technische universität
dortmund

# Algorithmen für das Stochastische Steinerbaumproblem auf Serien-Parallelen Graphen

Fritz Bökler

Algorithm Engineering Report
**TR12-1-002**
Juni 2012
ISSN 1864-4503

Fakultät für Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
**http://ls11-www.cs.uni-dortmund.de/**

technische universität
dortmund

Diplomarbeit

# Algorithmen für das Stochastische Steinerbaumproblem auf Serien-Parallelen Graphen

Fritz Bökler
April 2012

Gutachter:
Prof. Dr. Petra Mutzel
Dipl.-Inf. Bernd Zey

Technische Universität Dortmund
Fakultät für Informatik
Algorithm Engineering (Lehrstuhl 11)
http://ls11-www.cs.tu-dortmund.de

## Abstract

In this thesis we discuss the stochastic Steiner tree problem (SSTP) in the context of graphs of bounded treewidth. An undirected graph, a first-stage cost function and a set of scenarios described by a terminal set, a probability and a second-stage cost function are given. As the cost functions suggest, the problem consists of two stages: In the first stage we know all information given. In the second stage a scenario materializes with respect to its probability. The objective is to choose a set of edges $E_0$ in the first stage and for each scenario $k$ a set of edges $E_k$ in the second stage with the corresponding costs so that there is a subset $F_k \subseteq E_0$ in a way that $F_k \cup E_k$ forms a Steiner tree for scenario $k$, minimizing the expected cost.

We show that SSTP is **NP**-hard if the input is restricted to graphs whose treewidth is at most 3. Further, we show that there is a polynomial-time algorithm for SSTP if the input is restricted to graphs of treewidth at most 2, i.e. series-parallel graphs, and only $\mathcal{O}(\log_b |V|)$ scenarios are allowed for a suitable $b > 2$. To achieve this result we adopt a technique developed by J. A. Wald and C. J. Colbourn in [23].

## Zusammenfassung

In dieser Diplomarbeit wird das stochastische Steinerbaumproblems (SSTP) im Kontext von Graphen mit beschränkter Baumweite behandelt. Gegeben sind ein ungerichteter Graph, eine Kostenfunktion für die erste Phase und eine Menge von Szenarien, die jeweils durch eine Terminalmenge, eine Wahrscheinlichkeit und eine Szenarienkostenfunktion für die zweite Phase beschrieben sind. Wie die Kostenfunktionen vermuten lassen, besteht das Problem aus zwei Phasen: In der ersten Phase sind alle Informationen bekannt. In der zweiten Phase steht in Abhängigkeit seiner Wahrscheinlichkeit das Szenario fest, welches realisiert werden soll. Ziel ist es, eine Menge von Kanten $E_0$ in der ersten Phase und für jedes Szenario $k$ eine Menge von Kanten $E_k$ in der zweiten Phase mit den jeweiligen Kosten zu wählen, sodass für eine Menge $F_k \subseteq E_0$ die Kanten $F_k \cup E_k$ einen Steinerbaum für das Szenario $k$ bilden; dabei sollen die erwarteten Kosten minimiert werden.
In der vorliegenden Arbeit wird gezeigt, dass dieses Problem **NP**-schwierig ist, wenn die Eingabe auf Graphen mit Baumweite höchstens 3 eingeschränkt wird. Des Weiteren wird ein Polynomialzeitalgorithmus vorgestellt, der das SSTP löst, wenn die Eingabe auf Graphen mit Baumweite höchstens 2 – also serien-parallele Graphen – eingeschränkt wird und es für ein geeignetes $b > 2$ nur $\mathcal{O}(\log_b |V|)$ Szenarien gibt. Die hierbei verwendete Methode basiert auf dem Algorithmus von J. A. Wald und C. J. Colbourn aus [23].

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Das nach Jakob Steiner benannte Steinerbaumproblem ist eine ausführlich untersuchte Verallgemeinerung des Problems der Berechnung eines minimalen Spannbaumes. Gerade in Design- und Verbesserungsprozessen von Netzwerken spielt es eine große Rolle. Beispielsweise kann ein Übertragungsnetzbetreiber für den Strombereich zum Ziel haben, Teile seines Netzes durch das Austauschen von Leitungen an stärkere Anforderungen anzupassen. Statt das gesamte Übertragungsnetz auf einmal zu modernisieren, kann ein erster Schritt darin bestehen, einen einfach zusammenhängenden Teil des Netzes möglichst kostengünstig mit einer neuen Leitungstechnologie auszustatten.

Im Gegensatz zu der Berechnung eines Spannbaumes, welcher alle Knotenpunkte des Übertragungsnetzes verbindet, muss häufig nur eine bestimmte Teilmenge der Netzwerkknoten verbunden werden. Diese kann z. B. aus einem gewissen Kundenkreis bestehen oder nur die wichtigsten Verteiler enthalten. Das Problem, eine günstigste Auswahl an zu ersetzenden Leitungen zu finden, durch die alle Knoten dieser Teilmenge verbunden werden, ist das Steinerbaumproblem.

Zur mathematischen Beschreibung wird ein Netzwerk durch einen ungerichteten Graphen modelliert. Die zu verbindenden Knoten werden *Terminale* genannt und die Kosten, die durch das Ersetzen von Leitungen bzw. Kanten entstehen, werden mit einer Abbildung der Kanten in die reellen Zahlen beschrieben. Dieses auch kurz als *STP* bezeichnete Problem gehört zu den 21 klassischen Problemen, deren **NP**-Vollständigkeit R. Karp 1972 in [13] gezeigt hat. Aber auch in Lehrbüchern wie z. B. V. V. Vaziranis „Approximation Algorithms“[22] werden das STP und damit verwandte Probleme behandelt.

Im Kontext stochastischer Programmierung lässt sich nun das Problem verallgemeinern: Was ist, wenn zunächst gar nicht bekannt ist, welche Netzwerkknoten verbunden werden sollen? Beispielsweise können Vermutungen über Anforderungen angestellt werden, die in der Zukunft auf das Übertragungsnetz zukommen. Diese Vermutungen können – unter Zuhilfenahme statistischer Erhebungen und Modelle – durch eine endliche Menge von Szenarien angenähert werden.

Auf Basis dieser Informationen soll in einer ersten Phase bereits entschieden werden, für welche Leitungen sich eine Modernisierung lohnt, ohne konkret zu wissen welche Netzwerkknoten an das leistungsfähigere Netz angebunden werden müssen. Zu einem späteren Zeitpunkt hingegen, wenn sich herausstellt, welches der Szenarien zu realisieren ist, gilt es, die noch nicht verbundenen Ziele an das Netz anzuschließen. Die Kosten, die in dieser zweiten Phase entstehen, können sich allerdings verändert haben. Das Unternehmen kann nun z. B. daran interessiert sein, die erwarteten Kosten so gering wie möglich zu halten.

In der Arbeit „Linear Programming Under Uncertainty“[4] aus dem Jahr 1955 hat G. Dantzig für Probleme dieser Art ein allgemeines Modell mit zwei Phasen vorgestellt. Dieses Modell soll hier betrachtet werden. Das Problem kann an das STP anknüpfend modelliert werden: Statt einer Terminalmenge gibt es eine Menge von Szenarien, die jeweils ihre eigenen Terminalmengen besitzen. Jedes Szenario erhält zudem eine Abbildung von den Kanten in die reellen Zahlen, die die Kosten beschreibt, die in der zweiten Phase für das Wählen dieser Kante veranschlagt werden. Als Letztes gehört zu jedem Szenario die Wahrscheinlichkeit mit der es realisiert werden soll. In diesem stochastischen Steinerbaumproblem (kurz *SSTP*) ist das Ziel, die erwarteten Kosten zu minimieren.

Eine der ersten Erwähnungen des SSTP findet sich in [10] im Jahr 2004; in dieser Arbeit wird ein Approximationsalgorithmus mit konstanter Güte für eine Variante des SSTP vorgestellt, in der u. a. die Kosten in der zweiten Phase um einen konstanten Faktor gegenüber den Kosten in der ersten Phase steigen und ein gemeinsames Terminal in allen Szenarien existiert. Im Jahr 2005 wurde in [9] ein Approximationsalgorithmus vorgestellt, der diese Variante ohne die Bedingung eines gemeinsamen Terminals mit konstanter Approximationsgüte löst. In [8] aus dem Jahr 2007 hingegen wird mit Hilfe des *Label-Cover*-Problems aus [5] gezeigt, dass es keinen Approximationsalgorithmus für die hier beschriebene Variante des SSTP mit einer Güte besser als $2^{(\log n)^{1-o(1)}}$ geben kann, wenn $\mathbf{P} \neq \mathbf{NP}$ ist.

In der vorliegenden Arbeit wird – statt Bedingungen an das Problem selbst zu stellen – die Art der Netzwerke eingeschränkt, für die eine Lösung gefunden werden soll. Eine Klasse von Graphen, die gerade beim Design von Netzwerken eine wichtige Rolle spielt, sind *Isolated Failure Immune Networks* oder *IFI-Netwerke* die auf A. M. Farley in [6] zurückgehen. In einem Netzwerk wie dem des Übertragungsnetzbetreibers für den Strombereich ist es fatal, wenn durch den Ausfall eines Knotenpunktes oder einer Leitung Kunden nicht mehr versorgt werden können. Ein solcher Ausfall eines Knotens heißt im Jargon der IFI-Netzwerke *Site-Failure*, ein Ausfall einer Leitung *Line-Failure*.

In einem 2-zusammenhängenden Graphen bleibt nach einem *Site-* oder *Line-Failure* der Zusammenhang erhalten. Aber schon zwei Ausfälle können diesen zerstören. Ein IFI-Netzwerk hat dagegen die Eigenschaft, dass auch mehrere *Site-* oder *Line-Failures* den Zusammenhang des Netzwerkes nicht beeinflussen, sofern jeweils zwei Ausfälle isoliert voneinander vorkommen. „Isoliert“ zu sein bedeutet für zwei *Site-Failures*, dass sie nicht an der gleichen Leitung vorkommen dürfen; bei zwei *Line-Failures* darf nicht der gleiche Netzwerk-

knoten involviert sein. Und ein *Site-* und ein *Line-Failure* gelten als isoliert, wenn keiner der an der ausgefallenen Leitung liegenden Netzwerkknoten eine direkte Verbindung zu dem gestörten Knoten besitzt.

Für den Übertragungsnetzbetreiber war es bei der Konstruktion seines Netzes aller Wahrscheinlichkeit nach sinnvoll, Störungen durch isolierte Ausfälle zu vermeiden. Allerdings bedeuten zusätzliche Leitungen auch zusätzliche Kosten. Günstig für das Unternehmen ist es, nur so viele Leitungen zu ziehen, dass die o. g. Eigenschaft gerade zu erfüllen oder das Netzwerk sogar so zu konstruieren, dass es durch wenige weitere Leitungen zu einem IFI-Netzwerk erweitert werden kann.

Graphentheoretisch entspricht die Menge der minimalen IFI-Netzwerke, aus denen keine Kante entfernt werden darf, ohne dass ihre Eigenschaft verloren geht, genau der Graphenklasse der 2-Bäume (siehe auch [6]). Ein Graph, der durch das Hinzufügen von Kanten zu einem 2-Baum erweitert werden kann, ist ein partieller 2-Baum bzw. ein serien-paralleler Graph. J. A. Wald und C. J. Colbourn haben 1983 in [23] einen Algorithmus vorgestellt, der das STP auf serien-parallelen Graphen in linearer Laufzeit in der Anzahl der Knoten löst.

Der im Jahr 1986 von N. Robertson und P. D. Seymour in [19] eingeführte Begriff der Baumweite verallgemeinert diese Graphenklassen: Die Baumweite beschreibt, wie ähnlich ein Graph einem Baum ist. Jeder Baum hat Baumweite 1; je höher sie ist, desto weniger ähnelt der Graph einem Baum. Die Menge der partiellen $k$-Bäume entspricht dabei nach [16] genau der Menge der Graphen, deren Baumweite höchstens $k$ ist. Serien-parallele Graphen sind somit genau die Graphen mit Baumweite höchstens 2. Für das STP gibt es für jedes $k \in \mathbb{N}$ einen Algorithmus, der es auf Graphen mit Baumweite $k$ löst. Dies wird u. a. in [3] gezeigt.

Betrachtet man Graphen mit Baumweite höchstens $k$, so ist das SSTP auf Graphen, für die $k = 1$ ist – also auf Bäumen –, in linearer Zeit in der Anzahl der Knoten lösbar. Für $k > 1$ sind bisher keine Ergebnisse bekannt.

Das Ziel dieser Diplomarbeit besteht darin, einen möglichst effizienten Algorithmus für das SSTP auf serien-parallelen Graphen, also für den Fall $k = 2$, zu finden, der sich an dem Algorithmus von Wald und Colbourn orientiert. Zudem wird untersucht, ob es auch für Mengen von Graphen mit maximaler Baumweite $k > 2$ effiziente Algorithmen für das SSTP gibt.

## Aufbau dieser Arbeit

Nach der Einführung einiger Begriffe in Kapitel 2 behandelt Kapitel 3 grundlegende Eigenschaften serien-paralleler Graphen, partieller $k$-Bäume und des SSTP. Unter anderem werden Eigenschaften diskutiert, die in späteren Beweisen wichtig sind. Es wird ein Algorithmus mit Laufzeit $\mathcal{O}(kn)$ für das SSTP auf Bäumen vorgestellt sowie ein Algorithmus

für das SSTP auf serien-parallelen Graphen, dessen Laufzeit exponentiell in der Anzahl der Knoten des Graphen ist. Den Kern des Kapitels 3 bildet das Theorem 3.2.7 mit der Aussage, dass es für das SSTP auf Graphen mit Baumweite höchstens 3 unter der Voraussetzungen, dass $\mathbf{P} \neq \mathbf{NP}$ ist, keinen Polynomialzeitalgorithmus geben kann.

Die Technik, die in Kapitel 5 zur Konstruktion des Algorithmus für das SSTP auf serien-parallelen Graphen verwendet wird, ist an die Arbeit von J. A. Wald und C. J. Colbourn (siehe [23]) angelehnt. Der dort beschriebene Algorithmus wird in Kapitel 4 im Detail vorgestellt. Die in diesem Kapitel angewandte Beweistechnik und die Darstellung der Fälle wird in Kapitel 5 benutzt, um die dortigen Algorithmen zu beschreiben.

Für verschiedene Varianten des SSTP auf serien-parallelen Graphen werden in Kapitel 5 Algorithmen vorgestellt. In Abschnitt 5.3 erfolgt dann die Beschreibung des Algorithmus für das SSTP auf serien-parallelen Graphen sowie der Beweis der Korrektheit und der Laufzeit.

# Kapitel 2

# Definitionen und Notationen

## 2.1 Mengen

In dieser Arbeit werden Bezeichnungen verwendet, die zum Begriff der Menge gehören, aber keine einheitliche Notation besitzen. Diese sollen hier eingeführt werden.

**Definition 2.1.1** (Potenzmenge)**.** *Die Menge* $\mathcal{P}(M) := \{A \mid A \subseteq M\}$ *heißt* Potenzmenge *der Menge* $M$*.*

Gelegentlich wird als abkürzende Schreibweise $\mathcal{P}^k(M) := \{A \in \mathcal{P}(M) \mid |A| = k\}$ verwendet.

## 2.2 Graphentheorie

### 2.2.1 Allgemeines

Die hier folgenden Definitionen aus der Graphentheorie lehnen sich an die Inhalte der Vorlesung „Graphentheorie“ von Prof. T. Zamfirescu im Sommersemester 2009 an der TU Dortmund an.

**Definition 2.2.1** (Graph)**.** *Ein* (ungerichteter) Graph $G$ *ist ein Paar bestehend aus einer Menge von* Knoten $V$ *und einer irreflexiven, symmetrischen Relation* $E \subseteq V \times V$*, die* Kantenmenge *heißt.*

Die Knotenmenge eines Graphen $G$ wird auch mit $V(G)$ bezeichnet, die Kantenmenge mit $E(G)$. Die Elemente der Kantenmenge werden *Halbkanten* genannt. Eine *Kante* $\{u, v\}$ besteht aus den beiden Halbkanten $(u, v)$ und $(v, u)$. Es gilt $\{u, v\} \in E :\Leftrightarrow \{(u, v), (v, u)\} \subseteq E$. Da die Kantenrelation symmetrisch ist, existiert für jede Halbkante $(u, v) \in E$ auch die Halbkante $(v, u) \in E$ und somit $\{u, v\} \in E$. In den meisten Fällen können demnach die Kanten als Elemente der Kantenmenge angesehen werden. Eine Kante kann, wie die Notation suggeriert, als die Menge ihrer Endpunkte verwendet werden. In

einem Graphen $G$ heißen zwei Knoten $v, w \in V(G)$ *benachbart* oder *adjazent*, wenn eine Kante zwischen ihnen existiert, also $\{v, w\} \in E(G)$ gilt. Ein Knoten $v$ und eine Kante $e$ werden zueinander *inzident* genannt, wenn $v \in e$ ist.

Der Graph $K_n := (V, V \times V)$ mit $V = \{1, \ldots, n\}$ heißt *vollständiger Graph*. Ein Graph $(U \cup W, E)$ mit $V = \{1, \ldots, n\}$, $U \cup W = V$, $U \cap W = \emptyset$ und $E = (U \times W) \cup (W \times U)$ heißt *vollständig bipartit* und wird mit $K_{|U|,|W|}$ bezeichnet.

Für einen Graphen $G = (V, E)$, $U \subseteq V$ und $F \subseteq E \cap (U \times U)$ ist der Graph $(U, F)$ *Teilgraph* von G. Der Graph $G[U] := (U, E \cap (U \times U))$ mit $U \subseteq V$ wird der durch $U$ *induzierte Teilgraph* von $G$ genannt. Analog für Kanten heißt der Graph $G[F] := (\bigcup_{e \in F} e, F)$ mit $F \subseteq E$ der durch $F$ *induzierte Teilgraph* von $G$.

Der *Grad* eines Knotens $v$ wird mit $d(v) := |\{e \in E \mid v \in e\}|$ bezeichnet. Ein *Weg* ist ein Graph mit $V = \{v_1, \ldots, v_n\}$ und $E = \{\{v_i, v_{i+1}\} \in \mathcal{P}^2(V) \mid 1 \leq i \leq n-1\}$. Ein *Kreis* entspricht einem Weg mit einem zusätzlichen Kantenpaar $\{v_n, v_1\}$. Ist $K$ ein Kreis und $n = |V(K)|$, so wird $K$ auch *n-Eck* genannt. Ein Graph heißt *planar*, wenn er in einer Ebene so dargestellt werden kann, dass sich keine zwei Kanten schneiden. Gilt für zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$, dass $V_1 \cap V_2 = \emptyset$ ist, so werden $G_1$ und $G_2$ *knotendiskunkt* genannt.

**Definition 2.2.2** ($k$-Zusammenhang)**.** *Ein Graph $G$ heißt* $k$-zusammenhängend*, wenn es für je zwei Knoten $v, w \in G$ Wege $P_1, \ldots, P_k$ als Teilgraphen in $G$ gibt, sodass sowohl $\forall i \in \{1, \ldots k\} : v, w \in V(P_i)$ als auch $\forall i, j \in \{1, \ldots k\} : (V(P_i) \cap V(P_j)) \backslash \{v, w\} = \emptyset$ gilt.*

Die 1-zusammenhängenden Graphen werden *zusammenhängende Graphen* genannt. Um Teile eines Graphen zu bezeichnen, die $k$-zusammenhängend sind, wird der Begriff der *k-Zusammenhangskomponente* eingeführt: Für einen Graphen $G = (V, E)$ und $U \subseteq V$ heißt $G[U]$ eine *k-Zusammenhangskomponente*, wenn $G[U]$ $k$-zusammenhängend ist und es keine Menge $W$ mit $U \subset W \subseteq V$ gibt, sodass $G[W]$ $k$-zusammenhängend ist. Eine 1-Zusammenhangskomponente heißt auch *Zusammenhangskomponente* oder *Komponente*; eine 2-Zusammenhangskomponente wird *Block* genannt. Ein Knoten $v$ heißt *Artikulation*, wenn $G[V \backslash \{v\}]$ mehr Zusammenhangskomponenten besitzt als $G$. Allgemein ist eine $k$-elementige Knotenmenge $M$ ein *k-Separator*, wenn $G[V \backslash M]$ mehr Zusammenhangskomponenten als $G$ besitzt.

Ein zusammenhängender Graph, der keinen Kreis als Teilgraphen enthält, wird *Baum* genannt. Sind alle Zusammenhangskomponenten eines Graphen Bäume, so heißt der Graph *Wald*. Ein *aufspannender Teilgraph* eines Graphen $G = (V, E)$ ist ein zusammenhängender Teilgraph $(V, F)$ von $G$ mit $F \subseteq E$. Dementsprechend ist ein *aufspannender Teilbaum* eines Graphen ein aufspannender Teilgraph, der ein Baum ist.

Um zu beschreiben, wie sehr ein Graph einem Baum ähnelt, wird der Begriff der Baumweite eingeführt. Die Definition ist aus [19] entnommen.

**Definition 2.2.3** (Baumzerlegung)**.** *Sei $G = (V, E)$ ein Graph. Eine* Baumzerlegung *ist ein Paar $(M, T)$, wobei für ein $I \subseteq \mathbb{N}_0$ die Menge $M = \{X_i \subseteq V \mid i \in I\}$ und $T$ ein Baum mit $V(T) = I$ ist. Die Menge $M$ und der Baum $T$ müssen die folgenden Eigenschaften erfüllen:*

- $\bigcup_{i \in I} X_i = V$
- $\forall e \in E \; \exists i \in I : |e \cap X_i| = 2$
- $\forall i, j, k \in I$ : *Liegt $j$ auf dem Weg von $i$ nach $k$ in $T$, dann gilt $X_i \cap X_k \subseteq X_j$*

Die *Breite* einer Baumzerlegung ist $\max\{|X_i| - 1 \mid i \in I\}$. Die *Baumweite* eines Graphen ist die Breite derjenigen Baumzerlegung des Graphen mit der kleinsten Breite.

**Definition 2.2.4** (Isomorphie)**.** *Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sind* isomorph *zueinander, wenn es eine bijektive Abbildung $\pi : V_1 \mapsto V_2$ gibt, sodass für jede Kante $\{v, w\} \in E_1 : \{\pi(v), \pi(w)\} \in E_2$.*

Eine *$k$-Clique* ist ein zu dem $K_k$ isomorpher Graph. Ein Graph $G$ enthält eine $k$-Clique, wenn eine $k$-elementige Teilmenge $U$ der Knotenmenge existiert, sodass $G[U]$ isomorph zu $K_k$ ist.

Viele Graphenklassen lassen sich eindeutig über bestimmte Graphen charakterisieren, die nicht in einem Graphen dieser Klasse in einem gewissen Sinne enthalten sein dürfen. Im späteren Verlauf wird dies zur Charakterisierung serien-paralleler Graphen benötigt. Zentral für dieses Konzept ist der Begriff der Homöomorphie:

**Definition 2.2.5** (Homöomorphie)**.** *Zwei Graphen sind* homöomorph *zueinander, wenn ein Graph aus dem anderen durch mehrmalige Anwendung folgender Operation entsteht:*

*Ein Teilgraph $(\{u, v, w\}, \{\{u, v\}, \{v, w\}\})$ mit $d(v) = 2$ wird ersetzt durch den Graphen $(\{u, w\}, \{\{u, w\}\})$ oder umgekehrt.*

Ein *$M$-freier Graph* ist ein Graph, der keinen zu einem Graphen $M$ homöomorphen Teilgraphen enthält. So lassen sich später serien-parallele Graphen als genau die $K_4$-freien Graphen charakterisieren.

### 2.2.2 Serien-parallele Graphen

In dieser Arbeit werden serien-parallele Graphen, ähnlich wie in [12], über partielle $k$-Bäume definiert. Der Zusammenhang zu partiellen $k$-Bäumen ist notwendig, um das stochastische Steinerbaumproblem auf Graphenklassen mit unterschiedlicher maximaler Baumweite in Komplexitätsklassen einzuordnen.

Die Definitionen zu partiellen $k$-Bäumen folgen [2].

**Definition 2.2.6** ($k$-Baum)**.** *Der $K_k$ ist ein $k$-*Baum*. Wird in einem $k$-Baum für eine $k$-Clique ein Knoten hinzugefügt, der genau zu allen Knoten der $k$-Clique benachbart ist, so ist der neue Graph ebenfalls ein $k$-Baum.*

Ein aufspannender Teilgraph eines $k$-Baumes heißt *partieller $k$-Baum*. Die partiellen 2-Bäume werden auch *serien-parallele Graphen* (SP-Graphen) genannt.

## 2.3 Algorithmische Probleme

### 2.3.1 Problemklassen

Details zu Optimierungsproblemen und zur Bestimmung der Laufzeit von Algorithmen werden wie in [24] verwendet, hier aber nicht definiert.

Die zwei für das stochastische Steinerbaumproblem relevanten Komplexitätsklassen sind die Folgenden:

**Definition 2.3.1** (**P**)**.** *Ein algorithmisches Problem gehört zur Komplexitätsklasse* **P***, wenn es durch einen Algorithmus mit polynomieller maximaler Rechenzeit gelöst werden kann.[24]*

**Definition 2.3.2** (**NP**)**.** *Ein Entscheidungsproblem $L$ gehört zur Komplexitätsklasse* **NP***, wenn es einen nichtdeterministischen Algorithmus mit polynomieller maximaler Rechenzeit gibt, der jedes $x \in L$ auf mindestens einem erlaubten Rechenweg akzeptiert und jedes $x \notin L$ auf allen erlaubten Rechenwegen ablehnt.[24]*

Um Probleme untereinander zu vergleichen, bedarf es einer Relation auf Entscheidungsproblemen:

**Definition 2.3.3** (Polynomielle Reduktion)**.** *Das Entscheidungsproblem $A$ ist auf das Entscheidungsproblem $B$* polynomiell reduzierbar*, Notation $A \leq_p B$, wenn es eine in polynomieller Zeit berechenbare Funktion $f$ gibt, die Eingaben für $A$ auf Eingaben für $B$ so abbildet, dass für die zugehörigen Sprachen $L_A$ und $L_B$ gilt:*

$$\forall x : x \in L_A \Leftrightarrow f(x) \in L_B.$$

Auch diese Definition stammt aus [24].

Ist ein Problem **NP**-schwierig, so kann davon ausgegangen werden, dass der zeitliche Aufwand, dieses Problem zu lösen, vergleichsweise hoch ist.

**Definition 2.3.4** (**NP**-schwierig)**.** *Ein Entscheidungsproblem $A$ heißt* **NP**-schwierig*, wenn $C \leq_p A$ für jedes $C \in NP$.[24]*

Die Probleme, die für diese Diplomarbeit zentral sind, gehören einer Sonderform von Optimierungsproblemen an. Sie sind *kombinatorische Optimierungsprobleme* und als solche bestehen sie aus einer Menge von Instanzen:

**Definition 2.3.5** (Instanz eines kombinatorischen Optimierungsproblems)**.** *Eine* Instanz eines kombinatorischen Optimierungsproblems *besteht aus einer endlichen* Grundmenge $\Omega$*, einer* Gewichtungsfunktion $w : \Omega \mapsto \mathbb{R}$ *und einer Menge* zulässiger Lösungen[1] $\mathcal{S} \subseteq \Omega$*. Die* Zielfunktion *ist definiert durch* $v : \mathcal{S} \mapsto \mathbb{R}$ *mit* $v(s) := \sum_{\omega \in s} w(\omega)$*.*[2]

**Definition 2.3.6** (Kombinatorische Optimierungsprobleme)**.** *Bei einem* kombinatorischen Maximierungsproblem *ist für eine Instanz eine Lösung mit Wert* $\max_{s \in \mathcal{S}} v(s)$*, bei einem* kombinatorischen Minimierungsproblem *eine Lösung mit Wert* $\min_{s \in \mathcal{S}} v(s)$ *zu finden.*

### 2.3.2 Steinerbaumprobleme

Für diese Diplomarbeit sind einige kombinatorische Optimierungsprobleme grundlegend. Das Problem, von dem sich die anderen ableiten, ist das Steinerbaumproblem.

**Definition 2.3.7** (Steinerbaumproblem)**.** *Das* Steinerbaumproblem *(STP) ist ein kombinatorisches Minimierungsproblem. Für einen Graphen* $G = (V, E)$*, eine Kostenfunktion*[3] $c : E \mapsto \mathbb{R}^+$ *mit* $c((a, b)) = c((b, a))$ *für alle* $(a, b) \in E$ *und eine Terminalmenge* $T \subseteq V$ *ist eine Instanz* $(\Omega, w, \mathcal{S})$ *gegeben durch* $\Omega := E$ *und* $w := c$*. Eine zulässige Lösung ist die Kantenmenge eines Teilbaumes* $S = (V_S, E_S)$ *von* $G$ *mit* $T \subseteq V_S$*.*[4]

Als Repräsentation für eine Instanz des Steinerbaumproblems wird in dieser Diplomarbeit das Tupel $(G, c, T)$ verwendet. Das Problem besteht darin, einen Teilbaum von $G$ zu finden, der alle Terminale enthält und minimale Kosten besitzt. Ein solcher Baum wird *Steinerbaum* genannt. Die *Kosten eines Steinerbaumes* $S$ sind die Summe seiner Kantenkosten und werden mit $c(S)$ notiert. Zur Vereinfachung der Notation sollen allgemein für eine Kantenmenge $E$ und eine Kostenfunktion $c$ die Kosten dieser Menge mit $c(E) := \sum_{e \in E} c(e)$ definiert sein. Dann ist $c(S) := c(E(S))$.

Wie auch in den anderen Problem wird gefordert, dass für eine Halbkante $(a, b) \in E$ : $c((a, b)) = c((b, a))$ gilt. Dies stellt sicher, dass auch für eine Kante $\{a, b\} \in E$ der Ausdruck $c(\{a, b\}) := c(a, b)$ definiert ist.

**Definition 2.3.8** (Steinerwaldproblem)**.** *Beim kombinatorischen Minimierungsproblem des* Steinerwaldproblems *(SFP) ist für einen Graphen* $G = (V, E)$*, eine Kostenfunktion* $c : E \mapsto \mathbb{R}^+$ *mit* $c((a, b)) = c((b, a))$ *für alle* $(a, b) \in E$*, sowie eine Menge* $\mathcal{D}$ *von Paaren aus* $V \times V$ *eine Instanz* $(\Omega, w, \mathcal{S})$ *gegeben durch* $\Omega := E$ *und* $w := c$*. Die Menge der zulässigen Lösungen ist die Kantenmenge eines Teilgraphen* $S = (V_S, E_S)$ *von* $G$*, der ein Wald ist und bei dem für jedes Paar* $(x, y) \in \mathcal{D}$ *ein Weg zwischen* $x$ *und* $y$ *existiert.*

[1] Diese werden häufig auch nur als Lösung bezeichnet.
[2] Die Notation stammt aus [24], angelehnt an die Definition aus [18].
[3] $\mathbb{R}^+$ versteht sich hier als die Menge der echt positiven reellen Zahlen, also ohne 0.
[4] In Anlehnung an [23] sind zulässige Lösungen immer Bäume und Steinerbäume immer minimal.

Auch beim Steinerwaldproblem soll eine Instanz als Tripel $(G, c, \mathcal{D})$ beschrieben werden. Das Ziel bei der Lösung dieses Problems ist es, einen kostenminimalen Teilgraphen von $G$ zu finden, welcher die Paare aus $\mathcal{D}$ verbindet und einen Wald bildet. Aus den Paaren in $\mathcal{D}$ können durch Transitivität größere Mengen von zu verbindenden Knoten abgeleitet werden. Zum Beispiel entsteht durch die Paare $(x_1, x_2)$ und $(x_2, x_3)$ in $\mathcal{D}$ die Anforderung, dass $x_1$ mit $x_3$ verbunden werden muss. Daher ist es sinnvoll, die reflexive, symmetrische und transitive Hülle von $\mathcal{D}$ zu betrachten. Diese Relation ist eine Äquivalenzrelation und enthält als Äquivalenzklassen die Mengen der Knoten, die miteinander verbunden werden müssen. Hierauf wird in Abschnitt 3.2.3 näher eingegangen.

**Definition 2.3.9** ((Zweistufiges) stochastisches Steinerbaumproblem)**.** *Das* (zweistufige) stochastische Steinerbaumproblem *(SSTP) ist ein kombinatorisches Minimierungsproblem. Für*

- *einen Graphen* $G = (V, E)$*,*
- *eine Kostenfunktion* $c_0 : E \mapsto \mathbb{R}^+$ *mit* $c_0((a, b)) = c_0((b, a))$ *für alle* $(a, b) \in E$*,*
- *eine Anzahl* $K \in \mathbb{N}$ *von Szenarien sowie*
- *für jedes Szenario* $k \in \{1, \dots, K\}$
  - *eine Szenarienkostenfunktion* $c_k : E \mapsto \mathbb{R}^+$ *mit* $c_k((a, b)) = c_k((b, a))$ *für alle* $(a, b) \in E$*,*
  - *eine Menge von Terminalknoten* $T_k \subseteq V$ *und*
  - *eine Wahrscheinlichkeit* $p_k \in [0, 1]$ *mit* $\sum_{k=1}^{K} p_k = 1$

*ist eine Instanz* $(\Omega, w, \mathcal{S})$ *gegeben durch* $\Omega := \bigcup_{i \in \{0,\dots,k\}} \Omega_i$ *mit* $\Omega_i := E \times \{i\}$*. Für ein* $e_i := (e, i) \in \Omega$ *ist* $w(e_0) := c_0(e)$ *und für* $i \in \{1, \dots, k\}$ *ist* $w(e_i) := p_i c_i(e)$*.*

*Eine zulässige Lösung ist eine Menge* $S := \left(\bigcup_{i \in \{1,\dots,k\}} E_i\right) \cup E_0$ *mit* $E_0 \subseteq \Omega_0$ *und für jedes* $k \in \{1 \dots K\}$ *einer Menge* $E_k \subseteq \Omega_k$*, sodass für jedes* $k \in \{1 \dots K\}$ *und eine Menge* $F_k \subseteq E_0$ *der von den Kanten*[5] *in* $F_k \cup E_k$ *induzierte Teilgraph ein Baum ist und die Knoten aus* $T_k$ *enthält.*

Die Kanten in den Mengen $E_0, \dots, E_K$ werden zur Vereinfachung der Notation auch mit $\mathcal{E}(E_i)$ bezeichnet für $i \in \{0, \dots, K\}$. Ist eine Verwechslung ausgeschlossen, wird die Abbildung $\mathcal{E}$ implizit auf $E_i$ angewendet.

Aus der Definition kombinatorischer Optimierungsprobleme ergibt sich somit die Zielfunktion

$$\sum_{\omega \in S} w(\omega) = \sum_{\omega \in E_0} w(\omega) + \sum_{\omega \in E_1 \cup \dots \cup E_K} w(\omega) = \sum_{e \in \mathcal{E}(E_0)} c_0(e) + \sum_{k=1}^{K} p_k \sum_{e \in \mathcal{E}(E_k)} c_k(e).$$

[5] Hier ist jeweils die erste Komponente der Paare in $F_k \cup E_k$ gemeint.

Wie bei den Problemen zuvor wird auch hier eine Instanz nicht durch das Tripel $(\Omega, w, \mathcal{S})$ beschrieben, sondern durch das Tupel $(G, c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$.

Das Problem teilt sich, wie in der Einleitung angedeutet, in zwei Phasen auf: In der ersten Phase dürfen Kanten zu den in der Kostenfunktion $c_0$ festgelegten Konditionen „gekauft“ werden. Es ist aber noch nicht klar, welches der Szenarien später tatsächlich realisiert wird. Diese Informationen werden von den einzelnen Szenarien verkörpert: Für $1 \leq i \leq K$ setzt sich das Szenario $i$ zusammen aus

- der Wahrscheinlichkeit $p_i$, dass es eintrifft,
- der Terminalmenge $T_i$, deren Knoten in dem Fall, dass dieses Szenario realisiert wird, verbunden werden müssen und
- den Kosten $c_i$, die dann für das Wählen einer Kante entstehen, falls die Kante noch nicht in der ersten Phase gekauft wurde.

In der zweiten Phase steht fest, welches Szenario verwirklicht werden soll; allerdings sind die Kosten der Kanten möglicherweise gestiegen. Aus der Zielfunktion ergibt sich, dass die Kosten der gesuchten Lösung bezogen auf den Erwartungswert minimal sein sollen.

Wird die Eingabe auf partielle $k$-Bäume eingeschränkt, so werden diese Varianten der o. g. Probleme mit $\text{STP}^{\leq k}$, $\text{SFP}^{\leq k}$ und $\text{SSTP}^{\leq k}$ bezeichnet. Das Kernthema dieser Arbeit ist somit das Problem $\text{SSTP}^{\leq 2}$.

Diese sechs kombinatorischen Optimierungsprobleme sind auch Optimierungsprobleme im Sinne von [24]. Sie besitzen darüber hinaus jeweils eine Entscheidungsvariante. Diese enthält die Paare $(x, l)$, wobei $x$ die Instanzen kodiert und $l \in \mathbb{R}^+$ ist. Es sind alle Paare $(x, l) \in L$, wenn $x$ eine Lösung mit Kosten höchstens $l$ besitzt. Die Entscheidungsvarianten heißen $\text{STP}_{\text{dec}}$ bzw. $\text{STP}_{\text{dec}}^{\leq k}$, $\text{SFP}_{\text{dec}}$ bzw. $\text{SFP}_{\text{dec}}^{\leq k}$ und $\text{SSTP}_{\text{dec}}$ bzw. $\text{SSTP}_{\text{dec}}^{\leq k}$.

# Kapitel 3

# Vorüberlegungen

Einleitend werden einige Bemerkungen zu den Kernthemen dieser Arbeit diskutiert. Dazu geht es im ersten Abschnitt dieses Kapitels um (partielle) $k$-Bäume und im Speziellen um serien-parallele Graphen. Im zweiten Abschnitt werden Eigenschaften des SSTP betrachtet.

## 3.1 Über partielle $k$-Bäume

Im Mittelpunkt dieser Arbeit stehen serien-parallele Graphen. Die Hoffnung ist, dass ein Polynomialzeitalgorithmus für das SSTP existiert, wenn die zulässigen Lösungen auf diese Graphen eingeschränkt werden. Die Baumweite serien-paralleler Graphen beträgt höchstens 2. Analog gilt für partielle $k$-Bäume:

**Proposition 3.1.1.** *Die partiellen $k$-Bäume sind genau die Graphen mit Baumweite höchstens $k$.*

Der Beweis kann in [16] nachgelesen werden. Mit dieser Aussage ist es im späteren Verlauf der Arbeit möglich zu beweisen, dass das Finden einer Lösung für das SSTP auf Graphen mit Baumweite höchstens 3 **NP**-schwierig ist.

### 3.1.1 Serien-parallele Graphen

Wie viele Graphenklassen lassen sich auch die SP-Graphen durch eine Menge von Graphen charakterisieren, die nicht homöomorph zu einem Teilgraphen eines SP-Graphen sein dürfen.

**Proposition 3.1.2.** *Serien-parallele Graphen sind genau die $K_4$-freien Graphen.*

Der dazugehörige Beweis ist in [23] zu finden. Um einen Zusammenhang mit bekannteren Graphenklassen herzustellen, soll gezeigt werden, dass SP-Graphen eine Teilmenge der planaren Graphen sind:

**Proposition 3.1.3.** *Serien-parallele Graphen sind planar.*

Beweis. Angenommen es gäbe einen serien-parallelen, nicht planaren Graphen $G$. Somit enthielte dieser nach dem Satz von Kuratowski [14] einen zu $K_5$ oder $K_{3,3}$ homöomorphen Teilgraphen. Da beide Graphen aber $K_4$ als homöomorphen Teilgraphen besitzen (siehe Abb. 3.1), kann $G$ kein serien-paralleler Graph sein. □

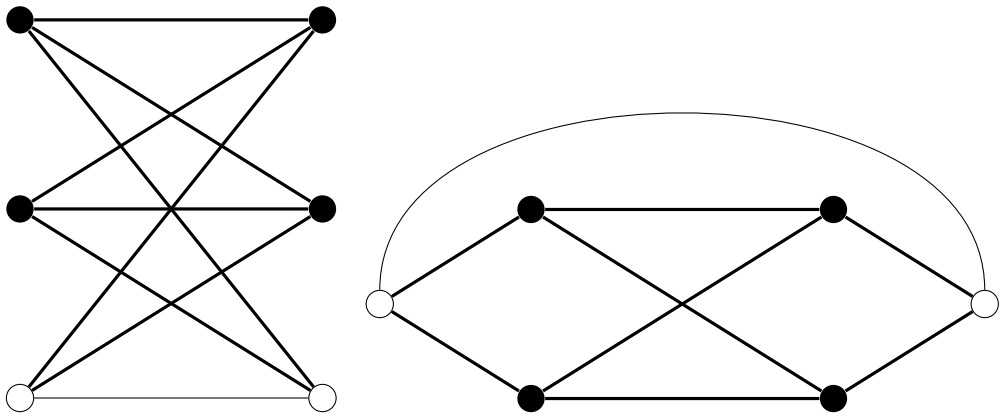


**Abbildung 3.1:** $K_{3,3}$ (in beiden Abbildungen dargestellt) enthält einen zu $K_4$ homöomorphen Teilgraphen.

Jeder Knoten eines 2-Baumes mit mindestens drei Knoten besitzt, wie aus der Definition ersichtlich, mindestens Grad 2. Für den Algorithmus von Wald und Colbourn ist eine wichtig Voraussetzung, dass es auch immer Knoten mit Grad genau 2 gibt. Diese Eigenschaft wird durch das folgende Lemma sichergestellt:

**Lemma 3.1.4.** *Sei $G$ ein 2-Baum mit $|V(G)| \geq 4$ und $G^*$ ein 2-Baum, der entsteht, wenn zu $G$ ein Knoten mit den in Definition 2.2.6 beschriebenen Kanten hinzugefügt wird. Sei $V_2(G) := \{v \in V(G) \mid d(v) = 2\}$, dann ist $|V_2(G)| \leq |V_2(G^*)|$.*

Beweis. Sei $G$ ein 2-Baum und $|V(G)| \geq 4$. Zuerst soll gezeigt werden, dass je zwei Knoten mit Grad 2 nicht benachbart sind. Hat der Graph weniger als zwei Knoten mit Grad 2, folgt dies sofort. Seien $u \in V_2(G)$ und $v \in V_2(G)$. Angenommen die Knoten $u$ und $v$ wären benachbart. Sei $x$ der zweite Nachbar von $u$ und $y$ der zweite Nachbar von $v$. Es gilt $x \neq y$, sonst ist $G = K_3$. Da $x$ und $v$ die einzigen Nachbarn von $u$ wären, müsste eine Kante $\{x, v\}$ existieren; sonst wäre $G$ kein 2-Baum. Dies ist ein Widerspruch dazu, dass $d(v) = 2$ ist. Diese Situation ist in Abb. 3.2a dargestellt.

Es sind also keine zwei Knoten mit Grad 2 benachbart. Sei $G^*$ ein Graph, der durch das Hinzufügen eines Knotens $x$ nach der induktiven Definition von 2-Bäumen aus $G$ entsteht. Die Kante der 2-Clique, die für das Einfügen gewählt wird, sei $\{u, v\}$. Ist $d(u) > 2$ und $d(v) > 2$, so ist $V_2(G^*) = V_2(G) \cup \{x\}$ und $|V_2(G^*)| = |V_2(G)| + 1$. Die Knoten $u$ und $v$ können nach den anfänglichen Überlegungen nicht beide Grad 2 besitzen. Es bleibt also o. B. d. A. der Fall zu klären, dass $d(u) = 2$ und $d(v) > 2$ ist. In $G^*$ gilt $d(u) > 2$; $d(v)$ bleibt größer als 2 und $d(x) = 2$. Somit ist $V_2(G^*) \backslash \{x\} = V_2(G) \backslash \{u\}$ und $|V_2(G^*)| = |V_2(G)|$. Eine Übersicht zu dieser Argumentation ist in Abb. 3.2b zu finden. □

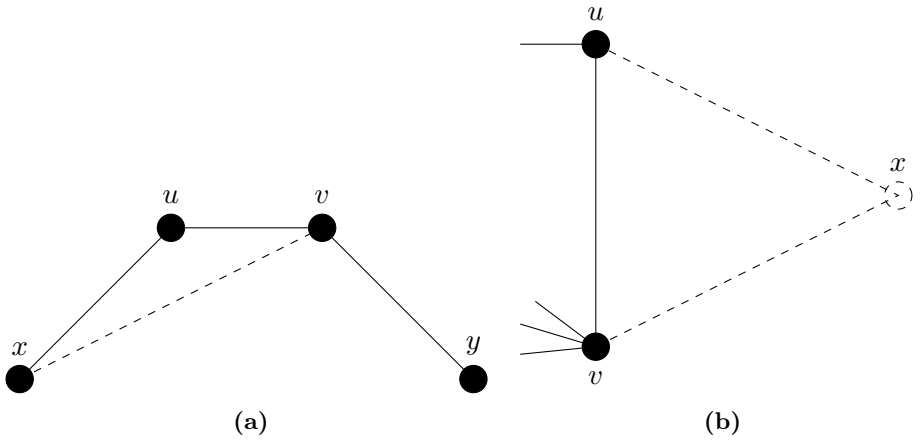



**Abbildung 3.2:** (a) Die Knoten mit Grad 2 sind nicht benachbart. (b) Die Situation für den neuen Knoten $x$

Der $K_3$ hat drei Knoten mit Grad 2; der 2-Baum mit vier Knoten hat zwei Knoten mit Grad 2. Daraus ergibt sich folgende Aussage, die für die Korrektheit des Algorithmus von Wald und Colbourn wichtig ist:

**Korollar 3.1.5.** *Jeder 2-Baum $(V, E)$ mit $|V| \geq 3$ besitzt mindestens zwei Knoten mit Grad genau 2.*

Demnach gibt es immer Knoten mit Grad echt kleiner als 3. Daraus resultiert:

**Korollar 3.1.6.** *Serien-parallele Graphen sind nicht 3-zusammenhängend.*

Die Analyse der Laufzeit des Algorithmus von Wald und Colbourn zeigt, dass die Initialisierung in Rechenzeit $\mathcal{O}(m)$ für einen Graphen mit $m$ Kanten ausgeführt werden kann. Um eine Rechenzeit von $\mathcal{O}(n)$ zu beweisen, ist es nötig, die Anzahl der Kanten eines serien-parallelen Graphen zu beschränken. Dafür kann festgestellt werden:

**Lemma 3.1.7.** *Ein 2-Baum mit $n$ Knoten hat genau $2n - 3$ Kanten.*

Beweis. Die Konstruktion eines 2-Baumes startet mit dem $K_2$. Dieser hat eine Kante und zwei Knoten. Für die restlichen $n - 2$ Knoten werden je zwei Kanten hinzugefügt. So folgt für die Anzahl Kanten: $m = 1 + 2(n - 2) = 1 + 2n - 4 = 2n - 3$ □

Da ein serien-paralleler Graph mit $n$ Knoten höchstens so viele Kanten besitzt wie ein 2-Baum mit $n$ Knoten, stellt dies eine obere Schranke für die Anzahl an Kanten eines serien-paralleler Graphen dar.

**Korollar 3.1.8.** *Ist $m$ die Anzahl der Kanten und $n$ die Anzahl der Knoten eines serien-parallelen Graphen, so gilt $m \leq 2n - 3$.*

## 3.2 Über das SSTP

Um sich dem stochastischen Steinerbaumproblem zu nähern, werden in diesem Abschnitt Aussagen zu dessen Lösbarkeit getroffen. In den ersten beiden Abschnitten geht es um obere Schranken der Komplexität: Im ersten Abschnitt wird ein Algorithmus für das SSTP vorgestellt, der in Zeit $\mathcal{O}(kn)$ eine Lösung findet, unter der Voraussetzung, dass der Eingabegraph ein Baum mit $n$ Knoten ist und $k$ Szenarien existieren. Im zweiten Abschnitt wird ein Algorithmus für das SSTP auf serien-parallelen Graphen dargestellt, der allerdings Rechenzeit $\mathcal{O}(4^n kn)$ benötigt. Dagegen wird im letzten Abschnitt eine untere Schranke der Komplexität des SSTPs im Kontext der Baumweite untersucht. Um einige Spezialfälle zu vermeiden, werden hier und im Verlauf dieser Arbeit nur zusammenhängende Graphen betrachtet.

### 3.2.1 Einschränkung der Eingabe auf Bäume

Es wird zunächst ein Algorithmus beschrieben, der das SSTP sehr effizient lösen kann. Für diesen Algorithmus muss aber die Menge der Eingaben so eingeschränkt werden, dass nur Graphen mit Baumweite 1 erlaubt sind.

**Algorithmus 3.1** Algorithmus für das SSTP auf Bäumen

**Eingabe:** Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ für das SSTP, $G$ ist ein Baum.

**Ausgabe:** eine Lösung für das SSTP mit minimalen Kosten

1: $F \leftarrow \emptyset$
2: **for** $i : 1 \leq i \leq K$ **do**
3: $\quad B_i \leftarrow$ BerechneSB$(G, (c_i \cdot p_i), T_i)$
4: $\quad G_i \leftarrow \emptyset$
5: **for all** $e \in \bigcup_{i \in \{1,\ldots,K\}} B_i$ **do**
6: $\quad S_e \leftarrow \emptyset$
7: $\quad$ **for** $i : 1 \leq i \leq K$ **do**
8: $\quad\quad$ **if** $e \in B_i$ **then**
9: $\quad\quad\quad S_e \leftarrow S_e \cup \{i\}$
10: $\quad$ **if** $\sum_{i \in S_e} p_i c_i(e) > c_0(e)$ **then**
11: $\quad\quad F \leftarrow F \cup \{e\}$
12: $\quad$ **else**
13: $\quad\quad$ **for all** $i \in S_e$ **do**
14: $\quad\quad\quad G_i \leftarrow G_i \cup \{e\}$
15: **return** $(F, G_1, \ldots, G_K)$

Der Algorithmus 3.1 berechnet in Zeilen 2 und 3 für jedes Szenario einen Steinerbaum mit Hilfe einer Funktion BerechneSB. Diese Funktion kann durch einen Algorithmus realisiert werden, der auf Bäumen einen Steinerbaum findet. Das kann beispielsweise durch den Algorithmus von Wald und Colbourn geschehen, der in Kapitel 4 vorgestellt wird. Für jede Kante $e$, die sich in mindestens einem dieser Bäume befindet, wird in den Zeilen 7 – 9 die Menge der Szenarien $S_e$ bestimmt, für die die Kante in einem optimalen Steinerbaum liegt. Dann werden für jede Kante die erwarteten Kosten für die zweite Phase berechnet. So kann in den Zeilen 10 – 14 entschieden werden, ob die Kante schon in der ersten Phase gewählt werden soll, oder ob es günstiger ist, die Kante in der zweiten Phase zu wählen.

**Proposition 3.2.1.** *Der Algorithmus 3.1 berechnet eine optimale Lösung für das SSTP$^{\leq 1}$ mit $k$ Szenarien auf Bäumen mit $n$ Knoten in Laufzeit $\mathcal{O}(kn)$.*

Beweis. Es wird zuerst die Korrektheit und danach die Laufzeit bewiesen. Für jedes Szenario gibt es genau einen Steinerbaum. Neben diesen Steinerbäumen können keine alternativen Wege genommen werden, weil es sich bei der Eingabe um einen Baum handelt. Dementsprechend stehen die Kosten für die zweite Phase fest, da für jede Kante bekannt ist, welcher Baum in welchem Szenario eine Kante verwenden muss. In welcher Phase eine Kante gewählt werden muss, kann also in Abhängigkeit davon entschieden werden, ob die Kosten der ersten Phase oder die erwarteten Kosten der zweiten Phase niedriger sind.

Die Berechnung von Steinerbäumen auf serien-parallelen Graphen – und somit auch auf Bäumen – ist, wie in Kapitel 4 gezeigt wird, in linearer Zeit in der Größe der Knotenmenge möglich. Das Berechnen der Steinerbäume muss für jedes Szenario einmal wiederholt werden. Für die Anzahl der Szenarien $k$ und die Anzahl der Knoten $n$ ergibt sich eine Laufzeit von $\mathcal{O}(kn)$. Das Bestimmen der Szenarienmenge für jede Kante ist in Zeit $\mathcal{O}(kn)$ möglich. Alle Kosten der zweiten Phase können ebenfalls in Laufzeit $\mathcal{O}(kn)$ berechnet werden. Insgesamt folgt so eine Laufzeit von $\mathcal{O}(kn)$. □

### 3.2.2 Erster Versuch auf serien-parallelen Graphen

Ein einfacher Algorithmus für das SSTP auf serien-parallelen Graphen kann daraus bestehen, die Kanten für die erste Phase auszuprobieren; d. h. für jede Kombination von Kanten für die erste Phase, Steinerbäume für jedes Szenario für die zweite Phase zu finden. Algorithmus 3.2 iteriert dafür über alle Teilmengen der Kantenmenge in Zeile 4. In der Funktion BerechneAlleSB in Zeile 6 werden alle Szenariensteinerbäume für eine solche Kantenmenge berechnet. Die Funktion STPAlgorithmus in Zeile 8 der Funktion BerechneAlleSB kann durch den Algorithmus von Wald und Colbourn realisiert werden. Die Kosten der Kanten, die für die erste Phase ausprobiert werden, werden dafür auf 0 gesetzt, damit der Algorithmus beim Berechnen der Steinerbäume diese Kanten ohne zusätzliche Kosten nutzen kann. Während des Suchens wird die beste Lösung $F$ und $S$,

und deren Wert min stets gespeichert. So kann nach jeder Iteration geprüft werden, ob eine bessere Lösung gefunden wurde.

**Algorithmus 3.2** Ein einfacher Algorithmus für das SSTP auf serien-parallelen Graphen

**Eingabe:** Instanz $I = (G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ für das SSTP, G ist ein serien-paralleler Graph.

**Ausgabe:** eine Lösung für das SSTP mit minimalen Kosten

```
1:  min ← ∞
2:  F ← ∅
3:  S ← ∅
4:  for all M ⊆ E do
5:      c ← c_0(M)
6:      B ← BerechneAlleSB(I, M)
7:      for k : 1 ≤ k ≤ K do
8:          for all e ∈ B_k do
9:              if e ∉ M then
10:                 c ← c + c_k(e) · p_i
11:     if c < min then
12:         min ← c
13:         F ← M
14:         S ← B
15: return (F, S)
```

```
1:  Funktion BerechneAlleSB(I, M)
2:      S ← ()
3:      for i : 1 ≤ i ≤ K do
4:          for all e ∈ E\M do
5:              c(e) ← c_i(e) · p_i
6:          for all e ∈ M do
7:              c(e) ← 0
8:          S ← (S_1, ..., S_|S|, STPAlgorithmus(G, c, T_i))
9:      return S
```

**Proposition 3.2.2.** *Der Algorithmus 3.2 berechnet eine optimale Lösung für das SSTP$^{\leq 2}$ mit k Szenarien auf serien-parallelen Graphen mit n Knoten in Laufzeit $\mathcal{O}(4^n kn)$*

Beweis. Im Laufe des Algorithmus werden in Zeile 4 alle Möglichkeiten für die *First-Stage*-Lösung $E_0$ ausprobiert. Die von dem Algorithmus von Wald und Colbourn gefunde-

nen Bäume sind Steinerbäume. Eine Lösung minimalen Wertes ist somit Teil der durchsuchten Menge und wird gefunden.

Ein wichtiges Detail ist die Zuweisung des Wertes 0 für die Kosten einer Kante in Zeile 7 der Funktion BerechneAlleSB. Die Definition des STP lässt nur echt positive Kantenkosten zu. Der Algorithmus von Wald und Colbourn berechnet aber auch auf solchen Instanzen eine kostenminimale Lösung, die dann allerdings Kanten enthalten kann, die nicht nötig sind. Diese haben aber Kosten 0 und können ignoriert werden. Das Herausfiltern der Kosten aller *First-Stage*-Kanten geschieht in Zeile 9 des Algorithmus.

Das Finden der Steinerbäume, also die Funktion BerechneAlleSB, benötigt Zeit $\mathcal{O}(kn)$, da der Algorithmus von Wald und Colbourn Rechenzeit $\mathcal{O}(n)$ benötigt (vgl. Kapitel 4). Die Menge der möglichen Kantenmengen, die in Zeile 4 des Algorithmus vollständig durchsucht wird, ist die Menge aller Teilmengen von $E$. Diese hat Kardinalität

$$|\mathcal{P}(E)| = 2^{|E|} \leq 2^{2n-3} \leq 4^n.$$

Die Zeilen 7–10 werden für eine Menge $M$ nur $\mathcal{O}(kn)$ mal durchlaufen. □

Bei der Laufzeit ist zu beachten, dass für einen zusammenhängenden Graphen schon $|E| \geq |V| - 1$ gilt und in diesem Fall $|\mathcal{P}(E)| \geq 2^{|V|-1} = \frac{2^{|V|}}{2} = \Omega(2^{|V|})$ ist. Dadurch sind exponentielle Laufzeiten des Algorithmus für die hier betrachteten Graphen immer notwendig.

### 3.2.3 Komplexitätstheoretische Einordnung im Baumweitenkontext

In diesem Abschnitt werden die Grenzen der effizienten Lösbarkeit im Bezug auf die Baumweite ausgelotet. Dass das SSTP auf Bäumen, also Graphen, deren Baumweite höchstens 1 ist, in polynomieller Zeit lösbar ist, wurde bereits in Abschnitt 3.2.1 gezeigt. Geht man die Baumweitenhierarchie nur ein wenig hinauf, verändert sich der Ausblick auf die Lösbarkeit des SSTP. In diesem Abschnitt wird die zentrale Aussage getroffen, dass sich das SSTP auf partiellen 3-Bäumen – also auf Graphen mit einer Baumweite von höchstens 3 – nicht in polynomieller Zeit lösen lässt, wenn $\mathbf{P} \neq \mathbf{NP}$ ist. Dies wird durch eine Reduktion des Steinerwaldproblems auf das stochastische Steinerbaumproblem bewiesen, die im Folgenden beschrieben wird.

Gegeben sei eine definitionsgemäße Eingabe für $\text{SFP}_{\text{dec}}$. Um eine $\text{SSTP}_{\text{dec}}$-Instanz zu konstruieren, wird der Graph übernommen und die Kostenfunktion der $\text{SFP}_{\text{dec}}$-Instanz als $c_0$ verwendet. Die Menge $\mathcal{D}$ wird als Relation aufgefasst und ihre reflexive, symmetrische und transitive Hülle mit $\widehat{\mathcal{D}}$ bezeichnet. Die Anzahl $K$ der Szenarien ergibt sich aus der Anzahl der Äquivalenzklassen von $\widehat{\mathcal{D}}$. Für jede dieser Äquivalenzklassen wird eine Terminalmenge $T_k$ mit den Knoten der Klasse zu der $\text{SSTP}_{\text{dec}}$-Instanz hinzugefügt. Für jedes $k \in \{1, \ldots, K\}$ und $e \in E$ sei $c_k(e) := \sum_{e \in E} K c_0(e) + 1$. Die Wahrscheinlichkeit jedes Szenarios wird auf $\frac{1}{K}$ gesetzt. Für die Wahrscheinlichkeiten gilt dann $\sum_{k=1}^{K} p_k = K \frac{1}{K} = 1$. Die

Kostenschranke der $\text{SFP}_{\text{dec}}$-Instanz wird ohne Änderung übernommen. Diese Reduktion wird im weiteren Verlauf mit $f$ bezeichnet.

**Lemma 3.2.3.** *Die Paare aus $\mathcal{D}$ einer $SFP_{dec}$-Instanz $(x, l)$ sind genau dann durch einen Teilgraphen $G^*$ von $G$ verbunden, wenn die Terminale jedes Szenarios von $f((x, l))$ durch $G^*$ verbunden sind.*

Beweis. „⇒“: Seien zunächst alle Paare aus $\mathcal{D}$ durch $G^*$ verbunden. Nach Konstruktion der Szenarien ist $(x_1, x_2) \in \widehat{\mathcal{D}}$ für je zwei verschiedene Knoten eines Szenarios $x_1$ und $x_2$. Gilt $(x_1, x_2) \in \mathcal{D}$, so gibt es einen Weg zwischen $x_1$ und $x_2$ in $G^*$, da alle Paare aus $\mathcal{D}$ verbunden sind. Ansonsten ist das Paar $(x_1, x_2)$ aus Gründen der Symmetrie oder der Transitivität zu $\widehat{\mathcal{D}}$ hinzugekommen. Im ersten Fall ist also $(x_2, x_1) \in \mathcal{D}$ und es gibt einen Weg, der beide Knoten verbindet. Im zweiten Fall ist das Paar wegen der Transitivität in $\widehat{\mathcal{D}}$ enthalten und es existiert eine Folge von Knoten $(v_0, \ldots, v_k)$ mit $(x_1, v_0), (v_k, x_2), (v_i, v_{i+1}) \in \mathcal{D}$ für $0 \leq i \leq k-1$. Weil alle Paare aus $\mathcal{D}$ verbunden sind, sind auch alle $v_i$ mit $v_{i+1}$ verbunden, $x_1$ mit $v_0$, $v_k$ mit $x_2$ und somit $x_1$ mit $x_2$. Der Teilgraph $G^*$ verbindet demnach die Terminale jedes Szenarios der $\text{SSTP}_{\text{dec}}$-Instanz.

„⇐“: Seien nun die Terminale jedes Szenarios durch den Graphen $G^*$ verbunden. Sei $(x_1, x_2)$ ein Paar aus $\mathcal{D}$. Die Knoten $x_1$ und $x_2$ sind dann in der gleichen Äquivalenzklasse von $\widehat{\mathcal{D}}$. Nach der obigen Konstruktion sind $x_1$ und $x_2$ in der Terminalmenge des gleichen Szenarios enthalten und somit in $G^*$ verbunden. □

Die Funktion $f$ mit der oben genannten Eigenschaft kann jetzt verwendet werden, um die folgende Proposition zu beweisen:

**Proposition 3.2.4.** *$SFP_{dec} \leq_p SSTP_{dec}$*

Beweis. Die Reduktion $f$ ist in polynomieller Zeit berechenbar. Es muss also noch für jede Eingabe $(x, l)$ gezeigt werden, dass $(x, l) \in \text{SFP}_{\text{dec}} \Leftrightarrow f((x, l)) \in \text{SSTP}_{\text{dec}}$ gilt.

„⇒“: Für jede Eingabe $(x, l) \in \text{SFP}_{\text{dec}}$ gibt es einen Teilgraphen $S = (V_S, E_S)$ des Eingabegraphen, in dem alle Paare aus $\mathcal{D}$ verbunden sind und die Summe der Kantenkosten höchstens $l$ ist. Somit existiert eine Lösung für die wie oben konstruierte $\text{SSTP}_{\text{dec}}$-Instanz durch folgende Überlegung: Die Kanten aus $E_S$ werden für $E_0$ gewählt und die $E_1, \ldots, E_K$ bleiben leer. Da $E_S$ alle Paare miteinander verbindet, sind mit Lemma 3.2.3 auch die Terminalmengen der Szenarien durch die Menge $E_S$ verbunden und für den Wert der Lösung ergibt sich:

$$\sum_{e \in \mathcal{E}(E_0)} c_0(e) + \sum_{k=1}^{K} p_k \sum_{e \in \mathcal{E}(E_k)} c_k(e) = \sum_{e \in \mathcal{E}(E_0)} c_0(e) = \sum_{e \in E_S} c(e) \leq l$$

„⇐“: Sei für $(x, l)$ die konstruierte Instanz $f((x, l)) \in \text{SSTP}_{\text{dec}}$. Es gibt also Mengen $E_0, \ldots, E_K$ von Kanten, die eine Lösung für das $\text{SSTP}_{\text{dec}}$ bilden, die zulässig ist und deren Kostensumme die Kostenschranke von $l$ nicht überschreitet.

Gibt es $i \in \{1, \ldots, K\}$, sodass $E_i \neq \emptyset$ ist, so ist es möglich eine Lösung kleineren Wertes zu finden, in der alle $e \in E_i$ mit $1 \leq i \leq K$ in der ersten Phase gewählt werden. Solche Kanten erzeugen dann geringere Kosten, da $p_i c_i(e) = \frac{1}{K}\left(\sum_{e \in E} K c_0(e) + 1\right) = \sum_{e \in E} c_0(e) + \frac{1}{K} > c_0(e)$ ist. Die Terminale der Szenarien bleiben aber weiterhin verbunden. Die Mengen $E_1, \ldots, E_K$ können demnach als leer angenommen werden.

Die Kanten aus $E_0$ verbinden die Terminalknoten der einzelnen Szenarien und mit Lemma 3.2.3 auch die Paare in $\mathcal{D}$. Allerdings ist $G[E_0]$ nicht notwendigerweise kreisfrei. Existiert aber ein Kreis, so kann eine Kante aus dem Kreis entfernt werden. Die neue Lösung hat geringere Kosten als die vorherige und verbindet die Paare in $\mathcal{D}$ auf die gleiche Weise. Der Graph $G[E_0]$ kann also als kreisfrei angenommen werden.

Aus dieser Lösung für das $\mathrm{SSTP}_{\mathrm{dec}}$ kann nun eine Lösung $S = (V_S, E_S)$ für das $\mathrm{SFP}_{\mathrm{dec}}$ als induzierter Teilgraph der Kantenmenge $E_0$ konstruiert werden. Für die Kostenschranke $l$ gilt dann:

$$\begin{aligned} l &\geq \sum_{e \in \mathcal{E}(E_0)} c_0(e) + \sum_{k=1}^{K} p_k \sum_{e \in \mathcal{E}(E_k)} c_k(e) \\ &= \sum_{e \in \mathcal{E}(E_0)} c_0(e) = \sum_{e \in E_S} c(e) \end{aligned}$$

□

**Proposition 3.2.5.** *Das $SSTP^{\leq 3}_{dec}$ ist* **NP***-schwierig.*

Beweis. Nach [1] ist das $\mathrm{SFP}^{\leq 3}_{\mathrm{dec}}$ **NP**-schwierig. Durch die in Proposition 3.2.4 gezeigte Reduktion gilt dies auch für das $\mathrm{SSTP}^{\leq 3}_{\mathrm{dec}}$. □

Für eine Instanz für das $\mathrm{SSTP}_{\mathrm{dec}}$ kann in polynomieller Zeit verifiziert werden, ob eine gegebene Lösung Kosten kleiner als $l$ besitzt; demnach ist $\mathrm{SSTP}_{\mathrm{dec}} \in \mathbf{NP}$. Auf partiellen 3-Bäumen ist dies ebenso möglich, woraus sich ergibt:

**Korollar 3.2.6.** *Das $SSTP^{\leq 3}_{dec}$ ist* **NP***-vollständig.*

Unter Zuhilfenahme eines Algorithmus für die Optimierungsvariante des SSTP ist eine $\mathrm{SSTP}_{\mathrm{dec}}$-Instanz leicht durch Berechnung des Optimums und Vergleich des Wertes mit $l$ zu lösen. Deshalb würde ein Polynomialzeitalgorithmus für das SSTP eingeschränkt auf partielle 3-Bäume $\mathbf{P} = \mathbf{NP}$ implizieren. Demnach gilt:

**Theorem 3.2.7.** *Wenn $\mathbf{P} \neq \mathbf{NP}$ ist, gibt es keinen deterministischen Algorithmus, der das $SSTP^{\leq 3}$ in polynomieller Rechenzeit löst.*

Mit dem obigen Theorem ist auch die Betrachtung von Baumweiten größer als 3 für deterministische Polynomialzeitalgorithmen uninteressant geworden. Zu untersuchen bleiben noch die Graphen mit Baumweite höchstens 2 – also die serien-parallelen Graphen. Um einen Algorithmus für diese Graphenklasse soll es in den folgenden Kapiteln gehen.

# Kapitel 4

# Algorithmus von Wald und Colbourn für das STP

In ihrer Ausarbeitung „Steiner Trees, Partial 2-Trees, and Minimum IFI Networks“[23] beschäftigen sich J. A. Wald und C. J. Colbourn mit dem STP auf serien-parallelen Graphen.[1] Ein zentraler Aspekt ihrer Arbeit ist die graphentheoretische Herangehensweise. Dabei wird unter anderem herausgestellt, dass es sich bei minimalen IFI-Netzwerken um 2-Bäume handelt.

Vor diesem Hintergrund stellen die Autoren einen Algorithmus vor, der das Steinerbaumproblem auf 2-Bäumen mit $n$ Knoten in Laufzeit $\mathcal{O}(n)$ löst. Um das Problem auf serien-parallelen Graphen zu behandeln, wird der Graph zunächst zu einem 2-Baum ergänzt.

## 4.1 Erweiterung eines SP-Graphen zu einem 2-Baum

Diese Erweiterung eines SP-Graphen zu einem 2-Baum geschieht nur durch das Hinzufügen von Kanten. Der Algorithmus, der für einen Eingabegraphen $(V, E)$ eine Kantenmenge $M$ berechnet, sodass $(V, E \cup M)$ ein 2-Baum ist, kann in zwei Schritte unterteilt werden:

Im ersten Schritt (vgl. Algorithmus 4.1) wird der zusammenhängende Eingabegraph zu einem 2-zusammenhängenden Graphen erweitert. Durch eine modifizierte Tiefensuche wird zuerst der DFS-Baum[2], die Menge der Artikulationen des Graphen und für jede Artikulation die Menge der angrenzenden Blöcke bestimmt. Der Algorithmus wählt daraufhin für jede Artikulation $v$ unter ihren Nachfolgern im DFS-Baum einen Knoten als Repräsentant $r_B$ für jeden Block $B$ aus. Zum Schluss wird je eine Kante zwischen dem Vorgänger $p$ von $v$ im DFS-Baum und allen Repräsentanten außer $r_B$ mit $p \in B$ zum Graphen und

[1] Dort werden SP-Graphen durchgehend als partielle 2-Bäume bezeichnet.

[2] Mit DFS-Baum ist der Baum bestehend aus den Knoten des Ursprungsgraphen und den T-Kanten der Tiefensuche gemeint.

zu $M$ hinzugefügt. Hat $v$ keinen Vorgänger im DFS-Baum, so kann ein beliebig gewählter Repräsentant diese Rolle übernehmen.

**Algorithmus 4.1** Erster Schritt des Algorithmus zur Erzeugung eines 2-Baumes aus einem SP-Graphen

**Eingabe:** ein zusammenhängender, serien-paralleler Graph $G = (V, E)$

**Ausgabe:** eine Menge $M \subseteq (V \times V) \backslash E$, sodass $(V, E \cup M)$ serien-parallel und 2-zusammenhängend ist

```
1:  M ← ∅
2:  T ← DFS(G)
3:  Berechne Artikulationenmenge A und für jedes v ∈ A die Menge der Blöcke B_v, die v
    enthalten
4:  for all v ∈ A do
5:      p ← Nil
6:      for all B ∈ B_v do
7:          Wähle Repräsentant r_B ∈ B aus Nachfolgern von v in T
8:          if p = Nil then
9:              if v ist Wurzel in T then
10:                 p ← r_B
11:             else
12:                 p ← Vorfahre von v in T
13:         if p ∉ B then
14:             Füge Kante {p, r_B} zu M und zu E hinzu
```

Im zweiten Schritt (vgl. Algorithmus 4.2) wird $M$ um diejenigen Kanten erweitert, durch deren Hinzufügen aus einem 2-zusammenhängenden, serien-parallelen Graphen ein 2-Baum konstruiert werden kann. Dazu werden zuerst alle Knoten mit Grad 2 in eine Queue eingefügt. Solange der Graph mehr als drei Knoten enthält, wird in jedem Schritt ein Knoten aus der Queue entfernt. Der Algorithmus prüft für diesen Knoten $v$, ob es zwischen dessen Nachbarn $u$ und $w$ eine Kante $\{u, w\} \in E$ gibt. Existiert sie nicht, wird die Kante zu dem Graphen und zu $M$ hinzugefügt. Daraufhin entfernt der Algorithmus den Knoten $v$ mit den zu ihm inzidenten Kanten aus dem Graphen und prüft, ob die Knoten $u$ und $w$ nun Grad 2 besitzen und in die Queue eingefügt werden können. Ist $|V| = 3$, so wird die Menge $M$ ausgegeben.

**Proposition 4.1.1.** *Der o. g. Algorithmus berechnet für einen SP-Graphen $(V, E)$ eine Menge $M$, sodass $(V, E \cup M)$ ein 2-Baum ist. Die Laufzeit ist linear in der Anzahl der Knoten.*

Beweis. Im ersten Schritt werden die Kanten so hinzugefügt, dass kein Teilgraph homöomorph zu $K_4$ entstehen kann. Der Graph, der durch das Hinzufügen der Kanten aus

**Algorithmus 4.2** Zweiter Schritt des Algorithmus zur Erzeugung eines 2-Baumes aus einem SP-Graphen

**Eingabe:** ein 2-zusammenhängender, serien-paralleler Graph $G = (V, E)$

**Ausgabe:** eine Menge $M \subseteq (V \times V) \backslash E$, sodass $(V, E \cup M)$ ein 2-Baum ist

```
1:  Q.enqueue(alle Knoten mit Grad 2)
2:  while |V| > 3 do
3:      v ← Q.dequeue()
4:      Ermittle Nachbarn u und w von v
5:      if {u, w} ∉ E then
6:          Füge Kante {u, w} zu M und zu E hinzu
7:      Entferne Knoten v und Kanten {v, w} sowie {v, u} aus G
8:      if d(u) = 2 ∧ u ∉ Q then
9:          Q.enqueue(u)
10:     if d(w) = 2 ∧ w ∉ Q then
11:         Q.enqueue(w)
12: return M
```

diesem Schritt resultiert, ist also genau dann ein SP-Graph, wenn der Ausgangsgraph ein SP-Graph war. Für jede Artikulation $v$ sind die angrenzenden Blöcke so verbunden worden, dass nach einem Entfernen von $v$ der Zusammenhang bestehen bliebe. Der Knoten $v$ ist nach dem ersten Schritt demnach keine Artikulation mehr. Da nur Kanten hinzugefügt werden, entstehen keine neuen Artikulationen. Der Graph besitzt nach Schritt Eins also keine Artikulationen mehr und ist somit 2-zusammenhängend.

Im zweiten Schritt wird der 2-zusammenhängende SP-Graph reduziert, indem die Definition von 2-Bäumen in umgekehrter Richtung befolgt wird: Statt Knoten hinzuzufügen, werden Knoten entfernt. Der Algorithmus fügt dabei an genau den Stellen Kanten ein, an denen nach der Definition vorher eine Kante hätte existieren müssen, um den aktuellen Knoten einzufügen. Der Algorithmus reduziert den Graphen bis zum $K_3$, der ein 2-Baum ist. Somit werden die noch fehlenden Kanten gefunden.

Zur Korrektheit bleibt zu zeigen, dass es für die Anwendung des zweiten Schrittes immer Knoten mit Grad 2 gibt. Dazu muss es nach dem ersten Schritt und nach jedem Entfernen eines Knotens im zweiten Schritt wieder Knoten mit Grad 2 geben. Für den ersten Teil soll gezeigt werden, dass jeder 2-zusammenhängende SP-Graph, der mindestens drei Knoten enthält, auch Knoten mit Grad 2 besitzt. Sei $G^*$ ein 2-zusammenhängender, serien-paralleler Graph mit $|V(G^*)| > 2$. Aufgrund des 2-Zusammenhangs besitzt jeder Knoten mindestens Grad 2. Angenommen $G^*$ hätte keinen Knoten mit Grad 2. Also besäße jeder Knoten mindestens Grad 3. Nun hat aber jeder 2-Baum nach Korollar 3.1.5

Knoten mit Grad 2 und $G^*$ ist ein aufspannender Teilgraph eines 2-Baumes. Das ist ein Widerspruch und $G^*$ muss Knoten mit Grad 2 besitzen.

Es ist noch unklar, ob nach einem Entfernen eines Knoten $v$ in Zeile 7 des zweiten Schrittes weiterhin Knoten mit Grad 2 existieren. Hierzu soll die Aussage bewiesen werden, dass der 2-Zusammenhang bestehen bleibt. Sei $G^*$ nun ein 2-zusammenhängender, serien-paralleler Graph mit $V(G^*) > 3$, aus dem schon Knoten entfernt wurden. Es soll die Situation vor Ausführung der Zeile 7 betrachtet werden. Führen für alle Paare von Knoten $x$ und $y$ aus $V(G^*)\backslash\{v\}$ zwei knotendisjunkte Wege nicht über $v$, so ist der Graph nach dem Entfernen von $v$ weiterhin 2-zusammenhängend. Seien also $P_1$ und $P_2$ zwei knotendisjunkte Wege zwischen $x$ und $y$ und sei $\{x, y\} \cap \{u, w\} = \emptyset$. O. B. d. A. enthalte $P_1$ den Knoten $v$. Aus $P_1$ kann dann ein neuer Weg konstruiert werden, indem der Knoten $v$ mit seinen inzidenten Kanten entfernt und die Kante $\{u, w\}$ hinzugefügt wird (siehe Abb. 4.1a). Der Weg $P_2$ kann diese Kante nicht verwendet haben, weil $u$ und $w$ Knoten aus $P_1$ sind.

Sei nun o. B. d. A. $x = u$ und $y = w$. Der Graph $G^*$ enthält mindestens einen weiteren Knoten $a$. Da $G^*$ 2-zusammenhängend ist, existieren zwei knotendisjunkte Wege zwischen $v$ und $a$ (vgl. Abb. 4.1b). Dann muss der Knoten $u$ auf einem Weg liegen und $w$ auf dem anderen. Daraus ergibt sich ein neuer Weg, der $u$ mit $w$ über $a$ verbindet. Somit ist nach jedem Entfernen eines Knotens $v$ der Graph wieder ein 2-zusammenhängender, serien-paralleler Graph.

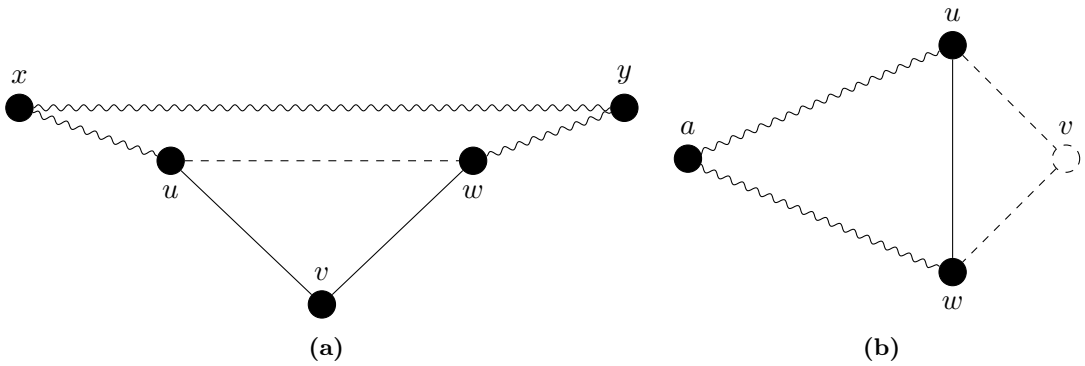



**Abbildung 4.1:** (a) Ein Weg über $v$ kann einen neuen Verlauf nehmen. (b) Spezialfall für $x = u$ und $y = w$

Das Finden der Blöcke und Artikulationen kann mit dem Algorithmus von Hopcroft und Tarjan aus [11] in linearer Rechenzeit in der Eingabelänge geschehen. Da ein SP-Graph mit $n$ Knoten höchstens $2n - 3$ Kanten besitzt, können im ersten Schritt nur $\mathcal{O}(n)$ viele Kanten hinzugefügt werden. In Zeile 14 wird in jedem Schleifendurchlauf eine neue Kante eingefügt. Würde bei der Betrachtung einer Artikulation $v$ eine Kante $\{p, r_B\}$ bereits existieren, so gäbe es eine Artikulation $w$, die entweder zwei Nachfolger $p$ und $r_B$ oder einen Vorgänger $p$ und einen Nachfolger $r_B$ im DFS-Baum besitzt. Im DFS-Baum gibt es für

jeden Knoten einen eindeutigen Weg von der Wurzel. Für den Knoten $r_B$ wären dann aber jeweils verschiedene Wege möglich (siehe Abb. 4.2). Die Zeilen 7–14 werden demnach $\mathcal{O}(n)$ mal ausgeführt. Im zweiten Schritt wird jeder Knoten genau einmal betrachtet und pro Knoten nur eine konstante Anzahl an Operationen durchgeführt. Daraus ergibt sich insgesamt eine Rechenzeit von $\mathcal{O}(n)$. □

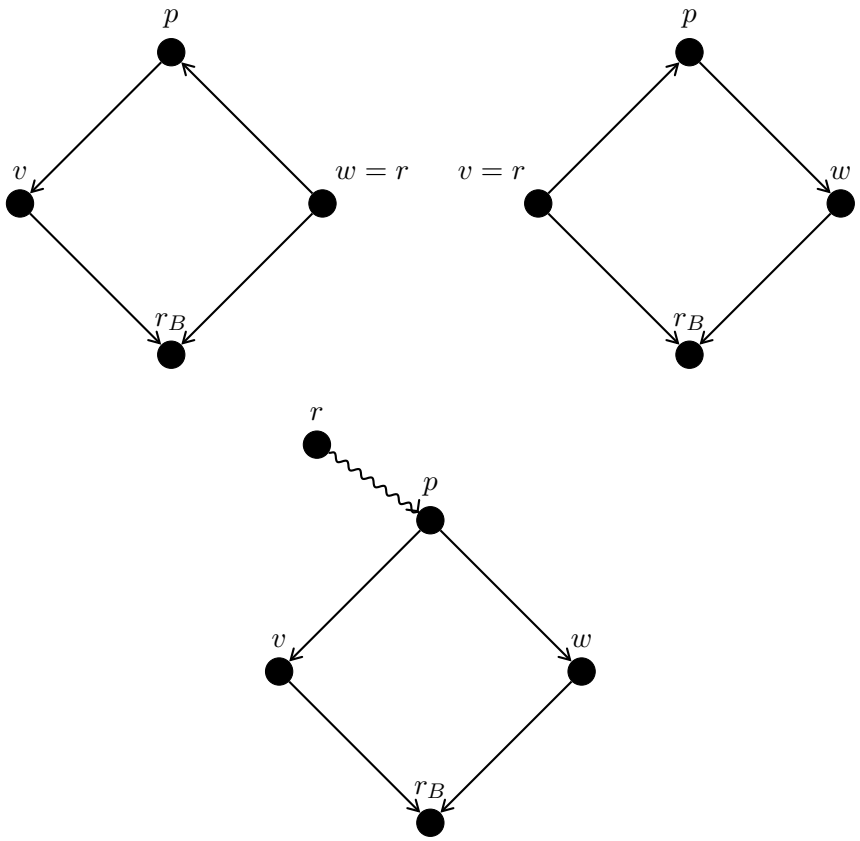



**Abbildung 4.2:** Im DFS-Baum gäbe es jeweils mehrere Wege von der Wurzel $r$ zu $r_B$.

## 4.2 Algorithmus zur Lösung des $STP^{\leq 2}$

Der Algorithmus von Wald und Colbourn reduziert nun, ähnlich wie schon zu Algorithmus 4.2 beschrieben, einen 2-Baum anhand seiner Definition zu $K_2$. Dabei werden für Dreiecke $vba$ mit $d(v) = 2$ Informationen über mögliche Steinerbäume dieses Teilgraphen in den Halbkanten[3] $(a, b)$ und $(b, a)$ gespeichert, bevor der Knoten $v$ entfernt wird (siehe Abb. 4.3). Diese Informationen lassen sich wiederum aus den Informationen, die für die Halbkanten in $\{a, v\}$ und $\{v, b\}$ gespeichert werden, berechnen. Informationen zu Steinerbäumen eines reduzierten Teilgraphen auf eine Halbkante $\alpha = (a, b)$ werden in sechs Variablen gespei-

[3]Hier werden die Kanten $\{a, b\}$ in die Halbkanten $(a, b)$ und $(b, a)$ aufgeteilt, weil für die Variablen die Richtungen der Halbkanten wesentlich sind.

chert: st($\alpha$), dt($\alpha$), ny($\alpha$), yn($\alpha$), nn($\alpha$) und none($\alpha$). Der Rahmen ist in Algorithmus 4.3 dargestellt.

**Algorithmus 4.3** Rahmenalgorithmus für den Algorithmus von Wald und Colbourn

**Eingabe:** Instanz $((V, E), c, T)$ für das STP, der Graph $G$ ist ein 2-Baum

**Ausgabe:** eine Lösung minimalen Wertes für das STP

```
1: for all α ∈ E do
2:     InitialisiereVariablen(α)
3: Q.enqueue(alle Knoten mit Grad 2)
4: while |V| > 2 do                    ▷ Solange noch mehr als 2 Knoten existieren
5:     v ← Q.dequeue()
6:     Ermittle Nachbarn a und b von v
7:     PasseVariablenAn(vab)
8:     Entferne Knoten v und Kanten {v, a} und {v, b} aus G
9:     if d(a) = 2 ∧ a ∉ Q then        ▷ Nachbarn hinzufügen, falls aktiv geworden
10:        Q.enqueue(a)
11:    if d(b) = 2 ∧ b ∉ Q then
12:        Q.enqueue(b)
13: return BerechneAusgabe
```

### 4.2.1 Beschreibung der Variablen

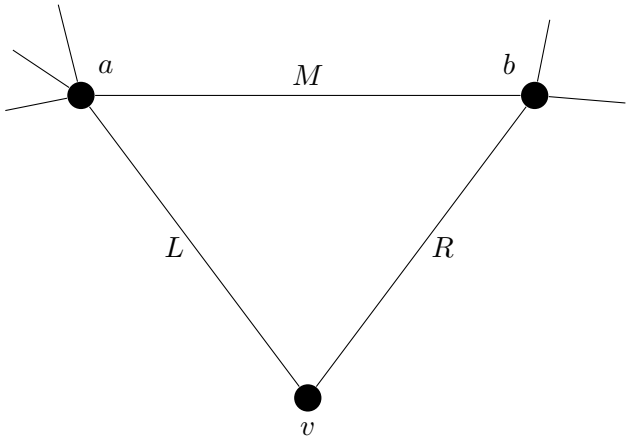



**Abbildung 4.3:** Zu reduzierendes Dreieck im Algorithmus von Wald und Colbourn

Zuerst soll die Interpretation der Variablen erläutert werden. Für eine STP-Instanz $(G = (V, E), c, T)$, in der $G$ ein SP-Graph ist, sei $\alpha = (a, b) \in E$ und $S$ der Teilgraph, der im Verlauf des Algorithmus auf die Kante $\alpha$ reduziert wurde. Der Wert der Variablen st($\alpha$) entspricht dann den Kosten eines Steinerbaumes für $S$, der sowohl $a$ als auch $b$ enthält.

Die Variable dt($\alpha$) speichert die Kosten zweier disjunkter Steinerbäume für $S$, welche alle Terminale aus $S$ enthalten und zusätzlich einer den Knoten $a$ und der andere den Knoten $b$. In ny($\alpha$) und yn($\alpha$) werden jeweils die Kosten eines Steinerbaumes für $S$ gespeichert, der $b$ enthält, aber $a$ nicht bzw. $a$ enthält, aber $b$ nicht. Der Wert der Variablen nn($\alpha$) entspricht den Kosten eines Steinerbaumes für $S$, der weder $a$ noch $b$ enthält. Die Variable none($\alpha$) hingegen speichert die Kosten eines Steinerbaumes, für den keine Kante aus $S$ gewählt werden darf.

Die Variablen ny, yn, nn und none können so im Verlauf des Algorithmus den Wert eines Steinerbaumes enthalten, der keine zulässige Lösung bildet. Beispielsweise ist der Steinerbaum, dessen Wert in ny($(a, b)$) gespeichert wird, niemals Teil einer zulässigen Lösung, wenn $a$ ein Terminal ist. In diesem Fall wird dieser Variablen ein Wert von mindestens $N := \sum_{e \in E} c(e) + 1$ zugewiesen. Für die Variable dt hingegen gilt die Prämisse, dass die Steinerbäume im Verlauf des Algorithmus noch verbunden werden.

### 4.2.2 Initialisierung der Variablen

Bevor neue Werte aus alten gewonnen werden können, müssen die Variablen für jede Kante aus $E$ initialisiert werden. Zu diesem Zeitpunkt wurde noch kein Teilgraph reduziert und die Variablen beziehen sich nur auf eine Kante.

Die einzige Möglichkeit in einem Graphen, der nur aus der Kante $(a, b)$ besteht, einen Steinerbaum zu finden, der sowohl $a$ als auch $b$ enthält, ist die Kante selbst zu wählen. Somit wird st($\alpha$) mit den Kosten für diese Kante initialisiert. Für die Variable dt($\alpha$) darf dagegen die Kante nicht gewählt werden, da sonst die Knoten $a$ und $b$ verbunden wären. Dafür entstehen keine Kosten. Allerdings darf die Variable dt nur dann Verwendung finden, wenn im späteren Verlauf des Algorithmus die Knoten $a$ und $b$ noch verbunden werden.

Für die restlichen vier Variablen darf die Kante $(a, b)$ ebenfalls nicht gewählt werden. Es kann aber passieren, dass Terminalknoten aus dem Steinerbaum ausgeschlossen werden. In diesem Fall wird die Variable mit $N$ initialisiert. Ansonsten entstehen auch hier keine Kosten. Die Initialisierung ist in Algorithmus 4.4 dargestellt.

### 4.2.3 Zusammensetzen der Variablen

Für die Beschreibung der Berechnung der Variablen wird davon ausgegangen, dass ein Dreieck *vba* wie in Abb. 4.3 betrachtet wird. Dabei werden die Halbkanten mit $M := (a, b)$, $L := (a, v)$ und $R := (v, b)$ benannt. Die Teilgraphen, die auf die Kanten $M$, $L$ und $R$ reduziert wurden, werden mit $S_M$, $S_L$ und $S_R$ bezeichnet.

#### Die Variable st

Um einen Baum zusammenzusetzen, der $a$ und $b$ verbindet gibt es vier Möglichkeiten, die jeweils durch eine Summe von Variablen beschrieben werden können. Der Wert der

**Algorithmus 4.4** Initialisierung der Variablen im Algorithmus von Wald und Colbourn

1: $\mathrm{st}(\alpha) \leftarrow c(\alpha)$

2: $\mathrm{dt}(\alpha) \leftarrow 0$

3: $\mathrm{ny}(\alpha) \leftarrow \begin{cases} N & a \in T \\ 0 & \text{sonst} \end{cases}$

4: $\mathrm{yn}(\alpha) \leftarrow \begin{cases} N & b \in T \\ 0 & \text{sonst} \end{cases}$

5: $\mathrm{nn}(\alpha) \leftarrow \mathrm{none}(\alpha) \leftarrow \begin{cases} N & a \in T \vee b \in T \\ 0 & \text{sonst} \end{cases}$

Variablen $\mathrm{st}(M)$ soll dem Wert derjenigen Möglichkeit entsprechen, die den kleinsten Wert hat. Somit kann $\mathrm{st}(M)$ berechnet werden durch

$$\begin{aligned} \mathrm{st}(M) \leftarrow \min\{&\mathrm{st}(M) + \mathrm{dt}(L) + \mathrm{st}(R), \\ &\mathrm{st}(M) + \mathrm{st}(L) + \mathrm{dt}(R), \\ &\mathrm{st}(M) + \mathrm{yn}(L) + \mathrm{ny}(R), \\ &\mathrm{dt}(M) + \mathrm{st}(L) + \mathrm{st}(R)\}. \end{aligned}$$

Bei den ersten drei Möglichkeiten wird die Verbindung über $S_M$ hergestellt. Dann darf über $S_L$ und $S_R$ keine Verbindung zwischen $a$ und $b$ existieren, da sonst ein Kreis entstehen würde. Die letzte Möglichkeit stellt die Verbindung über $S_L$ und $S_R$ her und lässt dabei eine Verbindung über $S_M$ nicht zu. Der dritte Fall lässt als einzige Möglichkeit den Knoten $v$ aus. Die Fälle sind in Abb. 4.4 dargestellt.

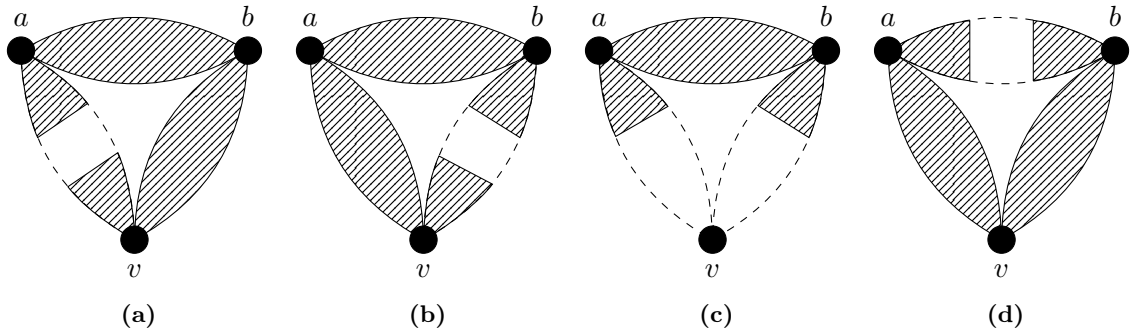



**Abbildung 4.4:** Die Fälle für st im Algorithmus von Wald und Colbourn für das STP: Die schraffierten Teile stellen vom Algorithmus gewählte Steinerbaumfragmente dar.

### Die Variable dt

Um den Wert dieser Variablen zu berechnen sind die folgenden drei Möglichkeiten in Betracht zu ziehen:

$$\begin{aligned}\mathrm{dt}(M) \leftarrow \min\{&\mathrm{dt}(M) + \mathrm{dt}(L) + \mathrm{st}(R),\\ &\mathrm{dt}(M) + \mathrm{st}(L) + \mathrm{dt}(R),\\ &\mathrm{dt}(M) + \mathrm{yn}(L) + \mathrm{ny}(R)\}\end{aligned}$$

Die Fälle sind analog zu den Fällen von st. Für die Kante $M$ darf aber nur dt gewählt werden: Würde st genommen, so wären die Knoten $a$ und $b$ verbunden. Die hier zu unterscheidenden Fälle sind in Abb. 4.5 zu sehen.

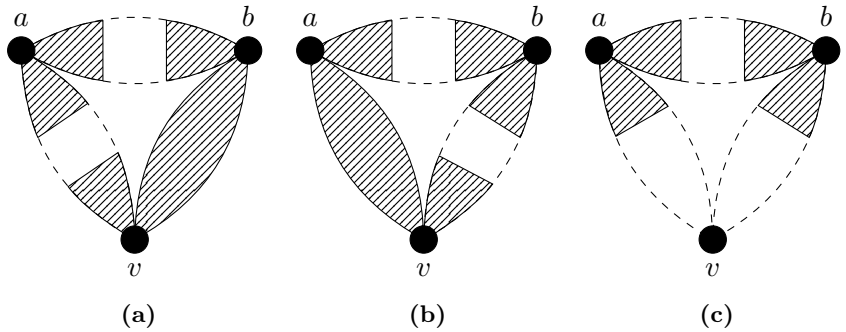



**Abbildung 4.5:** Die Fälle für dt im Algorithmus von Wald und Colbourn für das STP

### Die Variablen ny und yn

Bei diesen beiden Variablen soll jeweils ein Knoten in dem Steinerbaum der Teillösung enthalten sein und der andere nicht. Es genügt hier die Variable yn zu beschreiben; für die Variable ny verhalten sich die Fälle analog. Für die Kante $M$ lässt sich die Variable yn wie folgt berechnen:

$$\begin{aligned}\mathrm{yn}(M) \leftarrow \min\{&\mathrm{yn}(M) + \mathrm{st}(L) + \mathrm{yn}(R),\\ &\mathrm{yn}(M) + \mathrm{yn}(L) + \mathrm{none}(R)\}\end{aligned}$$

Für $M$ ist nur die Variable yn erlaubt. Es kann unterschieden werden, ob der Knoten $v$ im Steinerbaum enthalten ist oder nicht. Im ersten Fall muss zwischen $a$ und $v$ eine Verbindung in $S_L$ bestehen. Für $R$ darf nur yn gewählt werden, da $v$ im Steinerbaum enthalten sein soll, $b$ aber nicht. Im zweiten Fall soll $v$ nicht Teil des Steinerbaumes sein. Dann ist für $L$ nur noch yn möglich und aus $S_R$ darf keine Kante gewählt werden, da diese sonst keine Verbindung zur restlichen Lösung hätten. Die Fälle sind in Abb. 4.6 dargestellt.

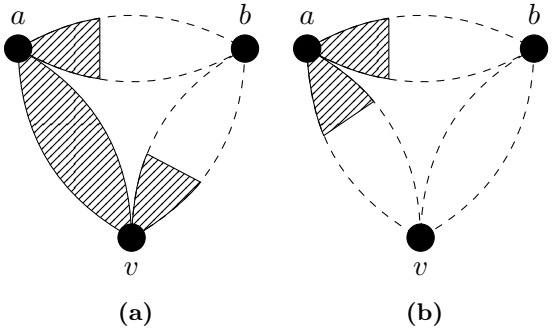



**Abbildung 4.6:** Die Fälle für yn im Algorithmus von Wald und Colbourn für das STP

### Die Variable nn

Bei dieser Variable dürfen sowohl $a$ als auch $b$ nicht im Steinerbaum vorkommen. Dafür gibt es die folgenden Möglichkeiten:

$$\begin{aligned}\mathrm{nn}(M) \leftarrow \min\{&\mathrm{nn}(M) + \mathrm{none}(L) + \mathrm{none}(R),\\ &\mathrm{none}(M) + \mathrm{nn}(L) + \mathrm{none}(R),\\ &\mathrm{none}(M) + \mathrm{none}(L) + \mathrm{nn}(R),\\ &\mathrm{none}(M) + \mathrm{ny}(L) + \mathrm{yn}(R)\}\end{aligned}$$

Die ersten drei Zeilen beschreiben die Fälle, in denen sich ein kompletter Steinerbaum, der die Knoten $a$, $b$ und $v$ nicht enthält, ausschließlich entweder in $S_M$, $S_L$ oder $S_R$ befindet. Grundsätzlich darf der Knoten $v$ aber im Steinerbaum enthalten sein; diese Situation wird von der vierten Zeile abgedeckt: Von $S_M$ wird keine Kante gewählt, dafür aber ein Steinerbaum, der den Knoten $v$ enthält und einen Teil von $S_L$ und $S_R$ abdeckt. Das gelingt durch die Variablen $\mathrm{ny}(L)$ und $\mathrm{yn}(R)$. Die Fälle sind in Abb. 4.7 abgebildet.

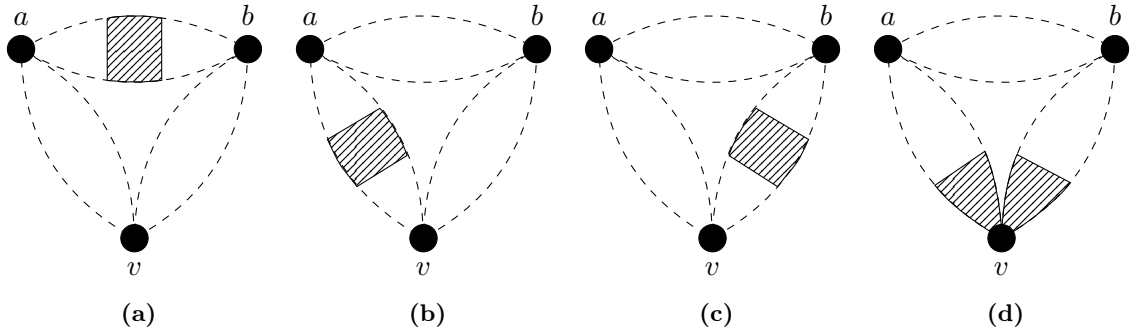



**Abbildung 4.7:** Die Fälle für nn im Algorithmus von Wald und Colbourn für das STP

#### Die Variable none

Als letzte Variable speichert none den Fall, dass keine Kante aus diesem Teilgraphen gewählt wird. Die Gleichung für die Berechnung hat folgende Form:

$$\text{none}(M) \leftarrow \text{none}(M) + \text{none}(L) + \text{none}(R)$$

Die einzige Möglichkeit, keine Kante in diesem Teilgraphen zu wählen, ist, in keinem Teilgraphen $S_M$, $S_L$ oder $S_R$ eine Kante zu wählen.

### 4.2.4 Berechnung der Ausgabe

Der Graph wird reduziert, bis $G = K_2$ ist. Zum Schluss bleibt also eine Kante $\{a, b\}$ bzw. die beiden Halbkanten $\alpha = (a, b)$ und $(b, a)$. Der Algorithmus gibt nun das Minimum der Menge $\{\text{st}(\alpha), \text{yn}(\alpha), \text{ny}(\alpha), \text{nn}(\alpha)\}$ aus.

Um nicht nur die Kosten, sondern auch die Kanten einer Lösung zu erhalten, kann für jede Variable gespeichert werden, welche Kanten bereits gewählt wurden. Wird eine neue Variable zusammengesetzt, entspricht ihre Kantenmenge der Vereinigung der Kantenmengen der Variablen, aus denen sie hervorgeht.

### 4.2.5 Analyse des Algorithmus

In diesem Abschnitt wird bewiesen, dass der Algorithmus von Wald und Colbourn in linearer Zeit in der Anzahl der Knoten einen Steinerbaum für einen serien-parallelen Graphen berechnet. Dazu sei $I = (G = (V, E), w, T)$ eine Instanz für das STP.

**Lemma 4.2.1.** *Die Laufzeit des Algorithmus von Wald und Colbourn ist $\mathcal{O}(|V|)$.*

Beweis. In der Initialisierung wird für jede Halbkante jeder der sechs Variablen ein Wert zugewiesen. Es gibt in einem 2-Baum nach Lemma 3.1.7 genau $2|V| - 3$ Kanten und somit $\mathcal{O}(|V|)$ Halbkanten. Das Testen, ob ein Knoten ein Terminal ist, und das Abfragen der Kostenfunktion kann in konstanter Zeit geschehen. Somit benötigt die Initialisierung eine Laufzeit von $\mathcal{O}(|V|)$.

Für jeden Knoten $v$ mit Grad 2 sucht der Algorithmus jeweils die beiden Nachbarn $a$ und $b$ und berechnet die sechs Variablen für die Halbkanten von $\{a, b\}$ neu. Danach entfernt der Algorithmus den Knoten $v$ mit den zu ihm inzidenten Kanten und fügt unter Umständen die Nachbarn in die Queue ein. Dies sind $\mathcal{O}(1)$ Operationen und insgesamt ergibt sich für das Berechnen der Variablen eine Laufzeit von $\mathcal{O}(|V|)$. Auch das Vergleichen der vier Variablen für die letzte Kante benötigt nur Laufzeit $\mathcal{O}(1)$. □

**Lemma 4.2.2.** *Für eine Instanz $I$, in der $G$ ein 2-Baum ist, berechnet der Algorithmus von Wald und Colbourn einen Steinerbaum.*

Beweis. Die Korrektheit wird in zwei Schritten bewiesen: Zuerst wird gezeigt, dass eine berechnete Lösung zulässig für das STP ist und danach, dass sie auch minimal ist.

Die von dem Algorithmus von Wald und Colbourn berechnete Lösung ist ein Baum, weil die Fallunterscheidungen in den einzelnen Variablen nur Bäume zulassen. Es wird an keiner Stelle ein Kreis geschlossen und der Zusammenhang der einzelnen Teillösungen wird ebenfalls in jedem Fall hergestellt.

Es muss noch gezeigt werden, dass jedes Terminal in der Lösung liegt. Während der Berechnung der Variablen wird ein Summand mit Kosten von $N = \sum_{e \in E} c(e) + 1$ aus der Initialisierung addiert, wenn ein Terminal ausgespart wird. Die Kosten der Lösung, die der Algorithmus von Wald und Colbourn berechnet, sind aber echt kleiner als $N$, da allein über die Variablen st und dt Spannbäume des Eingabegraphen getestet werden können. Diese haben Kosten, die echt kleiner als $N$ sind.

Die Minimalität der von dem Algorithmus berechneten Lösung beruht auf der Eigenschaft von 2-Bäumen, dass jede Kante $e = \{x, y\}$ ein 2-Separator ist. Die Komponenten sind dabei wieder 2-Bäume [20]. Es muss für jede Kante $e = \{x, y\}$ entschieden werden, welche Teilmenge der Endpunkte der Kante $e$ für den Steinerbaum gewählt wird. Dafür gibt es nur vier Möglichkeiten.

Sollen beide Knoten gewählt werden, so gibt es zwei Unterfälle: Werden die Knoten $x$ und $y$ in einer Komponente verbunden, so entspricht dies der Variablen st. Werden die Knoten andererseits in einer Komponente nicht verbunden, so wird der Wert dieses Steinerbaumes in der Variablen dt gespeichert.

Wird nur einer der beiden Knoten in einer Komponente in den Steinerbaum aufgenommen, so gibt es keine weitere Möglichkeit. Diese Fälle werden von den Variablen ny und yn abgedeckt.

Wird keiner der beiden Endpunkte von $e$ in einer Komponente gewählt, so gibt es wieder zwei Möglichkeiten: Erstens kann sich in einer Komponente kein Teil des Steinerbaumes befinden; dieser Fall wird durch die Variable none berechnet. Zweitens kann sich der Steinerbaum komplett in diesem Teilgraphen befinden. Den Wert eines solchen Steinerbaumes speichert die Variable nn.

So werden für jede Kante alle Möglichkeiten betrachtet, die Endpunkte auszuwählen. Bei dem Berechnen der Variablen wird dabei sichergestellt, dass ein Knoten entweder in allen oder in keiner Komponente für den Steinerbaum gewählt wird.

Bei der letzten Kante werden aus allen Möglichkeiten außer dt und none gewählt. Die Lösung, deren Wert die Variable dt speichert, ist für diese Kante unzulässig, da eine zusammenhängende Lösung gesucht wird. Die Lösung, deren Wert die Variable none speichert, kann zwar zulässig sein, wird aber von der Variable nn vollständig abgedeckt. □

**Theorem 4.2.3.** *Der Algorithmus von Wald und Colbourn berechnet für eine Instanz* $(G = (V, E), w, T)$ *für das* $STP^{\leq 2}$ *einen Steinerbaum in einer Laufzeit, die* $\mathcal{O}(|V|)$ *ist.*

BEWEIS. Bevor der Algorithmus für das STP beginnt, kann $G$ mit dem in Abschnitt 4.1 beschriebenen Algorithmus zu einem 2-Baum erweitert werden. Die Kanten aus $M$ erhalten dabei Kosten $N = \sum_{e \in E} c(e) + 1$. In einem zusammenhängenden Graphen existiert immer eine Lösung für das STP; die Lösung hat dabei höchstens die Kosten eines Spannbaumes für $G$ und ein solcher hat Kosten, die echt kleiner als $N$ sind. Der Algorithmus von Wald und Colbourn wird demnach nie eine Kante aus $M$ zur Lösung wählen.

Nach Proposition 4.1.1 ist die Erweiterung in linearer Zeit in der Anzahl der Knoten des Eingabegraphen möglich. Aus diesen Überlegungen folgt mit den Lemmata 4.2.1 und 4.2.2 die Behauptung. □

# Kapitel 5

# Algorithmen für das SSTP auf SP-Graphen

In diesem Kapitel wird ein Algorithmus für das stochastische Steinerbaumproblem auf serien-parallelen Graphen vorgestellt. Dieser beruht auf dem Algorithmus von Wald und Colbourn und berechnet ebenso Zwischenergebnisse aus Variablen. In dem von Wald und Colbourn beschriebenen Rahmen (vgl. Algorithmus 4.3) müssen drei Komponenten angepasst werden: zum Ersten die Initialisierung in Zeile 2, zum Zweiten das Anpassen der Variablen in jedem Schritt der Reduktion in Zeile 7 und als Letztes die Berechnung der Ausgabe aus den Variablen in Zeile 13.

Um die Methode, mit der der Algorithmus von Wald und Colbourn erweitert wurde, zu beschreiben, soll der Algorithmus zunächst an einer vereinfachten Variante des Problems vorgestellt werden. Sei für eine Instanz eine ursprünglich zulässige Lösung gegeben, die aus *First-Stage*-Kanten[1] $E_0$ und *Second-Stage*-Kanten $E_k$ für jedes Szenario $k$ mit $1 \leq k \leq K$ besteht. Bei dieser Variante ist diese Lösung nur dann zulässig, wenn sowohl $G[E_0]$ als auch $G[E_0 \cup E_k]$ für $1 \leq k \leq K$ zusammenhängend sind. Dieses Problem wird mit $\text{SSTP}^{\leq 2}_{\text{fs,ss}}$ bezeichnet.

Analog zu dieser Variante und dem $\text{SSTP}^{\leq 2}$ kann noch die Variante betrachtet werden, in der $G[E_0]$ zusammenhängend ist, aber $G[E_0 \cup E_k]$ nicht notwendigerweise ($\text{SSTP}^{\leq 2}_{\text{fs}}$) und die Variante, in der $G[E_0 \cup E_k]$ zusammenhängend ist, aber $G[E_0]$ nicht notwendigerweise ($\text{SSTP}^{\leq 2}_{\text{ss}}$). Daraus ergeben sich insgesamt vier Varianten. Sie und die Laufzeitschranken, die sich aus den hier vorgestellten Algorithmen ergeben, sind in Tabelle 5.1 dargestellt.

[1] In diesem Kapitel werden wegen der kompakteren Schreibweise häufig die englischen Begriffe für die erste und zweite Phase verwendet.

| | | $G[E_0]$ | |
|---|---|---|---|
| | | zusammenhängend | nicht notwendigerweise zusammenhängend |
| $G[E_0 \cup E_k]$ | zusammenhängend | $\text{SSTP}^{\leq 2}_{\text{fs,ss}} : \Theta(12^k n)$ | $\text{SSTP}^{\leq 2}_{\text{ss}} : \Theta(12^k n)$ |
| | nicht notwendigerweise zusammenhängend | $\text{SSTP}^{\leq 2}_{\text{fs}} : \Theta(16^k n)$ | $\text{SSTP}^{\leq 2} : \Theta(16^k n)$ |

**Tabelle 5.1:** Übersicht der Varianten mit den Laufzeitschranken der vorgestellten Algorithmen für einen SP-Graphen mit $n$ Knoten bei $k$ Szenarien

## 5.1 Algorithmus für das $\text{SSTP}^{\leq 2}_{\text{fs,ss}}$

Die Variablen setzen sich für die erste Variante wie folgt zusammen: Für jedes Szenario $k$ und jede Halbkante $\alpha$ gibt es eine Menge von Variablen, wie sie schon aus dem Algorithmus von Wald und Colbourn bekannt sind: $\text{st}_k(\alpha)$, $\text{dt}_k(\alpha)$, $\text{yn}_k(\alpha)$, $\text{ny}_k(\alpha)$, $\text{nn}_k(\alpha)$ und $\text{none}_k(\alpha)$. Die Bedeutungen der Variablen bleiben wie im ursprünglichen Algorithmus erhalten, beziehen sich aber alleine auf das Szenario $k$.

Sei $S_\alpha$ der auf die Halbkante $\alpha$ reduzierte Teilgraph. Beispielsweise wird dann in der Variablen $\text{st}_k(\alpha)$ der Wert einer günstigsten Lösung von $S_\alpha$ gespeichert, die aus einem Steinerbaum für die Terminalmenge $T_k \cap V(S_\alpha)$ besteht, welcher die Knoten $a$ und $b$ verbindet. Als Kostenfunktion wird dafür die mit der Wahrscheinlichkeit gewichtete Kostenfunktion dieses Szenarios verwendet. Diese Variablen werden als *Second-Stage*-Variablen bezeichnet.

Die mit Hilfe der *Second-Stage*-Variablen berechneten Steinerbäume enthalten keine *First-Stage*-Kanten. Sie können also unabhängig voneinander durch die im Abschnitt über den Algorithmus von Wald und Colbourn vorgestellten Regeln berechnet werden.

Zusätzlich werden für jede Halbkante $\alpha$ fünf Variablen bzw. Variablenklassen für die erste Phase hinzugefügt:

- Die Variablen $\text{st}(\alpha)$ und $\text{dt}(\alpha)$ sowie
- für jede Auswahl paarweise disjunkter Mengen ST, DT, YN $\subseteq \{1, \ldots, K\}$ mit ST $\dot\cup$ DT $\dot\cup$ YN $= \{1, \ldots, K\}$ die Variable $\text{yn}_{\text{ST,DT,YN}}(\alpha)$,
- für jede Auswahl paarweise disjunkter Mengen ST, DT, NY $\subseteq \{1, \ldots, K\}$ mit ST $\dot\cup$ DT $\dot\cup$ NY $= \{1, \ldots, K\}$ die Variable $\text{ny}_{\text{ST,DT,NY}}(\alpha)$ und
- für jede Auswahl paarweise disjunkter Mengen ST, DT, YN, NY, NN $\subseteq \{1, \ldots, K\}$ mit ST $\dot\cup$ DT $\dot\cup$ YN $\dot\cup$ NY $\dot\cup$ NN $= \{1, \ldots, K\}$ die Variable $\text{nn}_{\text{ST,DT,YN,NY,NN}}(\alpha)$.

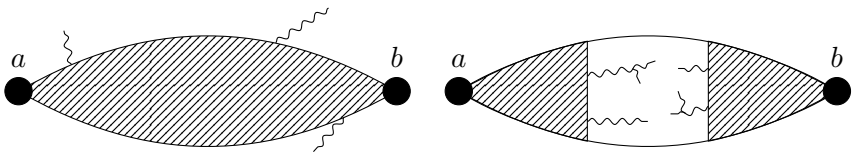
Diese Variablen werden *First-Stage*-Variablen genannt.

### 5.1.1 Beschreibung der Variablen

Die *First-Stage*-Variablen beschreiben jeweils einen Fall, der in der ersten Phase auftreten kann. Hier sei eine Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ für das $\text{SSTP}^{\leq 2}_{\text{fs,ss}}$ gegeben. Es sei $G$ ein 2-Baum, $\alpha = (a, b) \in E$ und $S_\alpha$ der Teilgraph, der im Laufe des Algorithmus auf die Halbkante $\alpha$ reduziert wurde.

**st** Die Variable $\text{st}(\alpha)$ speichert den Wert einer günstigsten Lösung für das $\text{SSTP}^{\leq 2}_{\text{fs,ss}}$ für $S_\alpha$, in der die Knoten $a$ und $b$ durch Kanten aus der ersten Phase verbunden sind. (Siehe Abb. 5.1a)

**dt** Der Wert der Variablen $\text{dt}(\alpha)$ entspricht den Kosten einer günstigsten Lösung für $S_\alpha$, in der die Kanten für die erste Phase zwei disjunkte Bäume bilden; einer dieser Bäume enthält $a$, der andere $b$ (Siehe Abb. 5.1b). Es ist ebenfalls erlaubt, dass keine Kante in der ersten Phase gewählt wird.




**(a)** $\text{st}((a, b))$ **(b)** $\text{dt}((a, b))$

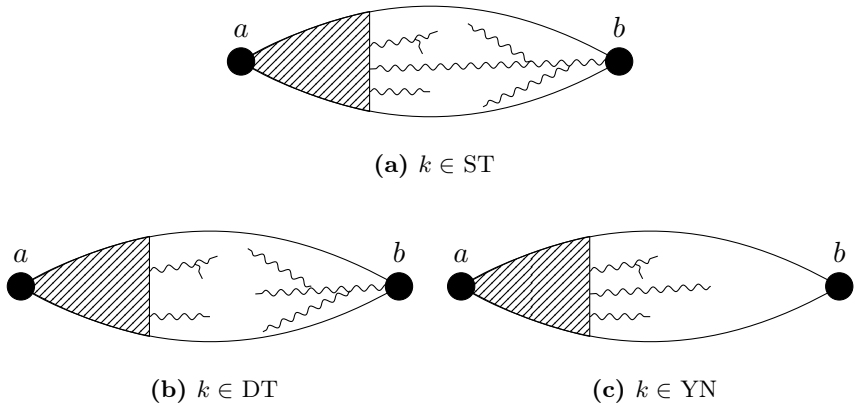
**Abbildung 5.1:** Illustration der Variablen st und dt: Der schraffierte Teil stellt *First-Stage*-Teile dar, während die geschlängelten Linien mögliche *Second-Stage*-Fragmente eines Szenarios zeigen.

**$\text{yn}_{\text{ST,DT,YN}}$** In $\text{yn}_{\text{ST,DT,YN}}(\alpha)$ werden die Kosten einer günstigsten Lösung $(E_0, \ldots, E_K)$ für $S_\alpha$ gespeichert, in der für $E_0 \neq \emptyset$ der Knoten $a$ in $G[E_0]$ ist, der Knoten $b$ hingegen nicht. An die Lösung werden je nach Zusammensetzung der Mengen ST, DT und YN weitere Bedingungen gestellt:

- Für $k \in$ ST müssen $a$ und $b$ in $G[E_0 \cup E_k]$ sein und ein Weg in $G[E_0 \cup E_k]$ existieren, der die Knoten $a$ und $b$ verbindet.
- Für kein $k \in$ DT darf es einen Weg in $G[E_0 \cup E_k]$ geben, der die Knoten $a$ und $b$ miteinander verbindet.
- Für kein Szenario $k \in$ YN darf der Knoten $b$ in $G[E_0 \cup E_k]$ enthalten sein.

Die Fälle sind in Abb. 5.2 zusammengefasst.

**$\text{ny}_{\text{ST,DT,NY}}$** Die Bedeutung dieser Variablen ist analog zu der Bedeutung der Variablen $\text{yn}_{\text{ST,DT,YN}}$.




**(a)** $k \in \mathrm{ST}$

**(b)** $k \in \mathrm{DT}$ **(c)** $k \in \mathrm{YN}$

**Abbildung 5.2:** Die Unterfälle für die Variable $\mathrm{yn}_{\mathrm{ST,DT,YN}}((a,b))$ für ein Szenario $k : 1 \leq k \leq K$

**$\mathrm{nn}_{\mathbf{ST,DT,YN,NY,NN}}$** Der Wert der Variablen $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}(\alpha)$ entspricht den Kosten einer günstigsten Lösung für $S_\alpha$, in der weder der Knoten $a$ noch der Knoten $b$ in $G[E_0]$ enthalten ist. Auch hier sind Bedingungen an die *Second-Stage*-Teillösung zu stellen. Zusätzlich zu den Bedingungen bei $\mathrm{yn}_{\mathrm{ST,DT,YN}}$ kommen noch die folgenden Fälle hinzu:

- Für kein Szenario $k \in \mathrm{NY}$ darf der Knoten $a$ in $G[E_0 \cup E_k]$ enthalten sein.
- Für kein $k \in \mathrm{NN}$ dürfen die Knoten $a$ oder $b$ in $G[E_0 \cup E_k]$ enthalten sein.

Die Bedingungen für die Variable $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}$ sind in Abb. 5.3 dargestellt.

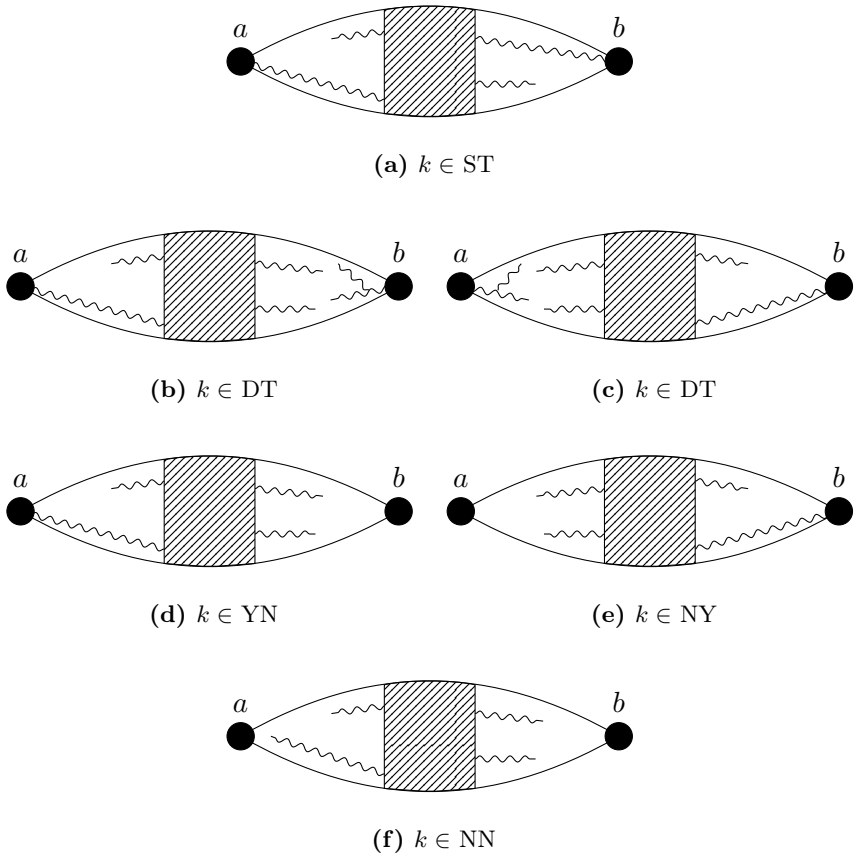
Sowohl bei den *First-* als auch bei den *Second-Stage*-Variablen ist es wie beim Algorithmus von Wald und Colbourn möglich, dass die Variablen Werte einer Lösung speichern sollen, die für die SSTP-Variante nicht zulässig ist. Diese Variablen erhalten dann mindestens einen Wert von

$$N := \sum_{e \in E} \max \left\{ \sum_{k=1}^{K} p_k c_k(e), c_0(e) \right\} + 1.$$

### 5.1.2 Initialisierung der Variablen

Die *Second-Stage*-Variablen werden wie im ursprünglichen Algorithmus initialisiert (siehe Algorithmus 5.1). Die Initialisierung der *First-Stage*-Variablen wird in diesem Abschnitt im Detail beschrieben. Sei dafür eine Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ für das $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$ gegeben. Der Graph $G$ sei ein 2-Baum. Die *First-Stage*-Variablen werden dann ebenfalls für jede Halbkante $\alpha = (a, b) \in E$ initialisiert (vgl. Algorithmus 5.2).

Der Wert der Variablen $\mathrm{st}(\alpha)$ wird auf $c_0(\alpha)$ gesetzt, weil die einzige Möglichkeit, die Knoten $a$ und $b$ durch einen Weg aus *First-Stage*-Kanten zu verbinden, darin besteht, diese Kante zu wählen. Die Variable $\mathrm{dt}(\alpha)$ wird auf 0 gesetzt, da sonst durch die Wahl der Kante die Knoten $a$ und $b$ miteinander verbunden wären.




**(a)** $k \in \mathrm{ST}$

**(b)** $k \in \mathrm{DT}$

**(c)** $k \in \mathrm{DT}$

**(d)** $k \in \mathrm{YN}$

**(e)** $k \in \mathrm{NY}$

**(f)** $k \in \mathrm{NN}$

**Abbildung 5.3:** Die Unterfälle für die Variable $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}((a,b))$ für ein Szenario $k : 1 \leq k \leq K$

Für die Variable $\mathrm{yn}_{\mathrm{ST,DT,YN}}(\alpha)$ darf die Halbkante $\alpha$ nicht in der ersten Phase gewählt werden; ansonsten wäre der Knoten $b$ in $G[E_0]$ enthalten. Der Wert der Variablen $\mathrm{yn}_{\mathrm{ST,DT,YN}}(\alpha)$ wird zu $N$, falls für ein Szenario $k \in \mathrm{YN}$ der Knoten $b \in T_k$ ist. Dann kommt dieses Terminal endgültig nicht mehr in der Lösung für dieses Szenario vor und die Lösung ist somit nicht zulässig. Ist $k \notin \mathrm{YN}$ und $b \in T_k$, so wird durch die *Second-Stage*-Bäume gesichert, dass der potentielle Terminalknoten $b$ in dem jeweiligen Szenario verbunden ist. Für Szenarien $k \in \mathrm{DT}$ gilt wieder die Prämisse, dass die Knoten $a$ und $b$ im Verlauf des Algorithmus noch verbunden werden und somit immer gültige Lösungen entstehen. Die Kosten, die sich dann ergeben, errechnen sich aus den Szenarien, die in der Menge ST vorkommen: Für $k \in \mathrm{ST}$ sollen die Knoten $a$ und $b$ durch *First-* und *Second-Stage*-Kanten verbunden sein, was nur möglich ist, wenn die Halbkante $\alpha$ in der zweiten Phase gewählt wird. Die Kosten sind demnach $\sum_{k \in \mathrm{ST}} p_k c_k(\alpha)$. Szenarien in DT kommen hier nicht weiter in Betracht, da durch das Wählen der Halbkante $\alpha$ in der zweiten Phase für $k \in \mathrm{DT}$ die Knoten $a$ und $b$ in dem Teilgraphen, der nur aus dieser Kante besteht, verbunden wären. Für die Variable $\mathrm{ny}_{\mathrm{ST,DT,NY}}$ gelten aus Symmetriegründen die gleichen Überlegungen.

**Algorithmus 5.1** Initialisierung der *Second-Stage*-Variablen in dem Algorithmus für die erste Variante des SSTP

**Eingabe:** eine Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ und die zu initialisierende Halbkante $\alpha = (a, b)$

1: **for** $k : 1 \leq k \leq K$ **do**

2: $\quad \mathrm{st}_k(\alpha) \leftarrow p_k c_k(\alpha)$

3: $\quad \mathrm{dt}_k(\alpha) \leftarrow 0$

4: $\quad \mathrm{yn}_k(\alpha) \leftarrow \begin{cases} N & b \in T_k \\ 0 & \text{sonst} \end{cases}$

5: $\quad \mathrm{ny}_k(\alpha) \leftarrow \begin{cases} N & a \in T_k \\ 0 & \text{sonst} \end{cases}$

6: $\quad \mathrm{none}_k(\alpha) \leftarrow \begin{cases} N & a \in T_k \vee b \in T_k \\ 0 & \text{sonst} \end{cases}$

Auch für die Variable $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}(\alpha)$ darf die Halbkante $\alpha$ nicht in der ersten Phase gewählt werden, da sonst sowohl $a$ als auch $b$ in $G[E_0]$ enthalten wären. Der Wert von $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}(\alpha)$ wird ebenfalls zu $N$, falls ein Terminal eines Szenarios endgültig nicht mehr in der Lösung vorkommen kann. Das geschieht erstens, wenn für ein Szenario $k \in \mathrm{NN}$ ein Knoten $a$ oder $b$ Terminalknoten ist, also $\{a, b\} \cap T_k \neq \emptyset$. Zweitens wird ein Terminal ausgelassen, wenn für $k \in \mathrm{YN}$ der Knoten $b$ in der Terminalmenge des Szenarios $k$ ist. Oder drittens – analog zum zweiten Fall – wird ein Terminal ausgespart, wenn für $k \in \mathrm{NY}$ der Knoten $a \in T_k$ ist. Die Kosten ergeben sich wie für die Variable $\mathrm{yn}_{\mathrm{ST,DT,YN}}(\alpha)$: Sie entsprechen der Summe der durch die Wahrscheinlichkeiten gewichteten *Second-Stage*-Kosten der Kante in allen Szenarien mit $k \in \mathrm{ST}$, da eine Verbindung zwischen $a$ und $b$ über diese Kante in diesem Szenario hergestellt werden muss. Wird kein Terminalknoten ausgeschlossen, so erzeugen Szenarien in den Mengen YN, NY und NN keine weiteren Kosten, weil für diese Szenarien keine Kante gewählt werden darf. Szenarien in der Menge DT erzeugen ebenfalls keine Kosten, da weder die Knoten $a$ oder $b$ ausgeschlossen werden – unter der Prämisse, dass die Knoten $a$ und $b$ noch verbunden werden –, noch die Halbkante $\alpha$ gewählt werden darf.

### 5.1.3 Zusammensetzen der Variablen

Um die Variablen im Laufe des Algorithmus aus alten Variablen zu berechnen, sind einige Fallunterscheidungen nötig. Wie im Algorithmus von Wald und Colbourn werden dafür Gleichungen angegeben. Die Bezeichnungen sind wie im ursprünglichen Algorithmus gegeben und beziehen sich auf das Dreieck *vba* aus Abb. 4.3. Die Halbkanten $(a, b)$, $(a, v)$ und

**Algorithmus 5.2** Initialisierung der *First-Stage*-Variablen in dem Algorithmus für die erste Variante des SSTP

**Eingabe:** eine Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ und die zu initialisierende Halbkante $\alpha = (a, b)$

1: $\mathrm{st}(\alpha) \leftarrow c_0(\alpha)$

2: $\mathrm{dt}(\alpha) \leftarrow 0$

3: **for all** $\mathrm{ST} \dot\cup \mathrm{DT} \dot\cup \mathrm{YN} = \{1, \ldots, K\}$ **do**

4: $$\mathrm{yn}_{\mathrm{ST,DT,YN}}(\alpha) \leftarrow \begin{cases} N & \exists k \in \mathrm{YN} : b \in T_k \\ \sum\limits_{k \in \mathrm{ST}} p_k c_k(\alpha) & \text{sonst} \end{cases}$$

5: **for all** $\mathrm{ST} \dot\cup \mathrm{DT} \dot\cup \mathrm{NY} = \{1, \ldots, K\}$ **do**

6: $$\mathrm{ny}_{\mathrm{ST,DT,NY}}(\alpha) \leftarrow \begin{cases} N & \exists k \in \mathrm{NY} : a \in T_k \\ \sum\limits_{k \in \mathrm{ST}} p_k c_k(\alpha) & \text{sonst} \end{cases}$$

7: **for all** $\mathrm{ST} \dot\cup \mathrm{DT} \dot\cup \mathrm{YN} \dot\cup \mathrm{NY} \dot\cup \mathrm{NN} = \{1, \ldots, K\}$ **do**

8: $$\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}(\alpha) \leftarrow \begin{cases} & \exists k \in \mathrm{YN} : b \in T_k \vee \\ N & \exists k \in \mathrm{NY} : a \in T_k \vee \\ & \exists k \in \mathrm{NN} : a \in T_k \vee b \in T_k \\ \sum\limits_{k \in \mathrm{ST}} p_k c_k(\alpha) & \text{sonst} \end{cases}$$

$(v, b)$ werden mit $M$, $L$ und $R$ bezeichnet und die auf sie reduzierten Teilgraphen mit $S_M$, $S_L$ und $S_R$.

### Die Variablen st und dt

Für die Variable st gibt es für die *First-Stage*-Teillösung die gleichen Möglichkeiten wie im Algorithmus von Wald und Colbourn, die schon in Abb. 4.4 zu sehen waren. Allerdings gibt es die Variablen yn und ny nicht in der Form, wie sie im ursprünglichen Algorithmus vorkommen. Stattdessen wird dafür eine Menge von Summen hinzugefügt, die hier als $\mathcal{S}^{\mathrm{st}}$ bezeichnet wird.

$$\begin{aligned} \mathrm{st}(M) \leftarrow \min(\{&\mathrm{st}(M) + \mathrm{dt}(L) + \mathrm{st}(R), && (5.1)\\ &\mathrm{st}(M) + \mathrm{st}(L) + \mathrm{dt}(R), && (5.2)\\ &\mathrm{dt}(M) + \mathrm{st}(L) + \mathrm{st}(R)\} && (5.3)\\ &\cup \mathcal{S}^{\mathrm{st}}) && (5.4) \end{aligned}$$

Die Menge $\mathcal{S}^{\mathrm{st}}$ bildet den Fall ab, in dem für die Halbkante $M$ die Variable st gewählt wird und für die Halbkanten $L$ und $R$ jeweils die Variablen $\mathrm{yn}_{\mathrm{ST}_L,\mathrm{DT}_L,\mathrm{YN}_L}(L)$ und $\mathrm{ny}_{\mathrm{ST}_R,\mathrm{DT}_R,\mathrm{NY}_R}(R)$ (vgl. Abb. 4.4d). Für die Variable $\mathrm{yn}_{\mathrm{ST}_L,\mathrm{DT}_L,\mathrm{YN}_L}(L)$ und die Variable

$\mathrm{ny}_{\mathrm{ST}_R,\mathrm{DT}_R,\mathrm{NY}_R}(R)$ sind aber nur bestimmte Auswahlen von Szenarien für die Mengen $\mathrm{ST}_L$, $\mathrm{DT}_L$ und $\mathrm{YN}_L$ sowie für die Mengen $\mathrm{ST}_R$, $\mathrm{DT}_R$ und $\mathrm{NY}_R$ möglich. Die Menge $\mathcal{S}^{\mathrm{st}}$ kann somit wie folgt beschrieben werden:

$$\begin{aligned}\mathcal{S}^{\mathrm{st}} := \{&\mathrm{st}(M)+\mathrm{yn}_{\mathrm{ST}_L,\mathrm{DT}_L,\mathrm{YN}_L}(L) + \mathrm{ny}_{\mathrm{ST}_R,\mathrm{DT}_R,\mathrm{NY}_R}(R) \mid \\ &\forall 1 \leq k \leq K : \\ &k \in \mathrm{YN}_L \wedge k \in \mathrm{NY}_R \vee \qquad (5.5)\\ &k \in \mathrm{ST}_L \wedge k \in \mathrm{DT}_R \vee \qquad (5.6)\\ &k \in \mathrm{DT}_L \wedge k \in \mathrm{ST}_R\} \qquad (5.7)\end{aligned}$$

Ist der *First-Stage*-Baum aus den Lösungen von $\mathrm{st}(M)$ und aus den induzierten Lösungen der Variablen $\mathrm{yn}_{\mathrm{ST,DT,YN}}(L)$ und $\mathrm{ny}_{\mathrm{ST,DT,NY}}(R)$ zusammengesetzt, gibt es für ein Szenario drei Möglichkeiten, wie die *Second-Stage*-Bäume gewählt werden können: Die erste Möglichkeit ist, dass diese Bäume – verbunden mit der *First-Stage*-Teillösung – den aktiven Knoten $v$ nicht verwenden (vgl. Zeile 5.5 und Abb. 5.4a). Dürfen die *Second-Stage*-Fragmente den Knoten $v$ beinhalten, so gibt es zwei weitere Fälle: Entweder kann die Verbindung über $S_L$ geschehen (vgl. Zeile 5.6 und Abb. 5.4b) oder der Knoten kann über $S_R$ angebunden werden (vgl. Zeile 5.7 und Abb. 5.4c).

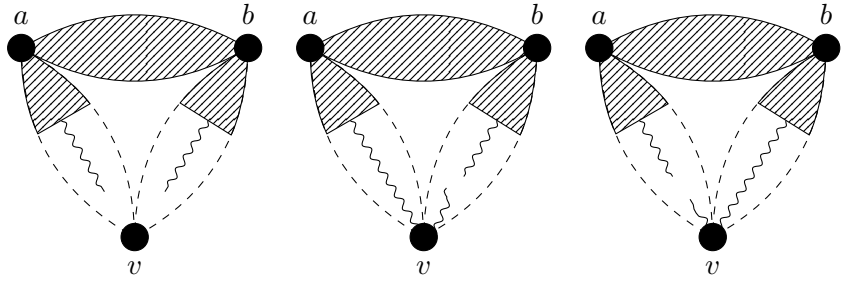
Für die Variable dt gilt dies im Wesentlichen analog zu der Variablen st. Es ist dabei jeweils der erste Summand durch $\mathrm{dt}(M)$ zu ersetzen (vgl. Illustration der Fälle in Abb. 4.5). Die Zeile 5.3 ist bei der Berechnung der Variablen dt allerdings nicht möglich, da sonst die Knoten $a$ und $b$ miteinander verbunden wären. Auch für die Variable dt gibt es eine Summe $\mathcal{S}^{\mathrm{dt}}$, die den Fall abdeckt, dass $v$ nicht in $G[E_0]$ enthalten ist; in dieser muss wiederum der erste Summand durch $\mathrm{dt}(M)$ ausgetauscht werden. Die Variable $\mathrm{dt}(M)$ kann also folgendermaßen berechnet werden:

$$\begin{aligned}\mathrm{dt}(M) \leftarrow \min(\{&\mathrm{dt}(M) + \mathrm{dt}(L) + \mathrm{st}(R),\\ &\mathrm{dt}(M) + \mathrm{st}(L) + \mathrm{dt}(R)\}\\ &\cup \mathcal{S}^{\mathrm{dt}})\end{aligned}$$

### Die Variablen yn und ny

Für die Variable yn sind ebenfalls in der ersten Phase dieselben Fälle möglich, wie im Algorithmus von Wald und Colbourn (vgl. Abb. 4.6). Beide Fälle greifen auf die Variable yn zurück und benötigen somit eine Fallunterscheidung in der zweiten Phase. Die Fälle werden wieder mit Mengen von Summen beschrieben. So lässt sich yn wie folgt berechnen:

$$\mathrm{yn}_{\mathrm{ST,DT,YN}}(M) \leftarrow \min\left(\mathcal{S}_1^{\mathrm{yn}}(\mathrm{ST},\mathrm{DT},\mathrm{YN}) \cup \mathcal{S}_2^{\mathrm{yn}}(\mathrm{ST},\mathrm{DT},\mathrm{YN})\right)$$




**(a)** Zeile 5.5 **(b)** Zeile 5.6 **(c)** Zeile 5.7

**Abbildung 5.4:** Fälle, die die Summen der Menge $\mathcal{S}^{\text{st}}$ für ein Szenario $k$ abdecken

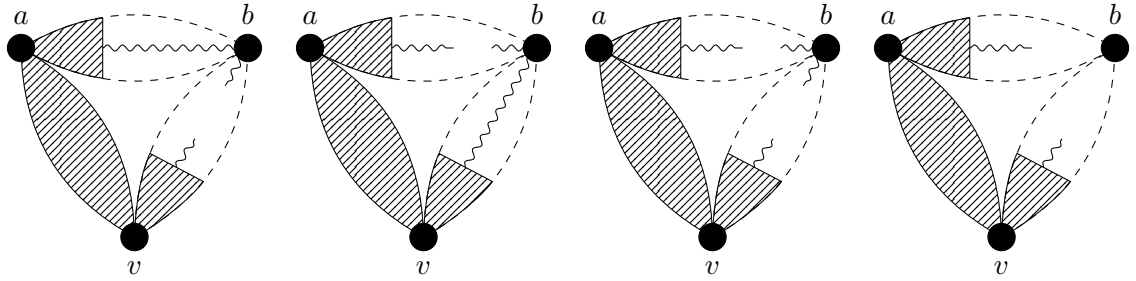
Die erste Menge von Summen $\mathcal{S}_1^{\text{yn}}(\text{ST}, \text{DT}, \text{YN})$ steht für die Möglichkeiten, den Knoten $v$ mit in den *First-Stage*-Teil aufzunehmen (vgl. Abb. 4.6a). Sie hat dann die Form

$$\begin{aligned}\mathcal{S}_1^{\text{yn}}(\text{ST}, \text{DT}, \text{YN}) := \{&\text{yn}_{\text{ST}_M,\text{DT}_M,\text{YN}_M}(M) + \text{st}(L) + \text{yn}_{\text{ST}_R,\text{DT}_R,\text{YN}_R}(R) \mid \\ &\forall k \in \text{ST} : \\ &\quad k \in \text{ST}_M \wedge k \in \text{DT}_R \vee && (5.8)\\ &\quad k \in \text{DT}_M \wedge k \in \text{ST}_R, && (5.9)\\ &\forall k \in \text{DT} : k \in \text{DT}_M \wedge k \in \text{DT}_R, && (5.10)\\ &\forall k \in \text{YN} : k \in \text{YN}_M \wedge k \in \text{YN}_R\}. && (5.11)\end{aligned}$$

Für Szenarien $k \in \text{ST}$ sind zwei Fälle in der zweiten Phase möglich: Die Verbindung zwischen $a$ und $b$ kann über die Teilgraphen $S_M$ oder $S_R$ geschehen. Für die jeweils andere Halbkante muss $k$ in $\text{DT}_R$ bzw. $k$ in $\text{DT}_M$ gewählt werden, da sonst ein Kreis entstehen würde (vgl. Abb. 5.5a und Zeile 5.8 bzw. Abb. 5.5b und Zeile 5.9).

Ist $k \in \text{DT}$, so darf es für das Szenario $k$ keinen Weg in $G[E_0 \cup E_k]$ geben, der die Knoten $a$ und $b$ miteinander verbindet. Da der Knoten $a$ in $G[E_0]$ enthalten ist, darf es demnach keine Verbindung zwischen $b$ und der *First-Stage*-Lösung geben. Von dem Knoten $b$ dürfen aber Bäume aus *Second-Stage*-Kanten ausgehen (vgl. Abb. 5.5c und Zeile 5.8). Diese werden im späteren Verlauf des Algorithmus mit der Teillösung aus Kanten der ersten Phase verbunden.

In dem Fall, dass $k \in \text{YN}$ ist, darf $b$ nicht in $G[E_0 \cup E_k]$ enthalten sein. Die Lösung ist wie für $k \in \text{DT}$ mit der Ausnahme, dass von $b$ keine *Second-Stage*-Bäume ausgehen dürfen (vgl. Abb. 5.5d und Zeile 5.11).




**(a)** $k \in \mathrm{ST}$: Zeile 5.8 **(b)** $k \in \mathrm{ST}$: Zeile 5.9 **(c)** $k \in \mathrm{DT}$: Zeile 5.10 **(d)** $k \in \mathrm{YN}$: Zeile 5.11

**Abbildung 5.5:** Fälle, die die Summen der Menge $\mathcal{S}_1^{\mathrm{yn}}$ für ein Szenario $k$ abdecken

Die zweite Menge von Summen $\mathcal{S}_2^{\mathrm{yn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})$ erfasst die Fälle, in denen $v$ nicht in $G[E_0]$ enthalten ist (vgl. Abb. 4.6b), und hat die Form

$$\mathcal{S}_2^{\mathrm{yn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN}) := \Bigg\{ \mathrm{yn}_{\mathrm{ST}_M, \mathrm{DT}_M, \mathrm{YN}_M}(M) + \mathrm{yn}_{\mathrm{ST}_L, \mathrm{DT}_L, \mathrm{YN}_L}(L) + \sum_{k=1}^{K} \mathrm{var}_k(R) \mid$$

$$\forall k \in \mathrm{ST}:$$

$$k \in \mathrm{ST}_M \wedge k \in \mathrm{YN}_L \wedge \mathrm{var}_k = \mathrm{ny}_k \vee \tag{5.12}$$

$$k \in \mathrm{ST}_M \wedge k \in \mathrm{ST}_L \wedge \mathrm{var}_k = \mathrm{dt}_k \vee \tag{5.13}$$

$$k \in \mathrm{ST}_M \wedge k \in \mathrm{DT}_L \wedge \mathrm{var}_k = \mathrm{st}_k \vee \tag{5.14}$$

$$k \in \mathrm{DT}_M \wedge k \in \mathrm{ST}_L \wedge \mathrm{var}_k = \mathrm{st}_k, \tag{5.15}$$

$$\forall k \in \mathrm{DT}:$$

$$k \in \mathrm{DT}_M \wedge k \in \mathrm{YN}_L \wedge \mathrm{var}_k = \mathrm{ny}_k \vee \tag{5.16}$$

$$k \in \mathrm{DT}_M \wedge k \in \mathrm{ST}_L \wedge \mathrm{var}_k = \mathrm{dt}_k, \tag{5.17}$$

$$k \in \mathrm{DT}_M \wedge k \in \mathrm{DT}_L \wedge \mathrm{var}_k = \mathrm{st}_k \vee \tag{5.18}$$

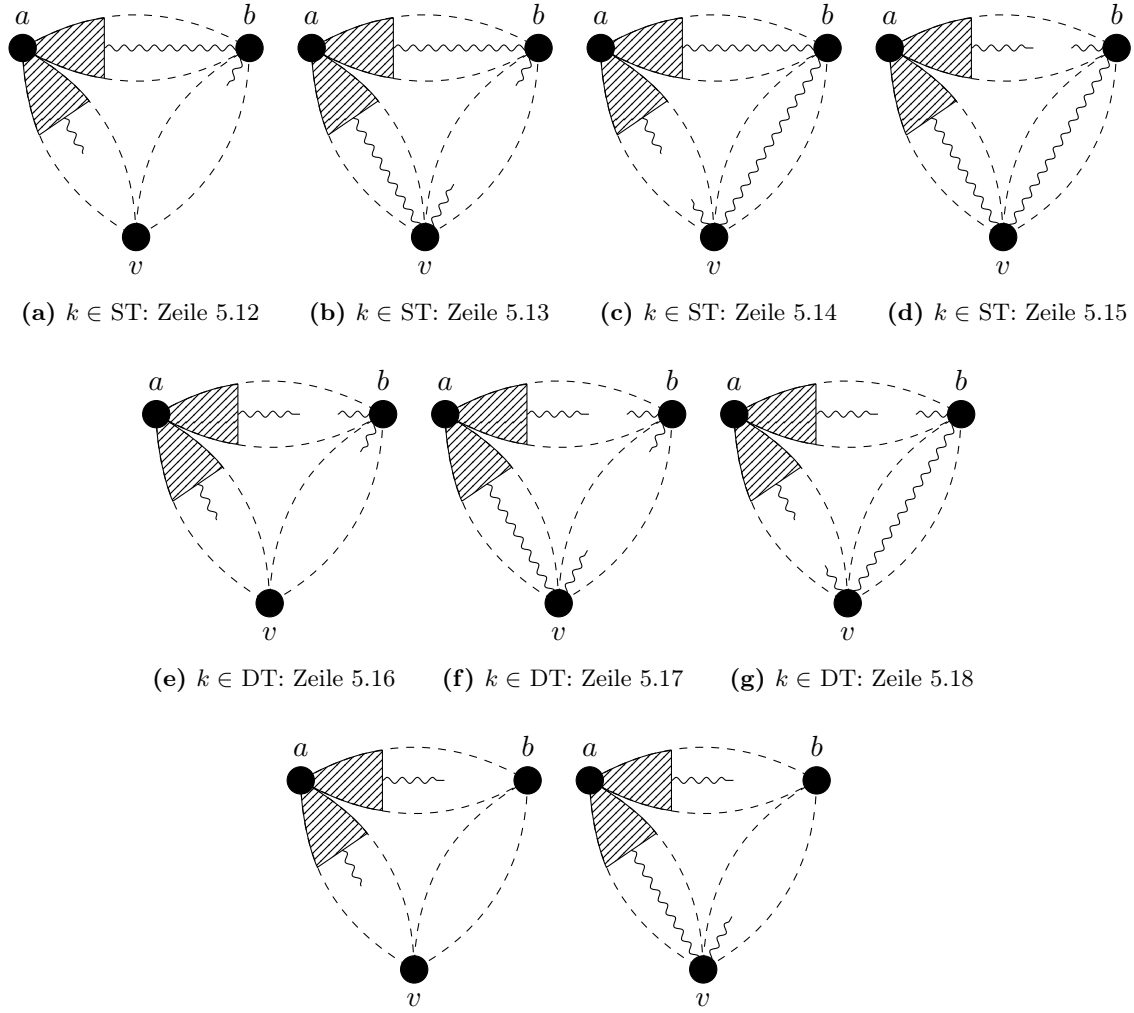
$$\forall k \in \mathrm{YN}:$$

$$k \in \mathrm{YN}_M \wedge k \in \mathrm{YN}_L \wedge \mathrm{var}_k = \mathrm{none}_k \vee \tag{5.19}$$

$$k \in \mathrm{YN}_M \wedge k \in \mathrm{ST}_L \wedge \mathrm{var}_k = \mathrm{yn}_k \Bigg\}. \tag{5.20}$$

Hierbei wird für die Halbkante $R$ für jedes Szenario eine *Second-Stage*-Variable gewählt. Dies geschieht durch die Variable $\mathrm{var}_k(R)$, welche für jedes Szenario $k$ jeweils für eine der *Second-Stage*-Variablen $\mathrm{st}_k(R)$, $\mathrm{dt}_k(R)$, $\mathrm{yn}_k(R)$, $\mathrm{ny}_k(R)$ oder $\mathrm{none}_k(R)$ stehen kann. Die Lösungen, für die diese Variablen stehen, sind für jedes Szenario unabhängig von den Lösungen für die anderen Szenarien. Die Werte können somit aufsummiert werden.

Ist ein Szenario $k \in \mathrm{ST}$, so existieren für $G[E_0 \cup E_k]$ die gleichen Fälle, wie im Algorithmus von Wald und Colbourn für die Steinerbäume, die für die Variable st unterschieden werden müssen (vgl. Abb. 5.6a–5.6d und Zeilen 5.12–5.15).




**(a)** $k \in$ ST: Zeile 5.12 **(b)** $k \in$ ST: Zeile 5.13 **(c)** $k \in$ ST: Zeile 5.14 **(d)** $k \in$ ST: Zeile 5.15

**(e)** $k \in$ DT: Zeile 5.16 **(f)** $k \in$ DT: Zeile 5.17 **(g)** $k \in$ DT: Zeile 5.18

**(h)** $k \in$ YN: Zeile 5.19 **(i)** $k \in$ YN: Zeile 5.20

**Abbildung 5.6:** Fälle, die die Summen der Menge $\mathcal{S}_2^{\text{yn}}$ für ein Szenario $k$ abdecken

Das Gleiche gilt für Szenarien $k \in$ DT und die Variable dt (vgl. Abb. 5.6e–5.6g und Zeilen 5.16–5.18) und für Szenarien $k \in$ YN und die Variable yn (vgl. Abb. 5.6h–5.6i und Zeilen 5.19–5.20).

### Die Variable nn

Die Fälle für die erste Phase sind für die Variable nn wie im Algorithmus von Wald und Colbourn gegeben (vgl. Abb. 4.7). Auch bei dieser Variablen sind einige Unterfälle zu unterscheiden. Die Berechnung benötigt vier Mengen von Summen, deren Minimum bestimmt werden muss:

$$\text{nn}_{\text{ST,DT,YN,NY,NN}}(M) \leftarrow \min \bigcup_{i=1}^{4} \mathcal{S}_i^{\text{nn}}(\text{ST}, \text{DT}, \text{YN}, \text{NY}, \text{NN})$$

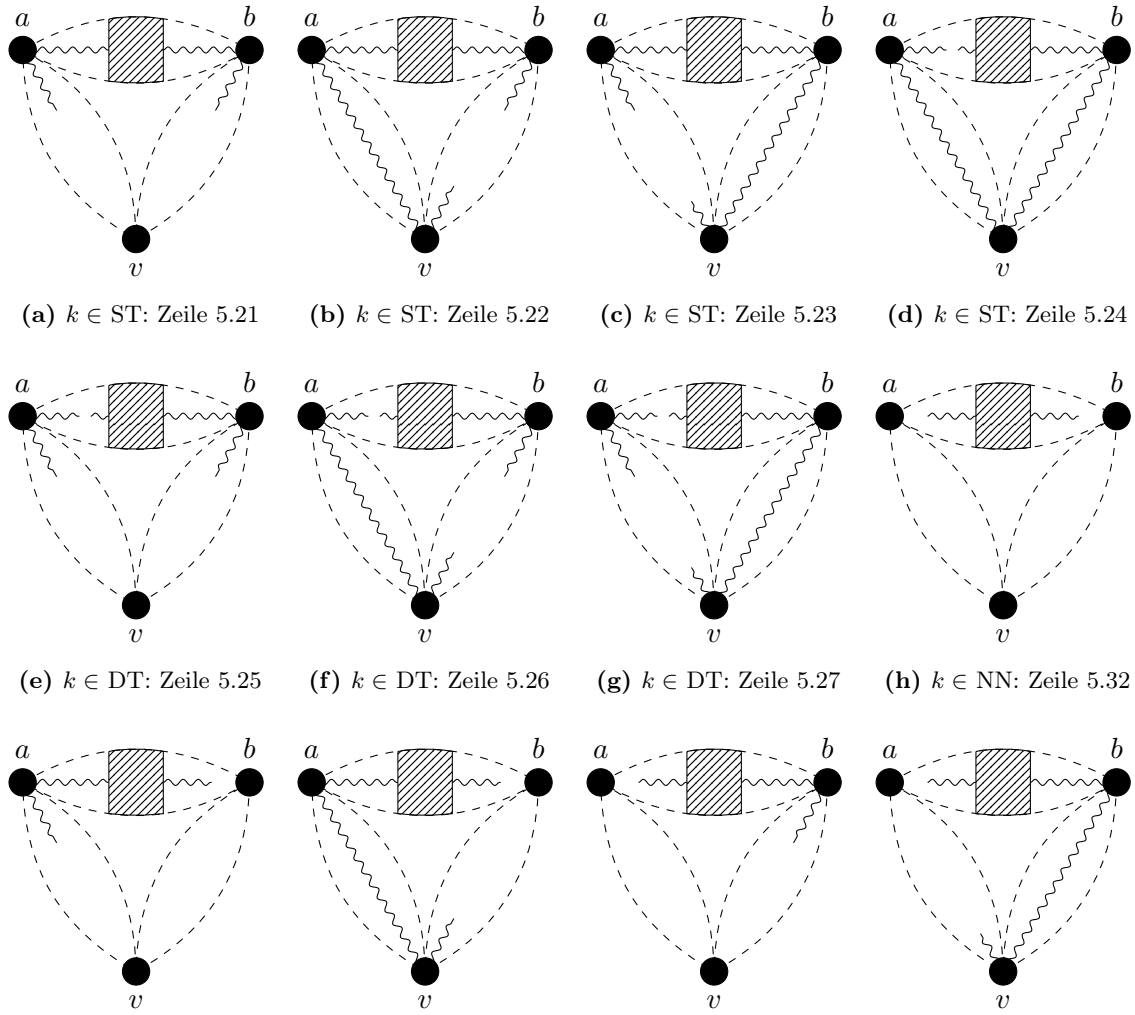
Die ersten drei Mengen stehen für die Fälle, in denen sich $E_0$ vollständig in $S_M$, $S_L$ oder $S_R$ befindet. Hier muss zwischen dem Teilgraphen $S_M$ und den Teilgraphen $S_L$ und $S_R$ unterschieden werden. Die vierte Menge bildet den Fall ab, dass Kanten aus $S_L$ und $S_R$ gewählt werden dürfen, sodass $v$ in $G[E_0]$ enthalten ist.

Die Menge $\mathcal{S}_1^{\text{nn}}$ beschreibt den Fall, dass sich die *First-Stage*-Kanten in $S_M$ befinden (vgl. Abb. 4.7a):

$$
\begin{aligned}
\mathcal{S}_1^{\text{nn}}(\text{ST},\text{DT},\text{YN},\text{NY},\text{NN}) := \Bigg\{ & \text{nn}_{\text{ST}_M,\text{DT}_M,\text{YN}_M,\text{NY}_M,\text{NN}_M}(M) + \\
& \sum_{k=1}^{K} \text{var}_k^L(L) + \sum_{k=1}^{K} \text{var}_k^R(R) \mid \\
& \forall k \in \text{ST} : \\
& \quad k \in \text{ST}_M \wedge \text{var}_k^L = \text{yn}_k \wedge \text{var}_k^R = \text{ny}_k \vee && (5.21) \\
& \quad k \in \text{ST}_M \wedge \text{var}_k^L = \text{st}_k \wedge \text{var}_k^R = \text{dt}_k \vee && (5.22) \\
& \quad k \in \text{ST}_M \wedge \text{var}_k^L = \text{dt}_k \wedge \text{var}_k^R = \text{st}_k \vee && (5.23) \\
& \quad k \in \text{DT}_M \wedge \text{var}_k^L = \text{st}_k \wedge \text{var}_k^R = \text{st}_k, && (5.24) \\
& \forall k \in \text{DT} : \\
& \quad k \in \text{DT}_M \wedge \text{var}_k^L = \text{yn}_k \wedge \text{var}_k^R = \text{ny}_k, && (5.25) \\
& \quad k \in \text{DT}_M \wedge \text{var}_k^L = \text{st}_k \wedge \text{var}_k^R = \text{dt}_k \vee && (5.26) \\
& \quad k \in \text{DT}_M \wedge \text{var}_k^L = \text{dt}_k \wedge \text{var}_k^R = \text{st}_k \vee && (5.27) \\
& \forall k \in \text{YN} : \\
& \quad k \in \text{YN}_M \wedge \text{var}_k^L = \text{yn}_k \wedge \text{var}_k^R = \text{none}_k \vee && (5.28) \\
& \quad k \in \text{YN}_M \wedge \text{var}_k^L = \text{st}_k \wedge \text{var}_k^R = \text{yn}_k, && (5.29) \\
& \forall k \in \text{NY} : \\
& \quad k \in \text{NY}_M \wedge \text{var}_k^L = \text{none}_k \wedge \text{var}_k^R = \text{ny}_k \vee && (5.30) \\
& \quad k \in \text{NY}_M \wedge \text{var}_k^L = \text{ny}_k \wedge \text{var}_k^R = \text{st}_k, && (5.31) \\
& \forall k \in \text{NN} : \\
& \quad k \in \text{NN}_M \wedge \text{var}_k^L = \text{none}_k \wedge \text{var}_k^R = \text{none}_k \Bigg\} && (5.32)
\end{aligned}
$$

Auch hier werden für die Halbkanten $L$ und $R$ Kombinationen von *Second-Stage*-Variablen durch Summen der Variablen $\text{var}_k^L(L)$ und $\text{var}_k^R(R)$ beschrieben. Für ein Szenario $k$ steht die Variable $\text{var}_k^L(L)$ für eine der *Second-Stage*-Variablen $\text{st}_k(L)$, $\text{dt}_k(L)$, $\text{yn}_k(L)$, $\text{ny}_k(L)$ oder $\text{none}_k(L)$ sowie die Variable $\text{var}_k^R(R)$ für $\text{st}_k(R)$, $\text{dt}_k(R)$, $\text{yn}_k(R)$, $\text{ny}_k(R)$ oder $\text{none}_k(L)$.

In den Zeilen 5.21–5.31 bzw. Abb. 5.7a–5.7g und 5.7i–5.7l sind für ein Szenario $k$ und für den Graphen $G[E_0 \cup E_k]$ wieder die Fälle aus dem Algorithmus von Wald und




**(a)** $k \in$ ST: Zeile 5.21 **(b)** $k \in$ ST: Zeile 5.22 **(c)** $k \in$ ST: Zeile 5.23 **(d)** $k \in$ ST: Zeile 5.24

**(e)** $k \in$ DT: Zeile 5.25 **(f)** $k \in$ DT: Zeile 5.26 **(g)** $k \in$ DT: Zeile 5.27 **(h)** $k \in$ NN: Zeile 5.32

**(i)** $k \in$ YN: Zeile 5.28 **(j)** $k \in$ YN: Zeile 5.29 **(k)** $k \in$ NY: Zeile 5.30 **(l)** $k \in$ NY: Zeile 5.31

**Abbildung 5.7:** Fälle, die die Summen der Menge $\mathcal{S}_1^{\text{nn}}$ für ein Szenario $k$ abdecken

Colbourn zu erkennen. Für alle Szenarien $k \in$ NN ist es in der zweiten Phase aber nur möglich Kanten aus $S_M$ zu wählen (vgl. Abb. 5.7h und Zeile 5.32). Werden Kanten aus $S_L$ oder $S_R$ für das Szenario $k$ in der zweiten Phase gewählt, so können diese nicht mehr mit den in der ersten Phase gewählten Kanten verbunden werden, da die Knoten $a$ und $b$ nicht verwendet werden dürfen.

Exemplarisch für die Fälle, in denen sich der *First-Stage*-Teil der Lösung in einem der Teilgraphen $S_L$ oder $S_R$ befindet (vgl. Abb. 4.7b und 4.7c), ist hier die Menge $\mathcal{S}_2^{\text{nn}}$ für den ersten der beiden Fälle angegeben. Für den zweiten Fall ist die Menge $\mathcal{S}_3^{\text{nn}}$ analog definiert.

$$\mathcal{S}_2^{\text{nn}}(\text{ST},\text{DT},\text{YN},\text{NY},\text{NN}) := \Bigg\{ \sum_{k=1}^{K} \text{var}_k^M(M) + \text{nn}_{\text{ST}_L,\text{DT}_L,\text{YN}_L,\text{NY}_L,\text{NN}_L}(L) + \sum_{k=1}^{K} \text{var}_k^R(R) \mid$$

$$\forall k \in \text{ST}:$$

$$\text{var}_k^M = \text{st}_k \wedge k \in \text{ST}_L \wedge \text{var}_k^R = \text{dt}_k \vee \tag{5.33}$$

$$\text{var}_k^M = \text{dt}_k \wedge k \in \text{ST}_L \wedge \text{var}_k^R = \text{st}_k \vee \tag{5.34}$$

$$\text{var}_k^M = \text{st}_k \wedge k \in \text{DT}_L \wedge \text{var}_k^R = \text{st}_k \vee \tag{5.35}$$

$$\text{var}_k^M = \text{st}_k \wedge k \in \text{YN}_L \wedge \text{var}_k^R = \text{ny}_k, \tag{5.36}$$

$$\forall k \in \text{DT}:$$

$$\text{var}_k^M = \text{dt}_k \wedge k \in \text{YN}_L \wedge \text{var}_k^R = \text{ny}_k \vee \tag{5.37}$$

$$\text{var}_k^M = \text{dt}_k \wedge k \in \text{DT}_L \wedge \text{var}_k^R = \text{st}_k \vee \tag{5.38}$$

$$\text{var}_k^M = \text{dt}_k \wedge k \in \text{ST}_L \wedge \text{var}_k^R = \text{dt}_k, \tag{5.39}$$

$$\forall k \in \text{YN}:$$

$$\text{var}_k^M = \text{yn}_k \wedge k \in \text{YN}_L \wedge \text{var}_k^R = \text{none}_k \vee \tag{5.40}$$

$$\text{var}_k^M = \text{yn}_k \wedge k \in \text{ST}_L \wedge \text{var}_k^R = \text{yn}_k, \tag{5.41}$$

$$\forall k \in \text{NY}:$$

$$\text{var}_k^M = \text{ny}_k \wedge k \in \text{NY}_L \wedge \text{var}_k^R = \text{st}_k, \tag{5.42}$$

$$\forall k \in \text{NN}:$$

$$\text{var}_k^M = \text{none}_k \wedge k \in \text{NN}_L \wedge \text{var}_k^R = \text{none}_k \vee \tag{5.43}$$

$$\text{var}_k^M = \text{none}_k \wedge k \in \text{NY}_L \wedge \text{var}_k^R = \text{yn}_k \Bigg\} \tag{5.44}$$

Die Zeilen 5.33–5.42 beschreiben wieder Fälle, wie sie auch im Algorithmus von Wald und Colbourn vorkommen. Für ein Szenario $k \in \text{NY}$ ist im Gegensatz zu den Mengen $\mathcal{S}_1^{\text{nn}}$ der Fall, in dem $v$ nicht in $G[E_k]$ enthalten ist, nicht möglich. Sonst entstünde die Forderung, dass der Knoten $b$ in $G[E_k]$ enthalten sein müsste, die Knoten $a$ und $v$ aber nicht. Dann wäre aber keine Verbindung zwischen dem *First-Stage*-Teil, der sich in $S_L$ befindet, und dem Knoten $b$ möglich (vgl. Abb. 5.8a).

Allerdings ist für Szenarien $k \in \text{NN}$ eine Möglichkeit gegenüber $\mathcal{S}_1^{\text{nn}}$ hinzugekommen: Neben dem Fall, in dem sich $E_0 \cup E_k$ vollständig in $S_L$ befindet (vgl. Zeile 5.43), gibt es auch den Fall, dass der Knoten $v$ in $G[E_k]$ ist und der *Second-Stage*-Baum zusätzlich Kanten aus $S_R$ verwenden darf (vgl. Abb. 5.8b und Zeile 5.44).

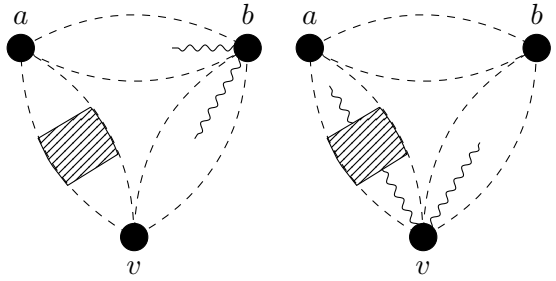



**Abbildung 5.8:** Ausgewählte Fälle, die die Summen der Menge $\mathcal{S}_2^{\text{nn}}$ für ein Szenario $k$ abdecken

Der letzte Fall für die Variable nn im Algorithmus von Wald und Colbourn beschreibt die Möglichkeit, dass $v$ in $G[E_0]$ ist und $G[E_0]$ Kanten aus $S_L$ und $S_R$ enthalten darf (vgl. Abb. 4.7d). Dieser Fall wird durch die folgende Menge von Summen abgedeckt:

$$\mathcal{S}_4^{\text{nn}}(\text{ST}, \text{DT}, \text{YN}, \text{NY}, \text{NN}) := \Bigg\{ \sum_{k=1}^{K} \text{var}_k(M) + \text{ny}_{\text{ST}_L,\text{DT}_L,\text{NY}_L}(L) + \text{yn}_{\text{ST}_R,\text{DT}_R,\text{YN}_R}(R) \mid$$

$$\forall k \in \text{ST}:$$

$$\text{var}_k = \text{st}_k \wedge k \in \text{ST}_L \wedge k \in \text{DT}_R \vee \tag{5.45}$$

$$\text{var}_k = \text{st}_k \wedge k \in \text{DT}_L \wedge k \in \text{ST}_R \vee \tag{5.46}$$

$$\text{var}_k = \text{dt}_k \wedge k \in \text{ST}_L \wedge k \in \text{ST}_R, \tag{5.47}$$

$$\forall k \in \text{DT}:$$

$$\text{var}_k = \text{dt}_k \wedge k \in \text{ST}_L \wedge k \in \text{DT}_R \vee \tag{5.48}$$

$$\text{var}_k = \text{dt}_k \wedge k \in \text{DT}_L \wedge k \in \text{ST}_R, \tag{5.49}$$

$$\forall k \in \text{YN}:$$

$$\text{var}_k = \text{yn}_k \wedge k \in \text{ST}_L \wedge k \in \text{YN}_R, \tag{5.50}$$

$$\forall k \in \text{NY}:$$

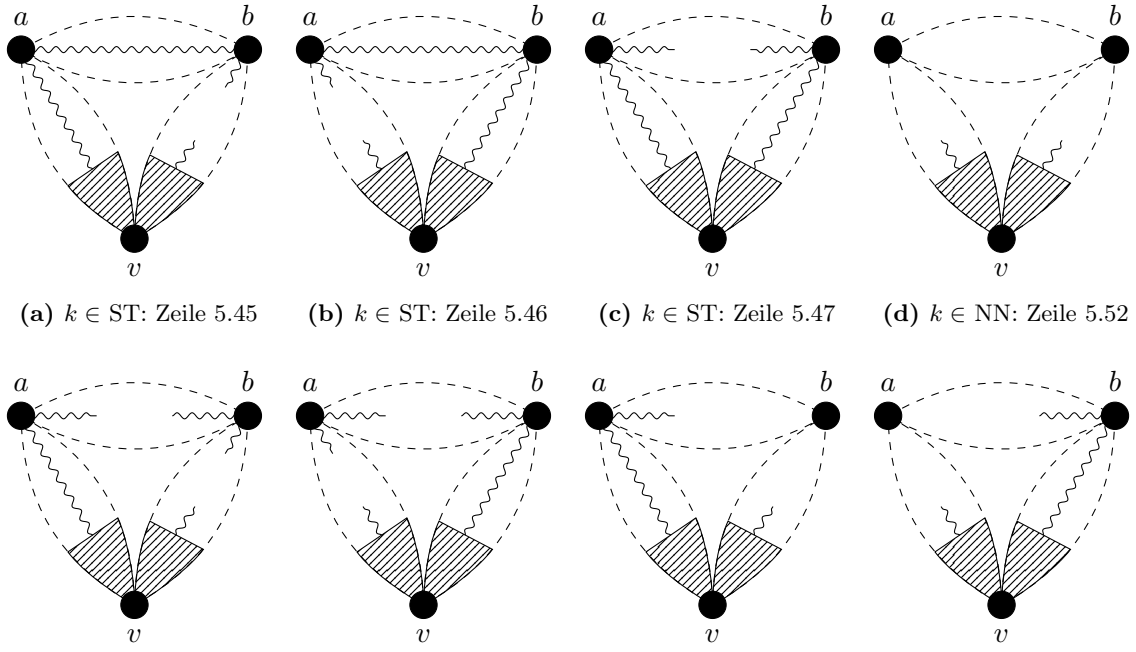
$$\text{var}_k = \text{ny}_k \wedge k \in \text{NY}_L \wedge k \in \text{ST}_R, \tag{5.51}$$

$$\forall k \in \text{NN}:$$

$$\text{var}_k = \text{none}_k \wedge k \in \text{NY}_L \wedge k \in \text{YN}_R \Bigg\} \tag{5.52}$$

Für Szenarien $k \in \text{ST}$ kann die Verbindung der Knoten $a$ und $b$ wie im Algorithmus von Wald und Colbourn hergestellt werden (vgl. Abb. 5.9a–5.9c sowie Zeilen 5.45–5.47). Der Fall für die Variable st, in dem der Knoten $v$ ausgelassen würde, ist allerdings nicht möglich, da dadurch auch die *First-Stage*-Teillösung ausgelassen werden müsste. Dieser Fall existiert auch für die Variablen dt, yn und ny und ist somit aus dem gleichen Grund für Szenarien $k \in \text{DT}$, $k \in \text{YN}$ und $k \in \text{NY}$ nicht möglich.

Für $k \in \mathrm{NN}$ ist nur der Fall zu betrachten, in dem sich noch *Second-Stage*-Kanten in den Teilgraphen $S_L$ und $S_R$ befinden (vgl. Abb. 5.9d und Zeile 5.52). Sonst wäre die Verbindung zu dem *First-Stage*-Teil der Lösung nicht möglich.




**(a)** $k \in \mathrm{ST}$: Zeile 5.45 **(b)** $k \in \mathrm{ST}$: Zeile 5.46 **(c)** $k \in \mathrm{ST}$: Zeile 5.47 **(d)** $k \in \mathrm{NN}$: Zeile 5.52

**(e)** $k \in \mathrm{DT}$: Zeile 5.48 **(f)** $k \in \mathrm{DT}$: Zeile 5.49 **(g)** $k \in \mathrm{YN}$: Zeile 5.50 **(h)** $k \in \mathrm{NY}$: Zeile 5.51

**Abbildung 5.9:** Fälle, die die Summen der Menge $\mathcal{S}_4^{\mathrm{nn}}$ für ein Szenario $k$ abdecken

### 5.1.4 Berechnung der Ausgabe

Wie im Algorithmus von Wald und Colbourn wird der Graph auf eine Kante $\{a, b\}$ reduziert, die aus den Halbkanten $(a, b) = \alpha$ und $(b, a)$ besteht. Der Algorithmus gibt dann das Minimum der folgenden Menge aus:

$$\{\mathrm{st}(\alpha)\} \cup \tag{5.53}$$

$$\{\mathrm{yn}_{\mathrm{ST,DT,YN}}(\alpha) \mid \mathrm{ST} \dot\cup \mathrm{YN} = \{1, \ldots, K\}, \mathrm{DT} = \emptyset\} \cup \tag{5.54}$$

$$\{\mathrm{ny}_{\mathrm{ST,DT,NY}}(\alpha) \mid \mathrm{ST} \dot\cup \mathrm{NY} = \{1, \ldots, K\}, \mathrm{DT} = \emptyset\} \cup \tag{5.55}$$

$$\{\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}(\alpha) \mid \mathrm{ST} \dot\cup \mathrm{YN} \dot\cup \mathrm{NY} \dot\cup \mathrm{NN} = \{1, \ldots, K\}, \mathrm{DT} = \emptyset\} \cup \tag{5.56}$$

$$\left\{ \sum_{k=1}^{K} \mathrm{var}_k(\alpha) \mid \forall k \in \{1, \ldots, K\} : \mathrm{var}_k(\alpha) = \min\{\mathrm{st}_k(\alpha), \mathrm{ny}_k(\alpha), \mathrm{yn}_k(\alpha), \mathrm{nn}_k(\alpha)\} \right\} \tag{5.57}$$

Auch in diesem Algorithmus kann das Berechnen der Variablen so modifiziert werden, dass die Wahl der Kanten gespeichert wird. Die Kantenmenge einer Variablen entspricht dann der Vereinigung der Kantenmengen der Variablen, aus denen sie hervorgeht. Dabei muss unterschieden werden, welche Kanten in der ersten und welche in der zweiten Phase

gewählt werden. Jede Variable erhält somit eine Menge von *First-Stage*-Kanten und für jedes Szenario eine Menge von *Second-Stage*-Kanten.

### 5.1.5 Analyse des Algorithmus

Die Analyse des Algorithmus geschieht analog zum Algorithmus von Wald und Colbourn in drei Schritten: Zuerst wird die Rechenzeit betrachtet, danach die Korrektheit auf 2-Bäumen und im letzten Schritt wird der Algorithmus aus Abschnitt 4.1 verwendet, um die Rechenzeit und die Korrektheit für das $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$ zu zeigen.

**Lemma 5.1.1.** *Für eine Instanz* $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ *für das SSTP, in der* $G$ *ein 2-Baum ist, hat der in diesem Abschnitt vorgestellte Algorithmus eine Laufzeit, die* $\Theta(12^K|V|)$ *ist.*

Beweis. Für die Variablen st und dt gibt es je eine Variable pro Halbkante. Für die Variablenklassen $\mathrm{yn}_{\mathrm{ST,DT,YN}}$ und $\mathrm{ny}_{\mathrm{ST,DT,NY}}$ hingegen gibt es so viele Variablen für jede Halbkante, wie es Auswahlen von Mengen ST, DT und $M \in \{\mathrm{YN}, \mathrm{NY}\}$ gibt, sodass $\mathrm{ST} \dot\cup \mathrm{DT} \dot\cup M = \{1, \ldots, K\}$ ist, also je $3^K$ Variablen. Ebenso lässt sich feststellen, dass es $5^K$ Variablen des nn-Typs pro Halbkante gibt. Bei der Initialisierung wird den Variablen jeder Halbkante jeweils ein Wert zugeordnet. Die Initialisierung jeder einzelnen Variable benötigt eine Laufzeit von $\Theta(K)$. Die fünf *Second-Stage*-Variablen für jedes Szenario und jede Halbkante machen dabei einen zusätzlichen Aufwand von $\mathcal{O}(K|V|)$ aus. Die Initialisierung hat somit eine Laufzeit von $\Theta(5^K K|V|)$.

Um die Variablen zu berechnen, müssen im Laufe des Algorithmus die Summen in den Mengen berechnet und ein Minimum bestimmt werden. Für die Variablen st und dt gibt es dabei zunächst 3 bzw. 2 Möglichkeiten und die Summen in $\mathcal{S}^{\mathrm{st}}$ bzw. $\mathcal{S}^{\mathrm{dt}}$. Es ist $|\mathcal{S}^{\mathrm{st}}| = |\mathcal{S}^{\mathrm{dt}}|$, da die Mengen die gleichen Fälle abdecken. Für jedes $k \in \{1, \ldots, K\}$ gibt es drei Möglichkeiten, wie die *Second-Stage*-Bäume zusammengesetzt werden. Damit ist $|\mathcal{S}^{\mathrm{st}}| = 3^K$.

Für eine Variable $\mathrm{yn}_{\mathrm{ST,DT,YN}}$ ist die Anzahl der Möglichkeiten, also die Kardinalitäten der Mengen $\mathcal{S}^{\mathrm{yn}}_1(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})$ und $\mathcal{S}^{\mathrm{yn}}_2(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})$, von den Mengen ST, DT und YN abhängig. Es ist

$$|\mathcal{S}^{\mathrm{yn}}_1(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})| = 2^{|\mathrm{ST}|}1^{|\mathrm{DT}|}1^{|\mathrm{YN}|}$$

und

$$|\mathcal{S}^{\mathrm{yn}}_2(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})| = 4^{|\mathrm{ST}|}3^{|\mathrm{DT}|}2^{|\mathrm{YN}|}.$$

Es gibt $\frac{K!}{a!b!c!}$ Möglichkeiten die $K$ Szenarien in Mengen aufzuteilen, die eine feste Größe von $|\mathrm{ST}| = a$, $|\mathrm{DT}| = b$ und $|\mathrm{YN}| = c$ besitzen. Wird nun über alle $|\mathcal{S}_1^{\mathrm{yn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})|$ summiert, ergibt sich:

$$\sum_{\mathrm{ST} \dot\cup \mathrm{DT} \dot\cup \mathrm{YN} = \{1,\dots,K\}} |\mathcal{S}_1^{\mathrm{yn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})| = \sum_{\mathrm{ST} \dot\cup \mathrm{DT} \dot\cup \mathrm{YN} = \{1,\dots,K\}} 2^{|\mathrm{ST}|} 1^{|\mathrm{DT}|} 1^{|\mathrm{YN}|}$$

$$= \sum_{a+b+c=K} \frac{K!}{a!b!c!} \cdot 2^a \cdot 1^b \cdot 1^c$$

$$= (2 + 1 + 1)^K = 4^K$$

Die vorletzte Identität folgt aus dem Multinomialtheorem. Somit erzeugen alle Mengen $\mathcal{S}_1^{\mathrm{yn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})$ zusammen $4^K$ Fälle. Durch die Mengen $\mathcal{S}_2^{\mathrm{yn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN})$ sind demnach zusammen $9^K$ Fälle zu unterscheiden. Für die Variable $\mathrm{ny}_{\mathrm{ST,DT,NY}}$ existieren aus Symmetriegründen die gleichen Fälle. Analog gibt es für die Mengen $\mathcal{S}_1^{\mathrm{nn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN}, \mathrm{NY}, \mathrm{NN})$, $\mathcal{S}_2^{\mathrm{nn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN}, \mathrm{NY}, \mathrm{NN})$ und $\mathcal{S}_3^{\mathrm{nn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN}, \mathrm{NY}, \mathrm{NN})$ jeweils zusammen $12^K$ und für die Mengen $\mathcal{S}_4^{\mathrm{nn}}(\mathrm{ST}, \mathrm{DT}, \mathrm{YN}, \mathrm{NY}, \mathrm{NN})$ zusammen $8^K$ Möglichkeiten.

Zur Berechnung der Ausgabe muss das Minimum aus $1 + 2 \cdot 2^K + 4^K + 4K$ Werten bestimmt werden, was eine Laufzeit von $\mathcal{O}(4^K)$ addiert. □

**Lemma 5.1.2.** *Der in diesem Abschnitt vorgestellte Algorithmus berechnet auf 2-Bäumen eine minimale Lösung für das* $SSTP_{fs,ss}^{\leq 2}$.

Beweis. Es muss gezeigt werden, dass die berechnete Lösung zulässig und minimal ist. Die *First-Stage*-Lösung, die der hier vorgestellte Algorithmus berechnet, ist ein Baum, da sie mit den gleichen Regeln wie im Algorithmus von Wald und Colbourn berechnet wird. Für jedes Szenario $k \in \{1, \dots, K\}$ ist durch die Fallunterscheidungen gesichert, dass die *Second-Stage*-Bäume mit dem *First-Stage*-Teil verbunden sind. Genauso sorgen die Fallunterscheidungen dafür, dass für jedes Szenario der Graph, der durch die Vereinigung aus *First-* und *Second-Stage*-Kanten induziert wird, keine Kreise enthält.

Ist bei der Initialisierung für ein Szenario $k$ ein Terminal $v \in T_k$ nicht in der *First-* oder *Second-Stage*-Lösung, so erhält diese Variable den Wert

$$N = \sum_{e \in E} \max \left\{ \sum_{k=1}^{K} p_k c_k(e), c_0(e) \right\} + 1.$$

Im Laufe des Algorithmus wird $N$ als Summand bei Variablen addiert, die mindestens eine Variable verwenden, welche für eine Lösung steht, die ein Terminalknoten in einem Szenario ausspart. Also haben alle Lösungen, die ein Terminal auslassen, mindestens Kosten $N$. Durch die Variablen st und dt können aber wie im Algorithmus von Wald und

Colbourn Spannbäume des Eingabegraphen getestet werden, die ausschließlich aus *First-Stage*-Kanten bestehen. Deren Kosten sind aber echt kleiner als $N$ und somit werden immer auch Lösungen gefunden, die alle Terminale in den einzelnen Szenarien verbinden.

Die Minimalität der Lösung lässt sich analog zu Lemma 4.2.2 zeigen. Dazu wird die zentrale Eigenschaft von 2-Bäumen verwendet, dass jede Kante $e = \{x, y\}$ ein 2-Separator des Graphen ist und dass die dadurch induzierten Komponenten wieder 2-Bäume sind [20]. Für jede Kante $e = \{x, y\}$ muss im Algorithmus entschieden werden, ob die Endpunkte in der ersten Phase, in der zweiten Phase oder gar nicht verwendet werden. In dem Algorithmus wird zuerst vollständig nach der ersten Phase unterschieden; dies geschieht genauso wie im Algorithmus von Wald und Colbourn. Die Fälle in der zweiten Phase sind dann Unterfälle der Fälle der ersten Phase.

Zunächst ist es möglich, beide Knoten in der ersten Phase zu wählen. Analog zu Lemma 4.2.2 kann nun differenziert werden, ob die Verbindung zwischen $x$ und $y$ in einer Komponente hergestellt wird – dafür steht die Variable st – oder ob der Zusammenhang in einer Komponente nicht hergestellt wird, was der Bedeutung der Variablen dt entspricht. Für die zweite Phase gibt es dabei keine Möglichkeiten, weil es keine weitere Verbindung zwischen $x$ und $y$ über die zweite Phase geben darf, da sonst ein Kreis geschlossen werden würde.

Wird nur einer der beiden Knoten in die erste Phase aufgenommen – was der Bedeutung der Variablen $\mathrm{yn}_{\mathrm{ST,DT,YN}}$ oder $\mathrm{ny}_{\mathrm{ST,DT,NY}}$ entspricht –, existieren für die erste Phase keine weiteren Fälle. Allerdings gibt es in der zweiten Phase jetzt zwei Möglichkeiten. Sei o. B. d. A. $x$ der Knoten, der in der ersten Phase verbunden wird. Es reicht diesen Fall für die Variable $\mathrm{yn}_{\mathrm{ST,DT,YN}}$ zu betrachten.

Entweder der Knoten $y$ wird in der zweiten Phase für ein Szenario $k$ angebunden oder $y$ kommt endgültig nicht mehr in der Lösung für dieses Szenario vor. Wird $y$ in einem Szenario $k \in \{1, \ldots, K\}$ nicht aufgenommen, so ist dieser Fall mit $k \in \mathrm{YN}$ abgedeckt. Wird $y$ für ein Szenario $k$ in die Lösung für die zweite Phase aufgenommen, so muss eine Verbindung in $G[E_0 \cup E_k]$ zwischen den Knoten $x$ und $y$ hergestellt werden. In der Komponente, in der die Verbindung hergestellt wird, kann dafür $k \in \mathrm{ST}$ gewählt werden, in den anderen $k \in \mathrm{DT}$.

Werden beide Knoten nicht in der ersten Phase gewählt, so ist zu unterscheiden, ob in einer Komponente keine Kante gewählt wird, oder ob sich alle Kanten der ersten Phase in einer Komponente befinden. Der Fall, in dem es optimal ist, in einer Komponente keine Kante in der ersten Phase zu wählen, wird durch die *Second-Stage*-Variablen nach Lemma 4.2.2 vollständig abgedeckt.

In dem anderen Fall befinden sich die *First-Stage*-Kanten vollständig in einer Komponente. Dieser Fall wird durch die Variablen $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}$ abgedeckt. Es muss dann zwischen den Fällen unterschieden werden, die sich daraus ergeben, welche Teilmenge der Knoten $x$ und $y$ in der zweiten Phase in welchem Szenario gewählt wird. Es entstehen für

die zweite Phase die gleichen Fälle wie im Algorithmus von Wald und Colbourn mit der Einschränkung, dass für jedes Szenario eine Verbindung zu $G[E_0]$ existieren muss.

Werden beide Knoten für ein Szenario $k$ gewählt, so müssen diese in einer Komponente verbunden werden; für diese kann $k \in \mathrm{ST}$ gewählt werden. In den anderen Komponenten werden die Knoten nicht verbunden, wofür $k \in \mathrm{DT}$ steht. Wird nur einer der beiden Knoten $x$ und $y$ gewählt, so werden diese Fälle durch $k \in \mathrm{YN}$ und $k \in \mathrm{NY}$ abgedeckt.

Eine Ausnahme stellt der Fall dar, in dem keiner der beiden Knoten in der zweiten Phase für das Szenario $k$ in einem Baum vorkommt. Ursprünglich waren die beiden Fälle zu unterscheiden, ob sich die Kanten vollständig in einer Komponente befinden, oder keine Kante in einer Komponente ist. Da $G[E_0 \cup E_k]$ zusammenhängend sein muss, dürfen sich die Kanten, die in der zweiten Phase gewählt werden, nur in der gleichen Komponente befinden wie die Kanten von $G[E_0]$. Andernfalls wäre eine Verbindung nur über die Knoten $x$ und $y$ möglich, die aber in diesem Fall weder in der ersten noch in der zweiten Phase gewählt werden. Dieser letzte Fall wird durch die Wahl von $k \in \mathrm{NN}$ abgedeckt.

Die Fallunterscheidung ist somit vollständig und es wird sichergestellt, dass ein Knoten in allen Komponenten entweder in der ersten Phase, in der zweiten Phase oder gar nicht in der Lösung vorkommt.

Zum Schluss wird die Ausgabe berechnet, indem das Minimum aller Variablen bestimmt wird, die für eine zulässige Lösung stehen. In der ersten Phase ist die Variable dt nicht zulässig, weil andernfalls die *First-Stage*-Teillösung nicht zusammenhängend wäre. In der zweiten Phase ist es für die Variablen $\mathrm{yn}_{\mathrm{ST,DT,YN}}$ und $\mathrm{ny}_{\mathrm{ST,DT,NY}}$ nur zulässig die Mengen $\mathrm{DT}= \emptyset$ zu wählen, da sonst für ein $k \in \mathrm{DT}$ die *Second-Stage*-Teillösung dieses Szenarios nicht zusammenhängend wäre oder keine Verbindung zur *First-Stage*-Teillösung bestünde. Dies gilt für die Variable $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}$ analog.

Den Fall, in dem es optimal ist, keine *First-Stage*-Kanten zu wählen, deckt Zeile 5.57 ab. Hier wird das gleiche Ergebnis berechnet, als wäre der Algorithmus von Wald und Colbourn für jedes Szenario einmal mit dessen gewichteter Kostenfunktion und den Terminalen aus diesem Szenario aufgerufen worden: Es wird für jedes Szenario $k$ gewählt, welche der Variablen $\mathrm{st}_k(\alpha)$, $\mathrm{ny}_k(\alpha)$, $\mathrm{yn}_k(\alpha)$ und $\mathrm{nn}_k(\alpha)$ den kleinsten Wert hat und diese kleinsten Kosten addiert. Dies ist zulässig, da die Szenarien erst mit der Wahl von *First-Stage*-Kanten voneinander abhängig werden. □

**Proposition 5.1.3.** *Für eine Instanz* $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \le k \le K})$ *für das* $SSTP^{\le 2}_{fs,ss}$ *berechnet der in diesem Abschnitt vorgestellte Algorithmus eine minimale Lösung. Seine Laufzeit ist* $\Theta(12^K|V|)$.

Beweis. Wie im Algorithmus von Wald und Colbourn kann der Algorithmus aus Abschnitt 4.1 dafür genutzt werden, einen serien-parallelen Graphen zu einem 2-Baum zu erweitern. Dabei erhalten die hinzugefügten Kanten Kosten

$$N = \sum_{e \in E} \max \left\{ \sum_{k=1}^{K} p_k c_k(e), c_0(e) \right\} + 1.$$

Die Argumentation verläuft damit genau wie in Theorem 4.2.3: In einem zusammenhängenden Graphen existiert immer eine Lösung für das $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$. Die Kosten dieser Lösung sind nach oben beschränkt durch die Kosten eines Spannbaumes für $G$, in dem für jede Kante das Maximum aus den erwarteten Kosten für die zweite Phase oder den Kosten für die erste Phase gewählt wird. Die Kosten dieses Spannbaumes sind aber echt kleiner als $N$ und somit wird dieser Algorithmus niemals Kanten auswählen, die durch den Algorithmus aus Abschnitt 4.1.1 hinzugekommen sind.

Durch Lemma 5.1.1 wird die Laufzeit gezeigt. Die Rechenzeit des Algorithmus zur Erweiterung des SP-Graphen zu einem 2-Baum aus Proposition 4.1.1 fällt dabei nicht ins Gewicht. Mit Lemma 5.1.2 folgt somit die Behauptung. □

## 5.2 Andere Varianten

Für die beiden Varianten $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs}}$ und $\mathrm{SSTP}^{\leq 2}_{\mathrm{ss}}$ existieren ähnliche Algorithmen wie der oben vorgestellte. In diesem Abschnitt wird kurz skizziert, welche Schwierigkeiten bei diesen Varianten auftreten und wie sie gelöst werden können.

### 5.2.1 Die Variante $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs}}$

Für die Variante $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs}}$ ist die *First-Stage*-Teillösung weiterhin zusammenhängend; für jedes Szenario darf aber die Lösung aus *Second-Stage*-Kanten auch disjunkt von der *First-Stage*-Lösung sein. Ist das für ein Szenario $k \in \{1, \ldots, K\}$ der Fall, so darf dieses Szenario in den *First-Stage*-Variablen nicht mehr hinzugenommen werden (vgl. Abb. 5.10). Dazu erhält jede *First-Stage*-Variable (auch die Variablen st und dt) eine Menge NONE, die alle Szenarien enthält, welche in der Komponente keine *Second-Stage*-Kanten besitzen dürfen.

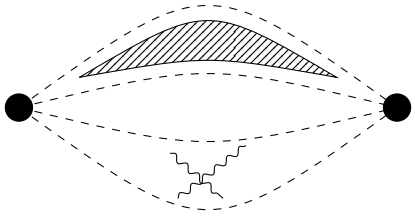


**Abbildung 5.10:** Die Szenarienmenge NONE wird zu jeder Variablen hinzugefügt.

Verwendet die *First-Stage*-Teillösung beide Endknoten einer Kante, so kann es ein Szenario $k$ geben, für das die folgenden Bedingungen gelten: Die *Second-Stage*-Kanten befinden sich ebenfalls in der Komponente der *First-Stage*-Teillösung und es gibt keine Verbindung zwischen der *First-Stage*- und der *Second-Stage*-Teillösung (vgl. Abb. 5.11). Dann erhalten die Variablen st und dt jeweils eine Menge NN. In allen anderen Variablen muss dieses Szenario in NONE enthalten sein, da für dieses Szenario im gesamten restlichen Graphen keine Kanten mehr gewählt werden darf.

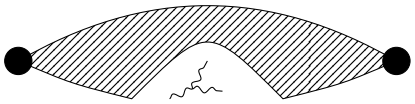


**Abbildung 5.11:** Die Szenarienmenge NN wird zu den Variablen st und dt hinzugefügt.

Wird aber nur einer der Knoten einer Kante in der ersten Phase verwendet, kann ein Szenario den anderen Knoten verwenden um eine Verbindung zwischen den Komponenten herzustellen (vgl. Abb. 5.12). Hat diese Szenarienlösung keine Verbindung zur Teillösung der ersten Phase, so muss sichergestellt werden, dass auch weiterhin die Kanten der ersten Phase von den Kanten dieses Szenarios unberührt bleiben. Aus diesem Grund erhält die Variable yn eine Menge NY und die Variable ny eine Menge YN.

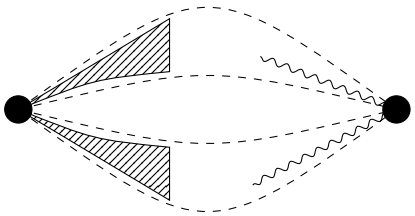


**Abbildung 5.12:** Die Szenarienmenge NY wird zu den Variablen yn hinzugefügt

Zu den Variablen yn und ny wird auch eine Menge NN hinzugefügt, weil für ein Szenario der Fall möglich ist, in dem sich die *Second-Stage*-Kanten in der Komponente der *First-Stage*-Teillösung befinden, diese aber nicht berühren.

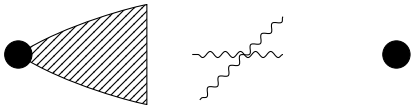


**Abbildung 5.13:** Die Szenarienmenge NN wird zu den Variablen yn hinzugefügt

Somit ergeben sich für eine Halbkante $\alpha$ die Variablen

- $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$,
- $\mathrm{dt}_{\mathrm{DT,NN,NONE}}(\alpha)$,

- $\text{yn}_{\text{ST,DT,YN,NY,NN,NONE}}(\alpha)$ und $\text{ny}_{\text{ST,DT,YN,NY,NN,NONE}}(\alpha)$ sowie
- $\text{nn}_{\text{ST,DT,YN,NY,NN,NONE}}(\alpha)$.

Die *Second-Stage*-Variablen werden wie im Algorithmus aus Abschnitt 5.1 behandelt. Die Initialisierung, das Zusammensetzen und die Ausgabe erfolgen ebenfalls analog zu diesem Algorithmus.

Mit den Argumenten aus Abschnitt 5.1.5 kann für das $\text{SSTP}_{\text{fs}}^{\leq 2}$ die folgende Proposition bewiesen werden:

**Proposition 5.2.1.** *Für das $SSTP_{fs}^{\leq 2}$ mit Instanzen $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ existiert ein Algorithmus, der eine minimale Lösung in Laufzeit $\Theta(16^K |V|)$ berechnet.*

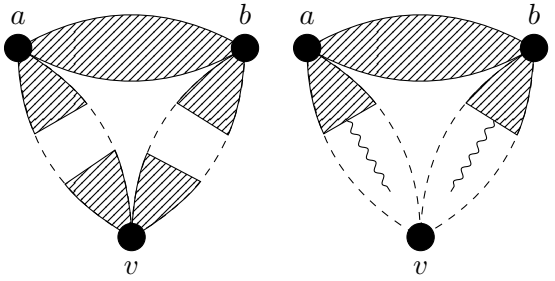
### 5.2.2 Die Variante $\text{SSTP}_{\text{ss}}^{\leq 2}$

In der Variante $\text{SSTP}_{\text{ss}}^{\leq 2}$ soll für jedes Szenario $k \in \{1, \ldots, K\}$ der Graph $G[E_0 \cup E_k]$ zusammenhängend sein. Die *First-Stage*-Kanten dürfen aber an jeder Stelle der *Second-Stage*-Teillösungen auftauchen, sofern sie mit allen Szenarienlösungen verbunden sind. Die Fälle aus der Variante $\text{SSTP}_{\text{fs}}^{\leq 2}$, in denen gesichert werden muss, dass die *First-Stage*-Teillösung keine Verbindung zu den *Second-Stage*-Kanten hat, sind hier demnach nicht nötig. Auch darf sich eine *Second-Stage*-Teillösung nicht in einer von einem 2-Separator $\{x, y\}$ induzierten Komponente ohne Verbindung zu den Knoten $x$ und $y$ oder einem *First-Stage*-Fragment befinden.

Schwierigkeiten entstehen allerdings an den Stellen, an denen die Fälle der ersten Phase unterschieden werden müssen. Beispielsweise darf die *First-Stage*-Teillösung für die Variable st nun zusätzlich eine von der restlichen *First-Stage*-Lösung disjunkte Komponente besitzen, die den Knoten $v$ enthält (vgl. Abb. 5.14a). In der zweiten Phase gibt es dazu die Einschränkung, dass diese *First-Stage*-Komponenten in jedem Szenario verbunden sein müssen.

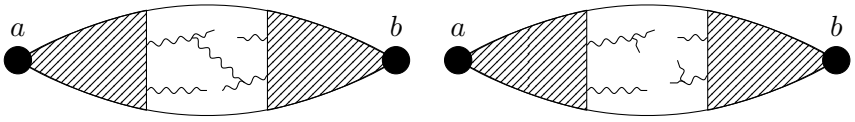
Das macht den Fall aus dem Algorithmus von Wald und Colbourn, in dem der Knoten $v$ ausgelassen wird, aber nicht überflüssig, da es günstiger sein kann, den Knoten $v$ in allen Szenarien auszusparen (vgl. Abb. 5.14b).

Die Variable dt erhält wieder die Interpretation, dass beide Endpunkte der Halbkante in der *First-Stage*-Teillösung enthalten sein müssen, es aber in dieser Komponente keine Verbindung alleine über *First-Stage*-Kanten zwischen ihnen geben darf. Für die zweite Phase ist nun aber zu unterscheiden, ob ein Szenario die disjunkten *First-Stage*-Fragmente miteinander verbindet oder nicht. In der Variante $\text{SSTP}_{\text{fs,ss}}^{\leq 2}$ ist dies nicht nötig, da in dieser die *First-Stage*-Teile später über *First-Stage*-Kanten verbunden werden müssen. In dem Algorithmus für diese Variante erhält die Variable dt daher die Mengen ST und DT (vgl. Abb. 5.15). Zudem ist analog zur Variablen st auch ein von der restlichen *First-Stage*-Teillösung disjunktes Fragment um den Knoten $v$ erlaubt, sodass ebenfalls ein zusätzlicher Fall für die *First-Stage*-Teillösung entsteht.




**(a)** **(b)**

**Abbildung 5.14:** (a) Ein neuer Fall für die Variable st (b) Es muss immer noch möglich sein, den Knoten $v$ in jedem Szenario auszusparen.




**(a)** $k \in$ ST **(b)** $k \in$ DT

**Abbildung 5.15:** Die Mengen ST und DT bei der Variablen dt für ein Szenario $k$

Für die Variable yn (und ny analog) kommt ebenfalls ein Fall hinzu, der ein disjunktes *First-Stage*-Fragment erlaubt, welches den Knoten $v$ enthält (vgl. Abb. 5.16). In der zweiten Phase sind dabei in jedem Szenario diese Fragmente miteinander zu verbinden.

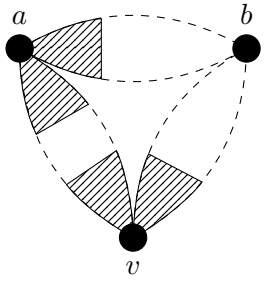



**Abbildung 5.16:** Ein neuer Fall für die Variable yn

Für die Variable nn sind in der zweiten Phase genau die Fälle zutreffend, die schon im Algorithmus aus Abschnitt 5.1 für die Variable $\text{nn}_{\text{ST,DT,YN,NY,NN}}$ diskutiert wurden. Allerdings ergeben sich für die erste Phase neun Möglichkeiten: Bezogen auf Abb. 4.3 und die üblichen Bezeichnungen kann sich je ein *First-Stage*-Fragment in den Teilgraphen $S_M$, $S_L$ und $S_R$ befinden und auch alle Kombinationen sind möglich. Zusätzlich darf sich ein *First-Stage*-Fragment um den Knoten $v$ befinden. Die Fälle sind in Abb. 5.17 dargestellt. Die Möglichkeiten in der zweiten Phase sind jeweils so eingeschränkt, dass die einzelnen *First-Stage*-Fragmente verbunden sein müssen.

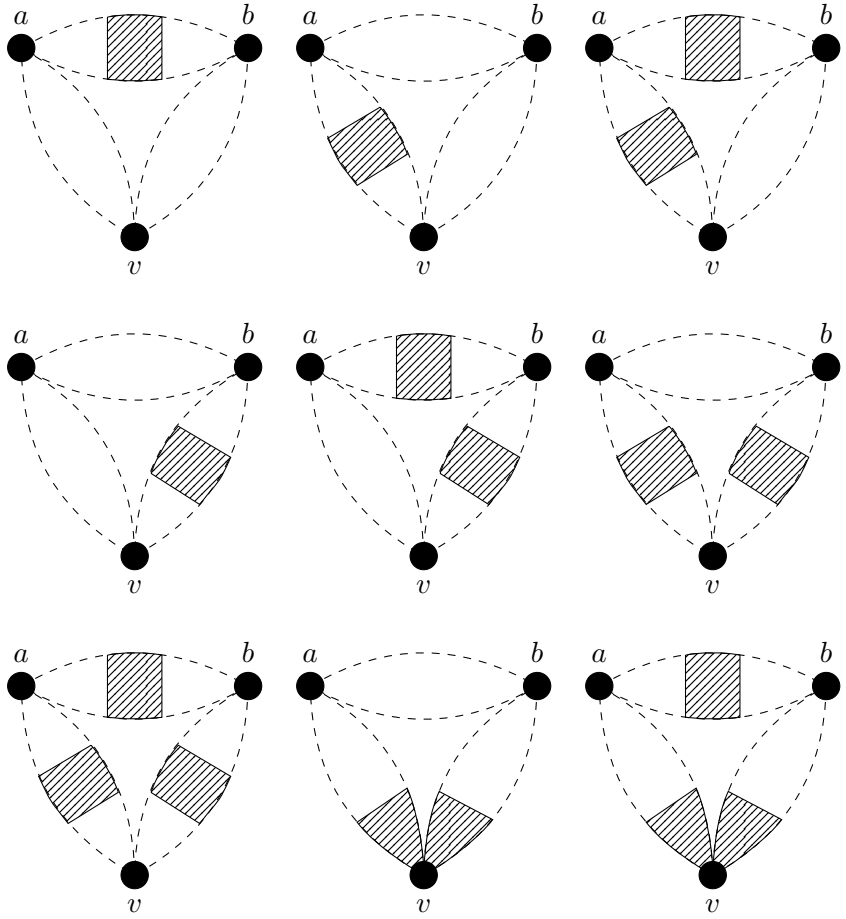



**Abbildung 5.17:** Alle Fälle für die Variable nn

Der Algorithmus für diese Variante nutzt also für eine Halbkante $\alpha$ die Variablen

- $\mathrm{st}(\alpha)$,
- $\mathrm{dt}_{\mathrm{ST,DT}}(\alpha)$,
- $\mathrm{yn}_{\mathrm{ST,DT,YN}}(\alpha)$ und $\mathrm{ny}_{\mathrm{ST,DT,NY}}(\alpha)$ sowie
- $\mathrm{nn}_{\mathrm{ST,DT,YN,NY,NN}}(\alpha)$.

Die *Second-Stage*-Variablen werden wie in dem Algorithmus für das $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$ behandelt, ebenso die Initialisierung und das Zusammensetzen der *First-Stage*-Variablen sowie die Ausgabe des Algorithmus.

Die Freiheiten für die erste Phase und die Bedingung, dass jedes Szenario mit den *First-Stage*-Fragmenten verbunden sein muss, lassen weniger Freiheiten für die zweite Phase. Für diese werden somit die Möglichkeiten im Vergleich zu dem Algorithmus aus Abschnitt 5.1 zumeist eingeschränkt. Die zusätzlichen Fälle in der ersten Phase verändern dabei nur konstante Faktoren. So ist für das $\mathrm{SSTP}^{\leq 2}_{\mathrm{ss}}$ die folgende Proposition nachweisbar:

**Proposition 5.2.2.** *Für das $SSTP_{ss}^{\leq 2}$ mit Instanzen $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ existiert ein Algorithmus, der eine minimale Lösung in Laufzeit $\Theta(12^K|V|)$ berechnet.*

## 5.3 Algorithmus für das SSTP$^{\leq 2}$

Der Algorithmus für das SSTP$^{\leq 2}$ kombiniert die Erkenntnisse aus den vorigen Abschnitten. Es sind keine *Second-* und nur zwei *First-Stage*-Variablen nötig. Für die Berechnung einer minimalen Lösung für das SSTP$^{\leq 2}$ gibt es für jede Halbkante $\alpha$ und

- für alle Mengen ST, NN, NONE $\subseteq \{1, \ldots, K\}$ mit ST $\dot\cup$ NN $\dot\cup$ NONE $= \{1, \ldots, K\}$ die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ und
- für alle Mengen ST, DT, YN, NY, NN, NONE $\subseteq \{1, \ldots, K\}$ mit ST $\dot\cup$ DT $\dot\cup$ YN $\dot\cup$ NY $\dot\cup$ NN $\dot\cup$ NONE $= \{1, \ldots, K\}$ die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha)$.

### 5.3.1 Beschreibung der Variablen

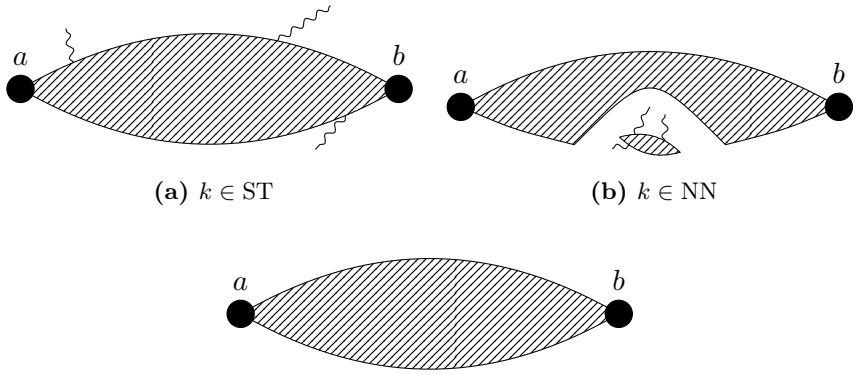
Die *First-Stage*-Variablen beschreiben weiterhin die Fälle, die in der ersten Phase auftreten können. Für eine SSTP$^{\leq 2}$-Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ sei $G$ ein 2-Baum, $\alpha = (a, b) \in E$ und $S_\alpha$ der im Verlauf des Algorithmus auf $\alpha$ reduzierte Teilgraph. In einer Lösung für das SSTP$^{\leq 2}$ ist die *First-Stage*-Teillösung $G[E_0]$ nicht notwendigerweise zusammenhängend; die Menge der Komponenten von $G[E_0]$ sei daher mit $\mathcal{C} = \{C_0, \ldots, C_r\}$ bezeichnet.

**$\mathrm{st}_{\mathrm{ST,NN,NONE}}$** In der Variablen $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ wird der Wert einer günstigsten Lösung (bestehend aus *First-Stage*-Kanten $E_0$ und *Second-Stage*-Kanten $E_k$ für jedes Szenario $k$) in $S_\alpha$ für das SSTP$^{\leq 2}$ gespeichert, in der die Knoten $a$ und $b$ durch *First-Stage*-Kanten verbunden sind. An die *Second-Stage*-Kanten werden je nach der Zusammensetzung der Mengen ST, NN und NONE weitere Bedingungen gestellt. Sei dazu o. B. d. A. $C_0$ die *First-Stage*-Komponente, die die Knoten $a$ und $b$ enthält.

- Für alle Szenarien $k \in$ ST mit $E_k \neq \emptyset$ muss $V(C_0) \cap V(G[E_k]) \neq \emptyset$ gelten.
- Für alle Szenarien $k \in$ NN muss $V(C_0) \cap V(G[E_k]) = \emptyset$ gelten.
- Für $k \in$ NONE ist $E_k = \emptyset$.

Die Fälle sind in Abb. 5.18 illustriert.

**$\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}$** Die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha)$ speichert den Wert einer günstigsten Lösung für $S_\alpha$, in der die Knoten $a$ und $b$ nicht in der gleichen Komponente von $G[E_0]$ enthalten sind. Hierbei ist es erlaubt, dass $a$ oder $b$ in keinem $C_i \in \mathcal{C}$ vorkommen. Dies ist möglich, wenn für die Halbkanten aller zu $a$ (bzw. $b$) inzidenten Kanten die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}$ gewählt wird. Sind die Knoten




**(a)** $k \in \mathrm{ST}$ **(b)** $k \in \mathrm{NN}$

**(c)** $k \in \mathrm{NONE}$

**Abbildung 5.18:** Die Unterfälle für die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}((a,b))$ für ein Szenario $k$: Der schraffierte Teil stellt *First-Stage*-Teile dar, während die geschlängelten Linien mögliche *Second-Stage*-Fragmente zeigen.

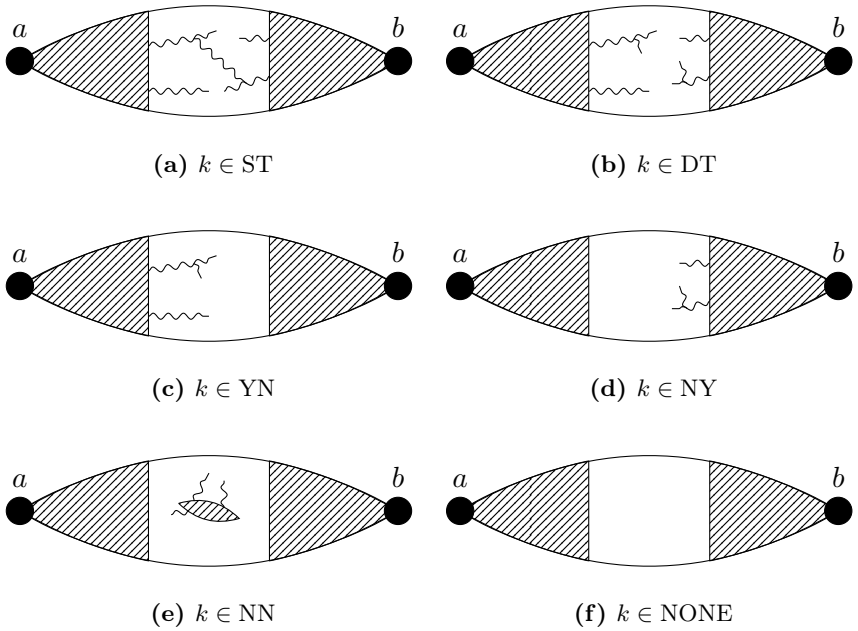
in einer eigenen *First-Stage*-Komponente, so sei o. B. d. A. $a \in V(C_0)$ und $b \in V(C_1)$. Gibt es kein $C_i \in \mathcal{C}$ mit $a \in C_i$, so sei mit $C_0$ der Graph benannt, der nur den Knoten $a$ enthält und analog $C_1$ für $b$. Zudem seien die Komponenten in $\mathcal{C}$ entsprechend umbenannt. Anhand der Mengen ST, DT, YN, NY, NN und NONE werden dabei weitere Bedingungen an diese Lösung gestellt:

- Für alle Szenarien $k \in \mathrm{ST}$ muss ein Weg in $G[E_0 \cup E_k]$ existieren, der die Knoten $a$ und $b$ miteinander verbindet.
- Für alle Szenarien $k \in \mathrm{DT}$ mit $E_k \neq \emptyset$ muss $(V(C_0) \cup V(C_1)) \cap V(G[E_k]) \neq \emptyset$ gelten und es darf keinen Weg in $G[E_0 \cup E_k]$ geben, der die Knoten $a$ und $b$ miteinander verbindet.
- Für alle Szenarien $k \in \mathrm{YN}$ mit $E_k \neq \emptyset$ muss $V(C_0) \cap V(G[E_k]) \neq \emptyset$ und $V(C_1) \cap V(G[E_k]) = \emptyset$ gelten.
- Für alle Szenarien $k \in \mathrm{NY}$ mit $E_k \neq \emptyset$ muss $V(C_0) \cap V(G[E_k]) = \emptyset$ und $V(C_1) \cap V(G[E_k]) \neq \emptyset$ gelten.
- Für alle Szenarien $k \in \mathrm{NN}$ muss $(V(C_0) \cup V(C_1)) \cap V(G[E_k]) = \emptyset$ sein.
- Für $k \in \mathrm{NONE}$ ist $E_k = \emptyset$.

Die Fälle für die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha)$ sind in Abb. 5.19 dargestellt.

### 5.3.2 Initialisierung der Variablen

Die Variablen werden ähnlich wie im Algorithmus aus Abschnitt 5.1 initialisiert. Sei für eine SSTP$^{\leq 2}$-Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ der SP-Graph $G$ ein 2-Baum und $\alpha = (a, b) \in E$.




**(a)** $k \in \mathrm{ST}$ **(b)** $k \in \mathrm{DT}$

**(c)** $k \in \mathrm{YN}$ **(d)** $k \in \mathrm{NY}$

**(e)** $k \in \mathrm{NN}$ **(f)** $k \in \mathrm{NONE}$

**Abbildung 5.19:** Die Unterfälle für die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}((a,b))$ für ein Szenario $k$

Bei der Variablen $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ und für Szenarien $k \in \mathrm{NN}$ oder $k \in \mathrm{NONE}$ darf der Graph $G[E_k]$ keine Verbindung zu der *First-Stage*-Komponente $C_0$ haben, die die Knoten $a$ und $b$ enthält. In diesem Fall werden die Knoten $a$ und $b$ im Szenario $k$ endgültig nicht mehr erreicht. Analog gilt für die Halbkante $\alpha$, aus der $C_0$ in diesem Fall besteht, dass die Knoten $a$ und $b$ endgültig nicht mehr in der Lösung für ein Szenario $k$ vorkommen, wenn $k \in \mathrm{NN}$ oder $k \in \mathrm{NONE}$ gewählt wird. Dann erhält die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ den Wert

$$N := \sum_{e \in E} \max \left\{ \sum_{k=1}^{K} p_k c_k(e), c_0(e) \right\} + 1.$$

Ansonsten werden die Variablen $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ für jede Auswahl an Mengen $\mathrm{ST} \dot\cup \mathrm{NN} \dot\cup \mathrm{NONE} = \{1, \ldots, K\}$ mit $c_0(\alpha)$ initialisiert, da die einzige Möglichkeit, die Knoten $a$ und $b$ in der ersten Phase miteinander zu verbinden, darin besteht, die Halbkante $\alpha$ in der ersten Phase zu wählen.

Für die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha)$ darf die Halbkante $\alpha$ nicht in der ersten Phase gewählt werden. Lässt die durch die Variable beschriebene Lösung ein Terminal in der zweiten Phase aus, so erhält die Variable den Wert $N$. Dies kann vorkommen, wenn für ein Szenario $k \in \mathrm{YN}$ der Knoten $b$ ein Terminalknoten ist oder für $k \in \mathrm{NY}$ der Knoten $a \in T_k$ ist oder für ein Szenario $k \in \mathrm{NN} \cup \mathrm{NONE}$ einer der Knoten $a$ oder $b$ ein Terminal in Szenario $k$ ist. Ansonsten erhält die Variable die Kosten $\sum_{k \in \mathrm{ST}} p_k c_k(\alpha)$, da für alle Szenarien in ST eine Verbindung zwischen $a$ und $b$ in der Lösung vorhanden sein muss und diese nur durch das Wählen von $\alpha$ möglich ist. Für Szenarien in DT darf die Halbkante $\alpha$

nicht gewählt werden, da sonst eine Verbindung zwischen $a$ und $b$ bestünde. Szenarien in den Mengen YN, NY, NN und NONE erzeugen keine weiteren Kosten.

**Algorithmus 5.3** Initialisierung der Variablen im Algorithmus für das SSTP$^{\leq 2}$

**Eingabe:** eine Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ und die zu initialisierende Halbkante $\alpha = (a, b)$

1: **for all** ST $\dot\cup$ NN $\dot\cup$ NONE $= \{1, \ldots, K\}$ **do**

2: $$\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha) \leftarrow \begin{cases} N & \exists k \in \mathrm{NN} \cup \mathrm{NONE} : a \in T_k \vee b \in T_k \\ c_0(\alpha) & \text{sonst} \end{cases}$$

3: **for all** ST $\dot\cup$ DT $\dot\cup$ YN $\dot\cup$ NY $\dot\cup$ NN $\dot\cup$ NONE $= \{1, \ldots, K\}$ **do**

4: $$\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha) \leftarrow \begin{cases} & \exists k \in \mathrm{YN} : b \in T_k \vee \\ N & \exists k \in \mathrm{NY} : a \in T_k \vee \\ & \exists k \in \mathrm{NN} \cup \mathrm{NONE} : a \in T_k \vee b \in T_k \\ \sum_{k \in \mathrm{ST}} p_k c_k(\alpha) & \text{sonst} \end{cases}$$

### 5.3.3 Zusammensetzen der Variablen

Die Gleichungen, die für das Berechnen der Variablen in jedem Schritt nötig sind, werden in diesem Abschnitt beschrieben. Sei dafür eine Situation wie in Abb. 4.3 gegeben. Die Bezeichnungen entsprechen denen aus den vorherigen Abschnitten: $M$, $L$ und $R$ bezeichnen die Halbkanten $(a, b)$, $(a, v)$ und $(v, b)$. Die auf diese Kanten bereits reduzierten Teilgraphen seien $S_M$, $S_L$ und $S_R$.

#### Die Variable st

Die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}(M)$ enthält den Wert einer günstigsten Lösung, in der zwischen den Knoten $a$ und $b$ ein Weg in $G[E_0]$ existiert. Im Gegensatz zu der ausführlich beschriebenen Variante für das SSTP$^{\leq 2}_{\mathrm{fs,ss}}$ sind jetzt andere Fälle für die Wahl von $E_0$ möglich. In Abb. 5.20 sind diese illustriert. Die Fälle in den Abb. 5.20a – 5.20c sind bereits aus dem Algorithmus von Wald und Colbourn bekannt.

Sei $\mathcal{C} = \{C_0, \ldots C_r\}$ die Menge der Komponenten von $G[E_0]$ und o. B. d. A. ist $C_0$ die Komponente, die die Knoten $a$ und $b$ enthält. Analog zu der Variante aus dem SSTP$^{\leq 2}_{\mathrm{ss}}$ ist es in dieser Variante möglich, dass eine *First-Stage*-Komponente $C_v \in \mathcal{C}$ mit $C_v \neq C_0$ den Knoten $v$ enthält (vgl. Abb. 5.20d). Es muss allerdings nicht wie bei der Problemvariante SSTP$^{\leq 2}_{\mathrm{ss}}$ dieser Fall von dem Fall unterschieden werden, in dem kein Szenario den Knoten $v$ verwendet. Dies lässt sich dadurch begründen, dass es im SSTP$^{\leq 2}$ auch Szenarien geben darf, welche die Komponente $C_v$ nicht verwenden. Dieser Fall deckt auch gleichzeitig die

Möglichkeit ab, dass $v$ nicht in $G[E_0]$ ist, indem für alle Halbkanten der Kanten, die zu $v$ inzident sind, die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}$ gewählt wird.

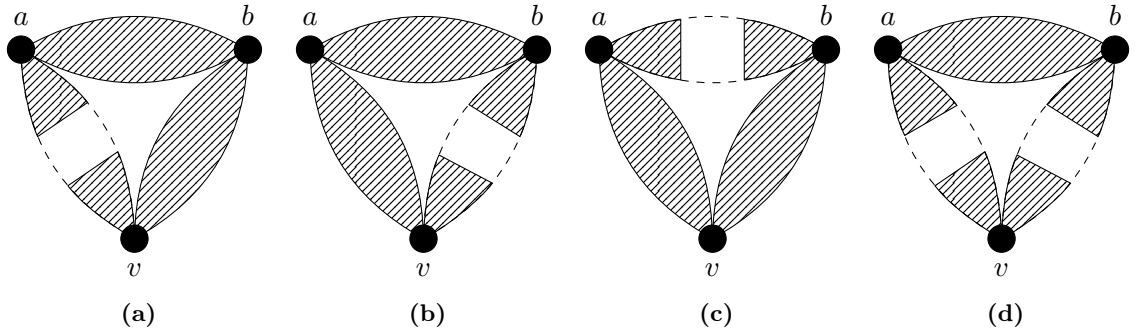



**Abbildung 5.20:** Die Fälle für die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}((a,b))$ in der ersten Phase: Der Fall (d) ist neu hinzugekommen.

Somit lässt sich der Wert der Variablen $\mathrm{st}_{\mathrm{ST,NN,NONE}}(M)$ durch vier Mengen $\mathcal{S}_i^{\mathrm{st}}$ bestimmen, die die Fallunterscheidung in der zweiten Phase beschreiben:

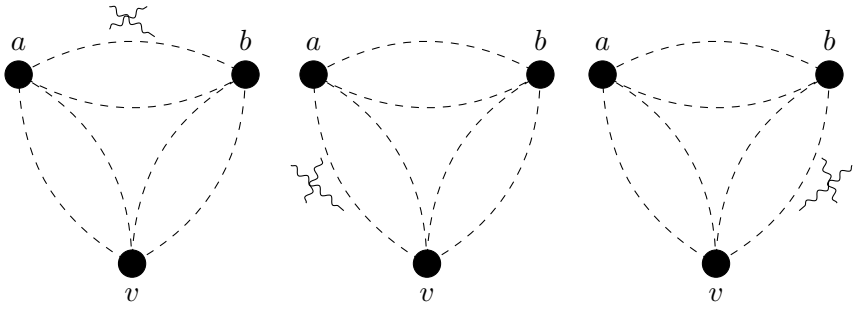
$$\mathrm{st}_{\mathrm{ST,NN,NONE}}(M) \leftarrow \min \bigcup_{i=1}^{4} \mathcal{S}_i^{\mathrm{st}}(\mathrm{ST,NN,NONE})$$

Die erste dieser Mengen $\mathcal{S}_1^{\mathrm{st}}$ beschreibt den Fall aus Abb. 5.20a. Die Existenz des Weges zwischen $a$ und $b$ wird durch die *First-Stage*-Kanten in $S_M$ gesichert. Zusätzlich sind die Knoten $b$ und $v$ über *First-Stage*-Kanten in $S_R$ miteinander verbunden. Im Teilgraphen $S_L$ ist hingegen keine Verbindung zwischen den Knoten $a$ und $v$ erlaubt. Damit hat diese Menge die Form

$$\begin{aligned}
\mathcal{S}_1^{\mathrm{st}}(\mathrm{ST,NN,NONE}) := \{&\mathrm{st}_{\mathrm{ST}_M,\mathrm{NN}_M,\mathrm{NONE}_M}(M) + \mathrm{dt}_{\mathrm{ST}_L,\mathrm{DT}_L,\mathrm{YN}_L,\mathrm{NY}_L,\mathrm{NN}_L,\mathrm{NONE}_L}(L) + \\
&\mathrm{st}_{\mathrm{ST}_R,\mathrm{NN}_R,\mathrm{NONE}_R}(R) \mid \\
&\forall k \in \mathrm{ST}: \\
&\quad k \in \mathrm{ST}_M \wedge k \in \mathrm{DT}_L \wedge k \in \mathrm{ST}_R, && (5.58) \\
&\forall k \in \mathrm{NN}: \\
&\quad k \in \mathrm{NN}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NONE}_R \vee && (5.59) \\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NN}_L \wedge k \in \mathrm{NONE}_R \vee && (5.60) \\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NN}_R, && (5.61) \\
&\forall k \in \mathrm{NONE}: \\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NONE}_R\}. && (5.62)
\end{aligned}$$

Die Fälle, die in den Zeilen 5.59 – 5.61 beschrieben sind, existieren für alle Fallunterscheidungen der zweiten Phase in diesem Algorithmus. Sie stehen für die Möglichkeiten, dass

sich die Kanten aus $E_k$ für das Szenario $k$ vollständig in dem Teilgraphen $S_M$, $S_L$ oder $S_R$ befinden und $G[E_k]$ keine Verbindung über $G[E_0 \cup E_k]$ zu einem der Knoten $a$, $b$ oder $v$ besitzt (vgl. Abb. 5.21). Auch der Fall, der in Zeile 5.62 beschrieben ist, existiert für alle Fallunterscheidungen: Für $k \in$ NONE darf in keinem Teilgraph $S_M$, $S_L$ oder $S_R$ eine *Second-Stage*-Kante gewählt werden.

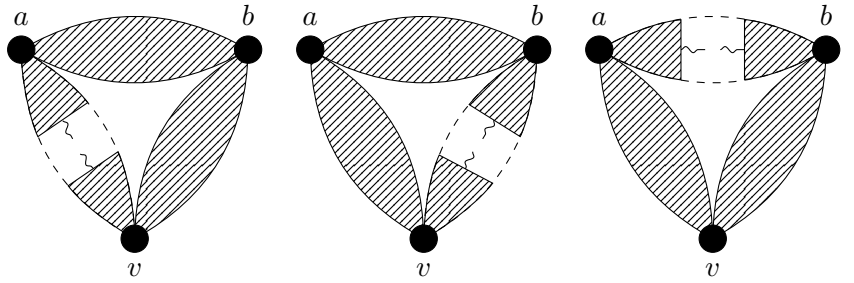



**(a)** $k \in \mathrm{NN}_M$ **(b)** $k \in \mathrm{NN}_L$ **(c)** $k \in \mathrm{NN}_R$

**Abbildung 5.21:** Die Fälle für ein Szenario $k \in$ NN für alle Mengen $\mathcal{S}$ in diesem Algorithmus: Die *Second-Stage*-Fragmente (geschlängelt dargestellt) befinden sich jeweils in einem der Teilgraphen $S_M$, $S_L$ oder $S_R$. Dies ist unabhängig von der Lage der *First-Stage*-Kanten.

Der einzige Fall, der für diesen *First-Stage*-Fall spezifisch ist, wird von Zeile 5.58 abgedeckt: Die Kanten der Szenarien $k \in$ ST müssen eine Verbindung zu $C_0$ haben. Dies geschieht, indem sie in jedem Teilgraphen $S_M$, $S_L$ und $S_R$ eine Verbindung zu den jeweiligen *First-Stage*-Komponenten besitzen, die einen der Knoten $a$, $b$ oder $v$ enthalten. Dabei darf in $S_L$ kein Weg zwischen $a$ und $v$ in $G[(E_0 \cup E_k) \cap E(S_L)]$ existieren, da sonst damit ein Kreis geschlossen wäre (vgl. Abb. 5.22a).

Die Mengen $\mathcal{S}_2^{\mathrm{st}}$ und $\mathcal{S}_3^{\mathrm{st}}$ sind analog definiert. Die für diese Mengen spezifischen Fälle sind in Abb. 5.22b und 5.22c dargestellt.




**(a)** $\mathcal{S}_1^{\mathrm{st}}$ **(b)** $\mathcal{S}_2^{\mathrm{st}}$ **(c)** $\mathcal{S}_3^{\mathrm{st}}$

**Abbildung 5.22:** Die für die Mengen $\mathcal{S}_1^{\mathrm{st}}$, $\mathcal{S}_2^{\mathrm{st}}$ und $\mathcal{S}_3^{\mathrm{st}}$ spezifischen Fälle für ein Szenario $k \in$ ST

Die letzte Menge von Summen $\mathcal{S}_4^{\mathrm{st}}$ beschreibt den Fall, der in Abb. 5.20d dargestellt ist: Die Knoten $a$ und $b$ werden durch *First-Stage*-Kanten in der Komponente $S_M$ verbunden. In diesem Fall kann es aber eine *First-Stage*-Komponente $C_v \in \mathcal{C}$ mit $C_v \neq C_0$ geben,

die den Knoten $v$ enthält. Gibt es keine Komponente $C_i \in \mathcal{C}$, die $v$ enthält, so sei $C_v$ der Graph, der nur aus dem Knoten $v$ besteht. Die Menge $\mathcal{S}_4^{\text{st}}$ hat dann die Form

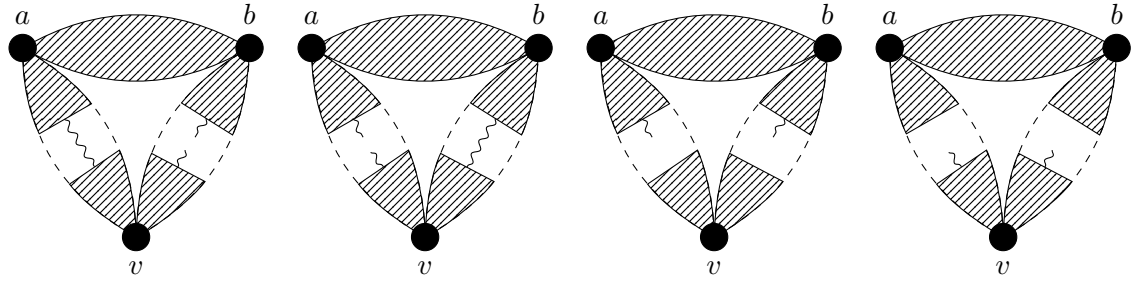
$$\begin{aligned}
\mathcal{S}_4^{\text{st}}(\text{ST}, \text{NN}, \text{NONE}) := \{&\text{st}_{\text{ST}_M,\text{NN}_M,\text{NONE}_M}(M)+ \\
&\text{dt}_{\text{ST}_L,\text{DT}_L,\text{YN}_L,\text{NY}_L,\text{NN}_L,\text{NONE}_L}(L)+ \\
&\text{dt}_{\text{ST}_R,\text{DT}_R,\text{YN}_R,\text{NY}_R,\text{NN}_R,\text{NONE}_R}(R) \mid \\
&\forall k \in \text{ST}: \\
&\quad k \in \text{ST}_M \wedge k \in \text{ST}_L \wedge k \in \text{DT}_R \vee && (5.63) \\
&\quad k \in \text{ST}_M \wedge k \in \text{DT}_L \wedge k \in \text{ST}_R \vee && (5.64) \\
&\quad k \in \text{ST}_M \wedge k \in \text{YN}_L \wedge k \in \text{NY}_R, && (5.65) \\
&\forall k \in \text{NN}: \\
&\quad k \in \text{NN}_M \wedge k \in \text{NONE}_L \wedge k \in \text{NONE}_R \vee \\
&\quad k \in \text{NONE}_M \wedge k \in \text{NN}_L \wedge k \in \text{NONE}_R \vee \\
&\quad k \in \text{NONE}_M \wedge k \in \text{NONE}_L \wedge k \in \text{NN}_R \vee \\
&\quad k \in \text{NONE}_M \wedge k \in \text{NY}_L \wedge k \in \text{YN}_R, && (5.66) \\
&\forall k \in \text{NONE}: \\
&\quad k \in \text{NONE}_M \wedge k \in \text{NONE}_L \wedge k \in \text{NONE}_R\}.
\end{aligned}$$

Für $k \in \text{NN}$ und $k \in \text{NONE}$ sind die gleichen Fälle wie für $\mathcal{S}_1^{\text{st}}$ möglich. Für $k \in \text{ST}$ sind zwei Fälle zu unterscheiden: Entweder wird für ein Szenario $k$ die Komponente $C_v$ über *Second-Stage*-Kanten mit $C_0$ verbunden oder nicht. Soll die Komponente $C_v$ mit $C_0$ über *Second-Stage*-Kanten verbunden werden, so kann diese Verbindung über $S_L$ oder $S_R$ geschehen. Dies wird von den Zeilen 5.63 und 5.64 abgedeckt (vgl. auch Abb. 5.23a und 5.23b). Soll die Komponente $C_v$ nicht von *Second-Stage*-Kanten des Szenarios $k$ mit der Komponente $C_0$ verbunden werden, so dürfen die *Second-Stage*-Kanten in den Teilgraphen $S_L$ und $S_R$ keine Verbindung zu $C_v$ besitzen (vgl. Abb. 5.23c). Diesen Fall bildet die Zeile 5.65 ab.

Für $k \in \text{NN}$ ist ein weiterer Fall möglich: Es ist denkbar, dass die *Second-Stage*-Kanten mit der Komponente $C_v$ verbunden sind, aber nicht mit der Komponente $C_0$. Dieser Fall ist in Zeile 5.66 und in Abb. 5.23d dargestellt.

### Die Variable dt

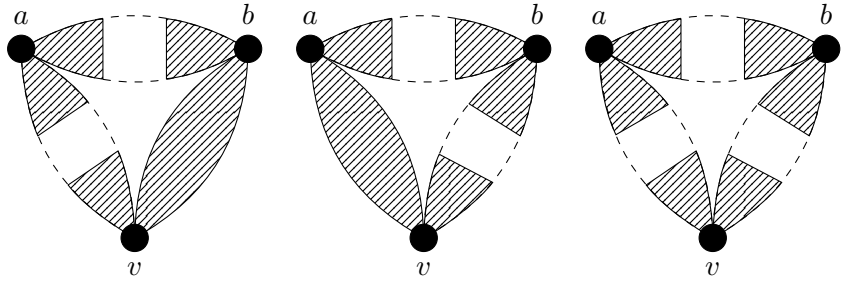
Die Variable $\text{dt}_{\text{ST,DT,YN,NY,NN,NONE}}(M)$ speichert den Wert einer günstigsten Lösung, in der zwischen $a$ und $b$ kein Weg aus *First-Stage*-Kanten existiert. Sei $\mathcal{C} = \{C_0, \ldots, C_r\}$ die Menge der Komponenten von $G[E_0]$ für eine solche Lösung. Es gelte o. B. d. A. $a \in C_0$ und $b \in C_1$, falls $a$ und $b$ in einer eigenen *First-Stage*-Komponente enthalten sind. Gibt es kein $C_i \in \mathcal{C}$, sodass $a \in C_i$ ist, so sei $C_0$ der Graph, der nur den Knoten $a$ enthält und die




**(a)** $k \in$ ST: Zeile 5.63 **(b)** $k \in$ ST: Zeile 5.64 **(c)** $k \in$ ST: Zeile 5.65 **(d)** $k \in$ NN: Zeile 5.66

**Abbildung 5.23:** Die spezifischen Fälle, die die Summen der Menge $\mathcal{S}_4^{\text{st}}$ für ein Szenario $k$ abdecken

Mengen in $\mathcal{C}$ seien dementsprechend umbenannt. Dies gilt für $C_1$ und $b$ analog. Auch bei dieser Variablen gibt es neue Fälle für die erste Phase. Die Möglichkeiten, die die Abb. 5.24a und 5.24b beschreiben, sind auch im Algorithmus von Wald und Colbourn für die Variable dt für einen Steinerbaum erlaubt. In dem Fall aus Abb. 5.24c existiert analog zum vierten Fall der Variable st eine Komponente $C_v \in \mathcal{C}$ mit $C_v \neq C_0$ und $C_v \neq C_1$ sowie $v \in V(C_v)$.




**(a)** **(b)** **(c)**

**Abbildung 5.24:** Die Fälle für die Variable $\text{dt}_{\text{ST,DT,YN,NY,NN,NONE}}((a,b))$ in der ersten Phase: Der Fall (c) ist neu hinzugekommen.

Auch diese Variable lässt sich durch das Minimum dreier Mengen von Summen $\mathcal{S}_i^{\text{dt}}$ berechnen, die die Fälle in der zweiten Phase unterscheiden:

$$\text{dt}_{\text{ST,DT,YN,NY,NN,NONE}}(M) \leftarrow \min \bigcup_{i=1}^{3} \mathcal{S}_i^{\text{dt}}(\text{ST}, \text{YN}, \text{NY}, \text{NN}, \text{NONE})$$

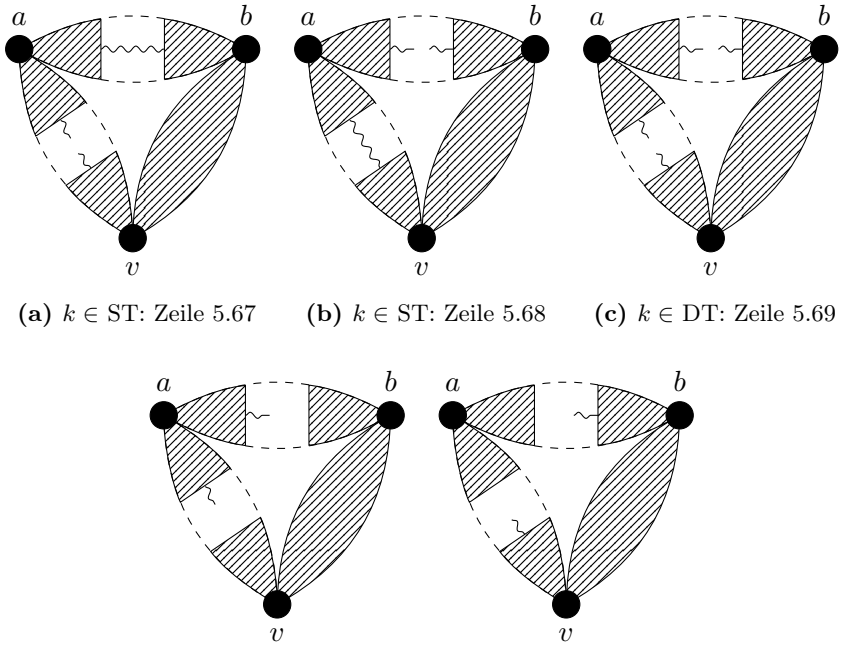
Die Menge $\mathcal{S}_1^{\mathrm{dt}}$ beschreibt den Fall aus Abb. 5.24a. Es darf einen Weg aus *First-Stage*-Kanten zwischen den Knoten $b$ und $v$ geben, nicht jedoch zusätzlich zwischen $a$ und $v$. Diese Menge hat dann die Form

$$
\begin{aligned}
\mathcal{S}_1^{\mathrm{dt}}(\mathrm{ST},\mathrm{DT},\mathrm{YN},\mathrm{NY},\mathrm{NN},\mathrm{NONE}) := \{&\mathrm{dt}_{\mathrm{ST}_M,\mathrm{DT}_M,\mathrm{YN}_M,\mathrm{NY}_M,\mathrm{NN}_M,\mathrm{NONE}_M}(M)+\\
&\mathrm{dt}_{\mathrm{ST}_L,\mathrm{DT}_L,\mathrm{YN}_L,\mathrm{NY}_L,\mathrm{NN}_L,\mathrm{NONE}_L}(L)+\\
&\mathrm{st}_{\mathrm{ST}_R,\mathrm{NN}_R,\mathrm{NONE}_R}(R) \mid\\
&\forall k \in \mathrm{ST}:\\
&\quad k \in \mathrm{ST}_M \wedge k \in \mathrm{DT}_L \wedge k \in \mathrm{ST}_R \vee \qquad (5.67)\\
&\quad k \in \mathrm{DT}_M \wedge k \in \mathrm{ST}_L \wedge k \in \mathrm{ST}_R, \qquad (5.68)\\
&\forall k \in \mathrm{DT}:\\
&\quad k \in \mathrm{DT}_M \wedge k \in \mathrm{DT}_L \wedge k \in \mathrm{ST}_R, \qquad (5.69)\\
&\forall k \in \mathrm{YN}:\\
&\quad k \in \mathrm{YN}_M \wedge k \in \mathrm{YN}_L \wedge k \in \mathrm{NONE}_R, \qquad (5.70)\\
&\forall k \in \mathrm{NY}:\\
&\quad k \in \mathrm{NY}_M \wedge k \in \mathrm{NY}_L \wedge k \in \mathrm{ST}_R, \qquad (5.71)\\
&\forall k \in \mathrm{NN}:\\
&\quad k \in \mathrm{NN}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NONE}_R \vee\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NN}_L \wedge k \in \mathrm{NONE}_R \vee\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NN}_R,\\
&\forall k \in \mathrm{NONE}:\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NONE}_R\}.
\end{aligned}
$$

Für Szenarien $k \in \mathrm{ST}$ kann die Verbindung zwischen $a$ und $b$ über $S_M$ oder über den Knoten $v$ geschlossen werden (vgl. Abb. 5.25a und 5.25b sowie Zeilen 5.67 und 5.68). Für $k \in \mathrm{DT}$ darf weder zwischen $a$ und $b$ noch zwischen $a$ und $v$ ein Weg existieren (vgl. Abb 5.25c und Zeile 5.69). Für $k \in \mathrm{YN}$ bzw. $k \in \mathrm{NY}$ dürfen die *Second-Stage*-Kanten ausschließlich mit der Komponente $C_0$ bzw. ausschließlich mit der Komponente $C_1$ verbunden sein (vgl. Abb. 5.25d und 5.25e sowie Zeilen 5.70 und 5.71).

Die Menge $\mathcal{S}_2^{\mathrm{dt}}$ deckt die Fälle aus Abb. 5.24b ab. Sie ist vollständig analog zu $\mathcal{S}_1^{\mathrm{dt}}$ definiert.

Die Menge $\mathcal{S}_3^{\mathrm{dt}}$ hingegen hat die meisten Freiheiten für die Kanten der zweiten Phase. Hier wird der Fall aus Abb. 5.24c abgedeckt. Es darf keine Wege aus *First-Stage*-Kanten geben, die die Knoten $a$, $b$ und $v$ miteinander verbinden. Allerdings darf es eine *First-Stage*-Komponente $C_v \in \mathcal{C}$ mit $C_v \neq C_0$ und $C_v \neq C_1$ geben, die den Knoten $v$ enthält (mit den üblichen Anmerkungen, falls $v$ in keiner Komponente $C_i$ vorkommt).




**(a)** $k \in$ ST: Zeile 5.67 **(b)** $k \in$ ST: Zeile 5.68 **(c)** $k \in$ DT: Zeile 5.69

**(d)** $k \in$ YN: Zeile 5.70 **(e)** $k \in$ NY: Zeile 5.71

**Abbildung 5.25:** Fälle, die die Summen der Menge $\mathcal{S}_1^{\mathrm{dt}}$ für ein Szenario $k$ abdecken

$$
\begin{aligned}
\mathcal{S}_3^{\mathrm{dt}}(\mathrm{ST},\mathrm{DT},\mathrm{YN},\mathrm{NY},\mathrm{NN},\mathrm{NONE}) := \{&\mathrm{dt}_{\mathrm{ST}_M,\mathrm{DT}_M,\mathrm{YN}_M,\mathrm{NY}_M,\mathrm{NN}_M,\mathrm{NONE}_M}(M)+\\
&\mathrm{dt}_{\mathrm{ST}_L,\mathrm{DT}_L,\mathrm{YN}_L,\mathrm{NY}_L,\mathrm{NN}_L,\mathrm{NONE}_L}(L)+\\
&\mathrm{dt}_{\mathrm{ST}_R,\mathrm{DT}_R,\mathrm{YN}_R,\mathrm{NY}_R,\mathrm{NN}_R,\mathrm{NONE}_R}(R) \mid\\
&\forall k \in \mathrm{ST}:\\
&\quad k \in \mathrm{ST}_M \wedge k \in \mathrm{ST}_L \wedge k \in \mathrm{DT}_R \vee\\
&\quad k \in \mathrm{ST}_M \wedge k \in \mathrm{DT}_L \wedge k \in \mathrm{ST}_R \vee\\
&\quad k \in \mathrm{ST}_M \wedge k \in \mathrm{YN}_L \wedge k \in \mathrm{NY}_R \vee\\
&\quad k \in \mathrm{DT}_M \wedge k \in \mathrm{ST}_L \wedge k \in \mathrm{ST}_R,\\
&\forall k \in \mathrm{DT}:\\
&\quad k \in \mathrm{DT}_M \wedge k \in \mathrm{ST}_L \wedge k \in \mathrm{DT}_R \vee\\
&\quad k \in \mathrm{DT}_M \wedge k \in \mathrm{DT}_L \wedge k \in \mathrm{ST}_R \vee\\
&\quad k \in \mathrm{DT}_M \wedge k \in \mathrm{YN}_L \wedge k \in \mathrm{NY}_R,\\
&\forall k \in \mathrm{YN}:\\
&\quad k \in \mathrm{YN}_M \wedge k \in \mathrm{ST}_L \wedge k \in \mathrm{YN}_R \vee\\
&\quad k \in \mathrm{YN}_M \wedge k \in \mathrm{YN}_L \wedge k \in \mathrm{NONE}_R,\\
&\forall k \in \mathrm{NY}:\\
&\quad k \in \mathrm{NY}_M \wedge k \in \mathrm{NY}_L \wedge k \in \mathrm{ST}_R \vee\\
&\quad k \in \mathrm{NY}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NY}_R,\\
&\forall k \in \mathrm{NN}:\\
&\quad k \in \mathrm{NN}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NONE}_R \vee\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NN}_L \wedge k \in \mathrm{NONE}_R \vee\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NN}_R \vee\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NY}_L \wedge k \in \mathrm{YN}_R,\\
&\forall k \in \mathrm{NONE}:\\
&\quad k \in \mathrm{NONE}_M \wedge k \in \mathrm{NONE}_L \wedge k \in \mathrm{NONE}_R\}
\end{aligned}
$$

Hierbei sind sämtliche Fälle aus dem Algorithmus von Wald und Colbourn für jedes Szenario möglich. Die Komponenten $C_0$, $C_1$ und $C_v$ können dabei als Stellvertreter für die Knoten $a$, $b$ und $v$ betrachtet werden.

### 5.3.4 Berechnung der Ausgabe

Am Schluss ist der Graph auf eine Kante $\{a, b\}$ reduziert. Diese Kante besteht aus den Halbkanten $\alpha := (a, b)$ und $(b, a)$. Der Algorithmus gibt dann das Minimum der Menge

$$\{\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha) \mid \mathrm{ST} \dot\cup \mathrm{NN} \dot\cup \mathrm{NONE} = \{1, \ldots, K\}\} \cup$$
$$\{\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha) \mid \mathrm{ST} \dot\cup \mathrm{YN} \dot\cup \mathrm{NY} \dot\cup \mathrm{NN} \dot\cup \mathrm{NONE} = \{1, \ldots, K\}, \mathrm{DT} = \emptyset\}$$

aus.

Um auch die Kanten der Lösung zu erhalten, kann wieder für jede Variable zusätzlich eine Menge für die *First-Stage*-Kanten und je eine Menge pro Szenario mit den *Second-Stage*-Kanten gespeichert werden. Diese Mengen ergeben sich dann als Vereinigung der Mengen der Variablen, aus denen sie hervorgeht.

### 5.3.5 Analyse des Algorithmus

**Lemma 5.3.1.** *Für eine Instanz $(G = (V, E), c_0, (T_k, c_k, p_k)_{1\leq k\leq K})$ für das SSTP, in der $G$ ein 2-Baum ist, hat der in diesem Abschnitt vorgestellte Algorithmus eine Laufzeit von $\Theta(16^K|V|)$.*

Beweis. Die Laufzeit kann analog zu Lemma 5.1.1 bestimmt werden. Für jede Halbkante $\alpha$ gibt es $3^K$ Variablen $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ und $6^K$ Variablen $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha)$. Demnach erfordert die Initialisierung eine Laufzeit von $\Theta(6^K K|V|)$.

Die Mengen $\mathcal{S}_1^{\mathrm{st}}$, $\mathcal{S}_2^{\mathrm{st}}$ und $\mathcal{S}_3^{\mathrm{st}}$ beschreiben jeweils – summiert über alle Möglichkeiten für die Mengen ST, NN und NONE – insgesamt $5^K$ Möglichkeiten. Für die Menge $\mathcal{S}_4^{\mathrm{st}}$ hingegen müssen für alle Möglichkeiten für die Mengen ST, NN und NONE insgesamt $8^K$ Werte berechnet werden.

Für die Variablen $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}$ sind für alle Mengen $\mathcal{S}_1^{\mathrm{dt}}$ und $\mathcal{S}_2^{\mathrm{dt}}$ jeweils zusammen $9^K$ Werte zu bestimmen und für die Mengen $\mathcal{S}_3^{\mathrm{dt}}$ schlussendlich insgesamt $16^K$ Werte. □

**Lemma 5.3.2.** *Der in diesem Abschnitt vorgestellte Algorithmus berechnet auf 2-Bäumen eine minimale Lösung für das SSTP.*

Beweis. Es muss gezeigt werden, dass die durch den Algorithmus berechnete Lösung zulässig und minimal ist. Sei $E_0$ die Menge der von dem Algorithmus berechneten *First-Stage*-Kanten und für jedes Szenario $k$ sei $E_k$ die Menge der vom Algorithmus berechneten *Second-Stage*-Kanten. Die Fallunterscheidungen bei der Berechnung der Variablen garantieren, dass für jedes Szenario $k$ der Graph $G[E_0 \cup E_k]$ ein Wald ist. Die Menge $F_k$ besteht aus allen Kanten aus $E_0$, die durch eine Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}$ gewählt werden, bei der $k \in$ ST ist. Die Fallunterscheidungen stellen ebenfalls sicher, dass für eine solche Kantenmenge $F_k$ der Graph $G[F_k \cup E_k]$ zusammenhängend ist.

Bei der Initialisierung erhält eine Variable, die ein Terminal eines Szenarios endgültig ausspart, den Wert

$$N = \sum_{e \in E} \max \left\{ \sum_{k=1}^{K} p_k c_k(e), c_0(e) \right\} + 1.$$

Beim Zusammensetzen von Lösungen wird dann ein Wert von $N$ addiert, wenn ein Terminal ausgelassen wird. Eine Lösung, die ein Terminal auslässt hat somit mindestens einen Wert von $N$. Bei dem Algorithmus können aber auch Spannbäume von $G$ getestet werden, die nur aus *First-Stage*-Kanten des Eingabegraphen bestehen (indem immer entweder in $S_L$ oder $S_R$ die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}$ gewählt wird). Diese haben aber Kosten, die echt kleiner als $N$ sind. Somit werden immer Lösungen gefunden, die alle Terminale in den einzelnen Szenarien verbinden.

Um die Minimalität zu zeigen muss der Beweis aus Lemma 5.1.2 leicht abgewandelt werden. Sei dazu $E_0^{\mathrm{opt}}$ die Menge der *First-Stage*-Kanten einer optimalen Lösung und für jedes Szenario $k$ sei $E_k^{\mathrm{opt}}$ die Menge der *Second-Stage*-Kanten für $k$ der gleichen Lösung. Sei $\mathcal{C} = \{C_0, \ldots, C_r\}$ die Menge der Komponenten von $G[E_0^{\mathrm{opt}}]$ und $k$ ein Szenario. Es sei $\{x, y\} \in E$; diese Kante ist nach [20] ein 2-Separator. Sei $\mathcal{H}$ die Menge der durch $\{x, y\}$ induzierten 2-Bäume.

Um zu zeigen, dass diese optimale Lösung gefunden werden kann, können nach der Form dieser Lösung Fälle unterschieden werden. Die erste Ebene dieser Fallunterscheidung deckt die Möglichkeiten ab, in denen zwischen den Knoten $x$ und $y$ in einem $H \in \mathcal{H}$ ein Weg aus *First-Stage*-Kanten existiert oder nicht.

1. In diesem Fall existiert in $H$ ein Weg aus *First-Stage*-Kanten zwischen $x$ und $y$. Somit sind beide Knoten in der gleichen Komponente von $G[E_0^{\mathrm{opt}}]$, o. B. d. A. sei dies $C_0$. Dazu kann in $H$ die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}$ gewählt werden. Für das Szenario $k$ wird nun unterschieden, ob in der optimalen Lösung innerhalb von $H$ eine Verbindung zu $C_0$ besteht oder nicht.

   (a) Ist $V(H) \cap V(G[E_k^{\mathrm{opt}}]) \cap V(C_0) \neq \emptyset$, so hat $E_k^{\mathrm{opt}}$ eine Verbindung zu $C_0$ in $H$. Dies wird durch die Wahl von $k \in \mathrm{ST}$ ermöglicht.

   (b) Ist das Gegenteil der Fall, also $V(H) \cap V(G[E_k^{\mathrm{opt}}]) \cap V(C_0) = \emptyset$, dann kann für die *Second-Stage*-Kanten $E_k^{\mathrm{opt}}$ unterschieden werden, ob sie sich vollständig in $H$ befinden oder ob keine Kante aus $E_k^{\mathrm{opt}}$ in $H$ enthalten ist. Es ist hingegen nicht möglich, dass sich nur manche *Second-Stage*-Kanten von $k$ in $H$ befinden und manche nicht, da eine Verbindung nur über $C_0$ existieren kann.

      i. Ist einerseits $E_k^{\mathrm{opt}} \subseteq E(H)$, dann wird der Fall durch die Wahl von $k \in \mathrm{NN}$ abgedeckt.

      ii. Andererseits kann $E_k^{\mathrm{opt}} \cap E(H) = \emptyset$ sein; dann befinden sich keine Kanten des Szenarios $k$ in $H$. Dies ist durch die Wahl von $k \in \mathrm{NONE}$ möglich.

2. Gilt das Gegenteil von Fall 1, so gibt es in der optimalen Lösung keinen Weg aus *First-Stage*-Kanten in $H$ zwischen $x$ und $y$. Dieser Fall wird durch die Wahl von $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}$ für $H$ abgedeckt. Ist $x$ in $G[E_0^{\mathrm{opt}}]$, so sei $C_x \in \mathcal{C}$ mit $x \in V(C_x)$ und sei $C_y$ analog für $y$ definiert. Ist ein Knoten $v \in \{x, y\}$ nicht in $G[E_0^{\mathrm{opt}} \cap E(H)]$, so sei $C_v$ der Graph, der nur den Knoten $v$ enthält. Für das Szenario $k$ können jetzt die Fälle unterschieden werden, in denen $E_k^{\mathrm{opt}}$ in $H$ eine Verbindung zu $C_x$, $C_y$, zu beiden oder zu keinem der Teilgraphen besitzt.

   (a) Ist $V(H) \cap V(C_x) \cap V(G[E_k^{\mathrm{opt}}]) \neq \emptyset$ und $V(H) \cap V(C_y) \cap V(G[E_k^{\mathrm{opt}}]) \neq \emptyset$, dann haben die *Second-Stage*-Kanten des Szenarios $k$ in $G[(E_0^{\mathrm{opt}} \cup E_k^{\mathrm{opt}}) \cap E(H)]$ eine Verbindung sowohl zu $x$ als auch zu $y$. Dabei gibt es zwei Möglichkeiten zu unterscheiden: Durch Kanten in $G[(E_k^{\mathrm{opt}} \cup E_0^{\mathrm{opt}}) \cap E(H)]$ können die Graphen $C_x$ und $C_y$ miteinander verbunden sein oder nicht.

      i. Es gibt einen Weg zwischen $x$ und $y$ in $G[(E_k^{\mathrm{opt}} \cup E_0^{\mathrm{opt}}) \cap E(H)]$. Genau dieser Fall wird durch $k \in \mathrm{ST}$ abgedeckt.

      ii. Es gibt keinen Weg zwischen $x$ und $y$ in $G[(E_k^{\mathrm{opt}} \cup E_0^{\mathrm{opt}}) \cap E(H)]$. Das wird durch die Wahl von $k \in \mathrm{DT}$ ermöglicht.

   (b) Ist $V(H) \cap V(C_x) \cap V(G[E_k^{\mathrm{opt}}]) \neq \emptyset$ und $V(H) \cap V(C_y) \cap V(G[E_k^{\mathrm{opt}}]) = \emptyset$, so ist der Graph $C_x$ an die Lösung für Szenario $k$ angebunden, der Graph $C_y$ hingegen nicht. Diese Situation wird durch die Wahl von $k \in \mathrm{YN}$ oder $k \in \mathrm{NY}$ abgebildet. Hier kommt es zusätzlich auf die Richtung der Halbkante an.

   (c) Ist $V(H) \cap V(C_x) \cap V(G[E_k^{\mathrm{opt}}]) = \emptyset$ und $V(H) \cap V(C_y) \cap V(G[E_k^{\mathrm{opt}}]) \neq \emptyset$, so ist dieser Fall analog zu dem vorherigen.

   (d) Ist zuletzt $V(H) \cap (V(C_x) \cup V(C_y)) \cap V(G[E_k^{\mathrm{opt}}]) = \emptyset$, d. h. keiner der Graphen $C_x$ oder $C_y$ werden im Szenario $k$ verwendet, gibt es zwei Möglichkeiten für $H$: Die Kanten aus $E_k^{\mathrm{opt}}$ können sich in $H$ befinden oder nicht. Es ist nicht möglich, dass sich nur einige Kanten aus $E_k^{\mathrm{opt}}$ in $H$ befinden, da sonst zwischen den Kanten aus $E_k^{\mathrm{opt}}$ eine Verbindung über $x$ oder $y$ hergestellt werden müsste; diese Knoten dürfen aber in diesem Fall nicht verwendet werden.

      i. Zum Einen kann $E_k^{\mathrm{opt}} \subseteq E(H)$ gelten. Die *Second-Stage*-Kanten für das Szenario $k$ befinden sich also vollständig in $H$. Dies wird durch die Wahl von $k \in \mathrm{NN}$ ermöglicht.

      ii. Zum Anderen kann $E_k^{\mathrm{opt}} \cap E(H) = \emptyset$ gelten, sodass in $H$ keine Kanten für das Szenario $k$ in der zweiten Phase gewählt werden dürfen. Dafür steht die Wahl von $k \in \mathrm{NONE}$.

Die Fallunterscheidungen beim Berechnen der Variablen stellen dabei sicher, dass nicht in zwei verschiedenen Teilgraphen aus $\mathcal{H}$ gleichzeitig die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}$ gewählt wird. Erlaubt ist hingegen, dass für alle 2-Bäume in $\mathcal{H}$ die Variable $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}$

gewählt wird. Die Fallunterscheidungen in den Berechnungen stellen zusätzlich sicher, dass die Knoten $x$ und $y$ konsistent in den gleichen Szenarien angebunden werden.

Die Ausgabe des Algorithmus ist das Minimum über alle Möglichkeiten der Halbkante $\alpha = (a, b)$, die eine zulässige Lösung bilden: Für die Variable $\mathrm{st}_{\mathrm{ST,NN,NONE}}(\alpha)$ sind alle Auswahlen von ST, NN und NONE gültig. Für $\mathrm{dt}_{\mathrm{ST,DT,YN,NY,NN,NONE}}(\alpha)$ sind auch alle Auswahlen von ST, YN, NY, NN und NONE gültig; allerdings muss DT $= \emptyset$ gewählt werden. Ist DT $\neq \emptyset$, so existieren sowohl Kanten aus $E_k$ an $C_0$ mit $a \in C_0$ und an $C_1$ mit $b \in C_1$, aber keine Verbindung zwischen diesen Kanten und diese Lösung wäre somit nicht zulässig. □

**Theorem 5.3.3.** *Für eine Instanz* $(G = (V, E), c_0, (T_k, c_k, p_k)_{1 \leq k \leq K})$ *für das* $SSTP^{\leq 2}$ *berechnet der in diesem Abschnitt vorgestellte Algorithmus eine minimale Lösung in Rechenzeit* $\Theta(16^K |V|)$.

Beweis. Als erster Schritt kann wie im Algorithmus von Wald und Colbourn der Algorithmus aus Abschnitt 4.1 dafür verwendet werden, einen serien-parallelen Graphen zu einem 2-Baum zu erweitern. Die Argumentation verläuft dabei genau wie im Beweis zu Proposition 5.1.3.

Lemma 5.3.1 zeigt die Laufzeit dieses Algorithmus und mit Lemma 5.3.2 folgt die Behauptung. □

# Kapitel 6

# Zusammenfassung und Ausblick

In Kapitel 3 wurde gezeigt, dass für das $\text{SSTP}^{\leq 3}$ kein Polynomialzeitalgorithmus existiert, wenn $\mathbf{P} \neq \mathbf{NP}$ ist. Dies hat weitreichende Konsequenzen: Für das STP sind für jedes $k \in \mathbb{N}$ Polynomialzeitalgorithmen bekannt, wenn die Eingabegraphen höchstens Baumweite $k$ besitzen (siehe z.B. [3]). Für das SSTP ist dies nicht möglich, wenn $\mathbf{NP} \neq \mathbf{P}$ ist. Ebenso ist es dann nicht möglich einen parametrisierten Algorithmus zu finden, der als Parameter ausschließlich die Baumweite verwendet.

Ob es einen Polynomialzeitalgorithmus für das $\text{SSTP}^{\leq 2}$ gibt, bleibt jedoch ein offenes Problem. Im Rahmen dieser Arbeit wurden einige Versuche unternommen, die **NP**-Schwere der Entscheidungsvariante des $\text{SSTP}^{\leq 2}$ nachzuweisen. Die dabei intensiver untersuchten Probleme waren

- *Label-Cover* [5],
- 3-SAT und NAE-SAT [21] sowie
- das *Bandwidth*-Problem [7, 17].

Bei den Versuchen, Reduktionen zu konstruieren, erwies sich ein $\text{SSTP}^{\leq 2}$-Orakel immer wieder als vermutlich nicht mächtig genug, eine Instanz dieser Probleme zu lösen, oder die Menge lösbarer Instanzen war nicht notwendigerweise **NP**-schwierig.

Auf der anderen Seite wurde in Kapitel 5 ein Polynomialzeitalgorithmus für Instanzen beschrieben, die aus $K$ Szenarien und einem SP-Graphen mit $n$ Knoten bestehen, der das $\text{SSTP}^{\leq 2}$ löst, falls nur $\mathcal{O}(\log_{16} n)$ Szenarien gegeben sind. Eine direkte Verbesserung dieses Algorithmus könnte möglich sein, indem die *Second-Stage*-Fälle durch geschickte Algorithmen reduziert werden, anstatt alle auszuprobieren. Es ist aber zu vermuten, dass dies nicht für alle Fälle möglich ist und so lediglich die Basis $b$ der Laufzeit $\mathcal{O}(b^K n)$ verbessert werden würde.

Für die Rechenzeiten sei zusammengefasst, dass die hier vorgestellten Algorithmen für die Varianten $\text{SSTP}^{\leq 2}$ und $\text{SSTP}^{\leq 2}_{\text{fs}}$ jeweils eine Laufzeit von $\Theta(16^K n)$ und die Algorithmen für die Varianten $\text{SSTP}^{\leq 2}_{\text{fs,ss}}$ und $\text{SSTP}^{\leq 2}_{\text{ss}}$ jeweils eine Laufzeit von $\Theta(12^K n)$ besitzen.

In [1] stellen die Autoren einen Algorithmus für das SFP vor, der polynomielle Laufzeit in der Länge der Eingabe besitzt, wenn die Eingabegraphen auf die Klasse der *Two-Terminal Series-Parallel Graphs* eingeschränkt werden. Es ist möglich, dass sich dieser Ansatz mit den Methoden aus Abschnitt 5.3 auf das SSTP übertragen lässt. Ebenso kann untersucht werden, ob sich diese Methoden so auf Graphen mit Baumweite höchstens $t$ übertragen lassen, dass für geeignete $a, b, c \in \mathbb{N}$ und Instanzen mit $n$ Knoten und $K$ Szenarien ein FPT-Algorithmus mit einer Laufzeit $\mathcal{O}(a^K b^t n^c)$ entsteht.

Eine andere Möglichkeit, die Problemvariante $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$ zu lösen, eröffnet die Diplomarbeit von D. Kurz [15]: In dieser Arbeit wird gezeigt, dass es eine Reduktion vom SSTP auf das gerichtete Steinerbaumproblem gibt, wenn sowohl $G[E_0 \cup E_k]$ als auch $G[E_0]$ zusammenhängend sein müssen. Es ist denkbar, dass so transformierte $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$-Instanzen in polynomieller Zeit in ihrer Länge lösbar sind. Scheitert dieser Ansatz, so könnte es sich als zielführend erweisen, diese Reduktion anzupassen: Möglicherweise lässt sich eine Reduktion finden, die $\mathrm{SSTP}^{\leq 2}_{\mathrm{fs,ss}}$-Instanzen auf eine Menge von Instanzen für das gerichtete Steinerbaumproblem abbildet, die in polynomieller Zeit lösbar sind.

# Abbildungsverzeichnis

# Algorithmenverzeichnis

# Literaturverzeichnis


[1] Bateni, M. ; Hajiaghayi, M. ; Marx, D.: Approximation Schemes for Steiner Forest on Planar Graphs and Graphs of Bounded Treewidth. In: *Journal of the ACM (JACM)* 58 (2011), Nr. 5, S. 21:1 – 21:37

[2] Brandstädt, A. ; Le, V. B. ; Spinrad, J. P.: *Graph Classes : A Survey.* SIAM Monographs on Discrete Mathematics and Applications, 1999

[3] Chimani, M. ; Mutzel, P. ; Zey, B.: Improved Steiner Tree Algorithms for Bounded Treewidth. In: *IWOCA* Bd. 7056, Springer-Verlag, 2011, S. 374–386

[4] Dantzig, G. B.: Linear Programming Under Uncertainty. In: *Management Science* 1 (1955), Nr. 3 und 4, S. 197 – 206

[5] Dinur, I. ; Safra, S.: On the Hardness of Approximating Label Cover. In: *Inf. Process. Lett.* 89 (1999), Nr. 5, S. 247 – 254

[6] Farley, A. M.: Networks immune to isolated failures. In: *Networks* (1981), Nr. 11, S. 255 – 268

[7] Garey, M. R. ; Graham, R. L. ; Johnson, D. S. ; Knuth, D. E.: Complexity Results for Bandwidth Minimization. In: *SIAM J. Appl. Math.* 34 (1978), S. 477 – 495

[8] Gupta, A. ; Hajiaghayi, M. ; Kumar, A.: Stochastic Steiner Tree with Non-uniform Inflation. In: *APPROX*, Springer-Verlag Berlin Heidelberg, 2007, S. 134 – 148

[9] Gupta, A. ; Pál, M.: Stochastic steiner trees without a root. In: *Proceedings of the 32nd international conference on Automata, Languages and Programming*, Springer-Verlag Berlin Heidelberg, 2005 (ICALP'05), S. 1051 – 1063

[10] Gupta, A. ; Ravi, R. ; Sinha, A.: An edge in time saves nine: LP rounding approximation algorithms for stochastic network design. In: *FOCS*, 2004, S. 218 – 227

[11] Hopcroft, J. ; Tarjan, R.: Algorithm 447 : Efficient algorithms for graph manipulation. In: *Communications of the ACM* 16 (1973), Nr. 6, S. 372 – 378

[12] Hwang, F. K. ; Richards, D. S. ; Winter, P.: *The Steiner Tree Problem.* Elsevier Science Publishers B.V., 1992

[13] Karp, R. M.: Reducibility Among Combinatorial Problems. In: *Complexity of Computer Computations.* Plenum Press, 1972, S. 85 – 103

[14] Kuratowski, C.: Sur le problème des courbes gauches en topologie. In: *Fundamenta Mathematicae* 15 (1930), S. 271 – 283

[15] Kurz, D.: *Parameterized Algorithms for Stochastic Steiner Tree Problems*, Technische Universität Dortmund, Diplomarbeit, 2012

[16] Leeuwen, J. van: Graph Algorithms. In: *Handbook of Theoretical Computer Science, Algorithms and Complexity* Bd. A. 1990, S. 525 – 631

[17] Papadimitriou, C. H.: The NP-Completeness of the bandwidth minimization problem. In: *Computing* 16 (1976), S. 263–270

[18] Papadimitriou, C. H. ; Steiglitz, K.: *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, 1998

[19] Robertson, N. ; Seymour, P. D.: Graph Minors. II. Algorithmic Aspects of Tree-Width. In: *Journal of Algorithms* 7 (1986), Nr. 3, S. 309 – 322

[20] Rose, D. J.: On simple characterizations of $k$-trees. In: *Discrete Math* 7 (1974), S. 317 – 322

[21] Schaefer, T. J.: The complexity of satisfiability problems. In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, ACM, 1978, S. 216 – 226

[22] Vazirani, V. V.: *Approximation Algorithms.* Springer-Verlag Berlin Heidelberg, 2001

[23] Wald, J. A. ; Colbourn, C. J.: Steiner Trees, Partial 2-Trees, and Minimum IFI Networks. In: *Networks* 13 (1983), Nr. 2, S. 159 – 167

[24] Wegener, I.: *Komplexitätstheorie.* Springer-Verlag Berlin Heidelberg, 2003

# Index