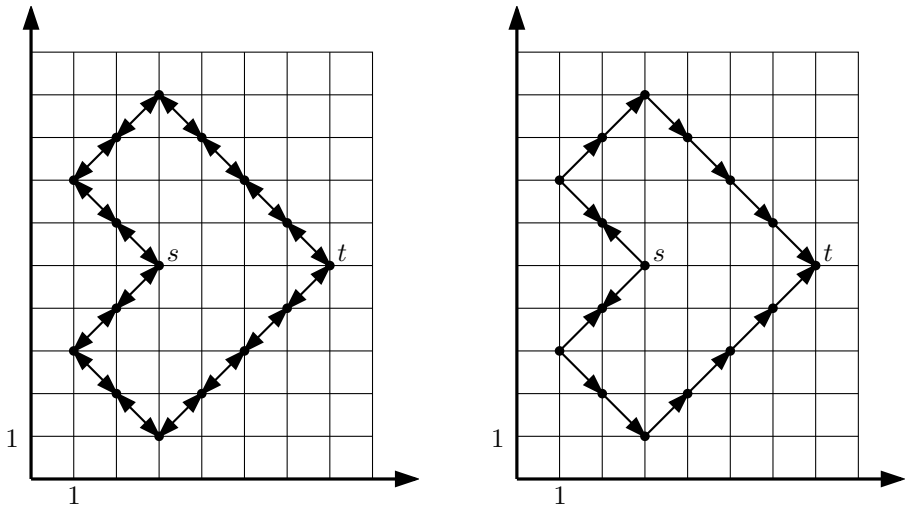



Abbildung 3.7: Beispiel eines Energiekostengraphen für ein Sensornetz mit $\mathcal{E}_{\max} = 2$ und $\alpha = 2$. Es gibt zwei Wege von $s$ nach $t$. Wird ein azyklischer Subgraph mit Algorithmus 4 berechnet, so gibt es keinen Weg mehr von $s$ nach $t$.

spiel dafür ist in Abbildung 3.7 zu sehen. Die obige Heuristik, die aus der Berechnung des azyklischen Subgraphen mittels Algorithmus 4 und anschließender exakter Lösung von $k$-EDPWMA besteht, wird im Folgenden ESAS-Heuristik genannt (Exaxt Solution in Acyclic Subgraph). Einige Versuche zum Abschneiden dieser Heuristik in zufälligen Sensornetzen finden sich in Kapitel refkap:experimente.

### 3.5.5 Approximationsalgorithmen

Da $k$-EDPWMA für allgemeines $k$ NP-vollständig ist, kann man nicht erwarten, einen deterministischen Polynomialzeitalgorithmus zu finden. Man ist stattdessen an polynomiellen Approximationsalgorithmen mit möglichst guten Approximationsfaktoren für das Optimierungsproblem interessiert. In [26] werden drei Approximationsalgorithmen vorgestellt:

- Der Naive-Dijkstra-Algorithmus und der LDESP-Algorithmus. Beim Naive-Dijkstra-Algorithmus werden sukzessive kürzeste Wege berechnet und die dazugehörigen

Kanten aus dem Graphen entfernt. Der LDESP-Algorithmus funktioniert nach dem gleichen Prinzip, doch werden die Kanten etwas anders gewichtet.

- Der Link-Disjoint Minimum-Weight-Algorithmus (LDMW): Es werden $k$ Wege minimalen Gewichts berechnet. Anschließend werden die redundanten Übertragungen entfernt.

Der WMA wird also in allen Fällen beim Suchen der Wege nicht berücksichtigt, sondern erst ausgenutzt, wenn die Wege bereits gefunden wurden.
Die beiden Algorithmen im ersten Punkt können beliebig schlechte Ergebnisse liefern, bzw. keine $k$ kantendisjunkten Wege finden, obwohl diese existieren. Deshalb werden diese Algorithmen in dieser Arbeit nicht weiter untersucht.

Für den LDMW-Algorithmus wurde in [26] eine obere Schranke für den Approximationsfaktor von $k$ gezeigt. Doch gibt es Beispiele, die diese Schranke erreichen, d. h. ist der Approximationsfaktor scharf?

**Satz 13.** *Es gibt kein $\delta > 0$ so dass der LDMW-Algorithmus einen Approximationsfaktor von $(1-\delta)k$ erreicht.*

*Beweis.* Wir betrachten die gewichteten gerichteten Graphen $D_k = (V_k, A_k)$ mit Knotenmenge

$$V_k = \{s,t\} \cup \{u_1, \ldots, u_k, w, v_1, \ldots, v_k\} \cup \{\bar{u_1}, \ldots, \bar{u_k}\},$$

Kantenmenge

$$A_k = \{(s,u_i), (u_i,w), (w,v_i), (v_i,t), (s,\bar{u}_i), (\bar{u}_i,t) \mid i = 1, \ldots, k\}$$

und Gewichtsfunktion

$$w_k(e) = \begin{cases} 1 & \text{falls } e = (w,v_i) \text{ oder } e = (\bar{u}_i,t) \text{ für ein } i \in \{1,\ldots,k\} \\ \epsilon & \text{sonst} \end{cases}$$

als Instanzen des Optimierungsproblems $k$-EDPWMA. In Abbildung 3.8 ist eine Darstellung der Struktur dieser Graphen zu sehen. Im Folgenden soll nun gezeigt werden, dass der Approximationsfaktor bei Anwendung des LDMW-Algorithmus auf die Graphen $D_k$ beliebig nahe an $k$ herankommt. Jeder mögliche $s$-$t$-Weg, der über einen der Knoten $u_1, \ldots, u_k$ geht, muss über $w$ und einen der Knoten $v_1, \ldots, v_k$ nach $t$ gehen. O.B.d.A. genügt es, die Wege $P_i =< s, u_i, w, v_i, t >$ mit $W(P_i) = 1 + 3\epsilon$ zu betrachten. Die Wege $\bar{P}_i =< s, \bar{u}_i, t >$ haben Gewicht $W(\bar{P}_i) = 1 + \epsilon$. Der LDMW-Algorithmus wird also die $k$ unteren Wege $\bar{P} = (\bar{P}_1, \ldots, \bar{P}_k)$ mit Gewicht $W(P) = (1+\epsilon)k$ und Energie $\mathcal{E}(P) = \epsilon + k$ wählen. Der Energieverbrauch der oberen $k$ Wege $P = (P_1, \ldots, P_k)$ ist hingegen $\mathcal{E}(P) = \epsilon + k\epsilon + 1 + k\epsilon = 1 + (2k+1)\epsilon$. Es folgt:

$$\frac{\mathcal{E}(\bar{P})}{\mathcal{E}(P)} = \frac{k + \epsilon k}{1 + (2k+1)\epsilon} \to k \quad (\epsilon \to 0)$$

Für jedes $\delta > 0$ existiert also ein $\epsilon > 0$, so dass

$$\mathcal{E}(\bar{P}) \geq (1-\delta)k\mathcal{E}(P)$$

□

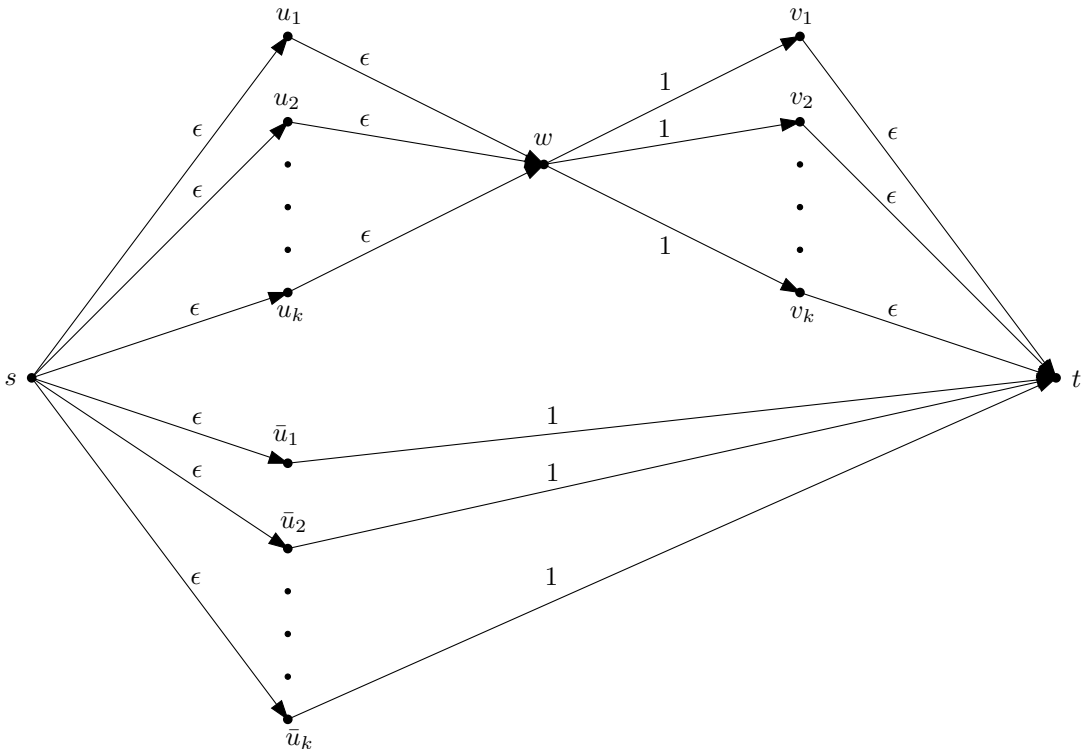



Abbildung 3.8: Der LDMW-Algorithmus hat Approximationsfaktor $k$.

## 3.6 Formulierung von $k$-EDPWMA als ganzzahliges lineares Programm

In den letzten Jahren sind für viele NP-vollständige Probleme Approximationsalgorithmen angegeben worden, die mit Methoden entwickelt wurden, die auf einer Formulierung des zu lösenden Problems als ganzzahliges lineares Programm basierten [1]. Eine der wichtigsten dieser Methoden ist die sogenannte Primal-Dual-Methode. Im Folgenden wird zuerst eine Formulierung von $k$-EDPWMA als ganzzahliges lineares Programm angegeben und deren Korrektheit gezeigt. Anschließend wird die Relaxation des ganzzahligen linearen Programms besprochen. In Abschnitt 3.6.3 wird kurz das Prinzip der Primal-Dual-Methode vorgestellt, bevor in Abschnitt 3.6.4 gezeigt wird, dass sich die Primal-Dual-Methode mit der hier vorgestellten Formulierung als ganzzahliges lineares Programm nicht dazu eignet, einen Approximationsalgorithmus für $k$-EDPWMA zu entwickeln.

### 3.6.1 Das ganzzahlige lineare Programm

Gegeben seien der gewichtete gerichtete Graph $D = (V, A)$ mit Gewichtsfunktion $w : A \to \mathbb{R}^+$, Start- und Zielknoten $s, t$ und die Anzahl $k$ der Wege. Für jede Kante $e_i$ wird eine Variable $x_i$ und für jeden Knoten $v_j$ eine Variable $z_j$ eingeführt ($i \in \{1, \ldots, n\}, j \in$

$\{1, \ldots, m\}$). Zur Vereinfachung der Schreibweise bedeute $x_e$ die zu einer Kante $e$ gehörende Variable, also $x_e = x_i$ falls $e = e_i$. Analog sei $z_v$ die zum Knoten $v$ gehörende Variable.

In [2] ist ein Beispiel aufgeführt, wie eine Zielfunktion, die die (nichtlineare) Maximierung max in der Zielfunktion enthält, durch Aufnahme eines neuen Constraints in eine lineare Zielfunktion umgewandelt werden kann. Wird dies auf das vorliegende Problem angewandt, so ergibt sich das folgende ganzzahlige lineare Programm:

$$\min f(x, z) = \sum_{v \in V} z_v \tag{3.5}$$

unter den Bedingungen, dass für alle $v \in V \setminus \{s, t\}$

$$\sum_{(u,v) \in A} x_{(u,v)} - \sum_{(v,u) \in A} x_{(v,u)} = 0 \tag{3.6}$$

gilt, während für $s$ bzw. $t$ die Constraints

$$\sum_{(u,s) \in A} x_{(u,s)} - \sum_{(s,u) \in A} x_{(s,u)} = -k \tag{3.7}$$

bzw.

$$\sum_{(u,t) \in A} x_{(u,t)} - \sum_{(t,u) \in A} x_{(t,u)} = k \tag{3.8}$$

gelten. Für jede Kante $e = (u, v) \in A$ wird folgender Constraint eingeführt:

$$z_u - x_e w(e) \geq 0 \tag{3.9}$$

Außerdem müssen die Definitionsbereiche der Variablen spezifiziert werden:

$$x_e \in \{0, 1\} \text{ für alle Kanten } e \in A \tag{3.10}$$

$$z_v \geq 0 \text{ für alle Knoten } v \in V. \tag{3.11}$$

Die Anzahl der Constraints ist $2(m + n)$, falls die Ganzzahligkeitsbedingung 3.10 als ein Constraint gezählt wird.

**Satz 14.** *Sei $(x^*, z^*)$ eine minimale Lösung des ganzzahligen linearen Programms. Dann können $k$ kantendisjunkte Pfade minimaler Energie $P^*$ mit Kantenmenge $E(P^*) = \{e \in E \mid x_e = 1\}$ konstruiert werden. Es gilt $\mathcal{E}(P^*) = f(x^*, z^*)$.*

*Beweis.* Der Beweis gliedert sich in drei Abschnitte: Zuerst wird gezeigt, wie aus der Lösung $(x^*, z^*)$ des ganzzahligen linearen Programms $k$ kantendisjunkte Wege $P^*$ gewonnen werden können. Anschließend wird gezeigt, dass $P^*$ Wege minimaler Energie sind. Schließlich wird die Gleichheit $\mathcal{E}(P^*) = f(x^*, z^*)$ gezeigt.

Aus den Bedingungen 3.6 - 3.8 folgt, dass die Abbildung $f : E \to \{0, 1\}$, $f(e) = x_e$ ein Fluss mit Wert $k$ ist. Aus der Ganzzahligkeitsbedingung 3.10 folgt, dass jede Kante $e$ nur Fluss $f(e) = 0$ oder $f(e) = 1$ führt. Analog zu der Konstruktion im Beweis von Satz 2

können zu diesem Fluss $k$ kantendisjunkte Wege $P^*$ mit $\mathcal{E}(P^*) = f(x^*, z^*)$ konstruiert werden .
Beh: Die Wege in $P^*$ sind Wege minimaler Energie.
Bew: Angenommen es gäbe $k$ kantendisjunkte Wege $P'$ mit $\mathcal{E}(P') < \mathcal{E}(P)$. Definiere $(x', z')$ durch

$$x'_e = \begin{cases} 1 & \text{falls } e \in E(P') \\ 0 & \text{sonst} \end{cases} \tag{3.12}$$

$$z'_v = \max_{(v,u)\in E} x^*_{(v,u)} w(v,u) \tag{3.13}$$

Es ist klar, dass dies eine zulässige Lösung des ganzzahligen linearen Programms ist. Dann gilt:

$$f(x', z') = \sum_{v\in V} z'_v = \mathcal{E}(P') < \mathcal{E}(P) = f(x^*, z^*)$$

Dies steht im Widerspruch zur Minimalität von $f(x^*, z^*)$. Also haben die $k$ kantendisjunkten Wege $P^*$ minimale Energie.
Beh: Für alle Knoten $v \in V$ gilt

$$z^*_v = \begin{cases} \max_{(v,u)\in E} x^*_{(v,u)} w(v,u) & \text{falls eine Kante } (v,u) \in E \text{ existiert} \\ 0 & \text{sonst} \end{cases}$$

Bew: Falls es keine Kante $(v,u) \in E$ gibt, folgt aus Constraint 3.11 $z_v \geq 0$. Angenommen $z^*_v > 0$. Setze $\hat{z} = (\hat{z}_1, \ldots, \hat{z}_n)$ mit

$$\hat{z}_i = \begin{cases} 0 & \text{falls } v_i = v \\ z_i & \text{sonst} \end{cases}$$

Falls es eine Kante $(v,u) \in E$ gibt, folgt aus den Constraints 3.9, dass $z^*_v \geq \max_{(v,u)\in E} x^*_{(v,u)} w(v,u)$. Angenommen $z_v > \max_{(v,u)\in E} x^*_{(v,u)} w(v,u)$. Setze $\hat{z} = (\hat{z}_1, \ldots, \hat{z}_n)$ mit

$$\hat{z}_i = \begin{cases} \max_{(v,u)\in E} x^*_{(v,u)} w(v,u) & \text{falls } v_i = v \\ z_i & \text{sonst} \end{cases}$$

In beiden Fällen ist ist $(x^*, \hat{z})$ eine zulässige Lösung des ganzzahligen linearen Programms mit $f(x^*, \hat{z}) = \sum_{v\in V} \hat{z}_v < \sum_{v\in V} z^*_v = f(x^*, z^*)$. Dies steht im Widerspruch zur Minimalität von $(x^*, z^*)$. Also gilt die Behauptung und, da $x_{(u,v)} = 0$ falls $u \notin V(P^*)$, folgt

$$z^*_v = \begin{cases} \max_{(v,u)\in P^*} w(v,u) & \text{falls } v \in V(P^*) \\ 0 & \text{sonst} \end{cases}$$

Daher:

$$f(x^*, z^*) = \sum_{v\in V} z^*_v = \sum_{v\in V(P^*)} \max_{(v,u)\in E(P^*)} w(v,u) = \mathcal{E}(P^*)$$

□

### 3.6.2 Das relaxierte LP

Bei der Relaxierung des obigen ganzzahligen linearen Programms genügt es nicht, die Ganzzahligkeitsbedingung 3.10 durch die Bedingung $x_e \geq 0$ zu ersetzen, wie dies bei vielen Problemen geschieht.
Setze $x_{(s,v_2)} = \alpha$. Aus der Bedingung 3.7 folgt $x_{(s,t)} = 2 - \alpha$ und aus 3.6 folgt $x_{(v_2,v_3)} =$

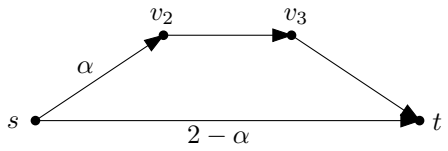



Abbildung 3.9: Beispiel zur Relaxation ohne Kapazitätsbeschränkung.

$x_{(v_3,t)} = \alpha$. Da $x_{(s,t)} \geq 0$ folgt: $\alpha \in [0, 2]$. Falls $1 \leq \alpha \leq 2$ sendet $s$ mit Energie $\alpha$ und die Zielfunktion hat in Abhängigkeit von $\alpha$ das Optimum $3\alpha$, wird also minimiert durch $\alpha = 1$ mit Wert 3. Gilt dagegen $0 \leq \alpha \leq 1$, so hat die Zielfunktion den Wert $2 - \alpha + 2\alpha = 2 + \alpha$. Das globale Minimum wird also für $\alpha = 0$ angenommen und beträgt 2. Der ganze Fluss fließt dann über die Kante $(s, t)$.
Es muss also für alle Kanten $e \in E$ die Bedingug $x_e \leq 1$ hinzugenommen werden. Das relaxierte Problem lautet also

$$\min \sum_{i=1}^{n} z_i \tag{3.14}$$

unter den Bedingungen

$$x_e \leq 1 \text{ für alle Kanten } e \in A \tag{3.15}$$

$$\sum_{(u,v)\in A} x_{(u,v)} - \sum_{(v,u)\in A} x_{(v,u)} = 0 \text{ für } v \in V \setminus \{s, t\} \tag{3.16}$$

$$\sum_{(u,s)\in A} x_{(u,s)} - \sum_{(s,u)\in A} x_{(s,u)} = -k \tag{3.17}$$

$$\sum_{(u,t)\in A} x_{(u,t)} - \sum_{(t,u)\in A} x_{(t,u)} = k \tag{3.18}$$

$$z_u - x_e w(e) \geq 0 \text{ für alle Kanten } e = (u, v) \in A \tag{3.19}$$

$$x_e \geq 0 \text{ für alle Kanten } e \in A \tag{3.20}$$

$$z_v \geq 0 \text{ für alle Knoten } v \in V. \tag{3.21}$$

### 3.6.3 Die Primal-Dual-Methode

Die Primal-Dual-Methode ist eine der wichtigsten Methoden, um für NP-vollständige Probleme, die sich als ganzzahliges lineares Programm formulieren lassen, Approximationsalgorithmen zu entwickeln. In dem Übersichtsartikel [13] wird sie als “Standardwerkzeug zur Entwicklung von Algorithmen für kombinatorische Optimierungsprobleme”

bezeichnet. In diesem Artikel wird die Anwendung der Primal-Dual-Methode zur Entwicklung von Approximationsalgorithmen auf eine Reihe sogenannter Netzwerk-Design-Probleme geschildert. In [29] ist die Anwendung auf eine Reihe weiterer NP-vollständiger Probleme beschrieben. In [21] wird die Primal-Dual-Methode zur Herleitung und Analyse von polynomiellen Algorithmen für kombinatorische Optimierungsprobleme verwendet. Die folgende kurze Beschreibung der Primal-Dual-Methode ist [1] entnommen.

Die Primal-Dual-Methode nutzt die Tatsache aus, dass das zu einem linearen Minimierungsproblem duale Problem ein lineares Maximierungsproblem ist und die Optima übereinstimmen. Ist also ein ganzzahliges Minimierungsproblem $ILP$ mit Relaxation $LP$ gegeben und bezeichnet $DLP$ das zu $LP$ duale Problem, so ist jede zulässige Lösung von $DLP$ kleiner oder gleich dem Minimum von $ILP$, das wiederum kleiner oder gleich jeder zulässigen Lösung von $ILP$ ist. Eine zulässige Lösung von $DLP$ bildet also eine untere Schranke für die optimale Lösung von $ILP$ und kann somit zur Abschätzung der Qualität einer approximativen Lösung zu $ILP$ verwendet werden. Ein Primal-Dual-Algorithmus nutzt diese Zusammenhänge der Lösungen der verschiedenen Optimierungsprobleme aus, um eine Approximationslösung zum ganzzahligen linearen Programm $ILP$ zu finden. Dazu werden simultan eine ganzzahlige Lösung $x$ von $ILP$ und eine zulässige Lösung $y$ von $DLP$ berechnet. In jedem Schritt werden $x$ und $y$ betrachtet und ein neues Paar von Lösungen $x'$ und $y'$ berechnet, wobei $x$ so verändert wird, dass nach endlich vielen Schritten eine zulässige Lösung gefunden wird und $y$ so, dass $y'$ näher am Optimum von (DLP) liegt als $y$. Der Algorithmus endet, wenn $x$ zulässig ist. Die Qualität der Lösung wird dann mit Hilfe der lezten Lösung $y'$ des dualen Problems abgeschätzt. Die Berechnung von $x'$ aus $x$ bzw. von $y'$ aus $y$ hängen natürlich vom konkreten Problem ab. In Abschnitt 3.6.4 wird jedoch gezeigt, dass man grundsätzlich nicht erwarten kann, mit der Primal-Dual-Methode und der obigen Formulierung von $k$-EDPWMA als ganzzahliges lineares Programm einen Approximationsalgorithmus entwickeln zu können.

### 3.6.4 Die Ganzzahligkeitslücke der LP-Relaxation

Als Maß für die Güte einer LP-Relaxation wird die Ganzzahligkeitslücke (engl. integrality gap) der Relaxation definiert:

**Definition 11.** Sei $\Pi$ ein als ganzzahliges Programm formuliertes Minimierungsproblem mit optimalem Zielfunktionswert OPT und Wert der LP-Relaxation $\mathrm{OPT}_f$. Die *Ganzzahligkeitslücke* der LP-Relaxation ist definiert als

$$\sup_{I \in \Pi} \frac{\mathrm{OPT}(I)}{\mathrm{OPT}_f(I)}$$

Je näher die Ganzzahligkeitslücke an 1 ist, desto besser ist die LP-Relaxation. Falls in der Analyse eines Approximationsalgorithmus, der auf einer LP-Relaxation beruht, die Lösung des Approximationsalgorithmus direkt mit dem Wert einer fraktionalen primalen Lösung oder dem einer zulässigen dualen Lösung verglichen wird, so kann man keinen Approximationsfaktor besser als die Ganzzahligkeitslücke erhalten (vgl. [29]).

**Satz 15.** *Die Ganzzahligkeitslücke der LP-Relaxation aus Abschnitt 3.6.2 ist unbeschränkt.*

*Beweis.* Sei $T_{m,n}$ der gerichtete vollständige Baum mit Grad $m$ und Höhe $n$, bei dem die Kanten von der Wurzel weg gerichtet sind. Die Wurzel werde mit $s$ bezeichnet. $G_{m,n} = (V_{m,n}, E_{m,n}; w)$ sei der gewichtete gerichtete Graph, der aus $T_{m,n}$ entsteht, indem ein Knoten $t$ und Kanten von den Blättern des Baumes zu $t$ hinzugefügt und alle Kantengewichte auf 1 gesetzt werden: $V_{m,n} = V(T_{m,n}) \cup \{t\}$ , $E_{m,n} = E(T_{m,n}) \cup \{(b,t) \mid b \text{ Blatt von } T_{m,n}\}$ und $w(e) = 1$ für alle Kanten $e \in E_{m,n}$. In Abbildung 3.10 ist als Beispiel der Graph $G_{3,3}$ abgebildet.
Für $m \geq k$ existieren in diesem Graphen $k$ Wege von $s$ nach $t$. Aus der Kantendisjunktheit folgt in diesem Fall, dass die Wege bis auf $s$ auch knotendisjunkt sind. Jeder Weg besucht außer $s$ noch $n-1$ weitere Knoten. Die Kosten von $k$ optimalen kantendisjunkten Wegen betragen also

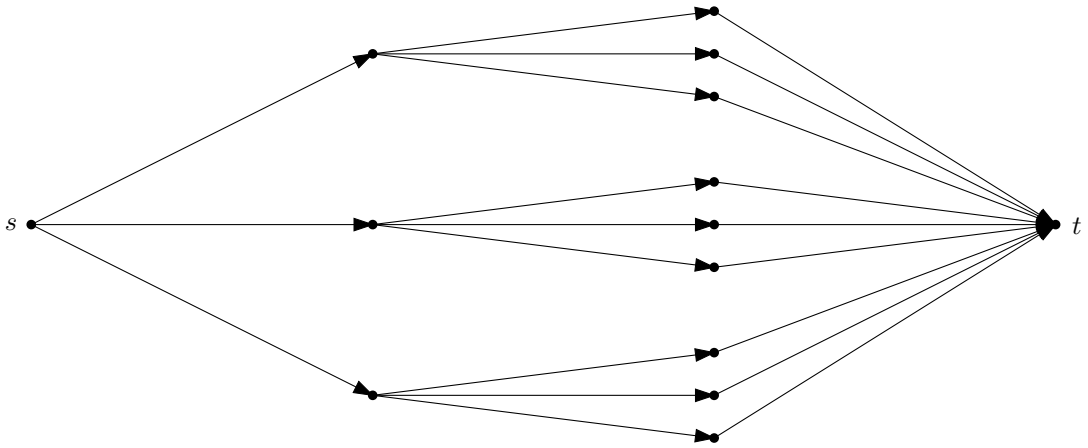



Abbildung 3.10: Der Graph $G_{3,3}$ wie im Beweis von Satz 15 definiert. Alle Kanten haben Gewicht 1.

$$\mathrm{OPT}(G_{m,n}) = 1 + k(n-1).$$

In der optimalen Lösung des relaxierten Problems hingegen kann der Fluss, der einen Knoten der Höhe $1, \ldots, n-1$ verlässt, auf $m$ Kanten verteilt werden. Diese Knoten senden also mit Energie $\frac{k}{m^h}$. Knoten der Höhe $n$ besitzen nur eine ausgehende Kante und senden daher mit Energie $\frac{k}{m^{n-1}}$. Es gibt $m^{h-1}$ Knoten der Höhe $h$ $(h = 1, \ldots, n)$. Die Gesamtkosten der nichtganzzahligen Lösung betragen also:

$$\begin{aligned}
\mathrm{OPT}_f(G_{m,n}) &= \sum_{i=1}^{n-1} m^{i-1} \frac{k}{m^i} + m^{n-1} \frac{k}{m^{n-1}} \\
&= k \frac{n-1}{m} + k \\
&= \frac{m+n-1}{m} k.
\end{aligned}$$

Berechnet man den Quotienten aus Definition 11, so ergibt sich:

$$\frac{\mathrm{OPT}(G_{m,n})}{\mathrm{OPT}_f(G_{m,n})} = \frac{m + mk(n-1)}{(m+n-1)k}.$$

Setzt man nun $n = m + 1$ folgt:

$$\frac{\mathrm{OPT}(G_{m,m+1})}{\mathrm{OPT}_f(G_{m,m+1})} = \frac{m + m^2k}{2mk} = \frac{m}{2} + \frac{1}{2k} \to \infty \quad (m \to \infty).$$

Die Ganzzahligkeitslücke ist also unbeschränkt. □

Wie oben ausgeführt, beruht jedoch die Abschätzung des Approximationsfaktors bei Algorithmen, die mit der Primal-Dual-Methode entwickelt wurden, auf dem Vergleich mit einer zulässigen dualen Lösung. Die Primal-Dual-Methode eignet sich also nicht dazu, mit Hilfe der angegebenen Formulierung des $k$-EDPWMA-Problems als ganzzahliges lineares Programm einen Approximationsalgorithmus zu entwickeln.

# 4 Das Single Power $k$-EDPWMA Problem

## 4.1 Einleitung

In Abschnitt 3.5.5 wurde gezeigt, dass der Approximationsfaktor des LDMW-Algorithmus $k$ ist. Für beliebiges $k$ gibt es Beispiele, in denen der Approximationsfaktor diesen Approximationsfaktor auch tatsächlich erreicht. Allerdings gibt es in diesen Beispielen sehr große Unterschiede in den Kantengewichten. Deshalb ist die Frage interessant, ob ein besserer Approximationsfaktor erreicht werden kann, wenn keine so großen Unterschiede in den Kantengewichten vorkommen können. Der Extremfall, dass alle Kanten dasselbe Gewicht haben, soll im folgenden untersucht werden.
Ein denkbares Anwendungsszenario für dieses Problem ist, dass eine Menge identischer Sensoren gegeben ist, die nicht mit verschiedenen Energiestufen senden können, sondern nur die Wahl zwischen senden oder nicht senden haben.
In der Literatur wurde die Idee, nur eine Energiestufe zuzulassen, in der ein Knoten senden kann, schon in [3] bei der Untersuchung des Broadcast-Problems verwendet. Es wurde gezeigt, dass das Broadcast-Problem auch bei nur einer Energiestufe NP-vollständig bleibt. Wie weiter unten gezeigt wird, gilt das auch für das $k$-EDPWMA-Problem.
In [17] wird das sogenannte $\Omega(1)$-Modell vorgeschlagen, bei dem Unit-Disk-Graphen betrachtet werden und der Abstand zwischen je zwei Knoten durch eine positive Konstante $d_0$ nach unten beschränkt ist. Unser Modell entspricht dem $\Omega(1)$-Modell mit unterer Schranke $d_0 = 1$.
Im Folgenden wird in Abschnitt 4.2 das Single Power $k$-EDPWMA-Problem formal definiert und gezeigt, dass es wie das $k$-EDPWMA-Problem NP-vollständig ist. In Abschnitt 4.3 wird gezeigt, wie eine Menge von $k$ kantendisjunkten Wegen mittels der Knoten, durch die alle Wege führen, zerlegt werden kann. Diese Zerlegung wird dabei helfen, in Abschnitt 4.4 eine obere Schranke für das Gewicht dreier inklusionsminimaler Wege minimaler Energie in Abhängigkeit von der Energie anzugeben. Mit dieser Schranke kann schließlich der Approximationsfaktor des LDMW-Algorithmus im Fall $k = 3$ exakt angegeben werden. Es wird sich zeigen, dass der Approximationsalgorithmus beim Single-Power-Problem im Fall $k = 3$ signifikant besser abschneidet als im $k$-EDPWMA-Problem mit beliebiger Energie.

## 4.2 Das Single Power $k$-EDPWMA Problem ($k$-SPEDPWMA)

**Definition 12.** Sei $D = (V, A)$ ein gerichteter Graph und $s, t \in V$. $k$-SPEDPWMA ist das Problem, $k$ kantendisjunkte Wege $P = \{P_1, \ldots, P_k\}$ zu finden, so dass $\mid V(P) \mid$ minimal ist.

Dieses Problem ist äquivalent zum Problem $k$-EDPWMA mit Graph $D = (V, A)$, Knoten $s, t$ und konstanter Gewichtsfunktion $w : A \to \mathbb{R}^+, w \equiv 1$, denn es gilt

$$\mathcal{E}(P) = \sum_{v \in V(P)} \max_{(v,u) \in E(P)} w(v, u) = \sum_{v \in V(P) \setminus \{t\}} 1 = \mid V(P) \mid -1$$

und die Konstante $-1$ spielt bei der Optimierung keine Rolle.

Es ist einfach zu sehen, dass $k$-SPEDPWMA NP-vollständig ist: Wird im Beweis zu Satz 6 $Q = 1$ gesetzt, so erhält man einen NP-Vollständigkeitsbeweis für $k$-SPEDPWMA. Allerdings ist es dann nicht mehr möglich, eine obere Schranke für die Approximierbarkeit wie im Beweis von Satz 8 zu zeigen. Da $k$-SPEDPWMA ein Spezialfall von $k$-EDPWMA ist, können natürlich dieselben Approximationsalgorithmen verwendet werden und eine obere Schranke für den Approximationsfaktor bleibt bestehen. Andererseits muss eine Schranke, die im allgemeinen Fall scharf ist, hier nicht mehr scharf sein. Für den LDMW-Algorithmus wird im folgenden gezeigt, dass im Fall $k = 3$ der Approximationsfaktor für $k$-SPEDPWMA besser ist als für $k$-EDPWMA.

## 4.3 Zerlegung von $k$ kantendisjunkten Wegen

Um die Schreibweise in den folgenden Beweisen zu vereinfachen, ist es sinnvoll, einige Notationen einzuführen:

**Notation 1.** Sei $P = \{P_1, \ldots, P_k\}$ eine Menge inklusionsminimaler kantendisjunkter Wege von $s$ nach $t$ und $s = v_1, v_2, \ldots, v_{l-1}, v_l = t$ die Knoten, die von allen Wegen besucht werden.

- $P_i(v_m, v_n)$ bezeichne das Wegstück des Weges $P_i$ zwischen den Knoten $v_m$ und $v_n$ ($i \in \{1, \ldots, k\}$, $m, n \in \{1, \ldots, l\}$).
- $\pi_i(v_m) = j$ falls $v_m$ als $j$-ter Knoten vom Grad $k$ von $P_i$ besucht wird

Anmerkungen: Da inklusionsminimale Wege zyklenfrei sind, sind $P_i(v_m, v_n)$ und $\pi_i(v_m)$ wohldefiniert.

**Satz 16.** *Seien $P_1, \ldots, P_k$ innklusionsminimale kantendisjunkte Wege von $s$ nach $t$ und $s = v_1, v_2, \ldots, v_{l-1}, v_l = t$ die Knoten, die auf allen $k$ Wegen liegen. Dann ist die Reihenfolge, in der diese Knoten auf den Wegen liegen, bei allen Wegen dieselbe.*

*Beweis.* O.B.d.A. seien die Knoten entsprechend ihrer Reihenfolge auf dem Weg $P_1$ indiziert, also $\pi_1(v_j) = j$ für alle $j \in \{1, \dots, l\}$. Angenommen es existieren $v_m$ und $v_n$ mit $m < n$ und $j \in \{1, \dots, k\}$, so dass $\pi_j(v_n) < \pi_j(v_m)$, d. h. $v_m$ liegt auf $P_1$ vor $v_n$ aber auf $P_j$ nach $v_n$. Dann lassen sich $P_1$ und $P_j$ folgendermaßen zerlegen:

$$P_1 = P_1(s, v_m) \oplus P_1(v_m, v_n) \oplus P_1(v_n, t)$$
$$P_j = P_j(s, v_n) \oplus P_j(v_n, v_m) \oplus P_j(v_m, t).$$

Ersetzt man nun $P_1$ und $P_j$ durch

$$P_1' = P_1(s, v_m) \oplus P_j(v_m, t)$$
$$P_j' = P_j(s, v_n) \oplus P_1(v_n, t),$$

so erhält man $k$ kantendisjunkte Wege $P' = (P_1', P_2, \dots, P_{j-1}, P_j', P_{j+1}, \dots, P_k)$ mit $E(P') \subset E(P)$. Dies ist ein Widerspruch zur Inklusionsminimalität von $P_1, \dots, P_k$. □

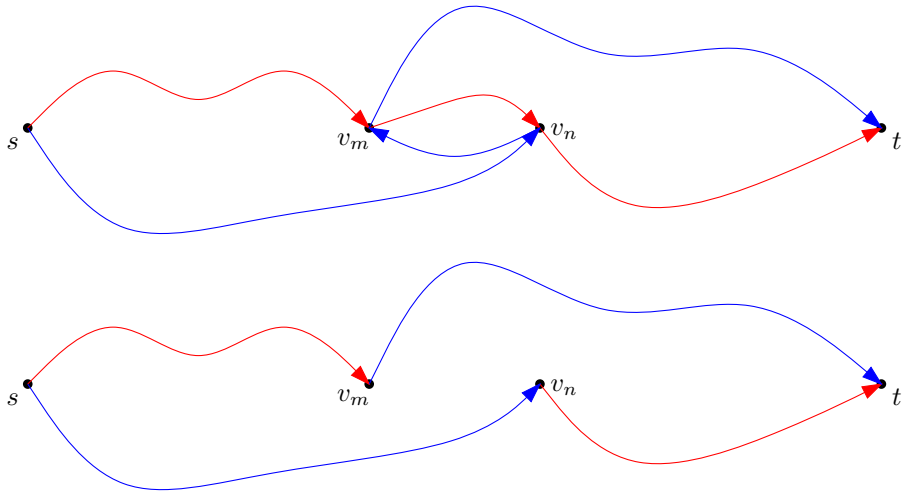



Abbildung 4.1: Oben sind die Wege $P_1$ und $P_j$ wie im Beweis zu Satz 16 dargestellt. Diese können wie unten abgebildet ersetzt werden.

**Satz 17.** *Seien $P = (P_1, \dots, P_k)$ kinklusionsminimale kantendisjunkte Wege von $s$ nach $t$, $s = v_1, v_2, \dots, v_{l-1}, v_l = t$ die Knoten, die auf allen $k$ Wegen liegen und $u$ ein weiterer Knoten, der auf mindestens einem Weg $P_i$ liegt. Dann gilt: Ist $u \in V(P_i(v_m, v_{m+1}))$ und $u \in V(P_j)$ , so auch $u \in V(P_j(v_m, v_{m+1}))$.*

*Beweis.* Angenommen $u \in V(P_i(v_m, v_{m+1}))$ und $u \in V(P_j(v_{m+1}, t))$. Dann können $P_i$ und $P_j$ folgendermaßen zerlegt werden:

$$P_i = P_i(s, v_m) \oplus P_i(v_m, u) \oplus P_i(u, v_{m+1}) \oplus P_i(v_{m+1}, t)$$
$$P_j = P_j(s, v_m) \oplus P_j(v_m, v_{m+1}) \oplus P_j(v_{m+1}, u) \oplus P_j(u, t)$$

Diese Zerlegung ist in Abbildung 4.2 oben dargestellt.
Damit können zwei kürzere Wege $P_i'$ und $P_j'$ definiert werden:

$$P_i' = P_i(s, v_m) \oplus P_i(v_m, u) \oplus P_j(u, t)$$
$$P_j' = P_j(s, v_m) \oplus P_j(v_m, v_{m+1}) \oplus P_i(v_{m+1}, t)$$

Diese Wege sind in Abbildung 4.2 unten dargestellt.
Werden $P_i$ und $P_j$ durch $P_i'$ bzw. $P_j'$ ersetzt, so erhält man $k$ Wege $P'$ mit $E(P') \subset E(P)$, da die Teilstücke $P_i(u, v_{m+1})$ und $P_j(v_{m+1}, u)$ nicht mehr benötigt werden. Dies steht im Widerspruch zur Inklusionsminimalität von $P$. Also $u \notin V(P_j(v_{m+1}, t))$.
Analog zeigt man $u \notin V(P_j(s, v_m))$. Folglich $u \in V(P_j(v_m, v_{m+1}))$. □

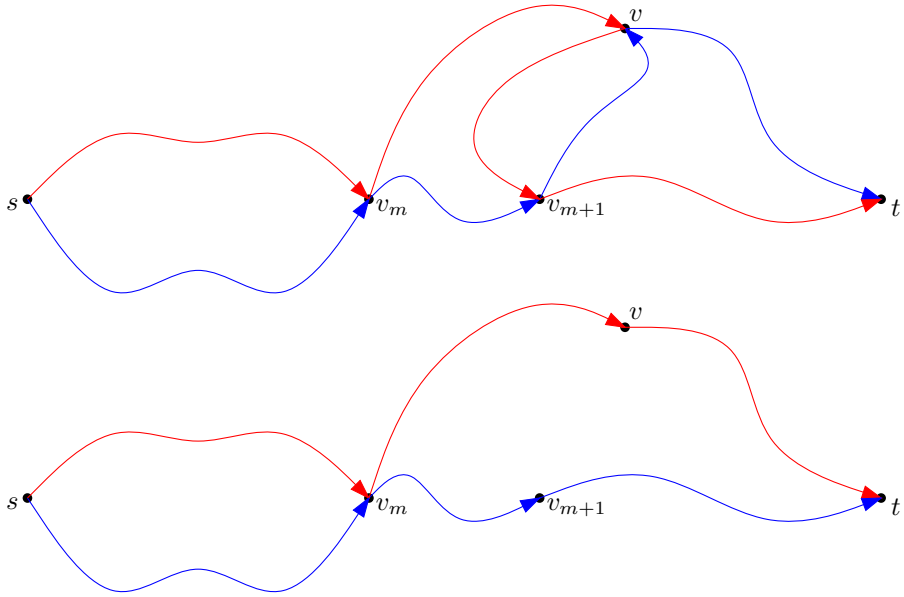



Abbildung 4.2: Oben sind die Wege $P_i$ und $P_j$ wie im Beweis von Satz 17 dargestellt. Diese können wie unten abgebildet ersetzt werden.

Aufgrund der beiden obigen Sätze ist es möglich, eine Menge $P$ von $k$ kantendisjunkten Wegen $P = \{P_1, \dots P_k\}$ mit Hilfe der Knoten, die von allen Wegen besucht werden, in Teilstücke zu zerlegen. Seien $s = v_0, v_1, \dots, v_l = t$ die Knoten, die (in dieser Reihenfolge) von allen $k$ Wegen besucht werden. $P(v_i, v_{i+1})$ bezeichne die Wegstücke zwischen $v_i$ und $v_{i+1}$:

$$P(v_i, v_{i+1}) = \{P_1(v_i, v_{i+1}), \dots, P_k(v_i, v_{i+1})\}$$

## 4.4 Gewicht und Energie bei drei Wegen

In diesem Abschnitt wird mit Hilfe der obigen Zerlegung inklusionsminimaler kantendisjunkter Wege gezeigt, dass für das Gewicht dreier Wege eine obere Schranke angegeben

werden kann, die nur von der Energie abhängt. Dies wird im nächsten Abschnitt dazu verwendet, den Approximationsfaktor des LDMW-Approximationsalgorithmus, angewandt auf $k$-SPEDPWMA, im Fall $k = 3$ exakt anzugeben.

**Satz 18.** *Seien $k = 3$ und $P = \{P_1, P_2, P_3\}$ drei inklusionsminimale kantendisjunkte Wege minimaler Energie.*
*Dann gilt: $W(P) \leq 2\mathcal{E}(P)$.*

*Beweis.* Seien $s = v_0, \ldots, v_l = t$ die Knoten aus $V(P)$, die von allen drei Wegen (in dieser Reihenfolge) besucht werden.
Wir betrachten zunächst einen einzelnen Abschnitt $P(v_i, v_{i+1})$. Die unmittelbaren Nachfolger von $v_i$ auf $P_1, P_2, P_3$ werden mit $u_1, u_2, u_3$ bezeichnet. Aus der Kantendisjunktheit von $P$ folgt, dass sie paarweise verschieden sind.
Beh.: Für jeden Abschnitt $P(v_i, v_{i+1})$ gilt: $W(P(v_i, v_{i+1}) \leq 2\mathcal{E}(P(v_i, v_{i+1})$
Zum Beweis ist eine Fallunterscheidung danach notwendig, ob $v_{i+1}$ direkter Nachfolger von $v_i$ auf einem der Wege ist oder nicht. Aus der Zerlegung der Wege folgt, dass höchstens einer der direkten Nachfolger von $v_i$ von allen Wegen besucht werden kann.

- **Fall 1:** $v_{i+1} \in \{u_1, u_2, u_3\}$, o.B.d.A. $v_{i+1} = u_2$
  In diesem Fall ist einer der direkten Nachfolger von $v_i$, o.B.d.A. $u_2$, der nächste Knoten, der von allen Wegen besucht wird. Nun wird eine Fallunterscheidung danach gemacht, ob einer der anderen Wege über den jeweils anderen direkten Nachfolger verläuft oder nicht.

  – **Fall 1.1:** $u_3 \notin V(P_1)$ und $u_1 \notin V(P_3)$
    In diesem Fall verlaufen die beiden noch übrigen Wege, $P_1$ und $P_3$, jeweils nur über $u_1$ und $u_3$. Dieser Fall ist in Abbildung 4.3 dargestellt. Es gilt dann:

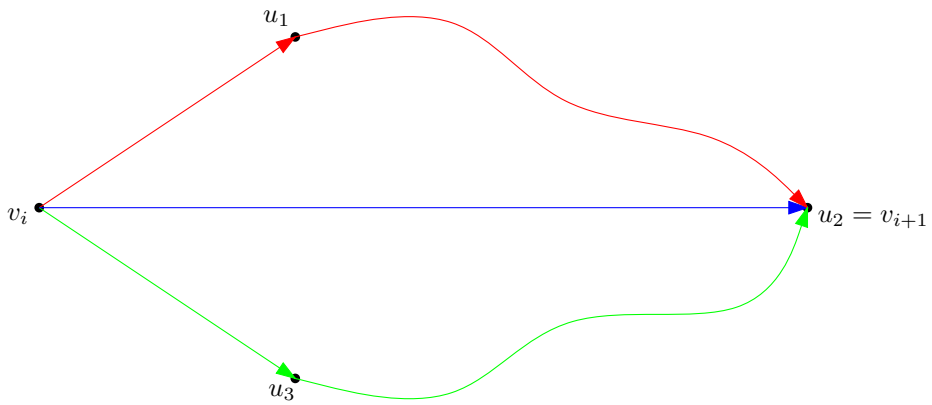



Abbildung 4.3: Fall 1.1 der Fallunterscheidung.

$$d_{\text{out}}(v_i) + d_{\text{out}}(u_1) + d_{\text{out}}(u_3) = 5$$

  – **Fall 1.2:** Genau ein Weg, o.B.d.A. $P_1$, geht durch $u_1$ und $u_3$.
    Dieser Fall ist in Abbildung 4.4 dargestellt und es gilt:

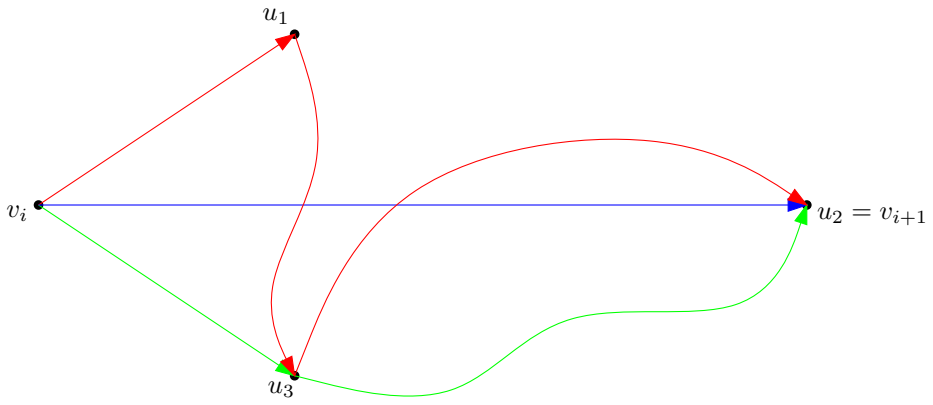



Abbildung 4.4: Fall 1.2 der Fallunterscheidung.

$$d_{\text{out}}(v_i) + d_{\text{out}}(u_1) + d_{\text{out}}(u_3) = 6$$

- **Fall 1.3:** Sowohl $P_1$ als auch $P_3$ gehen durch $u_1$ und $u_3$.
  Dann bilden $P_1(u_1, u_3)$ und $P_3(u_3, u_1)$ einen Zyklus. Dies ist ein Widerspruch zur Inklusionsminimalität von $P_1, P_2, P_3$!

Insgesamt gilt im Fall 1, d. h. wenn einer der direkten Nachfolger von $v_i$ auf einem der drei Wege $v_{i+1}$ ist, also:

$$\begin{aligned} W(P(v_i, v_{i+1})) &= |E(P(v_i, v_{i+1}))| \\ &= d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_3) \\ &\quad + \sum_{v \in V(i,i+1) \setminus \{v_i, v_{i+1}, u_1, u_3\}} d_{out}(v) \\ &\leq 6 + 2(|V(P(v_i, v_{i+1}))| - 4) \\ &= 2(|V(P(v_i, v_{i+1}))| - 1) \end{aligned}$$

- **Fall 2:** Im Fall 2 gehen durch keinen der direkten Nachfolger von $v_i$ alle drei Wege, also $v_{i+1} \notin \{u_1, u_2, u_3\}$.
  In diesem Fall werden wiederum drei Fälle unterschieden: In 2.1 gibt es einen Weg, der durch alle drei direkten Nachfolger von $v_i$ geht. In 2.2 gibt es keinen Weg, der durch alle drei Nachfolger geht, aber mindestens ein Weg geht durch zwei Nachfolger von $v_i$. In 2.3 geht jeder Weg nur durch genau einen direkten Nachfolger von $v_i$.

  - **Fall 2.1:** Es gibt einen Weg, o.B.d.A. $P_1$, der durch $u_1, u_2$ und $u_3$, o.B.d.A. in dieser Reihenfolge, verläuft.
    Dieser Fall ist in Abbildung 4.5 dargestellt. Da aus der Inklusionsminimalität der Wege folgt, dass es keine Zyklen geben kann, gilt:
    * $u_1 \notin V(P_2)$, sonst Zyklus $P_1(u_1, u_2) \oplus P_2(u_2, u_1)$.
    * $u_1 \notin V(P_3)$, sonst Zyklus $P_1(u_1) \oplus P_1(u_2, u_3) \oplus P_3(u_3, u_1)$

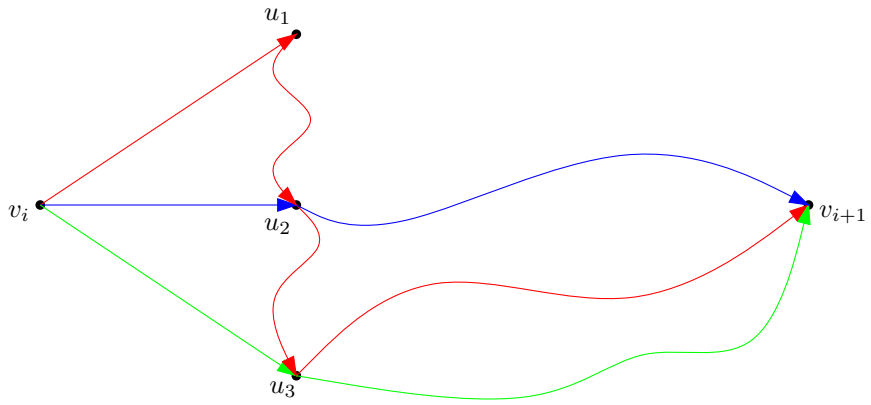



Abbildung 4.5: Fall 2.1: $P_1$ geht durch $u_1, u_2$ und $u_3$. Dann kann $P_2$ nur durch $u_2$ und $P_3$ nur durch $u_3$ gehen.

  - $u_2 \notin V(P_3)$, sonst Zyklus $P_1(u_2, u_3) \oplus P_3(u_3, u_2)$.
  - $u_3 \notin V(P_2)$, sonst $d_{\text{out}}(u_3) = 3$

  Daraus folgt, dass die Wege $P_2$ bzw. $P_3$ von den direkten Nachfolgern von $v_i$ nur $u_2$ bzw. $u_3$ besuchen können, also:

$$d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_2) + d_{out}(u_3) = 3 + 1 + 2 + 2 = 8$$

- **Fall 2.2:** Es gibt keinen Weg, der durch alle drei Knoten $u_1, u_2, u_3$ geht, aber mindestens ein Weg, o.B.d.A. $P_1$, verläuft durch zwei Knoten, o.B.d.A. $u_1, u_2$. Dann gelten auf jeden Fall:
  - $u_1 \notin V(P_2)$, sonst Zyklus $P_1(u_1, u_2) \oplus P_2(u_2, u_1)$.
  - $u_2 \notin V(P_3)$, sonst $d_{\text{out}}(u_2) = 3$.

  Es bleiben also für $P_2$ noch die Möglichkeiten, nur über $u_2$ zu verlaufen oder über $u_2$ und $u_3$. Für $P_3$ bleiben noch die Möglichkeiten über $u_3$ oder über $u_3$ und $u_1$.
  Wir machen zunächst eine Fallunterscheidung nach dem Verlauf von $P_2$, also danach, ob $u_3 \in V(P_2)$ oder nicht.
  - **Fall 2.2.1:** $P_2$ verläuft nicht über $u_3$, d. h. $u_3 \notin V(P_2)$. In diesem Fall bleiben für $P_3$ noch die Möglichkeiten über $u_3$ und $u_1$ oder nur über $u_3$. Wir machen nun eine Fallunterscheidung danach, ob $u_1 \in V(P_3)$ oder nicht.
    - **Fall 2.2.1.1:** $P_3$ verläuft nicht über $u_1$, also $u_1 \notin V(P_3)$. Dieser Fall ist in Abbildung 4.6 dargestellt und es gilt:

$$d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_2) + d_{out}(u_3) = 3 + 1 + 2 + 1 = 7$$

    - **Fall 2.2.1.2:** $P_3$ verläuft über $u_1$, also $u_1 \in V(P_3)$. Dieser Fall ist in Abbildung 4.7 dargestellt und es gilt:

$$d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_2) + d_{out}(u_3) = 3 + 2 + 2 + 1 = 8$$

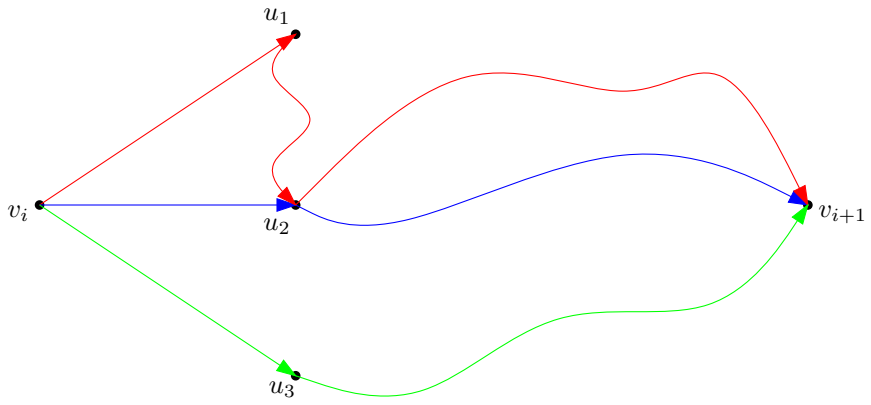



Abbildung 4.6: Fall 2.2.1.1 der Fallunterscheidung.

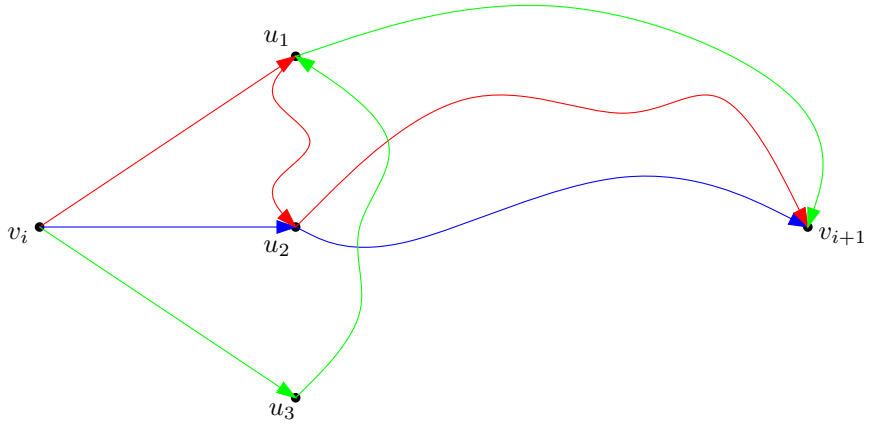



Abbildung 4.7: Fall 2.2.1.2 der Fallunterscheidung.

∗ **Fall 2.2.2:** $P_2$ verläuft über $u_3$, also $u_3 \in V(P_2)$. Insgesamt verläuft also $P_1$ über $u_1$ und $u_2$ und $P_2$ über $u_2$ und $u_3$. Dieser Fall ist in Abbildung 4.8 dargestellt.
Es muss dann gelten:
  · $u_2 \notin V(P_3)$, sonst Zyklus $P_2(u_2, u_3) \oplus P_3(u_3, u_2)$
  · $u_1 \notin V(P_3)$, sonst Zyklus $P_1(u_1, u_2) \oplus P_2(u_2, u_3) \oplus P_3(u_3, u_1)$
$P_3$ kann in diesem Fall also nicht über $u_1$ verlaufen und es gilt:

$$d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_2) + d_{out}(u_3) = 3 + 1 + 2 + 2 = 8$$

– **Fall 2.3:** Jeder der drei Wege verläuft nur durch einen der direkten Nachfolger von $v_i$, also $u_2, u_3 \notin V(P_1)$, $u_1, u_3 \notin V(P_2)$ und $u_1, u_2 \notin V(P_3)$. Dies ist der einfachste Fall und es gilt:

$$d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_2) + d_{out}(u_3) = 3 + 1 + 1 + 1 = 6$$

Auch im Fall 2, in dem keiner der direkten Nachfolger von $v_i$ von allen drei Wegen besucht wird, gilt also die folgende Ungleichung zwischen dem Gewicht und der

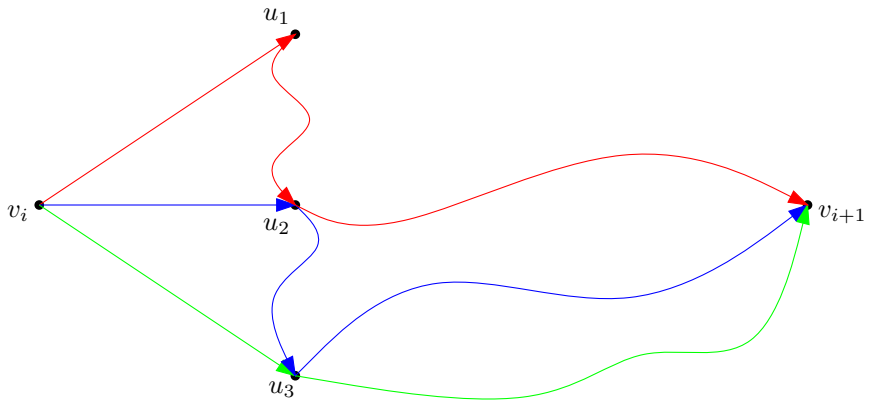



Abbildung 4.8: Fall 2.2.2 der Fallunterscheidung.

Anzahl der besuchten Knoten:

$$\begin{aligned} W(P(v_i, v_{i+1})) &= |E(P(v_i, v_{i+1}))| \\ &= d_{out}(v_i) + d_{out}(u_1) + d_{out}(u_2) + d_{out}(u_3) \\ &\quad + \sum_{v \in V(P(v_i, v_{i+1})) \setminus \{v_i, v_{i+1}, u_1, u_2, u_3\}} d_{out}(v) \\ &\leq 8 + 2(|V(P(v_i, v_{i+1}))| - 5) \\ &= 2(|V(P(v_i, v_{i+1}))| - 1) \end{aligned}$$

Wir haben also gezeigt, dass für jedes Teilstück $P(v_i, v_{i+1})$ die Ungleichung

$$W(P(v_i, v_{i+1})) \leq 2(\mid V(P(v_i, v_{i+1})) - 1) \mid)$$

gilt und damit für das Gewicht der ganzen Wegemenge:

$$W(P) = \sum_{i=0}^{l-1} W(P(v_i, v_{i+1})) \leq 2 \sum_{i=0}^{l-1} \mid V(P(v_i, v_{i+1})) \mid -2l.$$

Aufgrund von Satz 17 kann jeder Knoten $u \in V(P) \setminus \{v_0, \ldots, v_l\}$ genau einem Teilstück zugeordnet werden. Die Knoten $v_i$ für $i = 1, \ldots, l-1$ sind dagegen sowohl in $P_(v_{i-1}, v_i)$ als auch in $P_(v_i, v_{i+1})$. Daher gilt:

$$\sum_{i=0}^{l-1} \mid V(P(v_i, v_{i+1})) \mid = \mid V(P) \mid + l - 1$$

Durch Einsetzen ergibt sich schließlich die Aussage des Satzes:

$$W(P) \leq 2(\mid V(P) \mid + l - 1) - 2l = 2(\mid V(P) \mid -1) = 2\mathcal{E}(P)$$

□

## 4.5 Der Approximationsfaktor des LDMW-Algorithmus im Single-Power-Problem für $k = 3$

**Satz 19.** *Für $k = 3$ ist 2 eine obere Schranke für den Approximationsfaktor des LDMW-Algorithmus angewandt auf das Problem k-SPEDPWMA.*

*Beweis.* Zu einer energieminimalen Wegemenge $P^*$ lässt sich eine inklusionsminimale Wegemenge $P$ mit gleicher Energie konstruieren, d. h. $\mathcal{E}(P) = \mathcal{E}(P^*) = \mathcal{E}_{\text{opt}}$. Nach Satz 18 gilt $W(P) \leq 2\mathcal{E}(P) = 2\mathcal{E}_{\text{opt}}$. Da der Approximationsalgorithmus eine Lösung $P'$ minimalen Gewichts berechnet und die Energie einer Wegemenge nach Satz 1 immer kleiner ist als ihr Gewicht folgt $\mathcal{E}(P') \leq W(P') \leq W(P) \leq 2\mathcal{E}(P) = 2\mathcal{E}_{\text{opt}}$. □

**Satz 20.** *Für $k = 3$ ist der Approximationsfaktor des LDMW-Approximationsalgorithmus angewandt auf das Problem k-SPEDPWMA genau 2.*

*Beweis.* Nach Satz 19 ist 2 eine obere Schranke für den Approximationsfaktor. Um die Aussage zu zeigen genügt es also, ein Beispiel anzugeben, bei dem diese obere Schranke auch erreicht wird. Ein solches Beispiel ist in Abbildung 4.9 abgebildet. Der Weg $P_1 =$

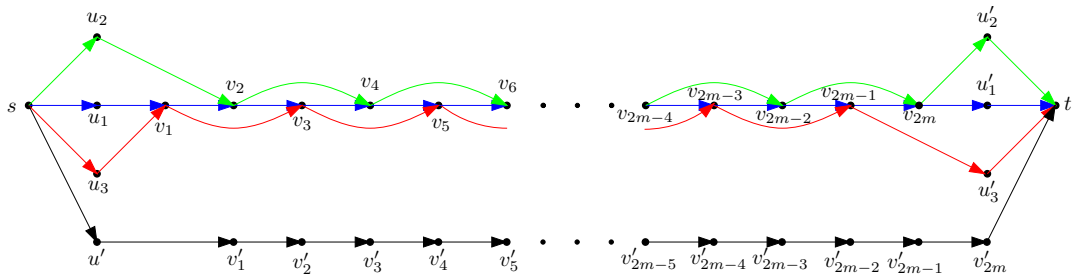



Abbildung 4.9: Die Lösung des Approximationsalgorithmus ist in diesem Beispiel um den Faktor 2 schlechter als die optimale Lösung.

$s - u_1 - v_1 - v_2 - \ldots - v_{2m} - u_1' - t$ (in Abbildung 4.9 blau dargestellt) enthält $2m + 3$ Kanten, also $W(P_1) = 2m + 3$. Der Weg $P_2 = s - u_2 - v_2 - v_4 - v_6 - \ldots - v_{2m-2} - v_{2m} - u_2' - t$ (in Abbildung 4.9 grün dargestellt) enthält $2 + (m - 1) + 2$ Kanten, also $W(P_2) = m + 3$. Der Weg $P_3 = s - u_3 - v_1 - v_3 - v_5 - \ldots - v_{2m-3} - v_{2m-1} - u_3' - t$ (in Abbildung 4.9 rot dargestellt) enthält ebenfalls $m + 3$ Kanten, also $W(P_2) = m + 3$. Der Weg $P_E = s - u' - v_1' - v_2' - \ldots - v_{2m}' - t$ (in Abbildung 4.9 schwarz dargestellt) enthält $2m+2$ Kanten, also $W(P_E) = 2m+2$. Der Approximationalgorithmus wählt nun die drei Wege mit minimalem Gewicht, also $P' = \{P_1, P_2, P_E\}$. Diese haben eine Energie von $\mathcal{E}(P') = \mid \{s, u_2, u_3, v_1, v_2, \ldots, v_{2m}, u_2', u_3', u', v_1', \ldots, v_{2m}'\} \mid = 4m + 6$. Andererseits ist die energieoptimale Lösung $P^* = \{P_1, P_2, P_3\}$. Diese hat die Energie $\mathcal{E}(P^*) = \mid \{s, u_1, u_2, u_3, v_1, v_2, \ldots, v_{2m}, u_1', u_2', u_3'\} \mid = 2m + 7$. Es folgt also:

$$\frac{\mathcal{E}(P')}{\mathcal{E}(P^*)} = \frac{4m + 6}{2m + 7} \to 2 \quad (m \to \infty)$$

Da $m$ beliebig groß gewählt werden kann, wird der Faktor 2 beliebig nahe angenähert, d. h. der Approximationsfaktor ist 2. □

Es konnte also gezeigt werden, dass der Approximationsfaktor des LDMW-Approximationsalgorithmus für das Single-Power-Problem im Fall $k = 3$ gleich 2 ist, während bei beliebig vielen Energiestufen nur ein Faktor von 3 erreicht werden kann.

## 4.6 Untere Schranke für den Approximationsfaktor für größere $k$

Während es gelungen ist, den Approximationsfaktor des LDMW-Algorithmus im Single-Power-Problem für den Fall $k = 3$ exakt anzugeben, ist dies für größere $k$ leider nicht gelungen. Da das Single-Power-Problem ein Spezialfall des allgemeineren $k$-EDPWMA-Problems ist, ist $k$ selbstverständlich auch hier eine obere Schranke für den Approximationsfaktor. Allerdings ist es im Rahmen dieser Arbeit nicht gelungen, ein Beispiel anzugeben, das zeigt, dass diese Schranke für größere $k$ auch im Single-Power-Fall scharf ist. Stattdessen soll im nächsten Satz durch Verallgemeinerung des Beispiels im Beweis von Satz 20 für bestimmte $k$ eine untere Schranke gezeigt werden, die sich aber von der oberen Schranke $k$ unterscheidet.

**Satz 21.** *Ist $k$ von der Form $k = \frac{l(l+1)}{2}$ für ein $l \in \mathbb{N}$, so ist $1+\frac{l}{2}$ eine untere Schranke für den Approximationsfaktor des LDMW-Algorithmus angewandt auf das $k$-SPEDPWMA-Problem.*

*Beweis.* Der Beweis erfolgt durch Angabe eines Beispiels, das eine Verallgemeinerung des Beispiels im Beweis von Satz 20 ist. Sei $n \in \mathbb{N}$ und $m = \frac{(l-1)l}{2}$. Zur Vereinfachung der Schreibweise werden die Abkürzungen $s = nl!$ und $r = \frac{nl!}{l-1}$ eingeführt. Wir definieren nun einen gerichteten Graphen $D = (\hat{V}, \hat{A})$ mit

$$\hat{V} = V \cup V_1' \cup \ldots \cup V_m'$$
$$\hat{A} = \bigcup_{i=1}^{l} \bigcup_{j=1}^{i} A_{ij} \cup A_1' \cup \ldots \cup A_m'$$

mit $V = \{s,t\} \cup \{u_{i,j}, u'_{i,j} \mid i = 1,\ldots,l; j = 1,\ldots,i\} \cup \{v_i \mid i = 1,\ldots,s\}$ $V_i' = \{v'_{i,1},\ldots,v'_{i,r}\}$ für $i = 1,\ldots,m$ für $i = 1,\ldots,m$.

Für jedes $i$ werden Kantenmengen $A_{i,j}$ definiert, für $j = 1$

$$A_{i,1} = \{(s,u_{i,1}),(u_{i,1},v_0),(v_0,v_i),(v_i,v_{2i}),\ldots,(v_{nl!-i},v_{nl!}),(v_s,u'_{i,1}),(u'_{i,1},t)\}$$

und für $j = 2,\ldots,i$

$$A_{i,j} = \{(s,u_{i,j}),(u_{i,j},v_{j-1}),(v_{j-1},v_{j-1+i}),\ldots,(v_{s+j-i},u'_{i,j}),(u'_{i,j},t)\}$$

außerdem für $i = 1,\ldots,m$

$$A_i' = \{(s,v'_{i,1}),(v'_{i,1},v'_{i,2}),\ldots,(v'_{i,r-1},v'_{i,r}),(v'_{i,r},t)\}$$

Die Mengen $A_{i,j}$ bzw. $A'_i$ können als Kanten von (kantendisjunkten) Wegen $P_{i,j}$ bzw. $P'_\nu$ aufgefasst werden ($i = 1, \ldots, l; j = 1, \ldots, i; \nu = 1, \ldots, m$). Da die $P_{i,j}$ im wesentlichen Kanten enthalten, die $i$ Knoten überspannen, sprechen wir im Folgenden von Wegen der "Schrittweite" $i$. Die Wege $P_{i,j}$ sind in Abbildung 4.10 die oberen, farbig dargestellten, Wege, während die Wege $P'_\nu$ die unteren, schwarz eingezeichneten, sind. Für das Gewicht der Wege, d. h. die Anzahl ihrer Kanten, gilt

$$W(P_{i,j}) = \begin{cases} 4 + \frac{s}{i} = 4 + \frac{nl!}{i} & \text{falls } j = 1 \\ 3 + \frac{s}{i} = 3 + \frac{nl!}{i} & \text{sonst} \end{cases} \tag{4.1}$$

$$W(P'_\nu) = r + 1 = 1 + \frac{nl!}{l-1} \tag{4.2}$$

für $(i = 1, \ldots, l; j = 1, \ldots, i; \nu = 1, \ldots, m)$.

Es ist klar, dass die Wege der "Schrittweite" $l$, also $P_{l,1}, \ldots, P_{l,l}$ $l$ kantendisjunkte Wege minimalen Gewichts in $\hat{D}$ sind (wenn auch nicht eindeutig). Falls die Zahl der (kantendisjunkten) Wege erhöht werden soll, so steigt das Gewicht bei Hinzunahme eines "oberen" Weges, also eines Weges, der über ein $u_{i,j}$ $(i = 1, \ldots, l; j = 1, \ldots, i)$ führt, mindestens um $3 + \frac{nl!}{l-1}$. Die (in der Zeichnung) unteren Wege $P'_\nu$ haben dagegen nur ein Gewicht von $1 + \frac{nl!}{l-1}$. Es werden also erst diese Wege genommen, bis alle diese Wege zur Wegemenge gehören. Insgesamt wurden dann $l + m = l + \frac{(l-1)l}{2} = \frac{l(l+1)}{2} = k$ Wege gefunden.

Die Wege $P_{\text{minG}} = (P_{l,1}, \ldots, P_{l,l}, P'_1, \ldots, P'_m)$ sind also $k$ kantendisjunkte Wege minimalen Gewichts. Die Energie dieser Wege beträgt

$$\begin{aligned} \mathcal{E}(P_{\text{minG}}) &= 1 + l + (s+1) + l + mr \\ &= 2 + 2l + nl! + \frac{l(l-1)}{2}\frac{nl!}{l-1} = \frac{nll!}{2} + nl! + 2l + 2 = (1 + \frac{l}{2})nl! + 2l + 2. \end{aligned}$$

Die Energie der $k$ kantendisjunkten Wege $\mathcal{E}(P_{\text{minE}}) = P_{1,1}, P_{2,1}, P_{2,2}, \ldots, P_{l,l}$ beträgt hingegen

$$\mathcal{E}(P_{\text{minE}}) = 1 + k + (s+1) + k = nl! + 2k + 2.$$

Für den Quotienten gilt also

$$\frac{\mathcal{E}(P_{\text{minG}})}{\mathcal{E}(P_{\text{minE}})} = \frac{(1 + \frac{l}{2})nl! + 2l + 2}{nl! + 2k + 2} \to 1 + \frac{l}{2} \quad (n \to \infty).$$

□

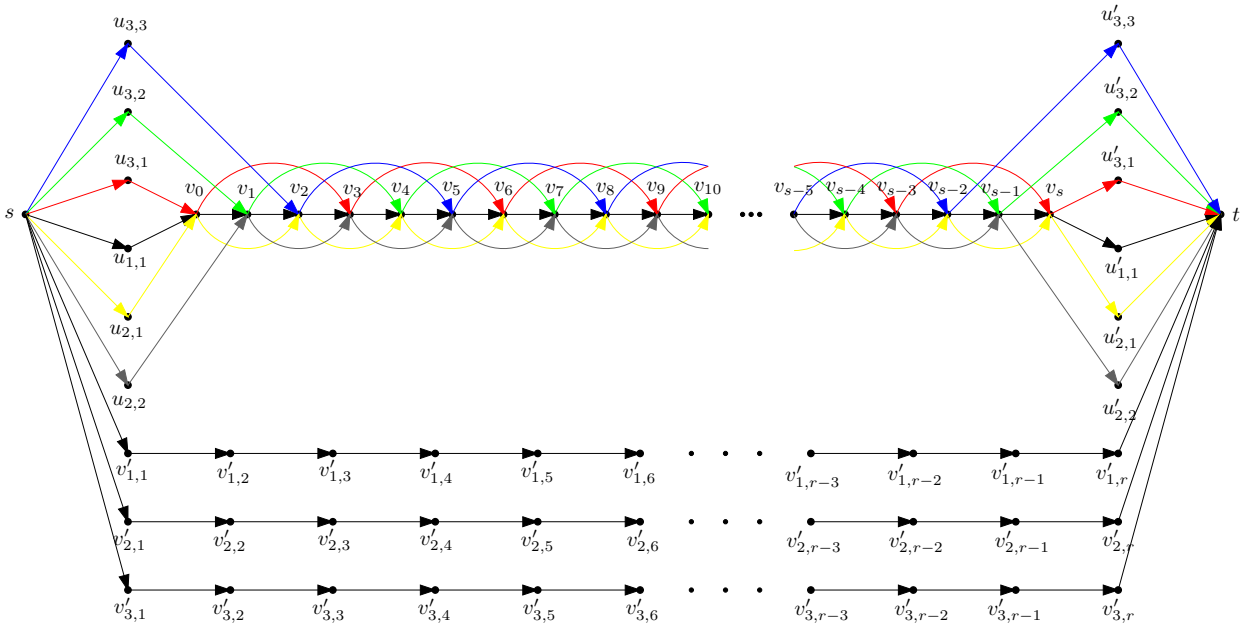



Abbildung 4.10: Der Graph $\hat{D}$ aus dem Beweis zu Satz 21 für $k = 6$, also $l = 3$ und $m = 2$. Die Wege der "Schrittweite 3", $P_{3,1}, P_{3,2}, P_{3,3}$, sind in rot, grün und blau eingezeichnet. Die Wege der "Schrittweite" 2, $P_{2,1}$ und $P_{2,2}$, sind in gelb und grau eingezeichnet und der Weg $P_{1,1}$ der "Schrittweite" 1 in schwarz. Die unteren schwarzen Wege sind die Wege $P'_1, P'_2$ und $P'_3$.

# 5 Experimentelle Ergebnisse

## 5.1 Hilfsmittel

Zur Duchführung der Experimente wurden die Java-Klassenbibliothek yFiles und der Grapheneditor yEd der Tübinger Firma yWorks GmbH verwendet. Die Bibliothek yFiles stellt effiziente Iplementierungen von Datenstrukturen für Graphen bereit, sowie eine Vielzahl von wichtigen Graphen- und Netzwerkalgorithmen [12]. Im Rahmen dieser Arbeit waren insbesondere die Algorithmen zur Berechnung maximaler Flüsse, Flüsse minimaler Kosten und längster Pfade von Bedeutung. Zur Berechnung maximaler Flüsse wird die Preflow-Push-Methode eingesetzt, die für ein Netzwerk mit $n$ Knoten und $m$ Kanten sowie einem maximalen Knotengrad $d_{\max}$ im schlimmsten Fall eine Laufzeit in $O(d_{\max}n^2\sqrt{m})$ hat. Zur Berechnung von Flüsen minimaler Kosten wird ein pseudopolynomieller Algorithmus mit einer Laufzeit in $O(m\log(c_{\max})(m + n\log n))$ verwendet, wobei $c_{\max}$ die maximale Kantenkapazität bezeichne. Da sämtliche hier verwendeten Netzwerke eine maximale Kantenkapazität $c_{\max} = 1$ haben, um die kantendisjunktheit der so gefundenen Wege sicherzustellen, ergibt sich für die Zwecke dieser Arbeit eine polynomielle Laufzeit in $O(m^2 + mn\log n)$.

yEd ist ein in Java implementierter Grapheneditor, der zur Erzeugung und Darstellung von Graphen verwendet werden kann, und durch sogenannte Module einfach erweiterbar ist. Im Rahmen dieser Arbeit entstanden die Module NetworkGenerator , FindSourceSink , CountDisjointPaths , ComputeMinCostPaths , DirectGraph und ExactAcyclicSolution , die in Abschnitt 5.2 vorgestellt werden sollen.

Zum Erzeugen der Diagramme in diesem Kapitel wurde die Statistiksoftware R verwendet [24], [30].

## 5.2 Die yEd-Module

### 5.2.1 Das Modul NetworkGenerator

Wie der Name schon sagt, ist das Modul NetworkGenerator für die Erzeugung von Sensornetzen zuständig. Dabei werden die folgenden Annahmen gemacht:

- Die Knoten befinden sich auf den Gitterpunkten eines regelmäßigen Gitters.
- Auf jedem Gitterpunkt kann sich höchstens ein Knoten befinden.
- Alle Knoten haben omnidirektionale Antennen und dieselbe maximale Sendeenergie.

- Die Knoten sind auf den Gitterpunkten gleichverteilt.

Als Parameter werden die Länge und Breite des Gitters, die Anzahl der Knoten, der Senderadius der Knoten und die Verlustkonstante $\alpha$ angegeben. Die Angabe des Senderadius $R$ der Knoten legt bei gegebenem $\alpha$ die maximale Sendeenergie $\mathcal{E}_{\max}$ der Knoten fest, denn es gilt: $\mathcal{E}_{\max} = R^{\alpha}$.

### 5.2.2 Das Modul FindSourceSink

Das Modul FindSourceSink dient zur Angabe von Quelle und Senke bzw. von Sender und Empfänger. Als Parameter werden die Knotennummern der entsprechenden Knoten verlangt. Falls die Knotennummern nicht existieren, so werden zwei Knoten zufällig als Quelle und Ziel ausgewählt. Die Quelle wird grün markiert, während das Ziel rot markiert wird.

### 5.2.3 Das Modul CountDisjointPaths

Das Modul CountDisjointPaths berechnet die maximale Anzahl kantendisjunkter Wege im Netzwerk bei gegebenem Start- und Zielknoten. Wie in Abschnitt 3.2 erwähnt, kann dies mit Algorithmen zur Berechnung maximaler Flüsse erfolgen. In Modul CountDisjointPaths kommt die in in Abschnitt 5.1 erwähnte Implementierung der Preflow-Push-Methode zum Einsatz. Die gefundenen Wege, die im Graphen rot markiert werden, während alle anderen Kanten in einem hellen Grau dargestellt werden, haben nicht minimales Gewicht.

### 5.2.4 Das Modul ComputeMinCostPaths

Das Modul ComputeMinCostPaths bekommt als Parameter die gewünschte Anzahl von Wegen und berechnet entsprechend viele Wege minimalen Gewichts bzw. gibt eine Fehlermeldung aus, falls so viele Wege nicht existieren. Wie beim Modul CountDisjointPaths werden die Kanten, die zu den gefundenen Wegen gehören, im Graphen rot dargestellt, während alle anderen Kanten in einem hellen Grau dargestellt werden. Aus Gründen der einfacheren Implementierbarkeit wurde hier nicht der Algorithmus von Suurballe implementiert, sondern das Problem als Minimalkostenflussproblem formuliert und mit dem in Abschnitt 5.1 erwähnten Algorithmus gelöst. Über die Konsole wird die Energie der Wege minimalen Gewichts, also der Wert einer LDMW-Approximation ausgegeben. Ein Beispiel für ein Netzwerk, in dem ein Start- und ein Zielknoten ausgezeichnet sind und die drei Wege minimalen Gewichts berechnet wurden, ist in Abbildung 5.1 dargestellt.

### 5.2.5 Die Module DirectGraph und ExactAcyclicSolution

Die Module DirectGraph und ExactAcyclicSolution implementieren zusammen die ESAS--Heuristik zur Bestimmung von $k$ kantendisjunkten Wegen aus Abschnitt 3.5.4. Sie setzen die Existenz einer Quelle und einer Senke voraus, können also erst nach Anwendung

von FindSourceSink angewandt werden. DirectGraph enthält eine Implementierung des Algorithmus zur Konstruktion azyklischer Subgraphen (Algorithmus 4). ExactAcyclicSolution setzt einen azyklischen Graphen mit gegebener Quelle und Senke voraus und berechnet in diesem die drei Wege minimaler Energie mit dem *EAS*-Algorithmus aus Abschnitt 3.5.3 . Die Beschränkung auf $k = 3$ liegt hauptsächlich an der Komplexität der Algorithmen für größere $k$. In Abbildung 5.2 ist das Ergebnis der Anwendung der beiden Module auf den Graphen aus Abbildung 5.1 zu sehen.

Leider ist die Speicherkomplexität der hier verwendeten Implementierung des exakten Algorithmus für azyklische Graphen sehr hoch: Für jedes Knotentripel aus Knoten einer Schicht werden die minimale Energie, mit der es erreichbar ist, sowie die drei Vorgängerkanten gespeichert. Insbesondere wird auch für solche Tripel Speicher reserviert, zu denen es keine drei Wege gibt. Durch Zugeständnisse bei der Laufzeit (die sich weniger als Problem herausgestellt hat) könnte der Speicherbedarf also reduziert werden, indem nur noch für Tripel Speicherplatz reserviert wird, die auch gemeinsam erreicht werden können. Es hat sich herausgestellt, dass es auf der verwendeten Hardware mit 512 MB Hauptspeicher bei etwas dichteren Graphen schnell zu einem Speicherüberlauf kommt. In der Tabelle in Abschnitt 5.3.3 sind einige Kombinationen von Knotenanzahl und Senderadius aufgeführt, für die es in der Regel mit dieser Hardware keine Probleme gab.

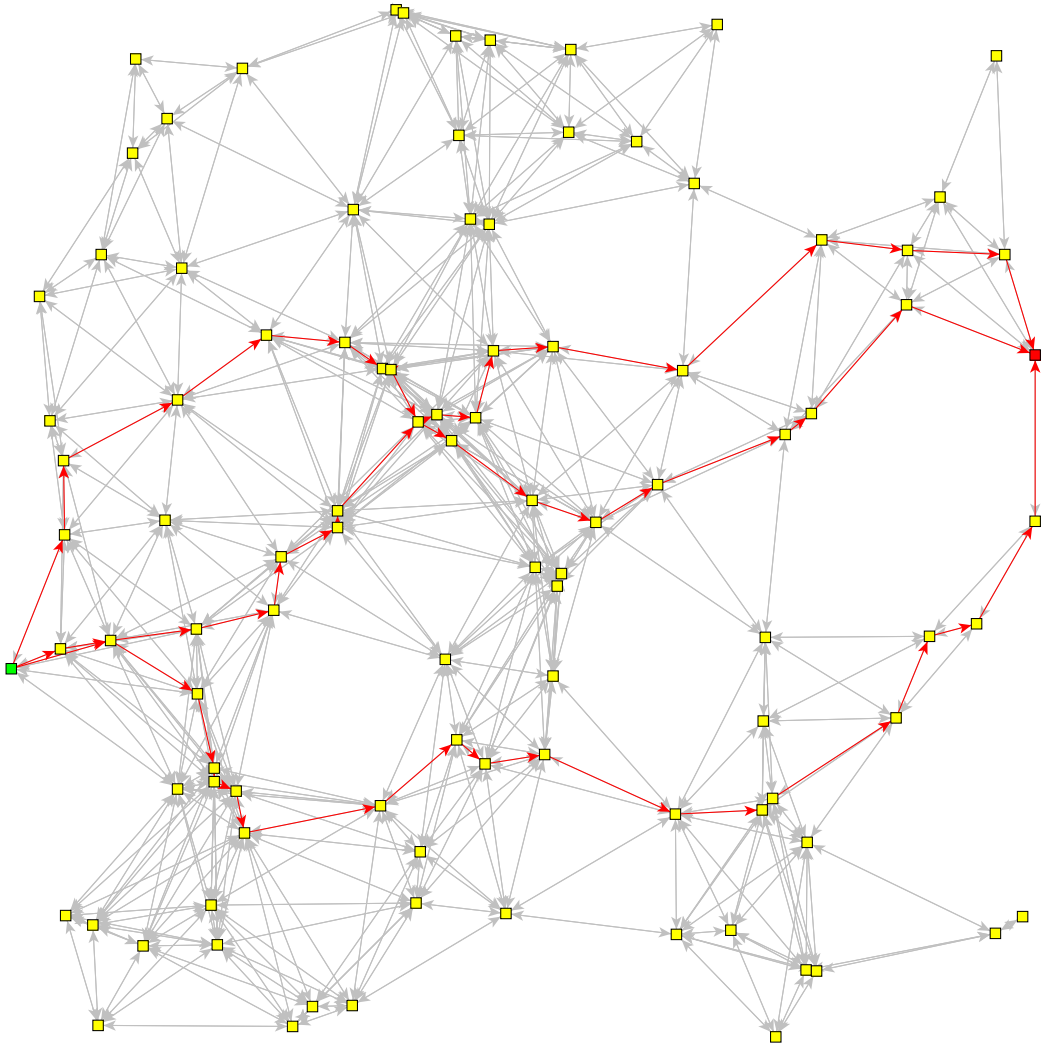


Abbildung 5.1: Ein Graph mit einem $1000 \times 1000$-Gitter, 100 Knoten Radius 200 und Verlustkonstante $\alpha = 2$. Start- bzw. Zielknoten sind grün bzw. rot markiert. Die roten Kanten gehören zu den drei Wegen minimalen Gewichts vom Start- zum Zielknoten. Die Energie dieser Wege beträgt 388861.

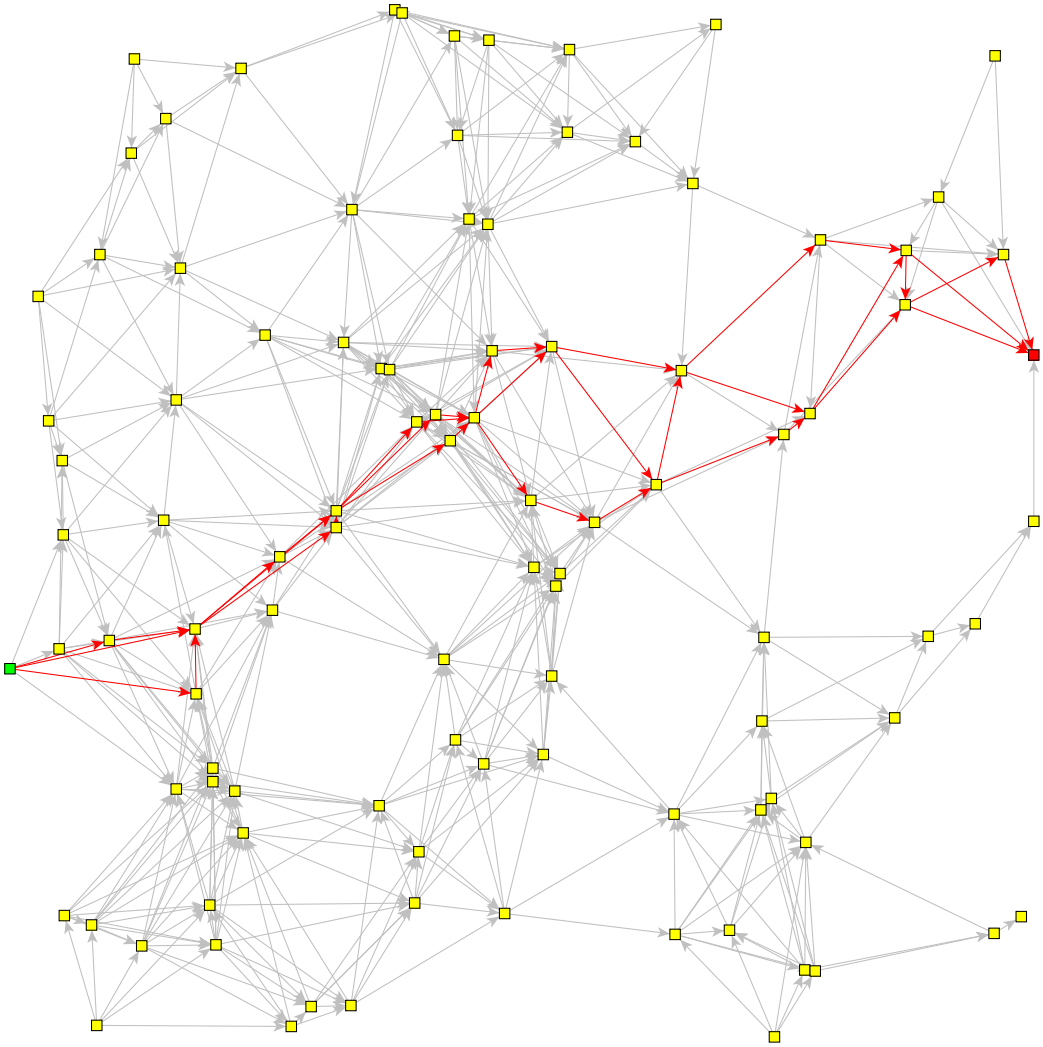


Abbildung 5.2: Der ayzklische Subgraph des Graphen aus Abbildung 5.1 mit den drei Wegen minimaler Energie vom Start- zum Zielknoten. Die Energie dieser Wege beträgt 295739. Dies bedeutet eine um knapp 24 Prozent niedrigere Energie gegenüber den Wegen minimalen Gewichts.

## 5.3 Experimente

Bei den folgenden Experimenten wurde ausnahmslos ein Netzwerkmodell mit omnidirektionalen Antennen verwendet. Außerdem wurde vereinfachend angenommen, dass alle Knoten dieselbe maximale Sendeenergie bzw. denselben Senderadius haben. Die Knoten wurden auf einem quadratischen Gitter der Größe 1000 entsprechend der Gleichverteilung auf den Gitterpunkten platziert. Als Verluskonstante wurde durchweg $\alpha = 2$ gesetzt.

### 5.3.1 Anzahl kantendisjunkter Wege

Zunächst wurde die Anzahl kantendisjunkter Wege zwischen zufällig gewähltem Start- und Zielknoten für verschiedene Anzahlen an Knoten und Radien untersucht. Für eine Anzahl $n$ von Knoten wurden die Daten wie folgt gewonnen: Beginnend mit einem Radius von 20 wurde der Radius in jedem Schritt um 50 erhöht. Für jeden Radius wurden 200 zufällige Graphen mit dem in Abschnitt 5.2.1 beschriebenen Verfahren erzeugt und in jedem Graphen zufällig ein Start- und ein Zielknoten bestimmt. Anschließend wurde die maximale Anzahl kantendisjunkter Wege zwischen Start- und Zielknoten mit der Methode aus CountDisjointPaths (vgl. Abschnitt 5.2.3) berechnet. Schließlich wurde für jede Kombination von Knotenzahl und Radius der Mittelwert über alle Zufallsgraphen gebildet. In Abbildung 5.3 ist die mittlere Anzahl kantendisjunkter Wege für Graphengrößen von 100, 400 und 700 Knoten in Abhängigkeit vom Radius abgebildet.

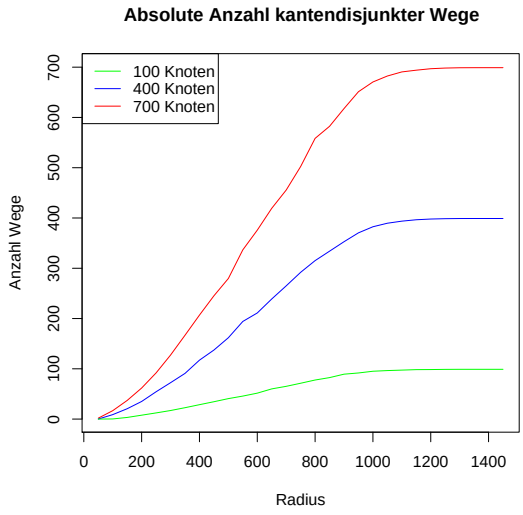



Abbildung 5.3: Die mittlere Anzahl kantendisjunkter Wege zwischen zufällig gewählten Start- und Zielknoten in zufälligen Graphen der Größen 100 Knoten (grün), 400 Knoten (blau) und 700 Knoten (rot).

Wie in Abschnitt 3.2 erklärt, ist die Anzahl kantendisjunkter Wege in einem Graphen mit $n$ Knoten durch $n - 1$ beschränkt. Bei einem Radius von 0 kann es natürlich kei-

nen Weg zwischen zwei Knoten geben, während es genau $n-1$ kantendisjunkte Wege zwischen je zwei Knoten gibt, wenn der Senderadius so gewählt wird, dass der Energiekostengraph ein vollständiger Graph ist (in unserem Beispiel also $1000\sqrt{2}$). Es ist nun interessant, wie sich die relative Anzahl kantendisjunkter Wege bezogen auf die maximale Anzahl kantendisjunkter Wege bei verschiedenen Graphengrößen in Abhängigkeit vom Senderadius verhält. In Abbildung 5.4 ist diese Größe für Graphengrößen von 100, 400 und 700 Knoten dargestellt.

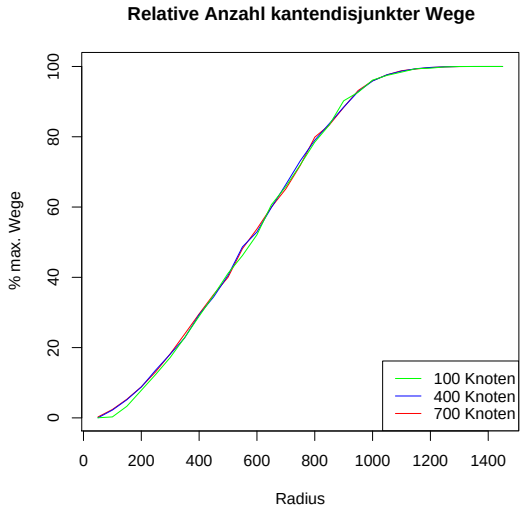



Abbildung 5.4: Die mittlere Anzahl kantendisjunkter Wege zwischen zufällig gewählten Start- und Zielknoten in zufälligen Graphen der Größen 100 Knoten (grün), 400 Knoten (blau) und 700 Knoten (rot) relativ zur maximalen Anzahl kantendisjunkter Wege.

Es ist deutlich zu sehen, dass die Kurven praktisch identisch sind. Lediglich für kleine Radien ist die relative Anzahl kantendisjunkter Wege für Graphen mit weniger Knoten kleiner als für Graphen mit mehr Knoten. Für nicht zu kleine Radien hängt die relative Anzahl kantendisjunkter Wege nur vom Senderadius ab, nicht jedoch von der Anzahl der Knoten im Graphen. Andererseits bedeutet dies, dass die absolute Anzahl kantendisjunkter Wege bei größeren Radien linear von $n-1$ abhängt.

### 5.3.2 Existenz von Wegen im azyklischen Subgraphen

In Abschnitt 3.5.4 wurde eine Heuristik zur Berechnung möglichst energieoptimaler kantendisjunkter Wege vorgestellt, die auf dem exakten Algorithmus für azyklische Graphen basiert. Um diesen Algorithmus anwenden zu können, muss für einen beliebigen, nicht azyklischen, Energiekostengraphen erst mit Algorithmus 4 ein geeigneter azyklischer Subgraph berechnet werden. Wie dort gezeigt kann es dabei vorkommen, dass es im azyklischen Subgraphen nicht mehr die geforderte Anzahl an kantendisjunkten Wegen

gibt, obwohl es sie im ursprünglichen Graphen gab. Hier soll deshalb für $k = 3$ untersucht werden, wie oft dieser Fall vorkommt.

Für verschiedene Knotenanzahlen wurde die relative Häufigkeit der Existenz von drei Wegen in Abhängigkeit vom Senderadius der Knoten untersucht. Dazu wurde der Senderadius in Schritten von 5 verändert. Für jede Kombination von Knotenanzahl und Radius wurden nun 200 Graphen zufällig erzeugt, indem die Knoten gleichverteilt auf die Gitterpunkte platziert wurden und anschließend die Kanten in Abhängigkeit von Radius berechnet wurden. Es wurden zufällig zwei verschieden Knoten als Quelle und Senke ausgezeichnet und durch Anwendung von Flussmethoden wie im Modul CountDisjointPaths (vgl. Abschnitt 5.2.3) die Anzahl der kantendisjunkten Wege bestimmt. War sie mindestens drei, wurde ein Zähler hochgezählt. Für jede betrachtete Kombination von Knotenanzahl und Radius konnte so durch Divison dieses Zählers durch die Anzahl der erzeugten Graphen, 200, die relative Häufigkeit der Existenz von drei Wegen berechnet werden.

In Abbildung 5.5 sind für 50 und 200 Knoten die relativen Häufigkeiten der Existenz von drei Wegen bei Radien zwischen 20 und 400 abgebildet. Es zeigt sich, dass es für jede Knotenzahl einen unteren Grenzradius $r_u$ und einen oberen Grenzradius $r_o$ gibt. Für Radien kleiner als $r_u$ bzw. größer als $r_o$ ist die relative Häufigkeit sowohl im ursprünglichen Graphen als auch im azyklischen Subgraphen gleich 0 bzw. 1. Dazwischen steigt die relative Häufigkeit an, wobei sie hier im azyklischen Subgraphen merklich kleiner ist. Je größer die Knotenanzahl, desto kleiner sind $r_u$ und $r_o$ und desto steiler ist der Anstieg zwischen $r_u$ und $r_o$.

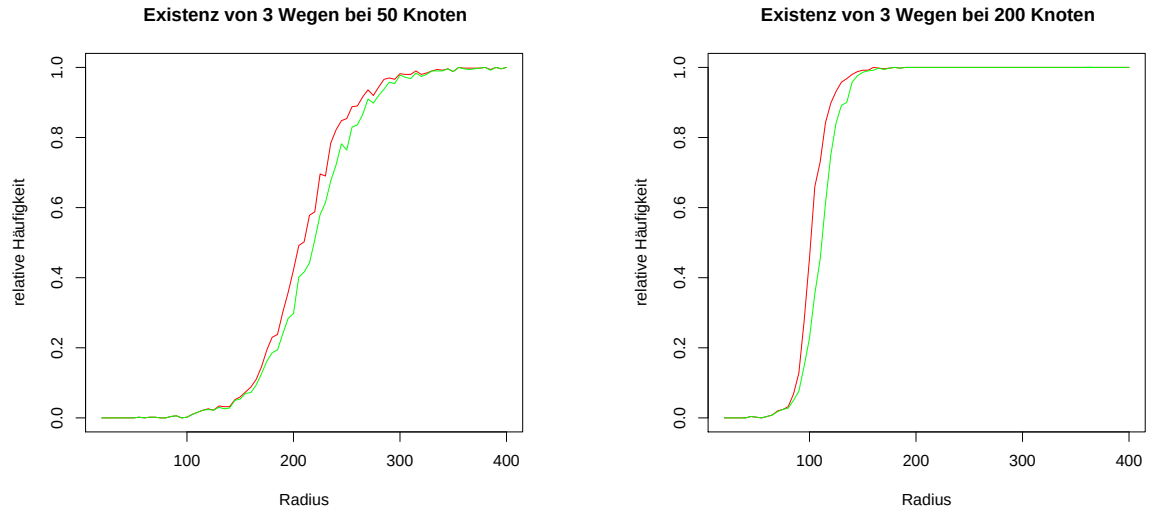



Abbildung 5.5: Relative Häufigkeit der Existenz von drei Wegen bei 50 und 200 Knoten in Abhängigkeit vom Senderadius. Die relative Häufigkeit im ursprünglichen Graphen ist rot eingezeichnet, während die relative Häufigkeit im azyklischen Subgraphen grün eingezeichnet ist.

Interessant ist in diesem Zusammenhang auch, welchen Einfluss die Sonderbehandlung der ein- und ausgehenden Kanten des Startknotens wie in Algorithmus 4 auf die Wahrscheinlichkeit der Existenz dreier Wege hat. Im Diagramm in Abbildung 5.6 ist am Beispiel von 100 Knoten die relative Häufigkeit der Existenz von drei Wegen in einem zufälligen Graphen, in seinem azyklischen Subgraphen nach Algorithmus 4 und in dem azyklischen Subgraphen, der gewonnen wird, indem nur die Kanten, die auf die Senke zulaufen, beibehalten werden, dargestellt. Die Daten wurden unter den gleichen Bedingungen wie oben gewonnen. Es wird deutlich, dass die Sonderbehandlung der ein- und ausgehenden Kanten des Senders einen großen Einfluss auf die Wahrscheinlichkeit der Existenz dreier Wege hat.

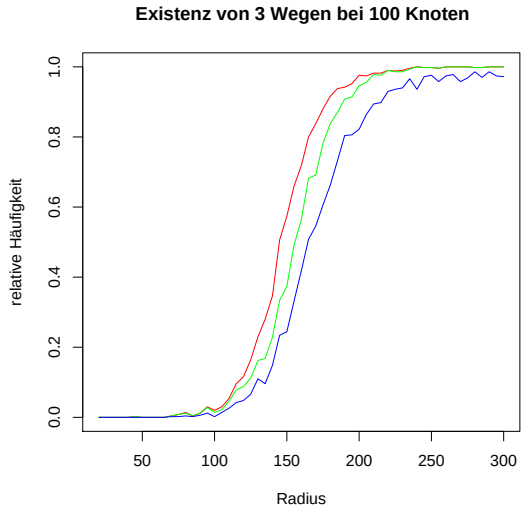



Abbildung 5.6: Die Sonderbehandlung der ein- und ausgehenden Kanten der Quelle hat großen Einfluss auf die Existenz dreier kantendisjunkter Wege. Es ist die relative Häufigkeit der Existenz dreier Wege bei 100 Knoten in Abhängigkeit vom Radius abgebildet. Rot ist im ursprünglichen Graphen, gün im azyklischen Graphen, bei dem die Kanten der Quelle gesondert behandelt werden und blau im azyklischen Subgraphen, in dem alle Kanten auf die Senke zulaufen.

### 5.3.3 Vergleich der ESAS-Heuristik mit dem LDMW-Algorithmus für $k = 3$

In Abschnitt 3.5.4 wurde die ESAS-Heuristik vorgestellt, die auf der Anwendung des exakten Algorithmus für azyklischen Graphen basiert. Dazu ist zuerst die Berechnung eines azyklischen Subgraphen notwendig, wobei Wege wegfallen können. Wie wir jedoch in Abschnitt 5.3.2 gesehen haben, gibt es für jede Graphengröße einen Senderadius, ab dem die Existenz von drei kantendisjunkten Wegen sehr wahrscheinlich wird. Selbst wenn jedoch drei Wege existieren, so kann doch kein Approximationsfaktor angegeben werden.

Der LDMW-Algorithmus wurde in Abschnitt 3.5.5 vorgestellt und basiert auf der Anwendung von Algorithmen zur Bestimmung von kantendisjunkten Wegen minimalen Gewichts. Falls es drei kantendisjunkte Wege in einem Graphen gibt, so findet der LDMW-Algorithmus auch drei Wege, die allerdings nicht energieminimal sein müssen. Für den LDMW-Algorithmus kann für allgemeine Graphen ein Approximationsfaktor von $k$ gezeigt werden.

Es ist nun interessant, zu untersuchen, welcher Algorithmus die besseren Ergebnisse liefert (falls überhaupt beide ein Ergebnis finden). Dazu wurden für verschiedene Kombinationen von Graphengröße (in Knoten) und Senderadius je ungefähr 100 Graphen erzeugt, zufällige Start- und Zielknoten gewählt und - falls es auch im azyklischen Subgraphen noch drei Wege vom Start- zum Zielknoten gab - beide Algorithmen angewandt.

Die folgende Tabelle enthält Werte für einige Kombinationen von Knotenanzahl und Radius. Die Spalten $\mathcal{E}_{\text{LDMW}}$ bzw. $\mathcal{E}_{\text{ESAS}}$ enthalten die durchschnittliche Energie von drei Wegen bei Einsatz des LDMW- bzw. ESAS-Algorithmus; in der Spalte Ersp steht, um wieviel Prozent $\mathcal{E}_{\text{ESAS}}$ kleiner ist als $\mathcal{E}_{\text{LDMW}}$. Die Spalten Min, Max, Drch und Var enthalten die minimale und maximale Ersparnis durch den ESAS-Algorithmus in den einzelnen Beispielen, die durchschnittliche Ersparnis und die Varianz der Energieeinsparungen.

| Knt. | Rad. | $\mathcal{E}_{\text{LDMW}}$ | $\mathcal{E}_{\text{ESAS}}$ | Ersp. | Min | Max | Drch | Var |
|---|---|---|---|---|---|---|---|---|
| 40 | 400 | 249503 | 214236 | 14,1 % | -5,5 % | 34,2 % | 11,6 % | 91,3 |
| 60 | 280 | 232501 | 196702 | 15,4 % | -13,0 % | 33,9 % | 13,8 % | 80,3 |
| 60 | 300 | 230660 | 193356 | 16,2 % | -5,0 % | 35,9 % | 14,9 % | 95,3 |
| 60 | 320 | 237026 | 194484 | 17,9 % | -1,9 % | 38,7 % | 16,1 % | 77,7 |
| 100 | 180 | 160701 | 140488 | 12,6 % | -6,5 % | 55,0 % | 12,5 % | 104,3 |
| 100 | 200 | 165523 | 139428 | 15,8 % | -14,1 % | 39,2 % | 14,5 % | 106,3 |
| 100 | 220 | 179506 | 147703 | 17,7 % | -3,8 % | 42,4 % | 16,4 % | 94,2 |

Es zeigt sich, dass Wege, die durch die ESAS-Heuristik gefunden wurden, im Durchschnitt deutlich weniger Energie benötigen als LDWM-Wege. Dass der gesparte Anteil bei den Durchschnitten (die Spalte Ersp) größer ist als der Durchschnitt der Einsparungen in den einzelnen Beispielen (die Spalte Drch) liegt wohl vor allem daran, dass die Einsparungen bei den Beispielen mit hohem Energieverbrauch höher ausfallen. Für eine feste Knotenzahl ist die durchschnittliche Energieeinsparung durch den ESAS-Algorithmus um so höher, je größer der Senderadius der Knoten. Es gibt allerdings auch Beispiele, bei denen der Energieverbrauch der ESAS-Lösung höher ist als der Energieverbrauch der LDMW-Lösung. Dies ist jedoch sehr selten der Fall und wie man in der Spalte Min sieht, fällt der Mehrverbrauch in der Regel moderat aus.

Vergleicht man die durch den ESAS-Algorithmus gefundenen Lösungen mit den LDMW--Approximationen, z. B. in den Abbildungen 5.1 und 5.2, so fällt auf, dass die Wege der ESAS-Lösung wesentlich enger beeinander verlaufen. Dadurch werden mehr Energieensparungen erzielt, indem Knoten von mehreren Wegen gemeinsam benutzt werden.

### 5.3.4 Die Komplexität der ESAS-Heuristik

Wie wir in Abschnitt 3.5.3 gesehen haben, ist die Zeitkomplexität des Algorithmus für azyklische Graphen mit $n$ Knoten und $m$ Kanten in $O(n^k m^k)$. Im schlimmsten Fall, d. h. falls $m \in O(n^2)$, ergibt sich schon für $k = 3$ eine Komplexität in $O(n^9)$.

Der Zeitbedarf des Algorithmus wird im wesentlichen durch die Anzahl der Kantentripel bestimmt, die betrachtet werden müssen. Außerdem wird eine Tabelle angelegt, die für jedes Knotentripel die minimale Energie bis dahin und die Vorgängerkanten speichert. Der Speicheraufwand wird also im wesentlichen von der Anzahl der Knotentripel bestimmt.

In Abbildung 5.7 ist die Anzahl der Knoten- bzw. Kantentripel in Abhängigkeit von der Anzahl der Knoten (bei konstanter Gittergröße 1000 und konstantem Radius 100) abgebildet. Die Daten wurden gewonnen, indem für jede Kombination von Knotenanzahl und Radius 200 zufällige Graphen erzeugt und zufällig Start- und Zielknoten gewählt wurden. Begonnen wurde mit 10 Knoten und die Knotenanzahl wurde solange um 10 erhöht, bis 1000 Knoten erreicht wurden.

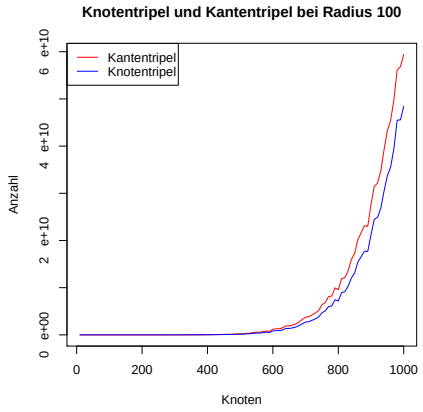



Abbildung 5.7: Die Anzahl der Knoten- bzw. Kantentripel in Abhängigkeit von der Anzahl der Knoten bei konstantem Radius 100.

Neben der Abhängigkeit der Anzahl der Knoten- und Kantentripel von der Anzahl der Knoten ist es sicherlich auch interessant, ihre Abhängigkeit vom Senderadius bei konstanter Knotenanzahl zu untersuchen. In Abbildung 5.8 ist für 50 und 100 Knoten die Anzahl der Knoten- bzw. Kantentripel vom Radius dargestellt. Die Daten wurden gewonnen, indem für jede Kombination von Knotenanzahl und Radius 200 zufällige Graphen erzeugt und zufällig Start- und Zielknoten gewählt wurden. Der Radius wurde in Schritten von 10 verändert. Begonnen wurde mit einem Radius von 20; der letzte Radius war 1400. Eine weitere Erhöhung des Radius ist dann nicht mehr sinnvoll, da in der Regel schon jeder Knoten jeden anderen erreichen kann (bei $1000\sqrt{2} \approx 1414$ ist dies garantiert). Auffällig ist hier vor allem die Kurve für 100 Knoten, die, nachdem sie einen Gipfel überschritten hat, wieder stark abfällt. Für diese Entwicklung wurde noch keine Erklärung gefunden.

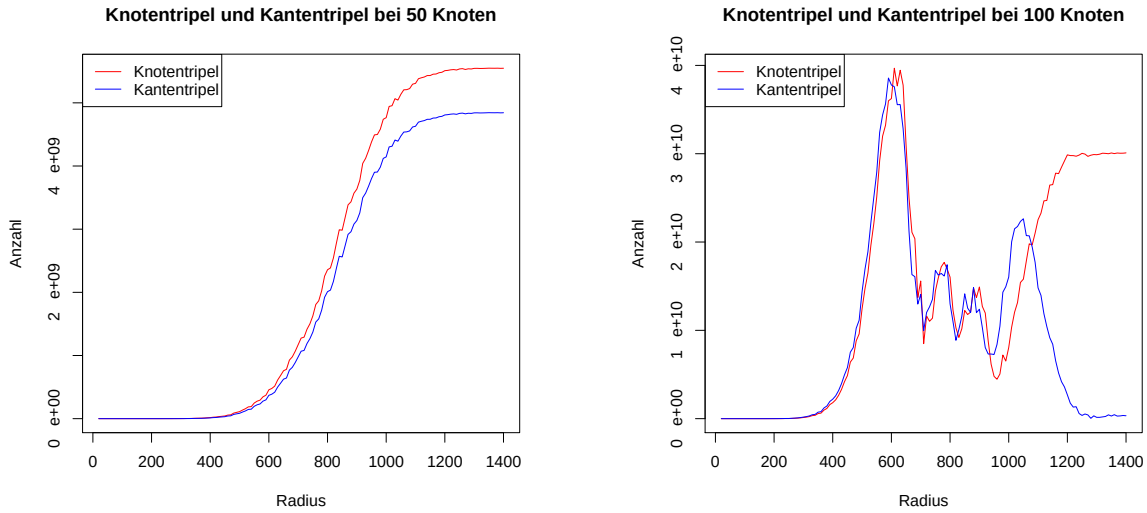



Abbildung 5.8: Die Anzahl der Knoten- bzw. Kantentripel in Abhängigkeit vom Radius bei konstanter Knotenanzahl 50 (links) bzw. 100 (rechts).

# 6 Zusammenfassung und Ausblick

Wir haben gesehen, dass $k$-EDPWMA, also das Problem, $k$ kantendisjunkte Wege minimalen Gewichts in einem gewichteten gerichteten Graphen zu finden, für allgemeine $k$ NP-vollständig ist. In [26] wurde gezeigt, dass es für 2-EDPWMA einen polynomiellen Alorithmus gibt. In dieser Arbeit konnte gezeigt werden, dass es für festes $k$ und azyklische Graphen einen polynomiellen Algorithmus mit einer Laufzeit in $O(n^k m^k)$ gibt, wobei $n$ für die Anzahl der Knoten und $m$ für die Anzahl der Kanten des Graphen steht. Die Komplexität für festes $k$ und allgemeine gewichtete gerichtete Graphen blieb aber leider ungelöst.

In dieser Arbeit wurde das Problem $k$-SPEDPWMA eingeführt, also das Problem, $k$ kantendisjunkte Wege minimaler Energie auf Graphen zu finden, in denen jeder Knoten nur zwei Energiestufen, o.B.d.A. 0 und 1, hat. Es wurde der LDMW-Approximationsalgorithmus für dieses Problem untersucht, und es konnte gezeigt werden, dass der Approximationsfaktor im Fall $k = 3$ gleich 2 ist. Außerdem konnte für größere $k$ von der Form $k = \frac{l(l+1)}{2}$ eine untere Schranke für den Approximationsfaktor von $1 + \frac{l}{2}$ gezeigt werden. Leider ist es nicht gelungen, für $k > 3$ eine bessere obere Schranke als $k$ für den Approximationsfaktor zu zeigen.

Aufbauend auf dem exakten Algorithmus für azyklische Graphen konnte eine Heuristik für Sensornetze, in denen die Koordinaten der Knoten gegeben sind, die sogenannte ESAS-Heuristik, angegeben werden. Für Netze mit omnidirektionalen Antennen in der euklidischen Ebene und $k = 3$ wurden einige Aspekte dieser Heuristik experimentell untersucht: Es zeigte sich, dass die ESAS-Heuristik in den meisten Fällen ein besseres Ergebnis als der LDMW-Approximationsalgorithmus liefert, und dass es für jede Knotenanzahl einen Senderadius gibt, ab dem die Existenz von drei Wegen auch im azyklischen Subgraphen, der in der Heuristik verwendet wird, praktisch sicher ist. Leider ist die Implementierung des exakten Algorithmus für azyklische Graphen schon im Fall $k = 3$ extrem speicherintensiv, so dass die Versuche auf weniger dichte Graphen beschränkt bleiben mussten. Da sich die Zeit in diesen Fällen weniger als Problem herausgestellt hat als der Speicherplatz, könnte eine Implementierung, die auf Kosten der Laufzeit Speicherplatz spart, sinnvoll sein. Aus Zeitgründen konnte dies aber nicht mehr in Angriff genommen werden. Da die Laufzeit der ESAS-Heuristik von der Laufzeit des exakten Algorithmus für azyklische Graphen bestimmt wird, wurde die Laufzeit und der Speicherplatzbedarf dieses Algorithmus untersucht. Die Abhängigkeit der Laufzeit und des Speicherbedarf von der Anzahl der Knoten bei festem Senderadius stellt keine Überraschung dar. Wird hingegen die Anzahl der Knoten festgehalten und der Senderadius variiert, so bietet sich ein überraschendes Bild: Laufzeit und Speicherbedarf steigen zuerst erwartungsgemäß an, überschreiten dann aber einen Gipfel und fallen wieder. Der Autor dieser Arbeit konnte sich diese Entwicklung bisher nicht erklären. Eine theoreti-

sche Untersuchung dieser und anderer Eigenschaften, wie z. B. der Wahrscheinlichkeit der Existenz von $k$ Wegen zwischen zwei beliebigen Knoten, zufälliger Sensornetze, z. B. mit den Methoden aus [22], wäre sicherlich interessant.

# Abbildungsverzeichnis

# Literaturverzeichnis


[1] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[2] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997.

[3] Mario Cagalj, Jean-Pierre Hubaux, and Christian Enz. Minimum-energy broadcast in all-wireless networks:: Np-completeness and distribution issues. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 172–182, New York, NY, USA, 2002. ACM Press.

[4] Wen-Tsuen Chen and Nen-Fu Huang. The strongly connecting problem on multihop packet radio networks. *IEEE Transactions on Communications*, 37(3):293–295, 1989.

[5] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[6] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. Reinform: Reliable information forwarding using multiple paths in sensor networks. In *LCN '03: Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, page 406, Washington, DC, USA, 2003. IEEE Computer Society.

[7] Giuseppe di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: algorithms for the visualization of graphs*. Prentice Hall, 1999.

[8] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, New York, NY, USA, 1999. ACM Press.

[9] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[10] Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.

[11] Michael R. Garey and David S. Johnson. *Computers and intractability.* Freeman, 1979.

[12] YWorks GmbH. yFiles 2.3 API Online-Dokumentation. Technical report, 2005. `www.yworks.com/products/yfiles/doc/api/index.html`.

[13] Michel X. Goemans and David P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In [15], 1997.

[14] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, 1999.

[15] Dorit S. Hochbaum. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, 1997.

[16] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen.* BI-Wiss.-Verl., 1994.

[17] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Dial-M)*, pages 24–33. ACM Press, 2002.

[18] Weifa Liang. Constructing minimum-energy broadcast trees in wireless ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 112–122, New York, NY, USA, 2002. ACM Press.

[19] Errol L. Lloyd, Rui Liu, Madhav V. Marathe, Ram Ramanathan, and S. S. Ravi. Algorithmic aspects of topology control problems for ad hoc networks. *Mob. Netw. Appl.*, 10(1-2):19–34, 2005.

[20] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1995.

[21] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Prentice Hall, 1982.

[22] Mathew Penrose. *Random geometric graphs.* Oxford University Press, 2004.

[23] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, 2000.

[24] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

[25] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, New York, NY, USA, 1998. ACM Press.

[26] Anand Srinivas and Eytan Modiano. Minimum energy disjoint path routing in wireless ad-hoc networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 122–133, New York, NY, USA, 2003. ACM Press.

[27] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[28] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.

[29] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[30] William N. Venables, David M. Smith, and R Development Core Team. *An Introduction to R*. 2005.

[31] Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *Mob. Netw. Appl.*, 7(6):481–492, 2002.

# Index