FernUniversität in Hagen

Fakultät für Wirtschaftswissenschaft

Bachelorarbeit

Die Lösung des restringierten Steinerbaum-Problems mittels lokaler Suchverfahren

eingereicht bei
Dr. Friedhelm Kulmann
Lehrstuhl für Betriebswirtschaftslehre,
insbesondere Quantitative Methoden
und Wirtschaftsmathematik

Vorname, Name: Wolfgang Skala
Matrikelnummer: 8485089
Studienjahr: WS 2014/15
Abgabedatum: 3. Februar 2015

Inhaltsverzeichnis

Verzeichnisse **iii**
Abbildungen . . . iii
Tabellen . . . iii
Probleme . . . iv
Algorithmen . . . iv
Abkürzungen . . . iv
Symbole . . . iv

1. Einleitung **1**

2. Varianten des Steinerbaum-Problems **3**
2.1. Das geometrische SBP . . . 3
2.2. Das graphentheoretische SBP . . . 8

3. Restringierte Steinerbaum-Probleme **14**
3.1. Überblick . . . 14
3.2. SBP mit Gewinn . . . 14
3.3. SBP mit Sprungbeschränkung . . . 17
3.4. SBP mit Ertrag, Budget- und Sprungbeschränkung . . . 21
3.5. Betriebswirtschaftliche Anwendungen des SBP-EBS . . . 24

4. Algorithmen zur Lösung des SBP-EBS **26**
4.1. Überblick . . . 26
4.2. Nachbarschaftssuche . . . 26
4.3. Gieriger Algorithmus . . . 30
4.4. Tabusuche . . . 33
4.5. Lokale Suche mit Ausbrechen . . . 35
4.6. Kritische Würdigung . . . 40

5. Zusammenfassung und Ausblick **41**

Anhang **42**

Literaturverzeichnis **52**

Eidesstattliche Versicherung **57**

Verzeichnisse

Abbildungen

1.1. Das Steinerbaum-Problem in einem Graphen 1

2.1. Das geometrische SBP und verwandte Probleme 5
2.2. Konstruktion des Steinerpunktes dreier Terminale 6
2.3. Beispiel für einen SB in einem Graphen 9
2.4. Das SBP im Kontext anderer Graphenprobleme 10
2.5. Der Algorithmus von Beasley 12
2.6. Die DFJ-Bedingungen 13

3.1. Das SBP mit Gewinn 15
3.2. Das SBP mit Sprungbeschränkung 18
3.3. Ein SBP mit Sprungbeschränkung, welches keine Lösung besitzt 18
3.4. Die MTZ-Bedingungen 20
3.5. Die Sprungbeschränkung im SBP-EBS 23

4.1. Beispiel einer Nachbarschaftssuche 29
4.2. Kreisvermeidung durch den gierigen Algorithmus 31
4.3. Ein gieriger Algorithmus zur Lösung des SBP-EBS 32
4.4. Züge der Tabusuche 34
4.5. $\mathrm{Zug}_2()$ der lokalen Suche mit Ausbrechen 38

A.1. Suchgraph für den gierigen Algorithmus 43
A.2. Lösungen des SBP-EBS durch die lokale Suche mit Ausbrechen 44

Tabellen

3.1. Anwendungen des SBP-EBS 25

4.1. Vergleich lokaler Suchverfahren zur Lösung des SBP-EBS 40

A.1. Details zu Lösungen der SBP-EBS-Instanzen 42

Probleme

1. Allgemeines Fermat-Torricelli-Problem 3
2. Geometrisches Steinerbaum-Problem 3
3. Geometrisches Minimalgerüstproblem 6
4. Problem des Handlungsreisenden 7
5. Melzak-Problem 7
6. Graphentheoretisches Steinerbaum-Problem 8
7. Verallgemeinertes Steinerbaum-Problem 9
8. Steinerbaum-Problem mit Gewinn 14
9. Steinerbaum-Problem mit Sprungbeschränkung 17
10. Steinerbaum-Problem mit Ertrag, Budget- und Sprungbeschränkung . . . 21

Algorithmen

1. Nachbarschaftssuche 27
2. Gieriger Algorithmus 31
3. Tabusuche 34
4. Iterierte lokale Suche 35
5. Lokale Suche mit Ausbrechen 36

Abkürzungen

DFJ	Dantzig-Fulkerson-Johnson
LSA	lokale Suche mit Ausbrechen
MTZ	Miller-Tucker-Zemlin
SB	Steinerbaum
SBP	(graphentheoretisches) Steinerbaum-Problem
SBP-EBS	Steinerbaum-Problem mit Ertrag, Budget- und Spungbeschränkung

Symbole

$\mathbf{A}$	Adjazenzmatrix
α	Gewichtungsfaktor für den Ertrag eines Knotens
β	Gewichtungsfaktor für die Kosten einer Kante

$\mathscr{C}$	Schnitt
c	Kanten- oder Pfeilgewicht, Kosten
c_{fix}	Fixkosten
$c_{\max}$	Budget
D	Definitionsmenge
$\mathscr{D}$	sprungbeschränkte kürzeste Entfernung
$\langle d_{\mathrm{Elite}} \rangle$	mittlere Hamming-Entfernung der Elitelösungen
d^{eukl}	euklidische Metrik
d^{hamm}	Hamming-Metrik
d^{manh}	Manhattan-Metrik
E	Kantenmenge
$\vec{E}$	Pfeilmenge
$\mathscr{E}$	Ertrag eines Graphen
e	Kante
$\vec{e}$	Pfeil
f	Bewertungs- oder Zielfunktion
$\mathscr{G}$	Graph
$\vec{\mathscr{G}}$	Digraph
$\mathscr{G}_{\mathrm{s}}$	Steinerbaum
$\vec{\mathscr{G}}_{\mathrm{s}}$	gerichteter Steinerbaum
$\mathscr{G}_{\mathrm{t}}$	Teilgraph
$\vec{\mathscr{G}}_{\mathrm{t}}$	Teildigraph
γ	Bewertung eines Zugs bei der Tabusuche
h	maximale Anzahl von Sprüngen
η_{qi}	Anzahl von Sprüngen zwischen den Knoten q und i
δ	Grad eines Knotens
K	Kantenfolge
$\mathscr{K}_{h+2}$	Menge der Ketten mit genau $h + 2$ Knoten
$k_{\mathrm{s}}, k_{\mathrm{l}}, k_{\mathrm{f}}$	Parameter für das Melzak-Problem
$\mathscr{L}$	Länge (Kosten) eines Graphen
m	Anzahl der Steinerpunkte eines geometrischen Steinerbaums
m	Anzahl der Kanten oder Pfeile eines Graphen
N	Menge der Nachbarn eines Knotens
N	Nachbarschaft, Nachbarschaftsstruktur
$\mathbb{N}$	Menge der natürlichen Zahlen
n	Anzahl der Terminale eines geometrischen Steinerbaums
n	Anzahl der Knoten eines Graphen

v	Nachbar
P	Menge der Vorgänger eines Knotens
$\vec{P}$	Pfeilfolge
P	Problem
p	Wahrscheinlichkeit
p_{Weg}	Ausfallwahrscheinlichkeit eines Weges
p_{Pfeil}	Ausfallwahrscheinlichkeit eines Pfeils
π	Strafwert
ϕ	Proportionalitätsfaktor
q	Quelle
$\mathbf{R}$	Verbindungsmatrix
$\mathbb{R}$	Menge der reellen Zahlen
r	Knotengewicht
r_{min}	Ertragsvorgabe
r_{max}	maximal möglicher Ertrag im SBP-EBS
r_{ij}	Anzahl von Verbindungen zwischen den Terminalen i und j
S^{Elite}	Menge von Elitelösungen
$\mathscr{S}$	Suchraum
S	Menge der Steinerpunkte oder -knoten
s	Steinerknoten, Steinerpunkt
s	Lösung
s^*	global optimale Lösung
s°	lokal optimale Lösung
σ	Störparameter
T	Terminalmenge
T	Tabuliste
t	Terminal
θ	Parameter für das probabilistische Eröffnungsverfahren
u	natürliche Entscheidungsvariable für einen Knoten
V	Knotenmenge
x	Entscheidungsvariable eines kombinatorischen Optimierungsproblems
x	binäre Entscheidungsvariable für eine Kante
y	binäre Entscheidungsvariable für einen Knoten
z	Anzahl der Züge eines lokalen Suchverfahrens

1. Einleitung

Die wirtschaftswissenschaftlichen Fakultäten der Universitäten Graz, Linz, München, Nürnberg-Erlagen, Salzburg, Stuttgart, Wien und Zürich planen ein langfristiges gemeinsames Forschungsprojekt, welches einen regen Austausch großer Datenmengen erfordert. Die Universitäten wenden sich daher an ein Telekommunikationsunternehmen, welches bereits ein Netz von Glasfaserkabeln zwischen den acht Städten betreibt (Abb. 1.1a). Die Universitäten wollen einzelne Leitungen derart anmieten, dass diese alle Fakultäten miteinander verbinden, wobei indirekte Verbindungen ausreichen. Die jährlichen Mietkosten einer Leitung sind proportional zu ihrer Länge. Aufgrund dieser Vorgaben bietet das Telekommunikationsunternehmen den Universitäten die in Abb. 1.1b rot gezeichneten Kabel an, weil das resultierende Verbindungsnetz jenes mit der geringsten Länge und daher mit den geringsten Mietkosten ist.

Nach mehreren Jahren fruchtbarer Zusammenarbeit müssen jedoch die Fakultäten in Stuttgart, München und Linz aufgrund ihrer prekären Budgetsituation ihre Beteiligung am Forschungsprojekt aufkündigen. Die verbleibenden Fakultäten stehen nun vor dem Problem, ein neues kostengünstigstes Verbindungsnetz zu finden. Da sie gute Erfahrungen mit dem bisherigen Anbieter gemacht haben, wollen sie weiterhin seine Kunden bleiben.

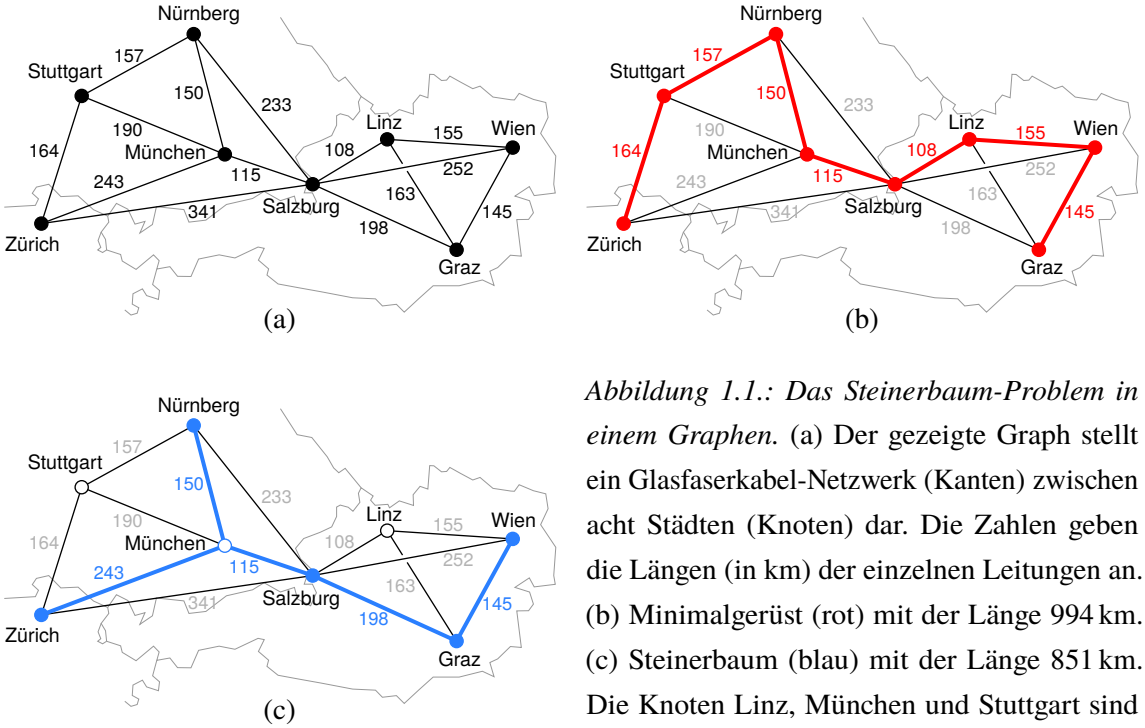


Abbildung 1.1.: Das Steinerbaum-Problem in einem Graphen. (a) Der gezeigte Graph stellt ein Glasfaserkabel-Netzwerk (Kanten) zwischen acht Städten (Knoten) dar. Die Zahlen geben die Längen (in km) der einzelnen Leitungen an. (b) Minimalgerüst (rot) mit der Länge 994 km. (c) Steinerbaum (blau) mit der Länge 851 km. Die Knoten Linz, München und Stuttgart sind Steinerknoten.

Das Telekommunikationsunternehmen schlägt nun das in Abb. 1.1c blau dargestellte Verbindungsnetz vor, welches die fünf Städte Graz, Nürnberg, Salzburg, Wien und Zürich verbindet und wiederum die geringsten Mietkosten verursacht.

⋆

Das einleitende Beispiel stellt das so genannte *Steinerbaum-Problem* (SBP) in einem Graphen vor (vgl. Hakimi, 1971, S. 114): Gegeben sei ein Graph aus Knoten (hier die wirtschaftswissenschaftlichen Fakultäten in den acht Städten) und bewerteten Kanten (hier die bestehenden Glasfaserkabel mit ihren Längen). Eine Teilmenge der Knoten (hier die fünf Fakultäten in Graz, Nürnberg, Salzburg, Wien und Zürich) soll nun auf kürzestem Wege verbunden werden. Der resultierende Teilgraph – der so genannte *Steinerbaum* (SB) – darf die restlichen Knoten enthalten (hier die drei Fakultäten in Stuttgart, München und Linz), muss aber nicht. Im gezeigten Beispiel enthält der SB den Knoten München, weil die sich ergebenden indirekten Verbindungen Salzburg–München–Nürnberg bzw. Salzburg–München–Zürich in Summe kürzer sind (508 km) als die direkten Verbindungen Salzburg–München und Salzburg–Zürich (zusammen 574 km). Die Knoten Linz und Stuttgart gehören hingegen *nicht* zum SB, weil sich dadurch seine Länge erhöhen würde.

Diese Arbeit beschäftigt sich mit dem Steinerbaum-Problem mit Ertrag, Budget- und Spungbeschränkung (SBP-EBS), einer restringierten Variante des SBP. Kapitel 2 enthält einen historischen Abriss der Entwicklung vom geometrischen SBP (Abschnitt 2.1) hin zum (einfachen) graphentheoretischen SBP (Abschnitt 2.2). Kapitel 3 führt im Anschluss zusätzliche Bedingungen ein, die das SBP einschränken können (Abschnitt 3.2 und 3.3), und formuliert mithilfe dieser Bedingungen das SBP-EBS (Abschnitt 3.4). Abschnitt 3.5 stellt betriebswirtschaftliche Anwendungen des SBP-EBS vor und zeigt somit, dass dieses Problem nicht nur von akademischem Interesse ist. Bereits das gewöhnliche SBP lässt sich vermutlich nicht effizient lösen (vgl. Garey *et al.*, 1977, S. 837). Daher werden vielfach Heuristiken eingesetzt, um mit angemessenem Rechenaufwand gute – wenngleich meist suboptimale – Lösungen für das SBP und verwandte Probleme zu ermitteln (vgl. Voß, 1992, S. 46–70). Kapitel 4 stellt drei solcher Heuristiken zur Lösung des SBP-EBS vor, die alle auf dem Prinzip der Nachbarschaftssuche basieren (Abschnitt 4.2): Einen gierigen Algorithmus (Abschnitt 4.3), eine Variante der Tabusuche (Abschnitt 4.4) und die lokale Suche mit Ausbrechen (Abschnitt 4.5). Abschnitt 4.6 vergleicht die vorgestellten Lösungsverfahren kritisch. Kapitel 5 fasst die Erkenntnisse der Arbeit zusammen und stellt aktuelle Entwicklungen im Bereich des SBP-EBS vor.

2. Varianten des Steinerbaum-Problems

2.1. Das geometrische SBP

Das geometrische SBP geht auf das Fermat-Torricelli-Problem zurück, welches Pierre de Fermat erstmals 1638 in einem Brief an René Descartes formulierte und welches Evangelista Torricelli später löste (vgl. Gueron und Tessler, 2002, S. 443). Fermats Problemstellung lautete: *„Datis tribus punctis, quartum reperire, a quo si ducantur tres rectæ ad data puncta, summa trium harum rectarum sit minima quantitas“* (Fermat, 1891, S. 153). („Man finde zu drei gegebenen Punkten einen vierten, sodass die Summe seiner Entfernungen zu den drei gegebenen Punkten ein Minimum annimmt“, Übers. v. Verf.) Das Fermat-Torricelli-Problem kann auf mehrere Punkte verallgemeinert werden:

Problem 1

Ein Ingenieur soll n Städte $v_1, \dots, v_n$ miteinander verbinden, die nicht auf einer Geraden liegen. Dazu möchte er von einem Punkt s aus eine gerade Leitung zu jeder Stadt verlegen. Wie muss der Ingenieur die Koordinaten von s wählen, um die Gesamtlänge der Leitungen zu minimieren? (nach Gergonne, 1810a, S. 196)

⋆

Allgemeines Fermat-Torricelli-Problem: *Zu einer endlichen Anzahl von Punkten $v_i \in \mathbb{R}^2$ $(i = 1, \dots, n)$ soll das Minimum der Funktion*

$$f : \begin{cases} \mathbb{R}^2 & \to & \mathbb{R} \\ s & \mapsto & \sum_{i=1}^{n} d^{\text{eukl}}(s, v_i) \end{cases}, \tag{2.1}$$

gefunden werden, wobei $d^{\text{eukl}}(s, v_i)$ die Entfernung zwischen s und v_i beschreibt (vgl. Nam, 2013, S. 2). s wird als Fermatpunkt des Polygons $(v_1, \dots, v_n)$ bezeichnet.

Eine alternative Verallgemeinerung des Fermat-Torricelli-Problems führt auf das geometrische SBP, welches auf Joseph Diaz Gergonne zurückgeht (vgl. Brazil *et al.*, 2014, S. 6–14) und durch das Buch *What is mathematics?* von Courant und Robbins (1941) bekannt wurde:

Problem 2

Ein Kanalsystem soll die Städte $v_1, \dots, v_n$ verbinden. Die geographische Lage der Städte ist bekannt und wird durch zwei Koordinatenwerte dargestellt. Die Gesamtlänge der Kanäle soll minimal werden (nach Gergonne, 1810b, S. 292).

⋆

> ***Geometrisches Steinerbaum-Problem:*** *Zu einer endlichen Anzahl von Punkten $v_i \in \mathbb{R}^2$ $(i = 1, \ldots, n)$ soll ein Baum minimaler Länge gefunden werden, der außer diesen Punkten noch beliebig viele andere Punkte $s_j \in \mathbb{R}^2$ $(j = 1, \ldots, m)$ enthalten darf. Ein Baum ist hier eine Teilmenge der $\binom{n+m}{2}$ möglichen Strecken zwischen den $n + m$ Punkten, sodass je zwei Punkte durch genau eine Folge von Strecken verbunden werden (vgl. Melzak, 1961, S. 144).*

Die zusätzlichen Punkte s_j heißen *Steinerpunkte*, die v_i heißen *Terminale*. Je nachdem, wie die Länge einer Strecke (und somit die Gesamtlänge des Baums) berechnet wird, werden das euklidische und das rektilineare SBP unterschieden. Das *euklidische SBP* (Abb. 2.1a) verwendet als Abstandsmaß die euklidische Metrik d^{eukl} auf $\mathbb{R}^2$, also

$$d^{\text{eukl}} : \begin{cases} \mathbb{R}^2 \times \mathbb{R}^2 & \to \quad \mathbb{R} \\ ((x_1, y_1), (x_2, y_2)) & \mapsto \quad \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \end{cases} . \tag{2.2}$$

Im Falle des *rektilinearen SBP* (Abb. 2.1b) ist ein Abstand hingegen durch die rektilineare oder Manhattan-Metrik

$$d^{\text{manh}} : \begin{cases} \mathbb{R}^2 \times \mathbb{R}^2 & \to \quad \mathbb{R} \\ ((x_1, y_1), (x_2, y_2)) & \mapsto \quad |x_2 - x_1| + |y_2 - y_1| \end{cases} \tag{2.3}$$

definiert (vgl. Hanan, 1966, S. 255). Euklidische Steinerbäume haben u. a. die folgenden Eigenschaften (vgl. Melzak, 1961, S. 146; Gilbert und Pollak, 1968, S. 4–5):

1. $m \leq n - 2$: Ein SB enthält höchstens $n - 2$ Steinerpunkte. Für $m = n - 2$ ergibt sich ein *vollständiger SB*; jedes Terminal ist dann Endpunkt genau einer Strecke.

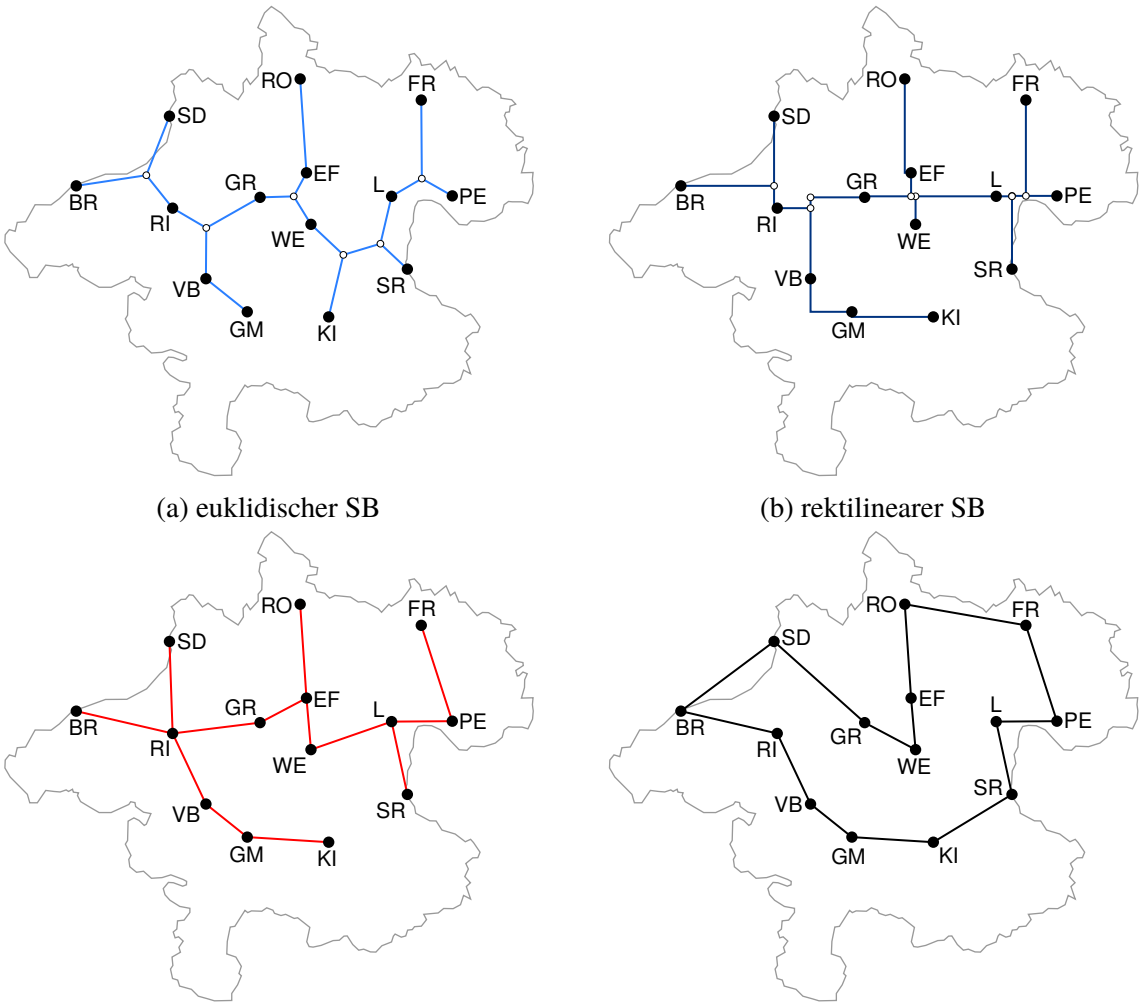
2. Jeder Steinerpunkt s_j ist Fermatpunkt des Dreiecks, welches von den Punkten $v \in N(s_j)$ gebildet wird. Die Menge $N(s_j)$ enthält die Nachbarn von s_j, also jene Punkte, die mit s_j durch eine Strecke verbunden sind.

3. $\forall v_i : \delta(v_i) \leq 3$: An jedem Terminal treffen sich höchstens drei Strecken, die paarweise einen Winkel von mindestens 120° einschließen. $\delta(v_i)$ bezeichnet den Grad eines Punktes (die Anzahl der Strecken, welche v_i als Endpunkt haben). Für rektilineare Steinerbäume gilt $\forall v_i : \delta(v_i) \leq 4$ (vgl. Hanan, 1966, S. 258).

4. $\forall s_j : \delta(s_j) = 3$: An jedem Steinerpunkt treffen sich genau drei Strecken, die paarweise einen Winkel von genau 120° einschließen. Für rektilineare Steinerbäume gilt $\forall s_j : \delta(s_j) = 3$ oder 4 (vgl. Hanan, 1966, S. 258).

5. $\forall s_j : x(s_j) \in \{x(v_i)\}$ und $y(s_j) \in \{y(v_i)\}$: Die Steinerpunkte eines rektilinearen SB liegen auf den Schnittpunkten des *Hanan-Gitters*, welches erzeugt wird, indem

horizontale und vertikale Geraden durch alle Terminale gezeichnet werden. $x(s_j)$ und $y(s_j)$ bezeichnen die x- bzw. y-Koordinaten von s_j. $\{x(v_i)\}$ und $\{y(v_i)\}$ sind die Mengen der x- bzw. y-Koordinaten der Terminale.



(a) euklidischer SB

(b) rektilinearer SB

(c) Minimalgerüst

(d) kürzeste Rundreise

Abbildung 2.1.: Das geometrische SBP und verwandte Probleme. Die 14 oberösterreichischen Bezirkshauptstädte seien als Punkte im $\mathbb{R}^2$ gegeben. (a, b) Der euklidische SB mit diesen Punkten als Terminale (schwarze Punkte) enthält sechs zusätzliche Steinerpunkte (weiß gefüllt), der rektilineare SB acht. Beide Steinerbäume wurden mit *GeoSteiner* v3.1 berechnet (vgl. Warme *et al.*, 2000, S. 81–116). (c) Minimalgerüst, welches in *Scilab* v5.5.1 mit dem Modul *Metanet* v0.6.2 bestimmt wurde. (d) Die kürzeste Rundreise des entsprechenden Problems des Handlungsreisenden wurde mithilfe von *Concorde* v03.12.19 ermittelt (vgl. Applegate *et al.*, 1998, S. 645–654).

Obwohl der Steinerpunkt für drei beliebig angeordnete Terminale leicht konstruiert werden kann (Abb. 2.2, vgl. Gilbert und Pollak, 1968, S. 9), sind sowohl das geometrische SBP mit der euklidischen als auch jenes mit der Manhattan-Metrik *NP*-vollständig. Folglich existiert vermutlich kein Algorithmus, der beliebig große Instanzen dieser Probleme effizient

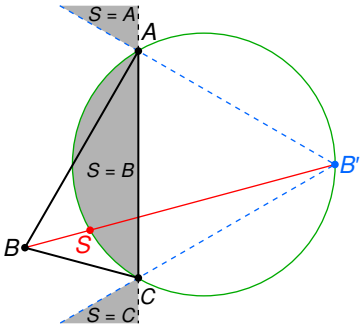


Abbildung 2.2.: Konstruktion des Steinerpunktes dreier Terminale. Das Dreieck ABC (schwarz) ist gegeben. (1) Punkt B' wird so gewählt, dass er mit den Punkten A und C ein gleichseitiges Dreieck bildet (blau). (2) Der Umkreis des Dreiecks $AB'C$ wird gezeichnet (grün). (3) Der Schnittpunkt der sog. Simpson-Linie BB' mit dem Umkreis ist der Steinerpunkt S (rot). Falls B in einer der grau gekennzeichneten Flächen liegt, ist S mit einem der gegebenen Punkte identisch (nach Gilbert und Pollak, 1968, S. 9).

lösen kann (vgl. Garey *et al.*, 1977, S. 837; Garey und Johnson, 1977, S. 827). Dennoch ist eine Reihe exakter Algorithmen bekannt, die selbst für mehrere tausend Terminale einen SB in vertretbarer Zeit ermitteln, wie z. B. der *GeoSteiner*-Algorithmus (vgl. Warme *et al.*, 2000, S. 81–116). Steinerbäume zu speziell angeordneten Terminalen haben manchmal eine besonders einfache Gestalt. Falls z. B. die Terminale ein regelmäßiges Polygon mit sechs oder mehr Ecken bilden, entspricht der SB dem Minimalgerüst: Er entsteht einfach durch Auslassen einer Kante des Polygons (vgl. Jarník und Kössler, 1934, S. 223–235; zitiert nach Korte und Nešetřil, 2001, S. 7–16; Du *et al.*, 1987, S. 65).

Das geometrische SBP ist mit den folgenden beiden Problemen verwandt (s. auch Abb. 2.1c, d): Zum einen vereinfacht sich das geometrische SBP für den Fall $m = 0$ auf das geometrische Minimalgerüstproblem.

Problem 3

> *Ein Telekommunikationsunternehmen will Telefonleitungen zwischen Washington, D. C., und den 48 Hauptstädten des US-Kernlands verlegen. Die Gesamtlänge der Kabel soll möglichst gering sein (vgl. Prim, 1957, S. 1389–1390).*
>
> ⋆
>
> ***Geometrisches Minimalgerüstproblem:*** *Zu einer endlichen Anzahl von Punkten $v_i \in \mathbb{R}^2$ $(i = 1, \ldots, n)$ soll ein Baum minimaler Länge gefunden werden, der nur diese Punkte enthält (vgl. Prim, 1957, S. 1389).*

Nach der *Gilbert-Pollak-Vermutung* unterschreitet die Länge des SB für eine gegebene Terminalmenge niemals das $\frac{\sqrt{3}}{2}$-fache der Länge des entsprechenden Minimalgerüsts (vgl. Gilbert und Pollak, 1968, S. 23–26). Der Bruch $\frac{\sqrt{3}}{2} \approx 0{,}866\,03$ wird als *Steinerverhältnis* bezeichnet. Daher stellt ein Minimalgerüst eine gute Approximation für einen SB dar (vgl. Hwang, 1992, S. 37).

Zum anderen hat das geometrische SBP eine gewisse Ähnlichkeit mit einem der wohl berühmtesten diskreten Optimierungsprobleme:

Problem 4

Eine Außendienstmitarbeiterin will Kunden in den Städten $v_1, \dots, v_n$ besuchen. Die Entfernungen je zweier Städte sind ihr bekannt. In welcher Reihenfolge soll sie die Kunden besuchen?

⋆

Problem des Handlungsreisenden: *Zu einer endlichen Anzahl von Punkten $v_i \in \mathbb{R}^2$ $(i = 1, \dots, n)$ soll ein Kreis minimaler Länge gefunden werden. Ein Kreis ist hier eine Teilmenge mit n der $\binom{n}{2}$ möglichen Strecken zwischen den gegebenen Punkten, sodass $\delta(v_i) = 2$ $(i = 1, \dots, n)$ und je zwei Punkte miteinander verbunden sind (vgl. Melzak, 1961, S. 144).*

Das Ziel der Probleme 3 und 4 besteht ebenfalls darin, eine kürzeste Verbindung von n Punkten zu finden. Lediglich die Restriktionen, die an die Struktur der Verbindungen gestellt werden, unterscheiden sie vom Fermat-Torricelli- und Steinerbaum-Problem. Melzak (1961, S. 144–145) schlug deshalb das folgende Problem vor, welches die Probleme 1–4 verallgemeinert:

Problem 5

n Städte sollen durch ein Straßennetz verbunden werden. Eine Längeneinheit Straße kostet 1 Geldeinheit. Eine Kreuzung, an der sich δ Straßen treffen, kostet innerhalb einer Stadt $k_s \times \delta$ Geldeinheiten und außerhalb einer Stadt $k_l \times \delta + k_f$ Geldeinheiten. k_s und k_l sind also variable Kosten, k_f hingegen Fixkosten. Wie soll das Straßennetz aussehen?

⋆

Melzak-Problem: *Zu einer endlichen Anzahl von Punkten $v_i \in \mathbb{R}^2$ $(i = 1, \dots, n)$ sollen (1) eine natürliche Zahl m, (2) weitere Punkte $s_1, \dots, s_m$ und (3) einen Baum T gefunden werden, der alle diese Punkte verbindet, sodass die Summe*

$$\mathscr{L}(T) + k_s \sum_{i=1}^{n} \delta(v_i) + k_l \sum_{j=1}^{m} \delta(s_j) + k_f\, m \tag{2.4}$$

minimal wird. $\mathscr{L}(T)$ bezeichnet die hier die Gesamtlänge aller Strecken des Baums (vgl. Melzak, 1961, S. 144–145).

Aus Problem 5 folgt für $k_s = k_l = k_f = 0$ das geometrische SBP: Mit diesen Kostenwerten ist lediglich die Gesamtlänge des Baums relevant, und die Anzahl an Steinerpunkten spielt keine Rolle. Mit $k_s = 0$ und $\max\{k_l, k_f\} \gg 1$ ergibt sich hingegen das geometrische Minimalgerüstproblem: In diesem Fall verursacht nämlich jeder zusätzliche Steinerpunkt hohe Kosten, weswegen die Lösung nur Terminale enthält. Wieder andere Werte für die Parameter k_s, k_l, k_f ergeben das Problem des Handlungsreisenden oder das Fermat-Torricelli-Problem (vgl. Melzak, 1961, S. 145).

2.2. Das graphentheoretische SBP

Das SBP erfährt eine erste Einschränkung durch den Übergang vom überabzählbaren $\mathbb{R}^2$ (mit Punkten, die durch Strecken verbunden werden) auf einen Graphen (aus Knoten, die durch Kanten verbunden werden). Während im ersteren Fall die Terminale und Steinerpunkte an jeder möglichen Position liegen können, sind sie im zweiteren Fall auf die Knotenmenge beschränkt – aus dem *diskreten* geometrischen SBP wird somit das *kombinatorische* graphentheoretische SBP (vgl. Hakimi, 1971, S. 114; Levin, 1971, S. 1477–1481).

Zunächst werden die zum Verständnis des graphentheoretischen SBP (im Folgenden nur noch kurz SBP genannt) notwendigen Begriffe eingeführt (vgl. Bondy und Murty, 1976, S. 1–26; Winter, 1987, S. 130–131). Grundlage dieses Problems ist ein *kantengewichteter ungerichteter Graph* $\mathcal{G} = [V, E, c]$, also ein Tripel aus (1) einer Knotenmenge $V = \{1, \dots, n\}$, (2) einer Kantenmenge $E = \{e_{ij} = [i,j] \mid e_{ij} \text{ verbindet Knoten } i \text{ und } j\}$ mit $|E| = m$, und (3) einer nichtnegativen Kantengewichtungsfunktion $c : E \to \mathbb{R}^{\geq 0}, e_{ij} \mapsto c_{ij}$. Da in einem ungerichteten Graphen das Paar $[i,j]$ ungeordnet ist, gilt $[i,j] \equiv [j,i]$. Falls das Symbol für eine Kante (e) mit nur einem Subskript erscheint (z. B. als e_i), ist die i-te Kante von E gemeint. Das entsprechende Kantengewicht $c(e_i)$ wird kurz als c_i bezeichnet. Die *Länge* $\mathcal{L}(\mathcal{G})$ eines Graphen entspricht der Summe seiner Kantengewichte: $\mathcal{L}(\mathcal{G}) = \sum_{e \in E} c(e)$. Ein *Teilgraph* $\mathcal{G}_\mathrm{t} = [V_\mathrm{t}, E_\mathrm{t}, c_\mathrm{t}] \subseteq \mathcal{G}$ ist ein Graph, für den gilt: $V_\mathrm{t} \subseteq V$, $E_\mathrm{t} \subseteq E$, $c_\mathrm{t} = c$ und $e_{ij} \in E_\mathrm{t} \Rightarrow i, j \in V_\mathrm{t}$. Ferner sind zwei Knoten i, j eines Graphen *verbunden*, wenn es in $\mathcal{G}$ eine *Kantenfolge* $K = (i, [i, i+1], i+1, \dots, j-1, [j-1, j], j)$ von i nach j gibt. Eine Kantenfolge ist eine *Kette*, wenn $\forall i, j \in K : i \neq j$ gilt, und ein *Kreis*, falls Anfangs- und Endknoten identisch sind (aber keine anderen Knoten). Ketten mit den Knoten $i, \dots, j$ werden auch kurz als $K = (i, \dots, j)$ geschrieben.

Mit diesen Begrifflichkeiten kann nun das SBP formuliert werden:

Problem 6

> *Welche Glasfaserverbindungen sollen die verbleibenden Fakultäten im Beispiel aus Kapitel 1 mieten, nachdem drei Fakultäten das Forschungsprojekt verlassen haben?*
>
> ⋆
>
> ***Graphentheoretisches Steinerbaum-Problem:*** *Gegeben sei ein kantengewichteter ungerichteter Graph* $\mathcal{G} = [V, E, c]$ *und eine nichtleere Teilmenge* $T \subseteq V$. *Dann soll ein Teilgraph* $\mathcal{G}_\mathrm{s} \subseteq \mathcal{G}$ *minimaler Länge* $\mathcal{L}(\mathcal{G}_\mathrm{s})$ *gefunden werden, der alle Knoten* $t \in T$ *verbindet (vgl. Voß, 1992, S. 45).*

Dieser sog. *Steinerbaum* $\mathcal{G}_\mathrm{s} = (V_\mathrm{s}, E_\mathrm{s}, c_\mathrm{s})$ muss also sämtliche *Terminale t* der Terminalmenge T verbinden. Er kann (muss aber nicht!) beliebig viele *Steinerknoten s* der Steinerknotenmenge $S = V \setminus T$ enthalten. Wie der Name schon andeutet, ist ein SB ein *Baum*, also ein kreisfreier Graph, in dem es genau eine Kette zwischen jedem Knotenpaar

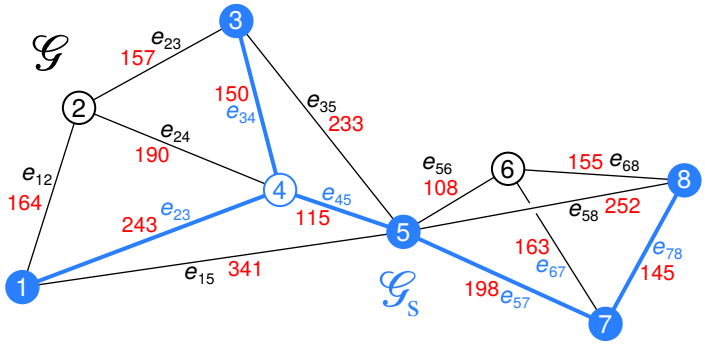


Abbildung 2.3.: Beispiel für einen SB in einem Graphen. Der Graph $\mathcal{G}$ besteht aus acht Knoten, $V = \{1, \ldots, 8\}$, und vierzehn Kanten, $E = \{e_{12}, \ldots, e_{78}\}$ (Kantengewichte in rot). Für die Terminalmenge $T = \{1, 3, 5, 7, 8\}$ (blaue Füllung) und die korrespondierende Steinerknotenmenge $S = \{2, 4, 6\}$ (weiße Füllung) ergibt sich der Steinerbaum $\mathcal{G}_s$ (blau). $\mathcal{G}_s$ enthält $n = 6$ Knoten, $m = n - 1 = 5$ Kanten und hat die Länge $\mathcal{L}(\mathcal{G}_s) = 851$.

gibt, und für den deshalb $m = n - 1$ gilt (vgl. Dreyfus und Wagner, 1972, S. 195–198). Abb. 2.3 greift das Beispiel aus Abb. 1.1 erneut auf, um das SBP formaler zu erläutern.

Das SBP lässt sich einerseits als Verallgemeinerung zweier klassischer Graphenprobleme ansehen, andererseits ist es wiederum der Spezialfall mehrerer übergeordneter Probleme (Abb. 2.4). Je nach der Anzahl der Terminale (vgl. Winter, 1987, S. 131) wird aus dem SBP

- für $|T| = 2$ das *Problem des kürzesten Wegs* zwischen zwei Knoten in einem Graphen. Für dieses Problem haben u. a. Dijkstra (1959, S. 270–271) und Floyd (1962, S. 345) bzw. Warshall (1962, S. 11–12) effiziente Algorithmen vorgeschlagen, welche den kürzesten Weg zwischen zwei bestimmten bzw. allen Knoten ermitteln; und
- für $|T| = n$ (sämtliche Knoten sind Terminale) das graphentheoretische *Minimalgerüstproblem*. Für dieses Problem sind ebenfalls effiziente Algorithmen bekannt, u. a. jene von Kruskal (1956, S. 49–50) und Prim (1957, S. 1391–1392).

Für $|T| = 1$ ist das SBP trivial – der SB besteht aus dem einzigen Terminal.

Während das Problem des kürzesten Wegs und das Minimalgerüstproblem als Spezialfälle des SBP gelten können, lautet eine mögliche Verallgemeinerung des SBP folgendermaßen:

Problem 7

> *Der Kunde einer Telefongesellschaft betreibt Büros an verschiedenen Standorten. Er verlangt, dass seine Büros paarweise durch Telefonleitungen verbunden werden. Um allerdings gegen Störfälle des Telefonnetzes gesichert zu sein, sollen manche Büro-Paare durch mehr als eine Leitung verbunden werden. Der Kunde legt für je zwei Büros i, j die Anzahl der Verbindungen mit r_{ij} fest. Welche Leitungen soll die Telefongesellschaft für diesen Kunden einsetzen? (nach Agrawal et al., 1995, S. 440).*

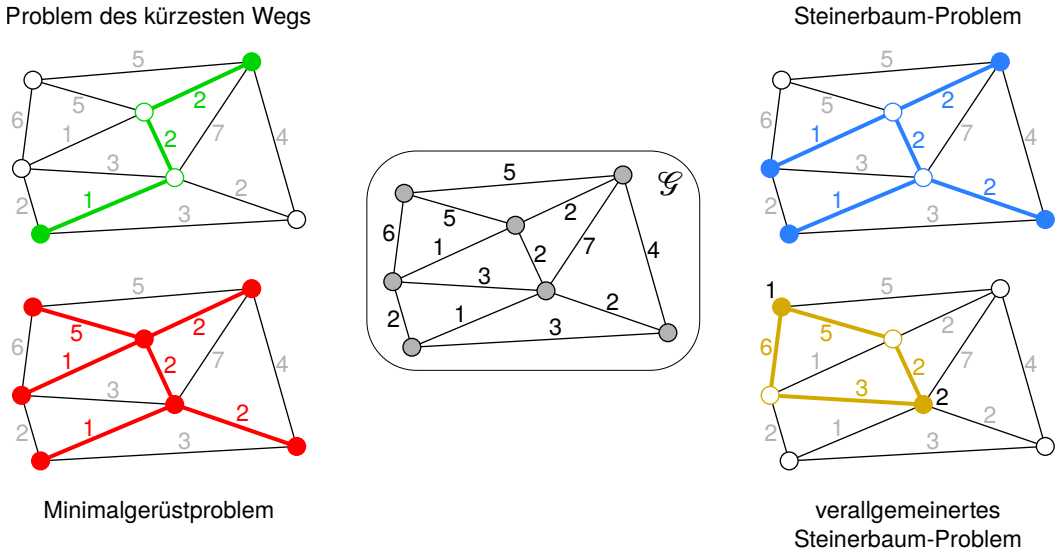


Abbildung 2.4.: Das SBP im Kontext anderer Graphenprobleme. Je nach Wahl der Terminale (gefüllt) und Steinerknoten (weiß) eines gegebenen Graphen $\mathcal{G}$ (Mitte) ergibt sich das Problem des kürzesten Wegs ($|T| = 2$), das Minimalgerüstproblem ($|T| = n$) oder das Steinerbaum-Problem ($2 < |T| < n$). Die ersteren beiden Probleme gelten deswegen als Spezialfälle des SBP, welches wiederum ein Spezialfall des verallgemeinerten SBP ist (hier mit $T = \{1, 2\}$ und $r_{12} = 2$). Eigene Darstellung nach einem Graphen von Winter (1992, S. 96).

⋆

Verallgemeinertes Steinerbaum-Problem: *Gegeben sei ein kantengewichteter ungerichteter Graph $\mathcal{G}$, eine nichtleere Terminalmenge $T = \{1, \ldots, t\} \subseteq V$ und eine $t \times t$-Matrix $\mathbf{R} = (r_{ij})$. Dann soll ein Teilgraph $\mathcal{G}_t \subseteq \mathcal{G}$ minimaler Länge gefunden werden, in dem alle Terminalpaare i, j für $i \neq j$ durch jeweils r_{ij} Ketten verbunden sind. Als zusätzliche Einschränkung dürfen die r_{ij} Ketten jeweils keine gemeinsamen Kanten haben (vgl. Winter, 1987, S. 160–162; Agrawal et al., 1995, S. 440).*

Wie zu erwarten geht Problem 7 in das SBP über, wenn $r_{ij} = 0$ für $i = j$ und $r_{ij} = 1$ sonst.

Modellierungen des SBP verwenden zumeist die Technik der *linearen Optimierung* (vgl. Maculan, 1987, S. 190–199). Im Folgenden werden zwei solcher Ansätze vorgestellt, nämlich (1) die Formulierung als Mengenüberdeckungsproblem von Aneja (1980, S. 168–169) und (2) die Darstellung als Minimalgerüstproblem in einem erweiterten Graphen von Beasley (1989, S. 2–3).

Variante (1), die Formulierung als *Mengenüberdeckungsproblem*, zerlegt zuerst die Knotenmenge V auf alle k möglichen Arten in zwei disjunkte Mengen V_1, V_2 mit $V_1 \cup V_2 = V$. Jede Teilmenge muss mindestens ein Terminal enthalten. Zu jeder Zerlegung wird der *Schnitt* $\mathscr{C}\langle V_1, V_2\rangle = \{e_{ij} \mid i \in V_1 \text{ und } j \in V_2\}$ ermittelt, also die Menge der Kanten, die in V_1 beginnen und in V_2 enden. Nun wird eine binäre $k \times m$-Matrix $\mathbf{A} = (a_{ij})$ folgendermaßen

definiert: Wenn der i-te Schnitt ($i = 1, \dots, k$) die Kante $e_j \in E$ ($j = 1, \dots, m$) enthält, ist $a_{ij} = 1$; alle anderen a_{ij} sind 0. Jeder Kante e_j wird außerdem eine binäre Variable x_j zugewiesen. Ein Wert von $x_j = 1$ bedeutet, dass die Lösung (d. h. der SB) die Kante e_j enthält; andernfalls hat die Variable den Wert 0. Die Definitionen $\mathbf{x} = (x_1, \dots, x_m)^\mathsf{T}$ und $\mathbf{c} = (c_1, \dots, c_m)^\mathsf{T}$ ergeben schließlich eine besonders einfache Modellierung des SBP als binäres lineares Programm:

$$\min \mathbf{c}^\mathsf{T}\mathbf{x} \tag{2.5a}$$

u. d. N.

$$\mathbf{A}\mathbf{x} \geq \mathbf{1}^m \tag{2.5b}$$

$$x_j \in \{0, 1\} \qquad j = 1, \dots, m\,. \tag{2.5c}$$

Die Zielfunktion (2.5a) minimiert die Länge des Steinerbaums $\mathscr{L}(\mathscr{G}_\mathrm{s})$ („so wenige Kanten wie möglich“). Bedingung (2.5b) stellt sicher, dass eine Lösung aus einer einzigen Zusammenhangskomponente besteht (d. h., das je zwei Terminale miteinander verbunden sind – „so viele Kanten wie nötig“). Durch das Zusammenspiel von (2.5a) und (2.5b) entsteht eine Lösung minimaler Länge, die keine „überflüssigen“ Verbindungen zwischen zwei Knoten aufweist, also ein Baum. Bedingung (2.5c) legt die x_j als binäre Variablen fest. $\mathbf{1}^m$ ist der Einheitsvektor des $\mathbb{R}^m$ (vgl. Aneja, 1980, S. 168).

Variante (2), die Darstellung des SBP als *Minimalgerüstproblem in einem erweiterten Graphen*, geht einen völlig anderen Weg. Sie erweitert zunächst den ursprünglichen Graphen $\mathscr{G} = [V, E, c]$ folgendermaßen zu einem Graphen $\mathscr{G}_0 = [V_0, E_0, c_0]$ (vgl. Ralphs und Galati, 2006, S. 105):

- $V_0 = V \cup \{0\}$ – Zur Knotenmenge V wird ein weiterer Knoten 0 hinzugefügt.
- $E_0 = E \cup \{[0, i] \mid i \in S \cup \{t\}\}$ – Die Kantenmenge E wird um die Kanten $[0, i]$ zwischen Knoten 0 und jedem Steinerknoten $i \in S$ sowie einem beliebigen Terminal $t \in T$ erweitert.
- $\forall i \in S \cup \{t\} : c_{0i} = 0$ – Jede neue Kante in E_0 erhält das Gewicht 0.

Nun wird das Minimalgerüst von $\mathscr{G}_0$ unter einer zusätzlichen Bedingung ermittelt: Wenn ein Steinerknoten durch eine Kante mit dem Knoten 0 verbunden ist, dann darf keine andere Kante mit diesem Steinerknoten inzident sein. Das resultierende Minimalgerüst hat die in Abb. 2.5 gezeigte Gestalt: Knoten 0 ist mit einigen Steinerknoten und dem Knoten t durch je eine Kante verbunden, und t ist – wenn die Kante $[0, t]$ weggelassen wird – Blatt eines Teilbaums. Beasley (1989, S. 2–3) zeigte, dass dieser Teilbaum der SB des ursprünglichen

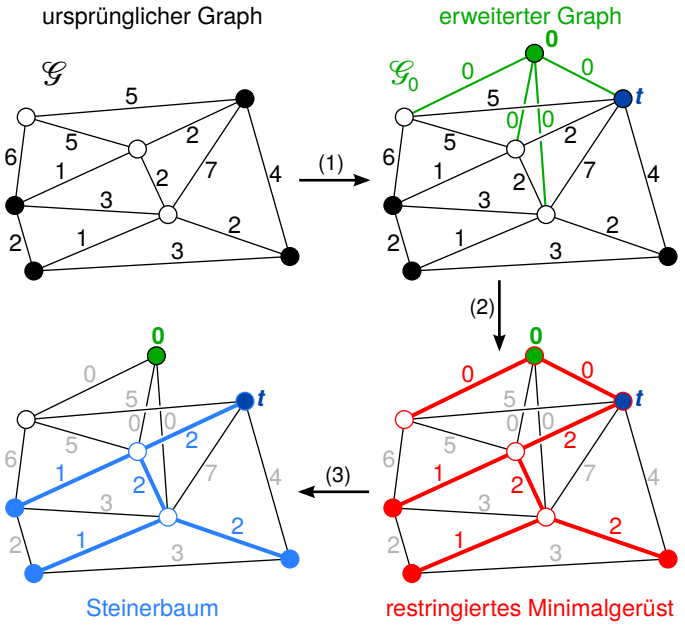


Abbildung 2.5.: Der Algorithmus von Beasley. Der Haupttext erläutert die folgenden Schritte näher: (1) Erweitern von $\mathcal{G}$ zu $\mathcal{G}_0$, (2) Lösen eines restringierten Minimalgerüstproblems in $\mathcal{G}_0$ und (3) Entfernen der Kante $[0, t]$. Jener Teilgraph, der nun den Knoten t enthält, ist der Steinerbaum von $\mathcal{G}$ (nach Beasley, 1989, S. 3; Maculan, 1987, S. 199).

Graphen $\mathcal{G}$ ist. Andernfalls könnte nämlich der Teilbaum durch einen kürzeren ersetzt und alle nicht verwendeten Steinerknoten mit Knoten 0 verbunden werden. Dies würde auf eine gültige Lösung des restringierten Minimalgerüstproblems führen, die noch dazu eine geringere Länge hätte – die ursprüngliche Lösung wäre also gar kein Minimalgerüst.

Dem restringierten Minimalgerüstproblem entspricht das folgende Modell:

$$\min \sum_{[i,j]\in E_0} c_{ij} x_{ij} \tag{2.6a}$$

u. d. N.

$$\sum_{[i,j]\in E_0} x_{ij} = |V_0| - 1 \tag{2.6b}$$

$$\sum_{[i,j]\in E(V_t)} x_{ij} \leq |V_t| - 1 \qquad V_t \subset V_0 \tag{2.6c}$$

$$x_{0i} + x_{ij} \leq 1 \qquad i \in S,\ j \in N(i) \tag{2.6d}$$

$$x_{ij} \in \{0, 1\} \qquad [i,j] \in E_0\ . \tag{2.6e}$$

Bedingung (2.6b) stellt sicher, dass die Lösung ein Baum ist (dieser muss eine Kante weniger als Knoten haben). Bei den Bedingungen (2.6c) handelt es sich um die sog. *Teiltourvermeidungsbedingungen von Dantzig, Fulkerson und Johnson* (DFJ-Bedingungen). Ursprünglich wurden diese Ungleichungen entwickelt, um Teiltouren im Problem des Handlungsreisenden zu vermeiden, weil sie Kreise in der Lösung verbieten. Da aber auch jeder Baum kreisfrei sein muss, werden die DFJ-Bedingungen ebenso verwendet, um verschiedene Baumprobleme zu modellieren. So enthält auch das Modell (2.6) für

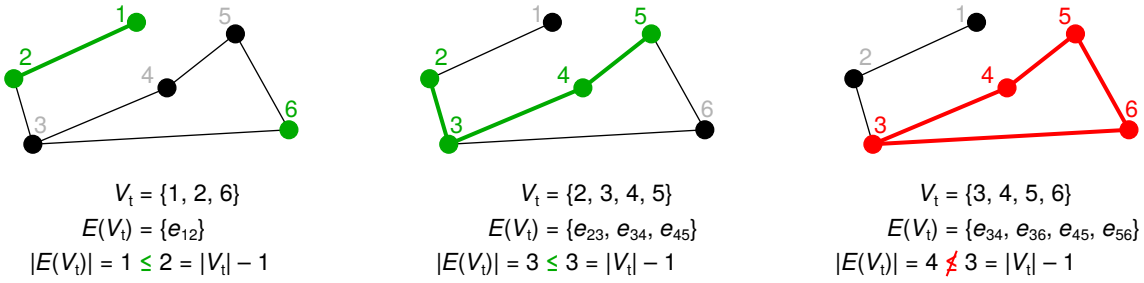


Abbildung 2.6.: Die DFJ-Bedingungen. Diese Menge von Ungleichungen verbietet, dass eine Teilmenge $V_t \subseteq V$ mit $|V_t| = n_t$ Knoten gleich viel oder mehr als n_t Kanten enthält. Das ist dann der Fall, wenn im von V_t induzierten Teilgraphen (farbig) mindestens ein Kreis vorkommt. Der gezeigte Graph verletzt die DFJ-Bedingungen zwar nicht für $V_t = \{1, 2, 6\}$ (links) oder $V_t = \{2, 3, 4, 5\}$ (Mitte), wohl aber für $V_t = \{3, 4, 5, 6\}$ (rechts).

jede mögliche nichtleere Knotenteilmenge $V_t \subset V_0$ eine Ungleichung der Form (2.6c) (Abb. 2.6). $E(V_t)$ ist die Menge der Kanten des von V_t induzierten Teilgraphen. (Ein induzierter Teilgraph oder *Untergraph* ist ein Teilgraph $\mathcal{G}_t = [V_t, E_t] \subseteq \mathcal{G}$, für den gilt: $\forall i, j \in V_t : [i, j] \in E \Rightarrow [i, j] \in E_t$.) Bildeten die Knoten in V_t einen Kreis, würde $E(V_t)$ genau $|V_t|$ Kanten enthalten und die Teiltourvermeidungsbedingung verletzen (vgl. Dantzig *et al.*, 1954, S. 397–398). Die Bedingungen (2.6d) gewährleisten, dass jeder durch eine Kante mit dem Knoten 0 verbundene Steinerknoten den Grad 1 hat. $N(i)$ ist die Menge der zu i benachbarten Knoten: $N(i) = \{j \mid [i, j] \in E\}$. Die Bedingungen (2.6e) legen schließlich die x_{ij} als binäre Variablen fest.

Obwohl die Modellierungen von Aneja (1980, S. 168–169) und Beasley (1989, S. 2–3) erlauben, exakte Lösungen für das SBP zu ermitteln, haben sie doch beide einen entscheidenden Nachteil: Die Anzahl der Nebenbedingungen (2.5b) bzw. (2.6c) steigt exponentiell mit der Knotenzahl (vgl. Aneja, 1980, S. 167; Beasley, 1989, S. 4). Bei n Knoten ist die Anzahl der möglichen Schnitte in Modell (2.5) sowie die Anzahl der möglichen Knotenteilmengen in Modell (2.6) jeweils von der Ordnung 2^{n-1}. (Bei 50 Knoten gibt es im ungünstigsten Fall schon $2^{49} \approx 10^{14}$ mögliche Knotenteilmengen!) Dadurch können bereits lineare Programme zu vergleichsweise kleinen SBP-Instanzen eine beachtliche Komplexität erreichen und nur noch mit hohem Rechenaufwand gelöst werden. Dies hat dazu geführt, dass eine Vielzahl von Heuristiken zur Lösung des SBP entwickelt wurde (vgl. Winter, 1987, S. 146–151; Voß, 1992, S. 46–70).

3. Restringierte Steinerbaum-Probleme

3.1. Überblick

Das im vergangenen Kapitel vorgestellte graphentheoretische SBP kann um unterschiedlichste Restriktionen erweitert werden. So enthält etwa das *Compendium on Steiner tree problems* von Hauptmann und Karpinski (2014) über 50 Varianten dieses Problems. Die folgenden Abschnitte beschreiben zwei dieser Restriktionen, nämlich die Budgetbeschränkung (Abschnitt 3.2) und die Sprungbeschränkung (Abschnitt 3.3). Abschnitt 3.4 fügt schließlich beide Beschränkungen zugleich zur Grundvariante des SBP hinzu und gelangt so zum SBP-EBS, dessen Anwendungen in Abschnitt 3.5 thematisiert werden.

3.2. SBP mit Gewinn

Die folgenden vom SBP abgeleiteten Probleme werden anhand eines *kanten- und knotengewichteten ungerichteten Graphen* formuliert. Ein solcher Graph $\mathcal{G} = [V, E, c, r]$ umfasst neben der schon bekannten Knotenmenge V, der Kantenmenge E und der Kantengewichtungsfunktion c noch eine *Knotengewichtungsfunktion* $r : V \to \mathbb{R}^{\geq 0}, i \mapsto r_i$ (vgl. Costa *et al.*, 2006, S. 99). Werden die Kantengewichte im betriebswirtschaftlichen Kontext als Kosten interpretiert, so entsprechen die Knotengewichte Erträgen. Daher werden im Folgenden die Summe der Knotengewichte eines Graphen als *Ertrag* $\mathscr{E}(\mathcal{G}) = \sum_{i \in V} r_i$ und die Summe der Kantengewichte als *Kosten* $\mathscr{L}(\mathcal{G}) = \sum_{[i,j] \in E} c_{ij}$ bezeichnet.

Problem 8

Der Betreiber eines Telefonnetzes muss entscheiden, welche Interessenten er an sein Netz anschließen soll. Der Betreiber weiß, über welche neuen Leitungen er die Kunden anschließen würde. Beim Errichten einer Leitung entstünden ihm Kosten, die proportional zur Leitungslänge sind. Der Betreiber kann außerdem einschätzen, wie lange jeder Interessent telefonieren würde, und erwartet daher einen bestimmten Ertrag pro Kunde. Welchen Interessenten soll der Betreiber überhaupt erst einen Vertrag anbieten? (nach Costa et al., 2006, S. 100)

⋆

Steinerbaum-Problem mit Gewinn: *Gegeben sei ein kanten- und knotengewichteter ungerichteter Graph $\mathcal{G} = [V, E, c, r]$ und eine (ggf. leere) Terminalmenge $T \subseteq V$. Dann soll ein Teilgraph $\mathcal{G}_s \subseteq \mathcal{G}$ gefunden werden, der alle Terminale verbindet und bei dem der Gewinn, also die Differenz zwischen Ertrag $\mathscr{E}(\mathcal{G}_s)$ und Kosten $\mathscr{L}(\mathcal{G}_s)$, ein Maximum*

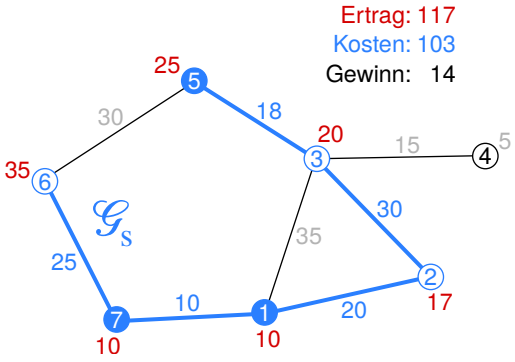


Abbildung 3.1.: Das SBP mit Gewinn. Der Ertrag eines Steinerknotens (weiße Füllung) beeinflusst, ob er in der Lösung vorkommt. $\mathcal{G}_s$ enthält Knoten 4 nicht, weil einem zusätzlichen Ertrag von 5 zusätzliche Kosten von 15 gegenüber stehen würden. Andererseits nimmt $\mathcal{G}_s$ „Umwege“ in Kauf, um den Gewinn zu steigern: e_{13} stellt zwar die kostenminimale Verbindung zwischen Knoten 1 und 3 dar (mit Kosten von 35). Anstatt dessen enthält $\mathcal{G}_s$ jedoch e_{12}, e_{23} und Knoten 2, weil die Kosten des Umwegs lediglich $20 + 30 - 17 = 33$ betragen.

> *annimmt (vgl. Costa et al., 2006, S. 99; Lucena und Resende, 2004, S. 277–278).*

Das SBP mit Gewinn sucht also einen Teilgraphen, der eine optimale Menge an Knoten enthält, und stellt somit gleichsam eine graphentheoretische Verkörperung des Wirtschaftlichkeitsprinzips dar (Abb. 3.1). Im obigen Beispiel enthält die optimale Menge an Knoten folglich nur jene Neukunden, die den Gewinn des Netzbetreibers steigern.

Ein wesentlicher Unterschied zum (einfachen) SBP besteht darin, dass die Terminalmenge T beim SBP mit Gewinn auch leer sein kann. In diesem Fall enthält der Graph ausschließlich Steinerknoten. Diese Variante wird in der Literatur auch unter dem Namen *preissammelndes SBP* behandelt (vgl. Costa *et al.*, 2006, S. 101). Während die Lösung eines einfachen SBP im Fall $T = \emptyset$ ein „leerer Teilgraph“ ist (weil dessen Länge null und somit minimal ist), enthält die Lösung des preissammelnden SBP mindestens einen Knoten, wenn nur $\exists i \in V : r_i > 0$. Mit einer nichtleeren Terminalmenge heißt das SBP mit Gewinn auch *knotengewichtetes SBP* (vgl. Costa *et al.*, 2006, S. 101).

Die Modellierung des SBP mit Gewinn als lineares Programm verwendet zwei Arten von binären Entscheidungsvariablen: Zum einen entspricht wieder eine Variable x_{ij} jeder Kante $[i,j] \in E$, zum anderen wird jedem Knoten $i \in V$ eine Variable y_i zugewiesen. y_i nimmt den Wert 1 an, falls die aktuelle Lösung den Knoten i enthält, und ist andernfalls 0. Diese Definitionen ergeben das folgende Modell (vgl. Lucena und Resende, 2004, S. 278–279):

$$\max \sum_{i \in V} r_i y_i - \sum_{[i,j] \in E} c_{ij} x_{ij} \tag{3.1a}$$

u. d. N.

$$\sum_{[i,j] \in E} x_{ij} = \sum_{i \in V} y_i - 1 \tag{3.1b}$$

$$\sum_{[i,j]\in E(V_t)} x_{ij} \leq \sum_{i\in V_t\setminus\{v\}} y_i \qquad v \in V_t,\ V_t \subseteq V,\ |V_t| \geq 2 \tag{3.1c}$$

$$y_i = 1 \qquad i \in T \tag{3.1d}$$

$$x_{ij} \in \{0,1\} \qquad [i,j] \in E \tag{3.1e}$$

$$y_i \in \{0,1\} \qquad i \in V \setminus T\ . \tag{3.1f}$$

Die Zielfunktion (3.1a) maximiert den Gewinn, d. h. die Differenz zwischen dem Gesamtertrag (Summe der Knotengewichte) und den Gesamtkosten (Summe der Kantengewichte). Falls das SBP mit Gewinn als Minimierungsaufgabe formuliert werden soll, lautet eine äquivalente Zielfunktion min $\sum_{[i,j]\in E} c_{ij}x_{ij} + \sum_{i\in V} r_i(1 - y_i)$. Diese Funktion minimiert dann die Summe aus Gesamtkosten und entgangenem Ertrag (vgl. Costa *et al.*, 2006, S. 100). Die für einen Baum notwendige (aber nicht hinreichende) Bedingung (3.1b) bewirkt, dass eine Lösung genau eine Kante weniger als Knoten enthält. Die Bedingungen (3.1c) sind *verallgemeinerte Teiltourvermeidungsbedingungen* (vgl. Goemans, 1994, S. 159). Für jede mögliche Knotenteilmenge $V_t \subseteq V$ mit mindestens zwei Knoten enthält das Modell $|V_t|$ Ungleichungen. Besonders anschauliche Ungleichungen ergeben sich für Teilmengen mit zwei Knoten: Der Menge $V_t = \{1,2\}$ entsprechen z. B. die beiden Ungleichungen $x_{12} \leq y_1$ und $x_{12} \leq y_2$. Die Lösung darf also nur dann die Kante zwischen den Knoten 1 und 2 enthalten, wenn sie auch diese beiden Knoten enthält. Wenn V_t nur aus Knoten besteht, die zur Lösung gehören, vereinfachen sich die Bedingungen (3.1c) zu den klassischen DFJ-Bedingungen (2.6c). Aufgrund Bedingung (3.1d) muss eine Lösung sämtliche Terminale enthalten. Die Bedingungen (3.1e) und (3.1f) legen die Variablen als binär fest. Alternativ zu (3.1e) können die Kantenvariablen x_{ij} sogar als reelle Zahlen im Intervall $[0;1]$ definiert werden ($0 \leq x_{ij} \leq 1, [i,j] \in E$) und nehmen dennoch nur die Werte 0 oder 1 an (vgl. Costa *et al.*, 2006, S. 101).

In manchen Fällen soll die beste Lösung des SBP mit Gewinn nicht die größte *Differenz* zwischen Ertrag und Kosten aufweisen, sondern das optimale *Verhältnis* dieser Größen. Dieser Erfordernis trägt das folgende Modell Rechnung, welches dem Modell (3.1) mit einer nichtlinearen Zielfunktion entspricht:

$$\max \frac{\sum_{i\in V} r_i y_i}{c_{\text{fix}} + \sum_{[i,j]\in E} c_{ij}x_{ij}} \quad \text{u. d. N.} \quad (3.1b)\text{–}(3.1f)\ . \tag{3.2}$$

c_{fix} sind Fixkosten, die unabhängig von der Gestalt des SB sind (vgl. Costa *et al.*, 2006, S. 104–105). Wenn in Modell (3.1) hingegen nur ein Summand in der Zielfunktion bleibt und der andere Summand als zusätzliche Restriktion ins Modell eingeht, ergeben sich zwei weitere Varianten des SBP mit Gewinn:

- Das *SBP mit Ertragsvorgabe* minimiert die Kosten des SB, während die Erträge eine

gegebene untere Schranke $r_{\min}$ erreichen müssen (vgl. Johnson *et al.*, 2000, S. 760):

$$\min \sum_{[i,j]\in E} c_{ij}x_{ij} \quad \text{u. d. N.} \quad (3.1b)\text{–}(3.1f) \quad \text{und} \quad \sum_{i\in V} r_i y_i \geq r_{\min} \,. \tag{3.3}$$

- Das *SBP mit Ertrag und Budgetbeschränkung* hingegen maximiert den Ertrag des SB, während die Kosten ein gegebenes Budget $c_{\max}$ nicht überschreiten dürfen (vgl. Costa *et al.*, 2006, S. 104):

$$\max \sum_{i\in V} r_i y_i \quad \text{u. d. N.} \quad (3.1b)\text{–}(3.1f) \quad \text{und} \quad \sum_{[i,j]\in E} c_{ij}x_{ij} \leq c_{\max} \,. \tag{3.4}$$

3.3. SBP mit Sprungbeschränkung

Voß (1999, S. 322–323) untersuchte eine völlig andere Art von Anforderungen an die Lösung des SBP, und zwar für Steinerbäume in gerichteten Graphen. (Zur folgenden Terminologie vgl. wiederum Bondy und Murty (1976, S. 171–172)). Ein *kantengewichteter Digraph* (gerichteter Graph) $\vec{\mathcal{G}} = \langle V, \vec{E}, c\rangle$ ist ein Tripel aus (1) einer Knotenmenge $V = \{q, 1, \dots, n-1\}$, (2) einer Pfeilmenge $\vec{E} = \{\vec{e}_{ij} = \langle i,j\rangle \mid \vec{e}_{ij}$ verbindet Knoten i und $j\}$ mit $|\vec{E}| = m$, und (3) einer nichtnegativen Pfeilgewichtungsfunktion $c : \vec{E} \to \mathbb{R}^{\geq 0}, \vec{e}_{ij} \mapsto c_{ij}$. Ein Knoten q zeichnet sich dadurch als *Quelle* aus, dass $\vec{\mathcal{G}}$ keine Pfeile mit q als Endknoten enthält. Im Gegensatz zu den Kanten eines ungerichteten Graphen sind die Pfeile eines Digraphen geordnete Paare, daher ist $\langle i,j\rangle \neq \langle j,i\rangle$ für $i \neq j$. Ein *Teildigraph* $\vec{\mathcal{G}}_{\mathrm{t}} \subseteq \vec{\mathcal{G}}$ ist analog zum Teilgraphen definiert. Ein Knoten j eines Digraphen $\vec{\mathcal{G}}$ ist von einem Knoten i aus *erreichbar*, wenn es in $\vec{\mathcal{G}}$ eine *Pfeilfolge* $\vec{P} = (i, \langle i, i+1\rangle, i+1, \dots, j-1, \langle j-1, j\rangle, j)$ von i nach j gibt. Eine Pfeilfolge ist ein *Weg*, wenn $\forall i,j \in \vec{P} : i \neq j$ gilt, und ein *Zyklus*, falls Anfangs- und Endknoten identisch sind (aber keine anderen Knoten). Wege mit den Knoten $i, \dots, j$ werden auch kurz als $\vec{P} = (i, \dots, j)$ geschrieben.

Problem 9

> *Ein Nachrichtendienst möchte das Netz eines Telefonieanbieters mieten, um seinen Kunden von seiner Zentrale aus Nachrichten zukommen zu lassen. Die laufenden Kosten sollen möglichst gering ausfallen. Dies würde der Nachrichtendienst grundsätzlich erreichen, indem er die jeweils kürzeste Verbindung von seiner Zentrale zu jedem Kunden mietet. Allerdings sollen die Kunden auch wirklich sämtliche an sie adressierten Nachrichten erhalten, was möglichst ausfallsichere Verbindungen zur Zentrale erfordert. Der Nachrichtendienst geht davon aus, dass jede Leitung im Netzwerk mit einer gewissen Wahrscheinlichkeit ausfallen kann, die unabhängig von der Leitungslänge und vom Ausfall einer anderen Leitung ist. Folglich darf nur eine beschränkte Anzahl an Leitungen zwischen der Zentrale und jedem Kunden liegen. Welches Netz soll der Nachrichtendienst*

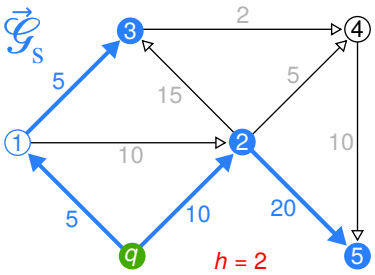


Abbildung 3.2.: Das SBP mit Sprungbeschränkung. Vom Teilbaum $\langle\{q, 2\}, \{e_{q2}\}\rangle$ aus ist es günstiger, Terminal 3 über den Weg $(q, 1, 3)$ zu erreichen als über den Weg $(2, 3)$. Daher enthält der gerichtete Steinerbaum $\vec{\mathcal{G}}_s$ den Steinerknoten 1. Obwohl $(2, 4, 5)$ der kürzeste Weg von Terminal 2 zu 5 ist, darf $\vec{\mathcal{G}}_s$ diesen Weg nicht enthalten, weil der Weg $(q, 2, 4, 5)$ die Sprungbeschränkung verletzt.

> *vom Telefonieanbieter mieten? (in Anlehnung an Voß, 1999, S. 321–322)*
>
> ⋆
>
> ***Steinerbaum-Problem mit Sprungbeschränkung:*** *Gegeben sei ein kantengewichteter Digraph $\vec{\mathcal{G}} = \langle V, \vec{E}, c\rangle$, eine nichtleere Terminalmenge $T \subseteq V$ und eine natürliche Zahl $h \leq |V|$. Dann soll ein Teildigraph $\vec{\mathcal{G}}_s \subseteq \vec{\mathcal{G}}$ mit minimalen Kosten $\mathcal{L}(\vec{\mathcal{G}}_s)$ gefunden werden, in dem alle Terminale von der Quelle aus erreichbar sind, wobei jeder entsprechende Weg höchstens h Pfeile umfasst (vgl. Voß, 1999, S. 322–323).*

Abb. 3.2 zeigt beispielhaft ein SBP mit Sprungbeschränkung. Der Name dieses Problems leitet sich von der Bezeichnung *Sprünge* (engl. *hops*) für die Anzahl von Pfeilen in einem Weg ab. Die maximal erlaubte Anzahl von Sprüngen kann ermittelt werden, wenn die einheitliche, unabhängige Ausfallwahrscheinlichkeit p_{Pfeil} einer Leitung bekannt ist (vgl. Voß, 1999, S. 322; Costa *et al.*, 2009, S. 141). Bei h Leitungen in einer Verbindung beträgt die Ausfallwahrscheinlichkeit für die gesamte Verbindung

$$p_{\text{Weg}} = 1 - (1 - p_{\text{Pfeil}})^h \,, \tag{3.5}$$

und wenn ein Grenzwert für p_{Weg} bekannt ist, ergibt sich h über die Umformung

$$h < \frac{\log(1 - p_{\text{Weg}})}{\log(1 - p_{\text{Pfeil}})} \,. \tag{3.6}$$

Verglichen mit den bisherigen Varianten des SBP besitzt das SBP mit Sprungbeschränkung bereits in manchen einfachen Graphen keine Lösung. Es kann nämlich der Fall eintreten, dass bereits jener Pfad von der Quelle zu einem Terminal mit der geringsten Anzahl von Kanten die Sprungbeschränkung verletzt (Abb. 3.3).

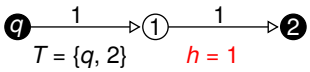


Abbildung 3.3.: Ein SBP mit Sprungbeschränkung, welches keine Lösung besitzt.

Auch das SBP mit Sprungbeschränkung kann als lineare Optimierungsaufgabe modelliert werden. Dazu wird analog zu oben jedem Pfeil $\langle i,j\rangle \in \vec{E}$ eine binäre Entscheidungsvariable x_{ij} und jedem Knoten $i \in V$ eine Entscheidungsvariable u_i zugeteilt. Für jeden Knoten $i \in V \setminus \{q\}$ bedeutet ein Wert von $u_i = 0$, dass i nicht in der aktuellen Lösung enthalten ist; ansonsten ist $u_i > 0$. Das resultierende Modell lautet:

$$\min \sum_{\langle i,j\rangle \in \vec{E}} c_{ij} x_{ij} \tag{3.7a}$$

u. d. N.

$$\sum_{i \in P(j)} x_{ij} = 1 \qquad j \in T \setminus \{q\} \tag{3.7b}$$

$$x_{jk} \leq \sum_{i \in P(j)} x_{ij} \qquad \langle j,k\rangle \in \vec{E},\ j \in S \tag{3.7c}$$

$$\sum_{i \in P(j)} x_{ij} \leq u_j \leq h \sum_{i \in P(j)} x_{ij} \qquad j \in V \tag{3.7d}$$

$$(h+1)x_{ij} + u_i - u_j + (h-1)x_{ji} \leq h \qquad \langle i,j\rangle \in \vec{E} \tag{3.7e}$$

$$x_{ij} \in \{0,1\} \qquad \langle i,j\rangle \in \vec{E}\,. \tag{3.7f}$$

Die Zielfunktion (3.7a) minimiert die Kosten des gerichteten Steinerbaums $\vec{\mathcal{G}}_s$. Die Bedingungen (3.7b) gewährleisten, dass jedes Terminal im Steinerbaum vorkommt und nur einen Vorgänger hat ($P(j)$ ist die Menge der Vorgänger von Knoten j: $P(j) = \{i \mid \langle i,j\rangle \in \vec{E}\}$). Aufgrund der Bedingungen (3.7c) darf nur dann ein Pfeil $\langle j,k\rangle$ von einem Steinerknoten j ausgehen, wenn j der Endknoten mindestens eines anderen Pfeils ist. Die Bedingungen (3.7d) schränken den Definitionsbereich einer jeden Knotenvariable u_j ein: u_j kann nur dann einen Wert ungleich 0 annehmen, wenn j der Endknoten eines Pfeils ist, und in diesem Fall ist $u_j \in [0;h]$; insbesondere ist $u_q = 0$, weil q per Definition keine Vorgänger hat. Die Bedingungen (3.7e) verbieten Zyklen in der Lösung. Es handelt sich dabei um die sog. *Teiltourvermeidungsbedingungen von Miller, Tucker und Zemlin* (MTZ-Bedingungen), welche ursprünglich ebenfalls für das Problem des Handlungsreisenden entwickelt wurden (vgl. Miller *et al.*, 1960, S. 326–328). Die Bedingungen (3.7f) schränken die x_{ij} auf binäre Werte ein.

In ihrer Grundform lauten die MTZ-Bedingungen für die Sprungbeschränkung

$$nx_{ij} + u_i \leq u_j + (n-1) \qquad i,j \in V \tag{3.8a}$$

$$1 \leq u_i \leq h \qquad i \in V\,. \tag{3.8b}$$

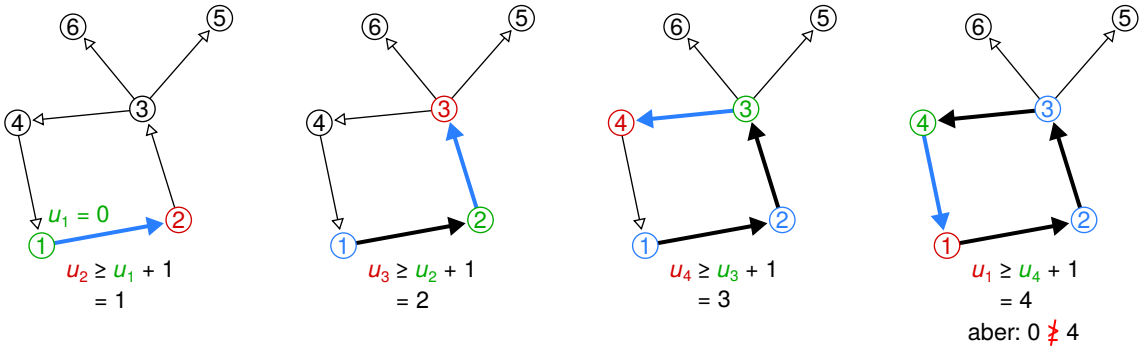


Abbildung 3.4.: Die MTZ-Bedingungen. Der gezeigte Graph enthält den Zyklus $(1, 2, 3, 4, 1)$. Nun wird beispielsweise die Knotenvariable u_1 mit dem Wert 0 initialisiert und die MTZ-Bedingung (3.8a) viermal angewendet (auf den jeweils blau gezeichneten Pfeil mit dem grünen Start- und roten Endknoten). Dadurch ergibt sich für den Knoten 1 die Ungleichung $u_1 \geq \big(((u_1+1)+1)+1\big)+1$ oder $0 \geq 4$, welche falsch ist.

Sie vermeiden Zyklen aufgrund der folgenden Überlegung (vgl. Gouveia, 1995, S. 961–962): Falls der Pfeil $\langle i, j \rangle$ zur aktuellen Lösung gehört ($x_{ij} = 1$), vereinfacht sich (3.8a) zu $u_i \leq u_j - 1$ oder $u_j \geq u_i + 1$. Diese Ungleichungen stellen also sicher, dass die Knotenvariablen entlang eines Wegs von Knoten zu Knoten größer werden. In einem Kreis aus k Knoten würde eine wiederholte Anwendung von (3.8a) zu der Ungleichung $u_i \leq u_i - k$ führen, welche für keine Werte von u_i und k erfüllt ist (Abb. 3.4). Darüber hinaus verbieten die MTZ-Bedingungen Wege mit mehr als h Pfeilen: Angenommen, der Weg $(q, 1, \ldots, i)$ enthielte $h + 1$ Pfeile. Dann enthält der Weg $(1, \ldots, i)$ genau h Pfeile, und wiederholte Anwendung von (3.8a) führt zu der Ungleichung

$$u_1 \leq u_i - h \,. \tag{3.9}$$

Wegen (3.8b) muss aber u. a. $u_1 \geq 1$ und $u_i \leq h$ gelten. Selbst wenn diese beiden Ausdrücke mit Gleichheit erfüllt sind, führt (3.9) auf die falsche Aussage $1 \leq 0$ (vgl. Gouveia, 1995, S. 961–962). Die alternative Version (3.7e) der MTZ-Bedingungen geht auf Desrochers und Laporte (1991, S. 29) zurück. Gouveia (1995, S. 961–962) zeigte, dass der Wert einer Knotenvariable u_i in diesem Fall genau der Anzahl von Kanten entspricht, die den Knoten i von der Quelle trennen. Für zwei benachbarte Knoten i, j, welche in der Lösung durch einen Pfeil verbunden werden ($x_{ij} = 1$), ergeben sich nämlich die beiden folgenden Ungleichungen ((3.7e) wird einmal auf das Knotenpaar (i, j) und einmal auf das Paar (j, i) angewendet):

$$(i,j): \quad h + 1 + u_i - u_j \leq h \qquad \text{oder} \quad u_i + 1 \leq u_j \tag{3.10a}$$

$$(j,i): \quad u_j - u_i + h - 1 \leq h \qquad \text{oder} \quad u_i + 1 \geq u_j \,. \tag{3.10b}$$

Daraus folgt $u_i + 1 = u_j$, und die Knotenvariablen zweier durch einen Pfeil verbundenen Knoten unterscheiden sich genau um den Wert 1. Falls das Knotenpaar i, j durch keinen Pfeil verbunden wird ($x_{ij} = x_{ji} = 0$), folgen aus den Bedingungen (3.7e) die beiden Ungleichungen

$$(i,j): \quad u_i - u_j \leq h \tag{3.11a}$$

$$(j,i): \quad u_j - u_i \leq h\,. \tag{3.11b}$$

Da die Knotenvariablen ohnehin nur Werte von 0 bis h annehmen können, sind diese beiden Ungleichungen stets erfüllt. Schließlich können aufgrund der Bedingungen (3.7e) niemals x_{ij} und x_{ji} gleichzeitig 1 sein:

$$(i,j): \quad h + 1 + u_i - u_j + h - 1 \leq h \qquad \text{oder} \quad u_i + h \leq u_j \tag{3.12a}$$

$$(j,i): \quad h + 1 + u_j - u_i + h - 1 \leq h \qquad \text{oder} \quad u_i - h \geq u_j\,. \tag{3.12b}$$

Das Zusammenfassen dieser beiden Ungleichungen ergibt $u_i + h \leq u_i - h$ oder $h \leq -h$, was für alle $h \in \mathbb{N}^{>0}$ falsch ist (vgl. Gouveia, 1995, S. 961–962).

3.4. SBP mit Ertrag, Budget- und Sprungbeschränkung

Die Kombination des SBP mit Ertrag und Budgetbeschränkung (Modell (3.4)) und des SBP mit Sprungbeschränkung (Modell (3.7)) ergibt eine betriebswirtschaftlich wichtige Verallgemeinerung dieser beiden Probleme, nämlich das *SBP mit Ertrag, Budget- und Sprungbeschränkung* (SBP-EBS).

Problem 10

> ***Steinerbaum-Problem mit Ertrag, Budget- und Sprungbeschränkung:*** *Gegeben sei ein kanten- und knotengewichteter (gerichteter oder ungerichteter) Graph $\mathcal{G} = [V, E, c, r]$ mit einem als Quelle q ausgezeichneten Knoten, eine natürliche Zahl $c_{\max}$ sowie eine natürliche Zahl $h \leq |V|$. Dann soll ein Teilgraph $\mathcal{G}_s \subseteq \mathcal{G}$ mit maximalem Ertrag $\mathcal{E}(\mathcal{G}_s)$ gefunden werden, der außerdem die folgenden drei Bedingungen erfüllt: (a) $\mathcal{G}_s$ enthält q; (b) die Kosten $\mathcal{L}(\mathcal{G}_s)$ betragen höchstens $c_{\max}$; und (c) jeder Knoten von $\mathcal{G}_s$ ist höchstens h Kanten von q entfernt (vgl. Costa et al., 2008, S. 68–69; Fu und Hao, 2014a, S. 209).*

Die Formulierung des SBP-EBS vermeidet den Begriff der Terminalmenge. Das einzige Terminal im klassischen Sinne ist die Quelle q, und die Lösung darf jeden anderen Knoten enthalten, solange sie einen maximalen Ertrag gewährleistet und den Nebenbedingungen genügt. Dafür verwendet die Formulierung auch im Falle eines ungerichteten Graphen den

Begriff der Quelle (als Synonym für das einzige Terminal). Folglich besitzt jedes SBP-EBS eine Lösung, die im Extremfall nur aus der Quelle besteht.

Costa *et al.* (2009, S. 142–147) schlugen mehrere Modellierungen des SBP-EBS vor, welche die bereits in den Abschnitten 3.2 und 3.3 vorgestellten Modelle erweitern. Ein lineares Programm für das SBP-EBS in einem ungerichteten Graphen $\mathcal{G}$, welches auf den DFJ-Bedingungen basiert, verwendet analog zu Modell (3.4) binäre Kanten- und Knotenvariablen x_{ij} bzw. y_i:

$$\max \sum_{i \in V} r_i y_i \tag{3.13a}$$

u. d. N.

$$\sum_{[i,j] \in E} x_{ij} \leq \sum_{i \in V} y_i - 1 \tag{3.13b}$$

$$\sum_{[i,j] \in E(V_\mathrm{t})} x_{ij} \leq \sum_{i \in V_\mathrm{t} \setminus \{v\}} y_i \qquad v \in V_\mathrm{t},\ V_\mathrm{t} \subseteq V,\ |V_\mathrm{t}| \geq 2 \tag{3.13c}$$

$$\sum_{[i,j] \in E} c_{ij} x_{ij} \leq c_{\max} \tag{3.13d}$$

$$\sum_{[i,j] \in K} x_{ij} \leq h \qquad K \in \mathcal{H}_{h+2} \tag{3.13e}$$

$$y_q = 1 \tag{3.13f}$$

$$x_{ij} \in \{0, 1\} \qquad [i,j] \in E \tag{3.13g}$$

$$y_i \in \{0, 1\} \qquad i \in V \setminus \{q\}\ . \tag{3.13h}$$

Die Zielfunktion (3.13a) und sämtliche Bedingungen außer (3.13e) wurden bereits in Abschnitt 3.2 erläutert (die Bedingungen (3.13f)–(3.13h) entsprechen den Bedingungen (3.1d)–(3.1f) mit $T = \{q\}$). Die einzige zusätzliche Bedingung (3.13e) ergänzt das Modell um die Sprungbeschränkung, und zwar vergleichbar mit den DFJ-Bedingungen. Die Menge $\mathcal{H}_{h+2} = \{K \mid K = (i_1 = q, \ldots, i_{h+2}), i \in V\}$ ist die Menge aller Ketten in $\mathcal{G}$, deren Anfangsknoten q ist und die genau $h+2$ Knoten enthalten. Da jede dieser Ketten $h+1$ Kanten enthält, verletzt sie gerade die Sprungbeschränkung. Eine gültige Lösung des SBP-EBS darf folglich zwar einige, aber eben nicht alle Kanten einer solchen Kette enthalten (Abb. 3.5). (Anmerkung: Das von Costa *et al.* (2009, S. 143) aufgestellte lineare Programm bedient sich einer Sprungbeschränkung der Form $\sum_{k=1}^{h+1} x_{i_k i_{k+1}} \leq h, (i_1 = q, \ldots, i_{h+2}) \in \mathcal{H}_{h+2}$. Die hier verwendete Variante (3.13e) ist aber äquivalent, weil $\mathcal{H}_{h+2}$ lediglich Ketten enthält, deren Knoten ja per Definition paarweise verschieden sind.)

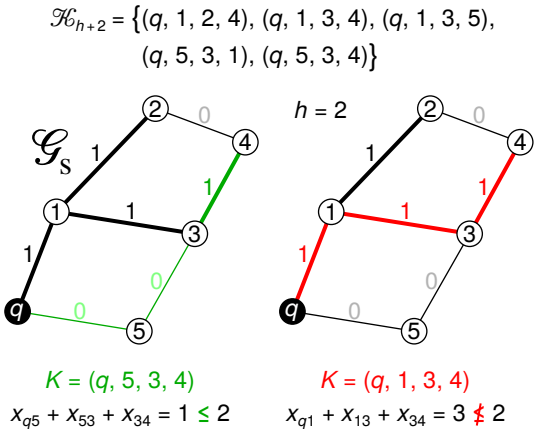


Abbildung 3.5.: Die Sprungbeschränkung im SBP-EBS. Im Graphen ist ein potentieller Steinerbaum $\mathcal{G}_s$ aus den Knoten $q, 1, \ldots, 4$ und den dicken Kanten eingezeichnet (alle Kanten sind mit dem Wert von x_{ij} beschriftet). $\mathcal{H}_{h+2}$ enthält fünf Ketten, von denen vier die entsprechende Ungleichung (3.13e) erfüllen, u. a. die Kette $(q, 5, 3, 4)$ (links). Allerdings kommen in $\mathcal{G}_s$ sämtliche Kanten der Kette $(q, 1, 3, 4)$ vor, weswegen der potentielle Steinerbaum die Sprungbeschränkung $h = 2$ verletzt (rechts).

Für einen gerichteten Graphen $\vec{\mathcal{G}}$ schlugen Costa *et al.* (2009, S. 144) u. a. das folgende Modell vor, welches die MTZ-Bedingungen beinhaltet:

$$\max \sum_{i \in V} r_i y_i \tag{3.14a}$$

u. d. N.

$$\sum_{i \in P(j)} x_{ij} = y_j \qquad j \in V \setminus \{q\} \tag{3.14b}$$

$$x_{jk} \leq \sum_{i \in P(j) \setminus \{k\}} x_{ij} \qquad \langle j,k \rangle \in \vec{E},\ j \neq q \tag{3.14c}$$

$$\sum_{\langle i,j \rangle \in \vec{E}} c_{ij} x_{ij} \leq c_{\max} \tag{3.14d}$$

$$u_q = 0 \tag{3.14e}$$

$$y_i \leq u_i \leq h y_i \qquad i \in V \setminus \{q\} \tag{3.14f}$$

$$(h+1)x_{ij} + u_i - u_j + (h-1)x_{ji} \leq h \qquad \langle i,j \rangle \in \vec{E} \tag{3.14g}$$

$$x_{ij} \in \{0,1\} \qquad \langle i,j \rangle \in \vec{E} \tag{3.14h}$$

$$y_q = 1 \tag{3.14i}$$

$$y_i \in \{0,1\} \qquad i \in V \setminus \{q\}\,. \tag{3.14j}$$

Die Nebenbedingungen dieses Modells entsprechen jenen, die bereits in Abschnitt 3.3 für das SBP mit Sprungbeschränkung erläutert wurden. Im Unterschied zu Modell (3.7) verwendet Modell (3.14) aber zwei verschiedene Entscheidungsvariablen für jeden Knoten i: Zum einen die binäre Variable y_i, die angibt, ob die aktuelle Lösung den Knoten enthält ($y_i = 1$) oder nicht ($y_i = 0$), zum anderen die reelle Variable u_i, deren Wert der Anzahl von Kanten zwischen i und q entspricht. Mithilfe der zusätzlichen Variablen y_i können manche

Bedingungen aus Modell (3.7) einfacher notiert werden: In einen (Steiner-)Knoten darf nur dann (höchstens) ein Pfeil münden, wenn dieser Knoten in der Lösung enthalten ist (Bedingungen (3.14b)). Ein Pfeil darf nur dann von einem Knoten ausgehen, wenn genau ein anderer Pfeil in diesen Knoten mündet, wobei kein Zyklus mit zwei Knoten entstehen darf (Bedingungen (3.14c)). Die Knotenvariable u_i darf nur dann einen Wert ungleich 0 annehmen, wenn i zur Lösung gehört, und ist dann auf das Intervall $[1; h]$ beschränkt (Bedingungen (3.14f)). Der Nachteil der zusätzlichen Entscheidungsvariablen ist, dass sie das Modell noch umfangreicher machen, als es ohnehin schon ist.

3.5. Betriebswirtschaftliche Anwendungen des SBP-EBS

Verschiedene betriebswirtschaftliche Planungsprobleme können als SBP-EBS oder als einer seiner Spezialfälle (SBP mit Gewinn, SBP mit Sprungbeschränkung) formuliert werden. So stellen etwa die Planung eines Fernwärmenetzes oder eines mehrstufigen Vertriebssystems zwei Anwendungsfälle des SBP-EBS dar. Im Folgenden werden diese Beispiele näher erläutert. Dabei wird jeweils in Klammern angegeben, wie die einzelnen Aspekte dieser Planungsprobleme in einem Graphen abgebildet werden können, sodass das Problem schließlich als SBP-EBS dargestellt wird. Tab. 3.1 fasst diese Analogien zusammen.

Fernwärmenetz In einer Stadt wird ein Fernwärmekraftwerk (Quelle) gebaut. Die Wärmeleitungen können aus baulichen Gründen lediglich entlang der Straßen (Kanten) verlegt werden. Kosten (Kantengewichte) entstehen durch das Verlegen der Leitungen und sind proportional zur Leitungslänge. Die Leitungen können sich an Straßenkreuzungen treffen (Knoten mit $r = 0$). Die Gebäude potentieller Abnehmer (Knoten mit $r > 0$) können grundsätzlich an das Fernwärmenetz angeschlossen werden, wenn sie sie den Ertrag des Betreibers mehren. Diesem stehen nur begrenzte finanzielle Reserven ($c_{\max}$) zur Verfügung, um sein Fernwärmenetzwerk zu errichten. Damit das Netzwerk ausfallsicher ist, darf außerdem nur eine begrenzte Anzahl von Leitungen (h) zwischen einem Kunden und dem Kraftwerk liegen (nach Ljubić *et al.*, 2004, S. 2). Analoge Planungsprobleme treten bei Telekommunikations-, Glasfaser- und Ad-hoc-Netzen auf (vgl. Costa *et al.*, 2009, S. 141; Canuto *et al.*, 2001, S. 50; Friedman und Parkes, 2002, S. 3–4).

Mehrstufiges Vertriebssystem Ein Hersteller von Handelsware (Quelle) plant sein Distributionsnetz. Da er die Distribution weitgehend auslagern will, möchte er verschiedene Absatzmittler (Kanten) mit dem Vertrieb seiner Waren beauftragen, die unterschiedlich hohe Provisionen verlangen (Kantengewichte). Jeder Absatzmittler betreut gewisse Kundenregionen (Knoten mit $r > 0$) und/oder fungiert seinerseits

als Intermediär (Knoten mit $r = 0$) für einen weiteren Absatzmittler. Die Vertriebsabteilung erwartet einen gewissen Absatz (Knotengewicht) in jeder Kundenregion. Allerdings stehen ihr nur begrenzte finanzielle Mittel ($c_{\max}$) zur Verfügung, um die Absatzmittler zu bezahlen. Darüber hinaus will die Abteilung die Anzahl der Vertriebsstufen (h) beschränken, um in engem Kontakt mit den Kunden zu bleiben. (Dieses Beispiel greift Überlegungen von Chawla *et al.* (2002, S. 1) auf, die sich mit Multicast-Netzwerken beschäftigten.)

Tabelle 3.1.: Anwendungen des SBP-EBS. Einzelne Aspekte der beiden im Text vorgestellten Planungsprobleme entsprechen verschiedenen Parametern des SBP-EBS.

	Fernwärmenetz	*Mehrstufiges Vertriebssystem*
q	Kraftwerk	Hersteller
$V \setminus \{q\}$	Straßenkreuzungen ($r_i = 0$), Gebäude ($r_i > 0$)	Verträge zwischen Absatzmittlern ($r_i = 0$), Kundenregionen ($r_i > 0$)
r_i	erwarteter Ertrag eines Abnehmers	Absatz in einer Region
$E, \vec{E}$	Leitungen	Absatzmittler
c_{ij}	Leitungskosten	Provision
$c_{\max}$	Budget für den Leitungsbau	Budget der Vertriebsabteilung
h	max. Anzahl von Leitungen zwischen Kraftwerk und Abnehmer	max. Anzahl von Distributionsstufen

4. Algorithmen zur Lösung des SBP-EBS

4.1. Überblick

Als *NP*-vollständiges Problem entzieht sich bereits die Grundform des SBP einem effizienten Lösungsalgorithmus (vgl. Karp, 1972, S. 95). Folglich gibt es vermutlich auch keine Algorithmen, um beliebig große Instanzen restringierter Steinerbaum-Probleme (wie etwa das SBP-EBS) effizient zu lösen. Verschiedene Autoren haben zwar exakte Lösungsverfahren für das SBP-EBS entwickelt, die u. a. auf Branch-and-Cut (vgl. Costa *et al.*, 2009, S. 146–156) oder Branch-and-Price (vgl. Sinnl, 2011, S. 35–52) basieren. Dennoch sind Heuristiken die Mittel der Wahl, um gute Lösungen für das SBP-EBS mit vergleichsweise wenig Rechenaufwand zu ermitteln, weil exakte Methoden bislang an großen Probleminstanzen scheitern (vgl. Costa *et al.*, 2009, S. 155–158; Sinnl, 2011, S. 53–74). Die Abschnitte 4.3 bis 4.5 stellen drei solche Heuristiken vor, Abschnitt 4.6 unterzieht sie einem kritischen Vergleich. Da alle diese Verfahren auf dem Prinzip der Nachbarschaftssuche fußen, erläutert Abschnitt 4.2 zunächst die Grundlagen dieses Optimierungsverfahrens.

4.2. Nachbarschaftssuche

Die kombinatorische Optimierung behandelt *Probleme* der Form $\mathsf{P} = (\mathcal{S}, f)$ (vgl. Blum und Roli, 2003, S. 269). Ein Problem ist also ein Tupel aus (1) einer Menge von möglichen Lösungen, dem *Suchraum* $\mathcal{S}$, und (2) einer *Bewertungs-* oder *Zielfunktion* $f : \mathcal{S} \to \mathbb{R}$, die jeder Lösung $s \in \mathcal{S}$ eine reelle Zahl (Bewertung) zuordnet. Eine *Lösung* s besteht aus n Variablen $x_1, \dots, x_n$, deren Definitionsmengen durch $D_1, \dots, D_n$ gegeben sind, und die einer Reihe von *Nebenbedingungen* unterliegen. Somit ist der Suchraum definiert als

$$\mathcal{S} = \left\{ s = \{x_1 = v_1, \dots, x_n = v_n\} \;\middle|\; \begin{array}{ll} v_i \in D_i & i = 1, \dots, n \\ h_j(x_1, \dots, x_n) = 0 & j = 1, \dots, J \\ g_k(x_1, \dots, x_n) \leq 0 & k = 1, \dots, K \end{array} \right\} . \tag{4.1}$$

Hier sind alle $J + K$ Nebenbedingungen Funktionen der Entscheidungsvariablen ($h, g : D_1 \times \cdots \times D_n \to \mathbb{R}$), die entweder null oder kleiner gleich null sein müssen (vgl. Yang, 2011, S. 77–78). Sowohl die Definitionsmengen als auch der Suchraum sind endlich oder zumindest abzählbar unendlich, weshalb Probleme dieser Klasse als „kombinatorisch“ bezeichnet werden. Das Ziel der kombinatorischen Optimierung besteht darin, eine Lösung $s^* \in \mathcal{S}$ mit einer „besten“ (z. B. möglichst niedrigen) Bewertung zu finden. Eine sol-

che *global optimale Lösung* ist daher durch $\forall s \in \mathcal{S} : f(s^*) \succ f(s)$ gekennzeichnet. Die Relation „$\succ$“ bedeutet hier „ist besser als“.

Damit Algorithmen zur Lösung kombinatorischer Optimierungsprobleme nicht alle möglichen Lösungen auf ihre Optimalität untersuchen müssen (was grundsätzlich möglich wäre), bedienen sie sich häufig des Konzepts der *lokalen Suche* oder *Nachbarschaftssuche* (vgl. Reiter und Sherman, 1965, S. 864–889). Dieser Lösungsansatz versucht, eine beste Lösung zu finden, indem er ausgehend von einer Startlösung solange „ähnliche“ Lösungen aufsucht, bis er keine bessere Lösung mehr finden kann. Welche Lösungen zu einer aktuellen Lösung s ähnlich sind, wird über die *Nachbarschaftsstruktur* N bestimmt:

$$N : \begin{cases} \mathcal{S} & \to & 2^{\mathcal{S}} \\ s & \mapsto & \{\nu_i \mid \nu_i = s \oplus \mathrm{Zug}_i(),\ i = 1, \dots, z\}\ . \end{cases} \tag{4.2}$$

$2^{\mathcal{S}}$ bezeichnet die Potenzmenge des Suchraums. Die Menge $N(s)$ ist die *Nachbarschaft* von s, ihre Elemente ν heißen *Nachbarn*. In einer Nachbarschaftsstruktur müssen Vorschriften bekannt sein, die angeben, wie ein Suchverfahren von der aktuellen Lösung zu jedem ihrer Nachbarn gelangt. Diese z Vorschriften oder *Operatoren* werden als *Züge* $\mathrm{Zug}_1(), \dots, \mathrm{Zug}_z()$ bezeichnet. Der symbolische Operator $\oplus$ zeigt an, dass die Anwendung des i-ten Zugs auf s den i-ten Nachbarn ergibt. Häufig gehört die aktuelle Lösung zu ihrer Nachbarschaft ($s \in N(s)$), was durch die Definition eines „neutralen“ Zugs $\mathrm{Zug}_0()$ ermöglicht wird, sodass $\forall s \in \mathcal{S} : s \oplus \mathrm{Zug}_0() = s$ gilt. Der Begriff der Nachbarschaft erlaubt auch die Definition eines *lokalen Optimums*. Eine Lösung s° ist hinsichtlich ihrer Nachbarschaft $N(s^\circ)$ lokal optimal, wenn $\forall s \in N(s^\circ) : f(s^\circ) \succ f(s)$ gilt (vgl. Blum und Roli, 2003, S. 269–270).

Ein möglicher Pseudocode für die Nachbarschaftssuche umfasst folgende Anweisungen:

Algorithmus 1: Nachbarschaftssuche

```
1  s = erzeugeStartlösung() // Initialisierung
2  repeat
3    wähle ein s' ∈ N(s) // Zug
4    ersetze s durch s'
5  until Abbruchbedingung = true
6  return s
```

Ein Durchlauf der `repeat`-Schleife wird als *Iteration* oder Schritt bezeichnet. Aus dieser bewusst einfach gehaltenen Beschreibung sind unzählige Varianten der Nachbarschaftssuche entwickelt worden (zu Übersichtsarbeiten vgl. Blum und Roli, 2003, S. 272–302; Yang, 2011, S. 77–84). Sie unterscheiden sich u. a. hinsichtlich der folgenden Aspekte:

- Initialisierung – Wie wird die Startlösung erzeugt?
- Nachbarschaft – Welche Lösungen sind benachbart?

- Züge – Welche Züge werden bevorzugt ausgeführt?
- Abbruchkriterium – Wann ist keine Verbesserung mehr möglich?

Das folgende Beispiel veranschaulicht die oben eingeführten (relativ abstrakten) Begriffe. Gegeben sei das Optimierungsproblem $\mathsf{P} = (\mathcal{S}, f)$ mit dem Suchraum $\mathcal{S} = \{(x, y) \mid x, y \in \{-10, 9, \dots, 9, 10\}\}$ und der Zielfunktion

$$f(x, y) = \frac{x - y}{2} + 4xy \exp \frac{-x^2 - y^2}{16}\,, \tag{4.3}$$

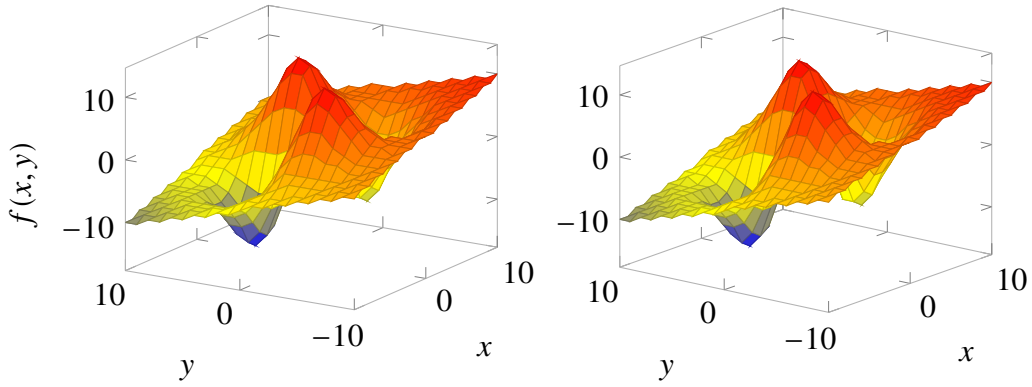
die minimiert werden soll. Der Suchraum umfasst also 441 Lösungen, nämlich die Tupel $(-10, -10)$ bis $(+10, +10)$. Abb. 4.1a zeigt den Graphen der Zielfunktion, die ein globales Optimum bei $(-3, 3)$ und mehrere lokale Optima besitzt, u. a. eines bei $(3, -3)$. Die Nachbarschaftsstruktur lautet explizit (unter Verwendung von Mengen)

$$N(x, y) = \{(x, y)\} \cup \begin{cases} \{(x+1, y),\ (x-1, y),\ (x, y+1),\ (x, y-1)\} & -9 \le x, y \le 9 \\ \{(x+1, y),\ (x, y+1),\ (x, y-1)\} & x = -10,\ -9 \le y \le 9 \\ \{(x+1, y),\ (x, y+1)\} & x = -10,\ y = -10 \\ \text{usw.} \end{cases} \tag{4.4}$$

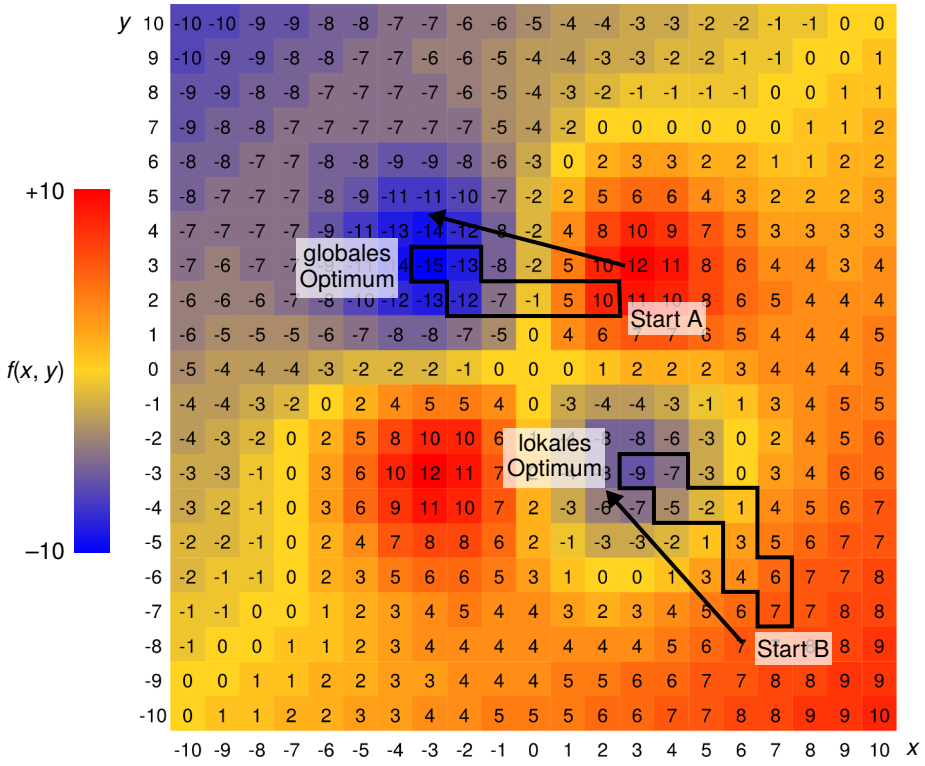
und implizit (unter Verwendung von Zügen)

$$\begin{aligned} N(x, y) &= \{(x', y') \mid (x', y') = (x, y) \oplus \mathtt{Zug}_i(),\ i = 0, \dots, 4\} \quad \text{mit} \\ \mathtt{Zug}_0() &: (x, y) \mapsto (x, y) \\ \mathtt{Zug}_1() \equiv \mathtt{Nord}() &: (x, y) \mapsto \begin{cases} (x, y+1) & y < 10 \\ (x, y) & \text{sonst} \end{cases} \\ \mathtt{Zug}_2() \equiv \mathtt{Süd}() &: (x, y) \mapsto \begin{cases} (x, y-1) & y > -10 \\ (x, y) & \text{sonst} \end{cases} \\ \mathtt{Zug}_3() \equiv \mathtt{Ost}() &: (x, y) \mapsto \begin{cases} (x+1, y) & x < 10 \\ (x, y) & \text{sonst} \end{cases} \\ \mathtt{Zug}_4() \equiv \mathtt{West}() &: (x, y) \mapsto \begin{cases} (x-1, y) & x > -10 \\ (x, y) & \text{sonst} \end{cases} . \end{aligned} \tag{4.5}$$

Ein erlaubter Zug entspricht also einem horizontalen oder vertikalen Schritt auf der in Abb. 4.1b dargestellten Fläche, solange der Definitionsbereich nicht verlassen wird. Die Startlösung wird zufällig gewählt. Ein Schritt besteht darin, dass der Algorithmus den Nachbarn mit dem kleinsten Zielfunktionswert wählt. Falls zwei oder mehr Nachbarn den gleichen Zielfunktionswert besitzen, wird jener Zug mit dem niedrigeren Index ausgeführt.



(a)



(b)

Abbildung 4.1.: Beispiel einer Nachbarschaftssuche. (a) Stereogramm des dreidimensionalen Funktionsgraphen zu (4.3). (b) Darstellung der kaufmännisch gerundeten Zielfunktion in der x, y-Ebene, wobei die Farben unterschiedlichen Funktionswerten entsprechen. Wenn Punkt A $= (2, 2)$ als Startlösung gewählt wird, erreicht der im Text beschriebene Algorithmus nach sechs Zügen tatsächlich das globale Optimum $s^* = (-3, 3)$ mit dem Zielfunktionswert -15. Falls der Algorithmus aber an Punkt B $= (7, -7)$ mit der lokalen Suche beginnt, terminiert er nach acht Schritten in einem lokalen Optimum $s^\circ = (3, -3)$ mit dem (suboptimalen) Zielfunktionswert -9.

Falls also beispielsweise sowohl $\mathtt{Zug}_1()$ als auch $\mathtt{Zug}_3()$ zu Nachbarn mit gleicher Bewertung führen, wird bevorzugt der nördliche Nachbar gewählt. Der Algorithmus endet, sobald er ein (lokales) Optimum erreicht. Wie aus Abb. 4.1b ersichtlich, ist aber nicht garantiert, dass dieser Algorithmus auch die global optimale Lösung findet – ein Problem, das allen Heuristiken anhaftet.

4.3. Gieriger Algorithmus

Der gierige Algorithmus von Costa *et al.* (2008, S. 70–72) findet einen budget- und sprungbeschränkten SB in einem ungerichteten Graphen $\mathscr{G}$. Dazu erzeugt er eine Folge von Bäumen in $\mathscr{G}$, welche die Budgetrestriktion (3.13d) und die Sprungrestriktion (3.13e) einhalten. Die anfängliche Lösung enthält lediglich die Quelle q, und in jeder Iteration wird die Lösung um die in einem gierigen Sinne „beste“ (d. h. kostengünstigste) Kette zu dem „besten“ (d. h. ertragreichsten) Knoten erweitert. Somit zählt dieser gierige Algorithmus zu den *Eröffnungsverfahren*. (Eine Anmerkung zur Terminologie: Im Kontext des einfachen SBP heißt ein Baum nur dann *Lösung* oder Steinerbaum schlechthin, wenn er die in Problem 6 beschriebenen Eigenschaften besitzt, also insbesondere *alle* Terminale einschließt. Bei der heuristischen Lösung des SBP-EBS ist eine Lösung hingegen *jeder* Baum, der die Ertrags- und Sprungbeschränkung nicht verletzt – also auch jener Baum, der lediglich aus der Quelle besteht.)

Damit ein Knoten j zur aktuellen Lösung $\mathscr{G}_\mathrm{t} = [V_\mathrm{t}, E_\mathrm{t}]$ überhaupt hinzugefügt werden kann, müssen drei Bedingungen erfüllt sein:

1. j muss den Ertrag der Lösung erhöhen: $r_j > 0$.
2. Höchstens h Kanten trennen j von q.
3. Die Kosten, die entstehen, wenn die Lösung um j ergänzt wird, dürfen das verbleibende Budget $c_\mathrm{max} - \mathscr{L}(\mathscr{G}_\mathrm{t})$ nicht überschreiten.

Die Bedingungen 2 und 3 finden ihren Niederschlag in der Ungleichung

$$\mathscr{D}(i, j, h - \eta_{qi}) \leq c_\mathrm{max} - \mathscr{L}(\mathscr{G}_\mathrm{t}) \qquad i \in V_\mathrm{t},\ j \notin V_\mathrm{t}\,. \tag{4.6}$$

$\mathscr{D}(i, j, h - \eta_{qi})$ sind die Kosten der kürzesten Kette zwischen einem Knoten i, der bereits zur aktuellen Lösung gehört, und dem Knoten j, der nicht zu Lösung gehört. Diese Kette darf höchstens $h - \eta_{qi}$ Kanten enthalten (η_{qi}: Anzahl der Kanten zwischen q und i). $\mathscr{D}$ kann mittels dynamischer Programmierung berechnet werden (vgl. Costa *et al.*, 2008, S. 70; Grötschel und Stephan, 2014, S. 230):

$$
\begin{aligned}
\mathscr{D}(i,i,0) &= 0 && i \in V \\
\mathscr{D}(i,j,0) &= +\infty && i,j \in V,\ i \neq j \\
\mathscr{D}(i,j,\eta) &= \min \left\{ \begin{array}{c} \mathscr{D}(i,j,\eta-1), \\ \min\limits_{k \in N(j)} \{\mathscr{D}(i,k,\eta-1) + c_{kj}\} \end{array} \right\} && i,j \in V,\ 1 \leq \eta \leq |V|-1\ .
\end{aligned} \tag{4.7}
$$

Die beste Kette verfügt über ein optimales Verhältnis von Ertrag zu Kosten, wobei Zähler und Nenner des folgenden Bruchs mit den Exponenten α bzw. β gewichtet werden:

$$
\max \frac{r_j^{\alpha}}{\mathscr{D}(i,j,h-\eta_{qi})^{\beta}} \qquad i \in V_t,\ j \notin V_t\ . \tag{4.8}
$$

Costa *et al.* (2008, S. 71) erhielten die besten Ergebnisse für $\alpha = 3$ und $\beta = 1$. Die sprungbeschränkte kürzeste Entfernung $\mathscr{D}$ muss nicht für alle i,j-Kombinationen berechnet werden: Es reicht, wenn die Gewichte aller Kanten von E_t auf null gesetzt werden und dann $\mathscr{D}(q,j,\eta)$ für alle $j \notin V_t$ und $1 \leq \eta \leq |V-1|$ berechnet wird. Außerdem können bei der Erweiterung der Lösung Kreise entstehen, welche vom gierigen Algorithmus aufgebrochen werden (Abb. 4.2).

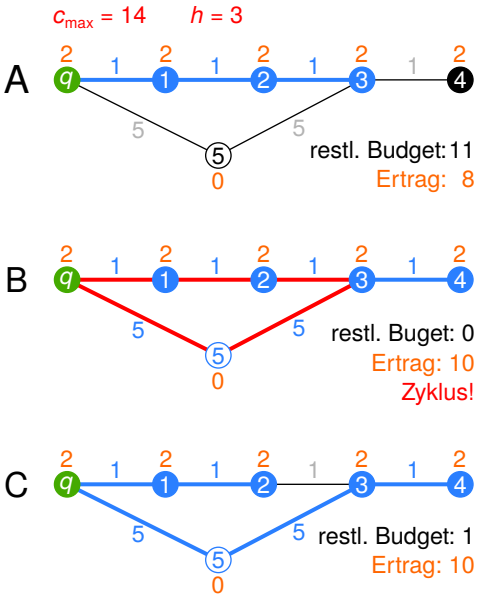


Abbildung 4.2.: Kreisvermeidung durch den gierigen Algorithmus. Die Lösung A kann nicht auf kostengünstigstem Weg um Knoten 4 erweitert werden, da die resultierende Kette zwischen q und 4 die Sprungbeschränkung verletzt. Daher fügt der Algorithmus die drei Kanten $[q,5]$, $[5,3]$ und $[3,4]$ hinzu, wodurch in Lösung B der Kreis $(q,1,2,3,5,q)$ entsteht. Der Algorithmus zerstört diesen Kreis, indem er einen Knoten v mit Grad 3 heraussucht (hier Knoten 3) und jene mit v inzidente Kante wieder aus der Lösung entfernt, welche zuerst zur Lösung hinzugefügt wurde (hier Kante $[2,3]$). Die Lösung C ist somit kreisfrei (nach Costa *et al.*, 2008, S. 71).

Der folgende Pseudocode fasst die oben erläuterten Schritte des gierigen Algorithmus zusammen (in Anlehnung an Costa *et al.*, 2008, S. 72). Abb. 4.3 zeigt, wie der Algorithmus eine kleine Probleminstanz löst.

Algorithmus 2: Gieriger Algorithmus

```
1 G_t = {q}, besteBewertung = 1 // Initialisierung
2 while besteBewertung ≠ 0 do
3   besteBewertung = 0, besteKosten = 0, besterKnoten = -1
4   for knoten not in V_t do
```

```
    kosten = 𝒟(q, knoten, h)
    bewertung = r_j^3 / c
    if 𝒦(𝒢_t) + kosten > c_max then
      bewertung = 0
    end if
    if bewertung ≥ besteBewertung then
      besteBewertung = bewertung
      besteKosten = kosten
      besterKnoten = knoten
    end if
  end for
  if besteBewertung > 0 then
    setze c_ij = 0 für alle Kanten der Kette zwischen q und knoten
    füge diese Kette zu 𝒢_t hinzu
    zerstöre etwaige Kreise in der Lösung
  end if
end while
return 𝒢_t
```

Der gierige Algorithmus ist eine Variante der Nachbarschaftssuche: Die Nachbarn der aktuellen Lösung sind all jene Bäume, die durch das Hinzufügen eines ertragreichen Knotens erzeugt werden können. Wie für einen gierigen Algorithmus üblich, führt das Suchverfahren jenen Zug aus, der in der aktuellen Nachbarschaft am vielversprechendsten ist, d. h. jenen mit dem besten Kosten-Nutzen-Verhältnis. Abb. A.1 im Anhang verdeutlicht dieses Vorgehen anhand des Suchgraphen zum Beispiel in Abb. 4.3.

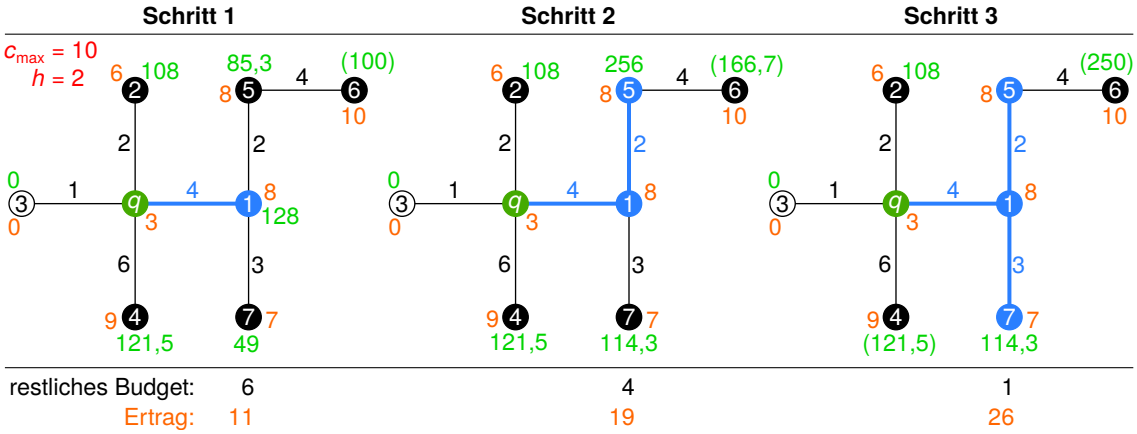


Abbildung 4.3.: Ein gieriger Algorithmus zur Lösung des SBP-EBS. Der Algorithmus fügt in den ersten beiden Schritten jeweils die im Sinne von Gl. (4.8) besten Knoten 1 bzw. 5 zur Lösung hinzu (orange, Knotengewichte; grün, gewichtetes Ertrag-Kosten-Verhältnis; blau, SB). In Schritt 3 würde Knoten 6 die Sprungbeschränkung und Knoten 4 die Budgetbeschränkung verletzen. Daher wählt der Algorithmus Knoten 7, der im Vergleich zu Knoten 2 eine bessere Bewertung erhält. Knoten 3 bringt keinen Ertrag ($r_3 = 0$) und wird daher niemals zu einer Lösung hinzugefügt.

4.4. Tabusuche

Die Tabusuche erweitert in ihrer einfachsten Form die Nachbarschaftssuche um eine *Tabuliste* $T(s) \subseteq N(s)$. Diese Liste bezeichnet Nachbarn der aktuellen Lösung s, die *tabu* sind – die aktuelle Iteration der lokalen Suche darf solche Nachbarn *nicht* als neue Lösung wählen. Die Tabuliste ergänzt die Nachbarschaftssuche also um ein „Gedächtnis“, welches Informationen aus den letzten t Iterationen speichert. Dadurch sollen insbesondere Zyklen vermieden werden (vgl. Glover, 1989, S. 190–206, 1990, S. 4–32).

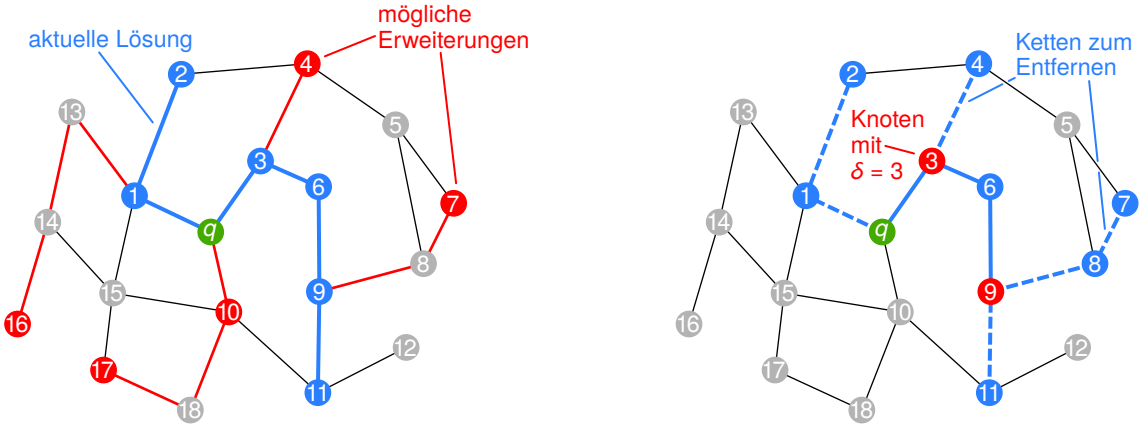
Im Gegensatz zu dem im letzten Abschnitt vorgestellten gierigen Algorithmus ist die Tabusuche von Costa *et al.* (2008, S. 74–76) zur Lösung des SBP-EBS ein *Verbesserungsverfahren*: Sie geht von einer Lösung dieses Problems aus und verbessert sie sukzessive. Die Tabusuche kennt drei Arten von Zügen:

1. Der Zug `Hinzufügen()` erweitert die aktuelle Lösung um einen ertragreichen Knoten und ähnelt einem Zug des oben vorgestellten gierigen Algorithmus. Allerdings darf dieser Zug vorübergehend die Budgetbeschränkung verletzten, damit die Tabusuche lokalen Minima entkommen und bislang unzugängliche Teile des Suchraums erforschen kann. Somit ist das Hinzufügen *ertrags*orientiert. Außerdem untersucht der Algorithmus in jeder Iteration nur einen zufällig ausgewählten Teil jener Knoten, die als Erweiterung infrage kommen (Abb. 4.4a). Hat das Suchverfahren erst einmal einen Knoten ausgewählt, so wird er einige Iterationen lang in die Tabuliste aufgenommen und kann folglich nicht durch den nachfolgend beschriebenen Zug aus der Lösung entfernt werden.

2. Der Zug `Entfernen()` ist das Gegenstück zum Zug Hinzufügen: Er entfernt einzelne Ketten aus Lösungen, falls diese die Budgetbeschränkung verletzen. Dieser Zug ist also *kosten*orientiert. Damit die Struktur der aktuellen Lösung durch dieses „Beschneiden” nur geringfügig gestört wird, zieht der Zug nur solche Ketten in Betracht, die von einem Blatt zum nächsten Knoten reichen, dessen Grad mindestens drei ist (Abb. 4.4b).

3. Der Zug `Zerstören()` wählt einen Knoten der aktuellen Lösung zufällig aus und entfernt den gesamten Teilbaum, dessen Wurzel der gewählte Knoten ist. Da dieser Zug tendenziell beträchtliche Änderungen in die aktuelle Lösung einführt, wird er nur ausgeführt, wenn der Algorithmus innerhalb von 100 Zyklen keine Lösungsverbesserung erzielen konnte.

In jeder Iteration berechnet der Algorithmus für alle Nachbarn $\mathcal{G}_{t\pm i}$ der aktuellen Lösung $\mathcal{G}_t$ einen *Strafwert* π, der proportional zum Ausmaß einer etwaigen Budgetverletzung ist:



(a) `Hinzufügen()` (b) `Entfernen()`

Abbildung 4.4.: Züge der Tabusuche. (a) Die aktuelle Lösung besteht aus dem blau gezeichneten Baum. Als Vorbereitung für den Zug `Hinzufügen()` wählt die Tabusuche einen Teil der Knoten, die noch nicht in der aktuellen Lösung enthalten sind (rot) und bestimmt jeweils die kostengünstigste Kette (rot), um jeden dieser Knoten mit der aktuellen Lösung zu verbinden. (b) In den letzten zwei Iterationen hat die Tabusuche die Knoten 4 und 7 zum Steinerbaum hinzugefügt. Im Rahmen des Zugs `Entfernen()` kommen nun vier Ketten in Frage (strichliert): $(q, 1, 2)$, $(3, 4)$, $(9, 8, 7)$ und $(9, 11)$. Falls die Tabuliste die Größe $t = 1$ hat (d. h., sie enthält die im vergangenen Zug hinzugefügte Kette), darf dieser Zug die Kette $(9, 8, 7)$ nicht entfernen.

$$\forall \mathcal{G}_{t\pm i} \in N(\mathcal{G}_t) : \pi(\mathcal{G}_{t\pm i}) = \max\left\{\phi \times (\mathcal{L}(\mathcal{G}_{t\pm i}) - c_{\max}),\, 0\right\} \tag{4.9}$$

($\phi > 0$: Proportionalitätsfaktor). Mittels dieses Strafwerts bewertet das Suchverfahren anschließend alle möglichen Hinzufügen- und Entfernen-Züge:

$$\gamma_{\pm i} = \pm r_i - \pi(\mathcal{G}_t) + \pi(\mathcal{G}_{t\pm i})\ . \tag{4.10}$$

$\gamma_{\pm i}$ ist die Bewertung des Zugs, der den Knoten i zur aktuellen Lösung hinzufügt bzw. aus der aktuellen Lösung entfernt; $\mathcal{G}_{t+i}$ bzw. $\mathcal{G}_{t-i} \in N(\mathcal{G}_t)$ sind die entsprechenden Nachbarn. Jener Zug mit der höchsten Bewertung wird ausgeführt. ϕ wird anschließend angepasst: Falls der Algorithmus in einer Iteration eine Lösung findet, welche die Budgetbeschränkung einhält, wird ϕ halbiert; im umgekehrten Fall (die neue Lösung verletzt die Budgetbeschränkung) wird ϕ verdoppelt (vgl. Costa *et al.*, 2008, S. 75).

Der folgende Pseudocode (nach Costa *et al.*, 2008, S. 76) entspricht den beschriebenen Abläufen:

Algorithmus 3: Tabusuche

```
1 G_t = erzeugeStartlösung() // z.B. mit dem gierigen Algorithmus
2 besterBaum = G_t, φ = 1 // Initialisierung
```

```
for zählvariable = 1, maxIterationen do
  berechne γ_{+i} für eine zufällig gewählte Teilmenge aller i ∉ V_t
  berechne γ_{-i} für alle in Frage kommenden Blätter i ∈ V_t
  G_t = arg max_{G_{t±i} ∈ N(G_t)} {γ_{±i}(G_{t±i})} // besten Zug ausführen
  setze diesen Zug für 5 Iterationen tabu
  if L(G_t) ≤ c_max then
    φ = φ/2
    if E(G_t) > E(besterBaum) then
      besterBaum = G_t
    end if
  else
   φ = 2φ
  end if
  if keine Lösungsverbesserung in 100 Iterationen then
    G_t = G_t ⊕ Zerstören()
  end if
end for
return besterBaum
```

4.5. Lokale Suche mit Ausbrechen

Die lokale Suche mit Ausbrechen (LSA) von Fu und Hao (2014a, S. 210–214) zur Lösung des SBP-EBS basiert auf der *iterierten lokalen Suche*. Diese Metaheuristik hat mit der (einfachen) wiederholten lokalen Suche gemeinsam, dass sie mehrmals hintereinander aufgerufen wird und dadurch eine Reihe von Lösungen liefert. Während die *wiederholte* lokale Suche aber in jedem Lauf von einer zufällig gewählten Startlösung ausgeht, verwendet die *iterierte* lokale Suche Informationen aus den vergangenen Läufen, um eine möglichst vielversprechende Startlösung zu wählen (vgl. Baxter, 1981, S. 815–819; Lourenço *et al.*, 2010, S. 322–327). Ein entsprechender Pseudocode lautet:

Algorithmus 4: Iterierte lokale Suche

```
chronik = ∅
s_0 = erzeugeStartlösung()
aktuellesOptimum = lokaleSuche(s_0)
repeat
  s_0 = störung(aktuellesOptimum, chronik)
  neuesOptimum = lokaleSuche(s_0)
  if neuesOptimum > aktuellesOptimum then // besser als
    aktuellesOptimum = neuesOptimum
  end if
  chronik = chronik ⊕ neuesOptimum
until Abbruchbedingung = true
return aktuellesOptimum
```

Die Funktion `störung()` (Zeile 5) ist das Markenzeichen der iterierten lokalen Suche: Sie erzeugt eine neue Startlösung, indem sie sowohl das aktuelle Optimum als auch Informationen aus vergangenen Iterationen berücksichtigt. Dazu werden die relevanten Informationen im Laufe jeder Iterationen in einer Chronik gespeichert, was der Operator $\oplus$ in Zeile 10 andeutet (vgl. Lourenço *et al.*, 2010, S. 325–327). (Bei der wiederholten lokalen Suche würde Zeile 5 hingegen einfach s_0 = `erzeugeStartlösung()` lauten.) Der Algorithmus zieht die Chronik auch zu Rate, wenn er in Zeile 7 entscheidet, ob die neue Lösung besser als die alte ist (Relation $\succ$).

Wie der folgende Algorithmus 5 bestätigt, ist die LSA von Fu und Hao (2014a, S. 210–214) letztendlich eine iterierte lokale Suche, die auf das SBP-EBS zugeschnitten wurde, wie es für eine *Meta*heuristik erforderlich ist (vgl. Blum und Roli, 2003, S. 271). Die wesentlichen Elemente dieses Algorithmus werden im Anschluss näher erläutert.

Algorithmus 5: Lokale Suche mit Ausbrechen

```
1  S^Elite = erzeugeElitelösungen()
2  σ = 1
3  s_0 = erzeugeStartlösung() // Anmerkung (1)
4  aktuellerBaum = lokaleSuche(s_0) // (2)
5  besterBaum = aktuellerBaum
6  repeat
7    s_0 = störung(aktuellerBaum, S^Elite, σ) // (3)
8    neuerBaum = lokaleSuche(s_0)
9    if neuerBaum ≻ besterBaum then // (4)
10     besterBaum = neuerBaum
11   end if
12   if zuNahe(neuerBaum, aktuellerBaum) then // (5)
13     σ = min{σ + 1, |T| − 1}
14   else
15     σ = max{σ − 1, 1}
16   end if
17   aktuellerBaum = neuerBaum
18 until Abbruchbedingung = true // (6)
19 return besterBaum
```

(1) Die LSA erzeugt zunächst eine Ausgangslösung mithilfe einer *probabilistischen Variante* des gierigen Algorithmus 2. Diese Variante bringt in jeder Iteration alle Knoten $j \notin V_t$, die zur aktuellen Lösung hinzugefügt werden können, nach Gl. (4.8) in eine Rangfolge. Rang 1 entspricht dabei dem Knoten mit dem besten Ertrag-Kosten-Verhältnis u. s. w. Wenn es k solche Knoten gibt, wählt der Algorithmus den Knoten auf Rang i mit der Wahrscheinlichkeit

$$p(i) = \begin{cases} (1-\theta)^{i-1} \times \theta & 2 \leq i \leq k \\ \theta + (1-\theta)^k & i = 1 \end{cases} \quad \text{, sodass} \quad \sum_{i=1}^{k} p(i) = 1 \,. \tag{4.11}$$

Der Parameter θ kann frei gewählt werden, wobei Fu und Hao (2014a, S. 215) die besten Ergebnisse mit $\theta = 0{,}3$ erzielten. Für $\theta = 1$ geht der probabilistische Algorithmus in den (deterministischen) gierigen Algorithmus 2 über. Die so erzeugte Startlösung ist ein so genannter *gesättigter* SB, der unter Beachtung der Budget- und Sprungrestriktion um keinen ertragreichen Knoten erweitert werden kann.

(2) Die sich anschließende *lokale Suche* verbessert die Startlösung, bis sie ein (meist lokales) Optimum erreicht. Die Nachbarschaftsstruktur ist durch drei Züge definiert:

$$\begin{aligned} \texttt{Zug}_1(i_1) &= \texttt{Löschen}(i_1) \oplus \texttt{Einfügen}() \\ \texttt{Zug}_2(i_1, i_2) &= \texttt{Löschen}(i_1) \oplus \texttt{Löschen}(i_2) \oplus \texttt{Einfügen}() \\ \texttt{Zug}_3(i_1, i_2, i_3) &= \texttt{Löschen}(i_1) \oplus \texttt{Löschen}(i_2) \oplus \texttt{Löschen}(i_3) \oplus \texttt{Einfügen}() \\ & \quad i_1, i_2, i_3 \text{ sind Blätter der aktuellen Lösung.} \end{aligned}$$

Der Teilzug `Löschen`(i) entfernt die gesamte Kette zwischen Blatt i und dem nächsten Knoten j mit $\delta(j) > 2$ oder $r_j > 0$ aus der aktuellen Lösung. Der Teilzug `Einfügen`$()$ entspricht dem probabilistischen gierigen Algorithmus; er erweitert eine ungesättigte Lösung um ertragreiche Knoten, bis die Lösung wieder gesättigt ist (Abb. 4.5). Da jeder Zug mit `Einfügen`$()$ abschließt, besteht der Suchraum ausschließlich aus gesättigten Steinerbäumen. Die Nachbarschaftsstruktur, die sich aufgrund der Züge 1–3 ergibt, hat allerdings einen gravierenden Nachteil: Wenn die aktuelle Lösung k Blätter enthält, ist die Anzahl ihrer Nachbarn von der Ordnung k^3. Bei umfangreichen Probleminstanzen würde die LSA also sehr lange dauern. Daher haben Fu und Hao (2014a, S. 213) ein Verfahren entwickelt, um die Anzahl von Nachbarn auf die Ordnung k zu reduzieren.

(3) Da das Ergebnis der Nachbarschaftssuche zumeist ein lokales Optimum darstellt, bricht das Suchverfahren nun mittels der Funktion `störung()` bzw. des Zugs `Stören()` aus diesem Optimum aus – daher der Name „lokale Suche mit Ausbrechen“. Die Störfunktion bezieht ihre Informationen aus einer Chronik, die aus zwei Elementen besteht. Zum einen bestimmt der *Störparameter* $\sigma \in [1\,;\,|T|-1]$, wie stark sich die neue Startlösung vom aktuellen gesättigten SB unterscheidet ($|T|$: Anzahl der Terminale, d. h. der Knoten mit $r > 0$). Zum anderen bedient sich die Störfunktion einer *Menge von Elitelösungen* S^{Elite}, um das Ausbrechen möglichst in die „richtige“ Richtung zu lenken, d. h. in Richtung des globalen Optimums. S^{Elite} wird zu Beginn der LSA mittels der Funktion `erzeugeElitelösungen()` erzeugt (zu Details vgl. Fu und Hao, 2014a, S. 213). Der Zug `Stören()` hat die Gestalt

$$\texttt{Stören}(\sigma, S^{\text{Elite}}) = \sigma \otimes \texttt{ProbabilistischesLöschen}(S^{\text{Elite}}) \oplus \texttt{Einfügen}() \,.$$

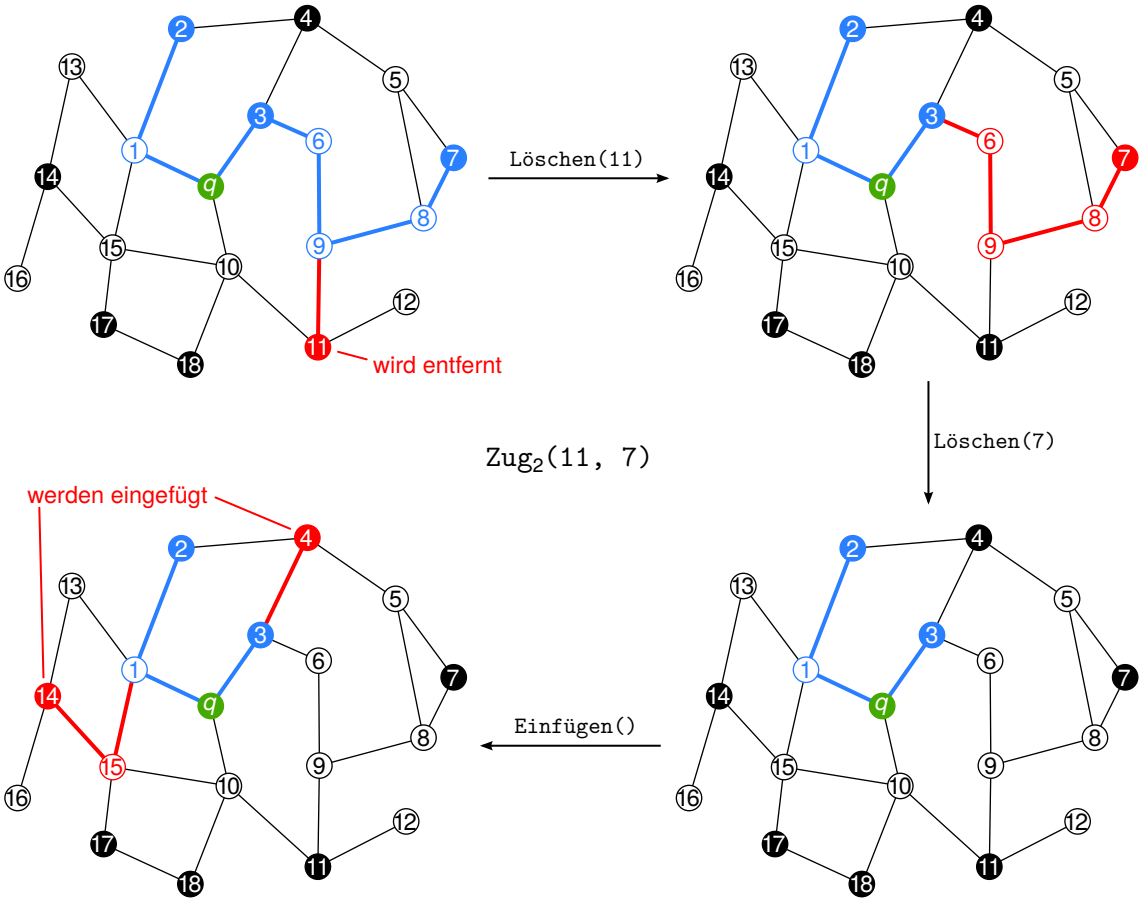


Abbildung 4.5.: Zug$_2$() der lokalen Suche mit Ausbrechen. Die beiden Teilzüge `Löschen(11)` und `Löschen(7)` entfernen zunächst die Terminale 11 bzw. 7 (rot). Der erste Teilzug stoppt an Knoten 9 (weil $\delta(9) > 2$), der zweite Teilzug an Knoten 3 (weil $r_3 > 0$). Im Anschluss fügt der Teilzug `Einfügen()` solange Ketten zwischen dem ungesättigten SB und nicht enthaltenen Terminalen hinzu, bis das Budget erschöpft ist. Hier wurde der beschnittene SB um die beiden Ketten $(1, 15, 14)$ und $(3, 4)$ ergänzt (in Anlehnung an Fu und Hao, 2014a, S. 212).

Der Teilzug `ProbabilistischesLöschen()`, der σ-mal ausgeführt wird, stellt eine probabilistische Version des oben erläuterten Teilzugs `Entfernen()` dar: Pro Aufruf entfernt er ein Terminal aus der Lösung, und zwar mit einer umso geringeren Wahrscheinlichkeit, je häufiger es in den Elitelösungen vorkommt. Angenommen, die binäre Variable $y_t(s)$ gebe für jede Elitelösung $s \in S^{\text{Elite}}$ an, ob sie Terminal t enthält ($y_t(s) = 1$) oder nicht ($y_t(s) = 0$). Dann entfernt `ProbabilistischesLöschen()` t mit der Wahrscheinlichkeit

$$p(t) = 1 - \frac{\sum_{s \in S^{\text{Elite}}} y_t(s)}{|S^{\text{Elite}}|} \,. \tag{4.12}$$

(4) Die neue Startlösung wird ebenfalls durch die unter (2) beschriebene Nachbarschaftssuche verbessert. Das neue Optimum ersetzt die beste bisher gefundene Lösung, falls es

einen höheren Ertrag oder bei gleichem Ertrag geringere Kosten aufweist.

(5) Die LSA aktualisiert die Chronik, indem sie den Störparameter σ anpasst: Falls das neue Optimum dem alten zu ähnlich ist, wird σ um 1 erhöht, damit das Ausbrechen zu Beginn der nächsten Iteration ein weiter entferntes Optimum erreichen kann. Ansonsten wird σ um 1 vermindert, damit die LSA die Umgebung der aktuellen Lösung näher erkundet. Damit findet sich hier das Prinzip wieder, dass eine Metaheuristik das optimale Verhältnis zwischen Intensivierung und Diversifizierung wählen soll (vgl. Blum und Roli, 2003, S. 271). Ob zwei Steinerbäume $\mathcal{G}_{t1}, \mathcal{G}_{t2}$ ähnlich sind, hängt vom Verhältnis ihrer Hamming-Entfernung $d^{\text{hamm}}(\mathcal{G}_{t1}, \mathcal{G}_{t2})$ zur durchschnittlichen Hamming-Entfernung der Elitelösungen ab. Letztere ist definiert als

$$\langle d_{\text{Elite}} \rangle = \frac{\sum\limits_{\mathcal{G}_{ti}, \mathcal{G}_{tj} \in S^{\text{Elite}},\, i<j} d^{\text{hamm}}(\mathcal{G}_{ti}, \mathcal{G}_{tj})}{\frac{1}{2} \times |S^{\text{Elite}}| \times \left(|S^{\text{Elite}}| - 1\right)} \,. \tag{4.13}$$

Falls also das Verhältnis $d^{\text{hamm}}(\mathcal{G}_{t1}, \mathcal{G}_{t2}) / \langle d_{\text{Elite}} \rangle$ einen bestimmten Wert unterschreitet (hier: 0,3), sind der alte und der neue SB zu ähnlich. Die *Hamming-Entfernung* zweier Teilgraphen $\mathcal{G}_{t1} = [V_{t1}, E_{t1}]$ und $\mathcal{G}_{t2} = [V_{t2}, E_{t2}]$ entspricht der Anzahl an Kanten, die nur in E_{t1}, aber nicht in E_{t2} enthalten sind und umgekehrt (vgl. Hamming, 1950, S. 154–155; Banks und Carley, 1994, S. 124–125):

$$d^{\text{hamm}}(\mathcal{G}_{t1}, \mathcal{G}_{t2}) = \tfrac{1}{2} \,\text{Spur}\Big[\left(\mathbf{A}(\mathcal{G}_{t1}) - \mathbf{A}(\mathcal{G}_{t2})\right)^2 \Big] \,. \tag{4.14}$$

Die $n \times n$-Matrix $\mathbf{A}(\mathcal{G}) = (a_{ij})$ ist die *Adjazenzmatrix* des Graphen $\mathcal{G}$, welcher aus n Knoten und der Kantenmenge E besteht: $a_{ij} = 1$, falls $[i,j] \in E$, und $a_{ij} = 0$ sonst.

(6) Die zentrale `repeat`-Schleife der LSA endet, falls eine der drei folgenden Bedingungen erfüllt ist: (a) Eine Lösung erreicht jenen Ertrag $r_{\max} = \sum_{i:\mathcal{D}(q,i,h)<\infty} r_i$, der angesichts der Sprungbeschränkung maximal möglich ist. (b) Der Algorithmus kann keine Lösungsverbesserung mehr erzielen, nachdem er eine festgelegte Anzahl neuer lokaler Optima aufgesucht hat (hier: 100). (c) Die maximal erlaubte Rechenzeit wurde überschritten.

Die LSA konnte das globale Optimum mehrerer standardisierter Probleminstanzen aus der OR-Library (vgl. Beasley, 1989, S. 8–14, 1990, S. 1069–1072) ermitteln, oder – falls dieses Optimum unbekannt war – zumindest Lösungen guter Qualität liefern (vgl. Fu und Hao, 2014a, S. 215–220). Um die Funktionsweise der LSA für diese Arbeit zu illustrieren, wurden mit ihrer Hilfe sechs Instanzen der restringierten SBP *steinb1* (Abb. A.2a im Anhang) und *steinb13* (Abb. A.2f) gelöst. Beasley (1989, S. 8–14) hat diese Probleme ursprünglich für das einfache SBP entworfen, und Costa *et al.* (2008, S. 76) haben sie für das SBP-EBS mit Knotengewichten versehen. Die Abb. A.2b–e zeigen die Ergebnisse für *steinb1* mit verschiedenen Werten von h und $c_{\max}$. Lösungen der größeren Instanz

steinb13 sind aus Abb. A.2g und h ersichtlich. Tab. A.1 fasst die in diesen Probleminstanzen verwendeten Parameter und die Eigenschaften der entsprechenden Lösungen zusammen.

4.6. Kritische Würdigung

Die LSA dürfte die derzeit beste Heuristik zur Lösung des SBP-EBS darstellen (Tab. 4.1). Fu und Hao (2014a, S. 215–220) demonstrierten die Leistungsfähigkeit dieses Algorithmus anhand der C-Serie von Steinerbaum-Problemen in der OR-Library (vgl. Beasley, 1989, S. 10). Dabei handelt es sich um 20 Graphen mit jeweils 500 Knoten und zwischen 625 und 12 500 Kanten. Bei den eher kleinen Instanzen mit 625 Kanten (z.B. *steinc1*), deren Optima bekannt sind, fand die LSA häufiger das globale Optimum als sämtliche Heuristiken von Costa *et al.* (2008, S. 70–76). Darüber hinaus war die mittlere Rechenzeit der LSA kürzer als jene des erfolgreichsten Algorithmus von Costa *et al.* (2008, S. 74–76), welchen die Tabusuche mit 10 000 Iterationen darstellt. Außerdem wiesen die lokal optimalen Lösungen der LSA im Durchschnitt den höchsten Ertrag auf. Die LSA erzielte ebenfalls bessere Ergebnisse als die Algorithmen von Costa *et al.* (2008, S. 70–76) bei größeren Instanzen (mit bis zu 2 500 Kanten, z. B. *steinc8*) und selbst bei den größten Instanzen (mit bis zu 12 500 Kanten, z. B. *steinc20*). Für jene Instanzen, deren globales Optimum unbekannt ist, fand die LSA Lösungen mit einem höheren Ertrag als dem besten bisher bekannten (vgl. Fu und Hao, 2014a, S. 216–217).

Tabelle 4.1.: Vergleich lokaler Suchverfahren zur Lösung des SBP-EBS. 60 von *steinc1–5* abgeleitete Instanzen des SBP-EBS wurden durch den gierigen Algorithmus, die Tabusuche mit 10 000 Iterationen und die LSA gelöst. Die mittlere Qualität der lokalen Optima gibt an, wie groß ihr durchschnittlicher Ertrag im Vergleich zum Ertrag im globalen Optimum ist. Die Daten stammen aus Fu und Hao (2014a, S. 216).

Algorithmus	*gefundene globale Optima*	*mittlere Qualität der lokalen Optima*	*mittlere Zeit (s)*
Gieriger Algorithmus	28	90,32 %	0,02
Tabusuche (10 000 Iterationen)	30	95,74 %	145,55
LSA (1 Lauf)	35	97,22 %	17,70
LSA (10 Läufe)	43	98,52 %	79,51

5. Zusammenfassung und Ausblick

Das graphentheoretische SBP, welches auf geometrische Fragestellungen des frühen 17. Jahrhunderts zurückgeht (vgl. Gueron und Tessler, 2002, S. 443), ist seit den 1970er Jahren Gegenstand zahlreicher Forschungsarbeiten (vgl. Hauptmann und Karpinski, 2014). In seiner einfachsten Form fragt das SBP nach der kostengünstigsten Möglichkeit, eine Teilmenge der Knoten eines kantengewichteten Graphen durch einen Baum – den Steinerbaum – zu verbinden (vgl. Hakimi, 1971, S. 114). Nicht nur für das SBP ohne zusätzliche Beschränkungen, sondern auch für seine restringierten Varianten sind zahlreiche praxisrelevante Anwendungen bekannt, wie in Kapitel 3 anhand mehrerer Beispiele gezeigt wurde. Insbesondere wurden in Abschnitt 3.5 Anwendungsbereiche des SBP mit Ertrag, Budget- und Sprungbeschränkung vorgestellt (vgl. Costa *et al.*, 2008, S. 68–69). Da sämtliche Steinerbaum-Probleme zur Klasse der *NP*-vollständigen Probleme gehören (vgl. Karp, 1972, S. 95), sind Heuristiken oftmals die Algorithmen der Wahl zur Lösung komplexer SBP-Instanzen (vgl. Voß, 1992, S. 46–70). Kapitel 4 stellte mehrere solcher Heuristiken vor, die alle auf dem Prinzip der lokalen Suche basieren und gute Lösungen für das SBP-EBS liefern. Insbesondere wurde die lokale Suche mit Ausbrechen von Fu und Hao (2014a, S. 210–214) detailliert behandelt, und ihre Leistungsfähigkeit anhand von standardisierten Probleminstanzen der OR-Library demonstriert (Abb. A.2).

Methoden zur Lösung des SBP-EBS bleiben ein aktives Feld wirtschaftswissenschaftlicher Forschung. So arbeiten beispielsweise Fu und Hao (2014b) an einem *memetischen Algorithmus*, der die Nachbarschaftssuche um die typischen Elemente eines genetischen Algorithmus erweitert (u. a. um eine Population von Lösungen und einen Crossover-Operator). Diese memetische Suche dürfte noch bessere Ergebnisse als die LSA liefern. Die Biologie könnte auch zu weiteren Algorithmen inspirieren, die graphentheoretische Probleme wie das SBP-EBS lösen. Auf welch überraschende Weise sie dies manchmal tut, haben Tero *et al.* (2010a, S. 439–442) anhand des geometrischen SBP gezeigt: Der Schleimpilz *Physarum polycephalum* wächst auf der Suche nach Nahrung zunächst flächig, bildet dann aber ein Netzwerk aus feinen Schläuchen zwischen gefundenen Nahrungsquellen. Um die Gesamtlänge dieses Netzwerks zu minimieren, führt *P. polycephalum* Steinerpunkte ein, an denen sich mehrere Verbindungen treffen (vgl. Tero *et al.*, 2010a, S. 439–440). In Anlehnung an dieses Verhalten haben Tero *et al.* (2010b, S. 109–123) ein mathematisches Modell zur Lösung des geometrischen SBP formuliert. Vielleicht könnte ein solches Modell mit entsprechenden Modifikationen auch für das SBP-EBS verwendet werden.

Anhang

Tabelle A.1.: Details zu Lösungen der SBP-EBS-Instanzen. V, T und E sind die Knoten-, Terminal- bzw. Kantenmengen des gegebenen Graphen $\mathcal{G}$. V_s und T_s sind die entsprechenden Mengen der jeweiligen Lösung $\mathcal{G}_s$.

Instanz	$\|V\|$	$\|T\|$	$\|E\|$	$\mathcal{L}(\mathcal{G})$	$\mathcal{C}(\mathcal{G})$	$c_{\max}$	h	$\|V_s\|$	$\|T_s\|$	$\mathcal{L}(\mathcal{G}_s)$	$\mathcal{C}(\mathcal{G}_s)$	*Abb. A.2*
steinb1	50	9	63	359	467	71	3	3	2	9	140	(b)
						71	6	16	8	68	403	(c)
						71	9	17	7	70	431	(d)
						35	6	11	6	33	341	(e)
steinb13	100	17	125	717	785	143	9	30	15	142	745	(g)
						71	6	13	9	71	452	(h)

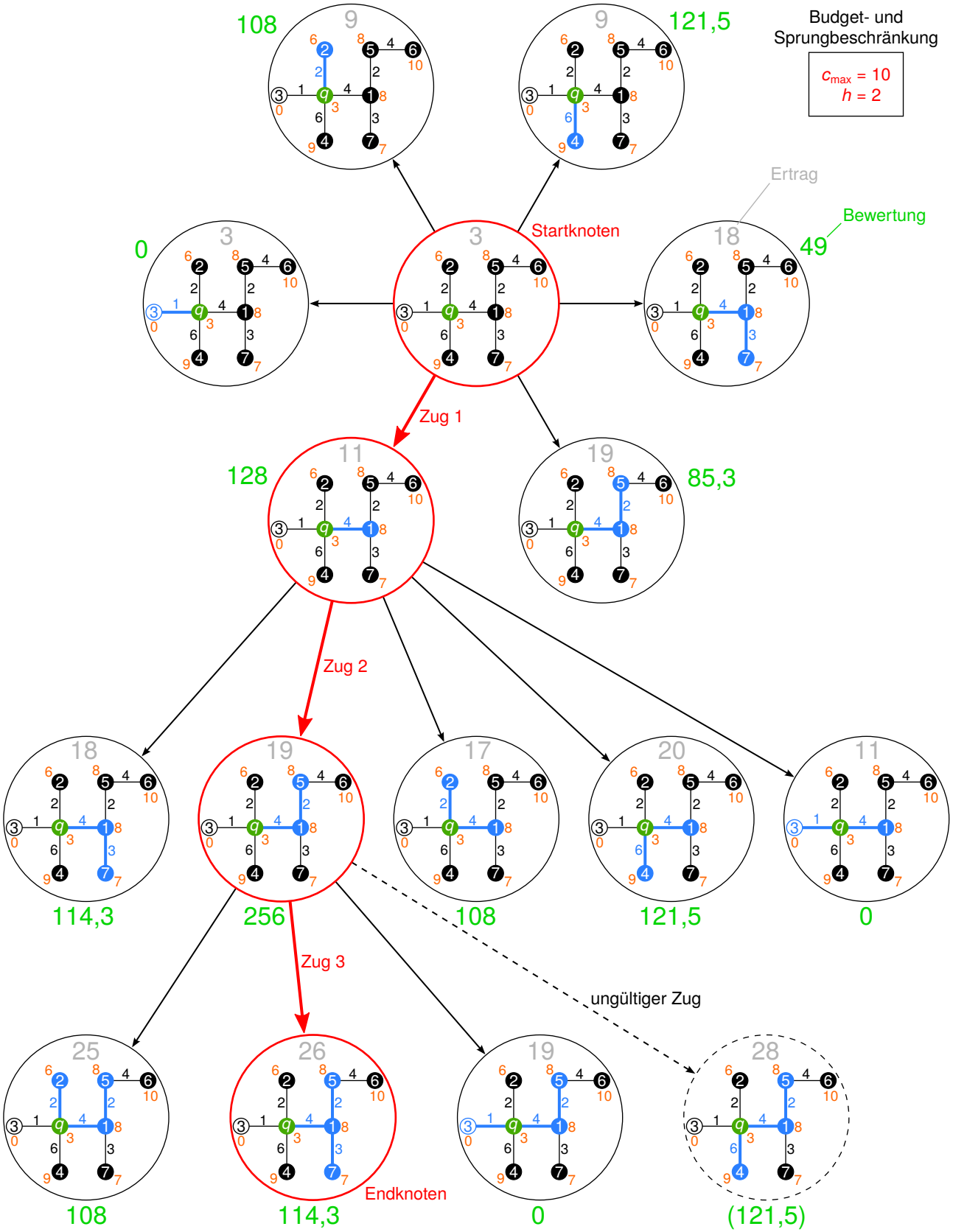
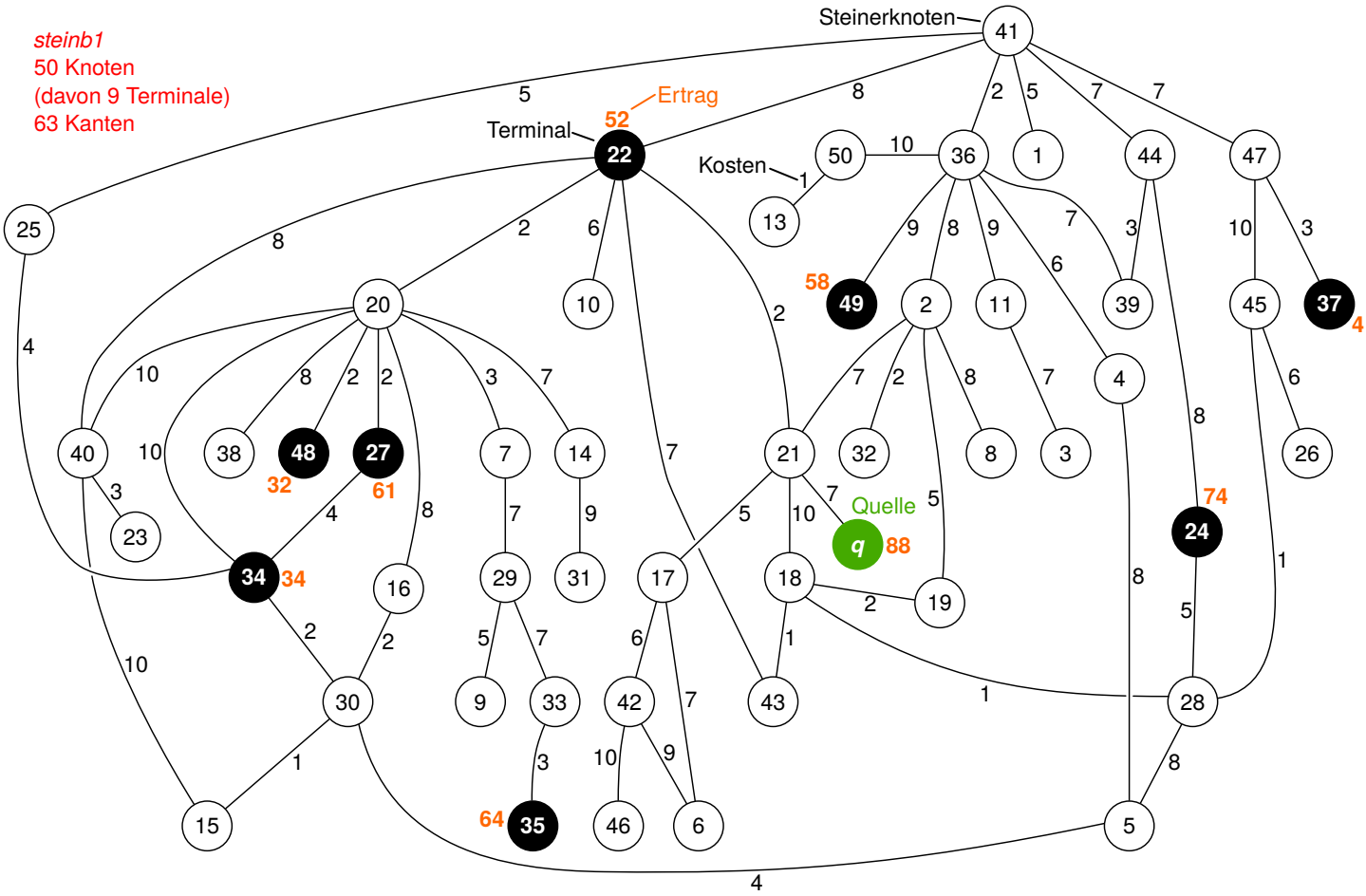
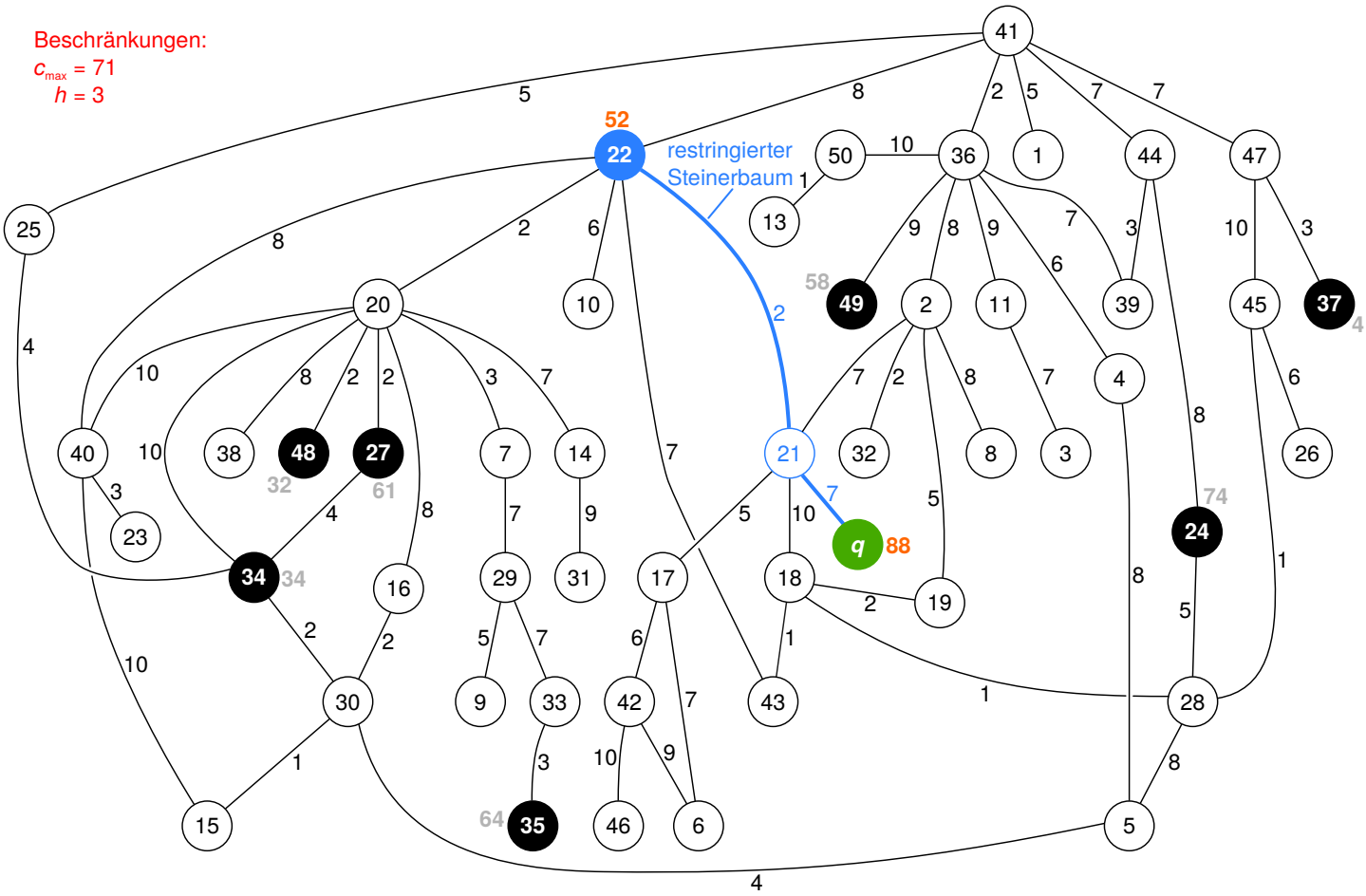


Abbildung A.1.: Suchgraph für den gierigen Algorithmus. Die Knoten in diesem Digraphen stellen die einzelnen Lösungen dar, Pfeile entsprechen den möglichen Zügen. Jeder Knoten ist außerdem mit dem Ertrag (grau) und Zielfunktionswert (grün) der entsprechenden Lösung beschriftet. Der gierige Algorithmus 2 wählt die Lösungen entlang des rot eingezeichneten Wegs und gelangt in drei Zügen vom Start- zum Endknoten.



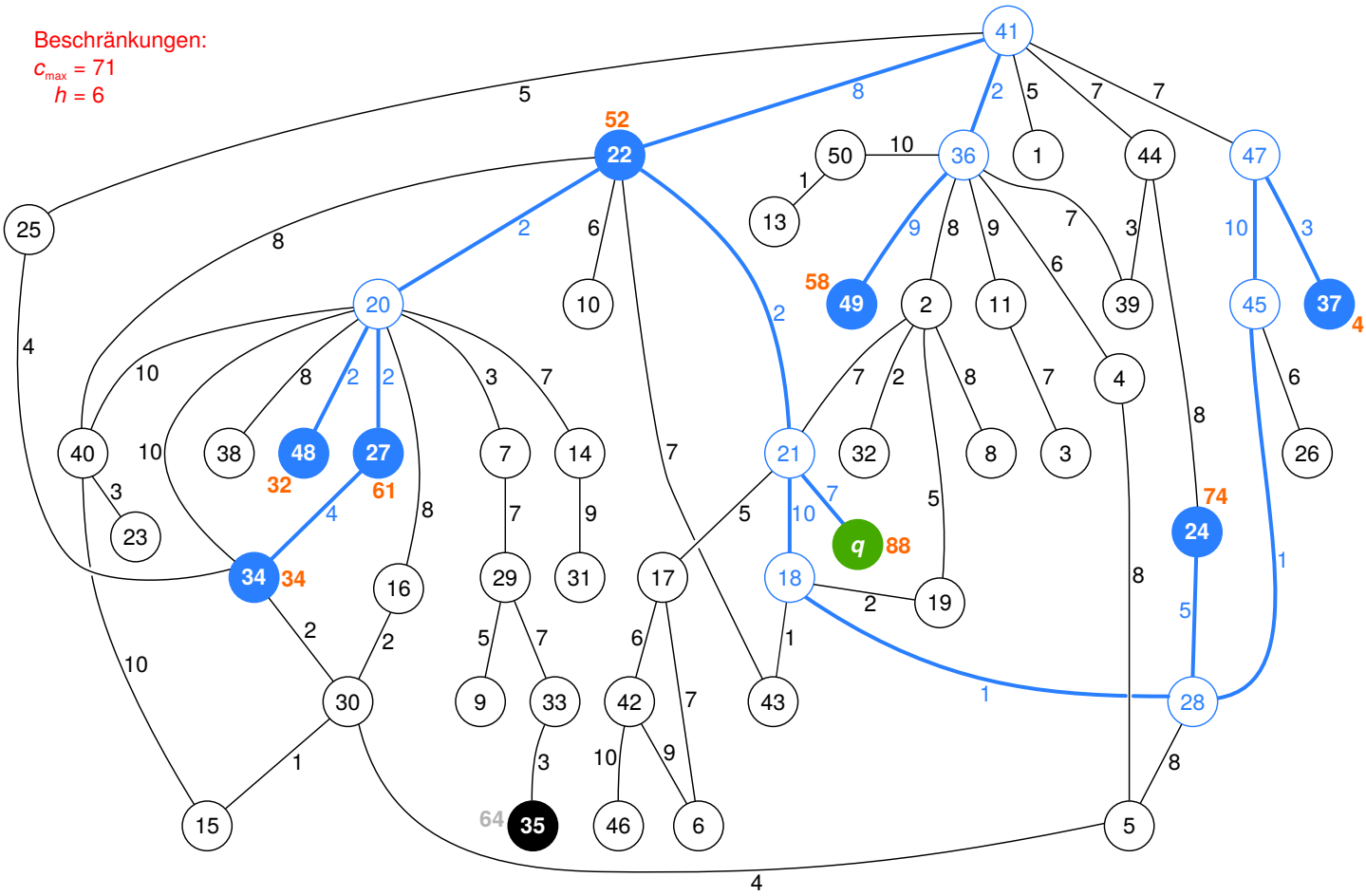
(a)

Abbildung A.2.: Lösungen des SBP-EBS durch die lokale Suche mit Ausbrechen. Die SBP-Instanzen *steinb1* (a) und *steinb13* (f) aus der OR-Library (vgl. Beasley, 1989, S. 8–14, 1990, S. 1069–1072) wurden folgendermaßen zu Instanzen des SBP-EBS erweitert (vgl. Costa *et al.*, 2008, S. 76): Den Terminalen wurden zufällige Erträge zugewiesen, die innerhalb des Intervalls [0; 100] gleichverteilt sind. Steinerknoten erhielten einen Ertrag von 0. Jede Instanz ist außerdem durch verschiedene Werte für die Parameter $c_{\max}$ und h gekennzeichnet. Der Algorithmus von Fu und Hao (2014a, S. 210–214) löste vier bzw. zwei Instanzen, die auf diese Weise von *steinb1* (b–e) oder *steinb13* (g, h) abgeleitet wurden. Die restringierten Steinerbäume sind jeweils blau hervorgehoben. Tab. A.1 enthält Details zu allen gezeigten Instanzen und deren Lösungen. Die Graphen wurden zuerst mit dem Programm *dot* aus dem Paket *Graphviz* v2.38.0 (vgl. Ellson *et al.*, 2002, S. 483–484) gezeichnet. Im Anschluss wurde das Layout eines jeden Graphen von Hand verbessert, um Kantenüberschneidungen möglichst zu vermeiden.



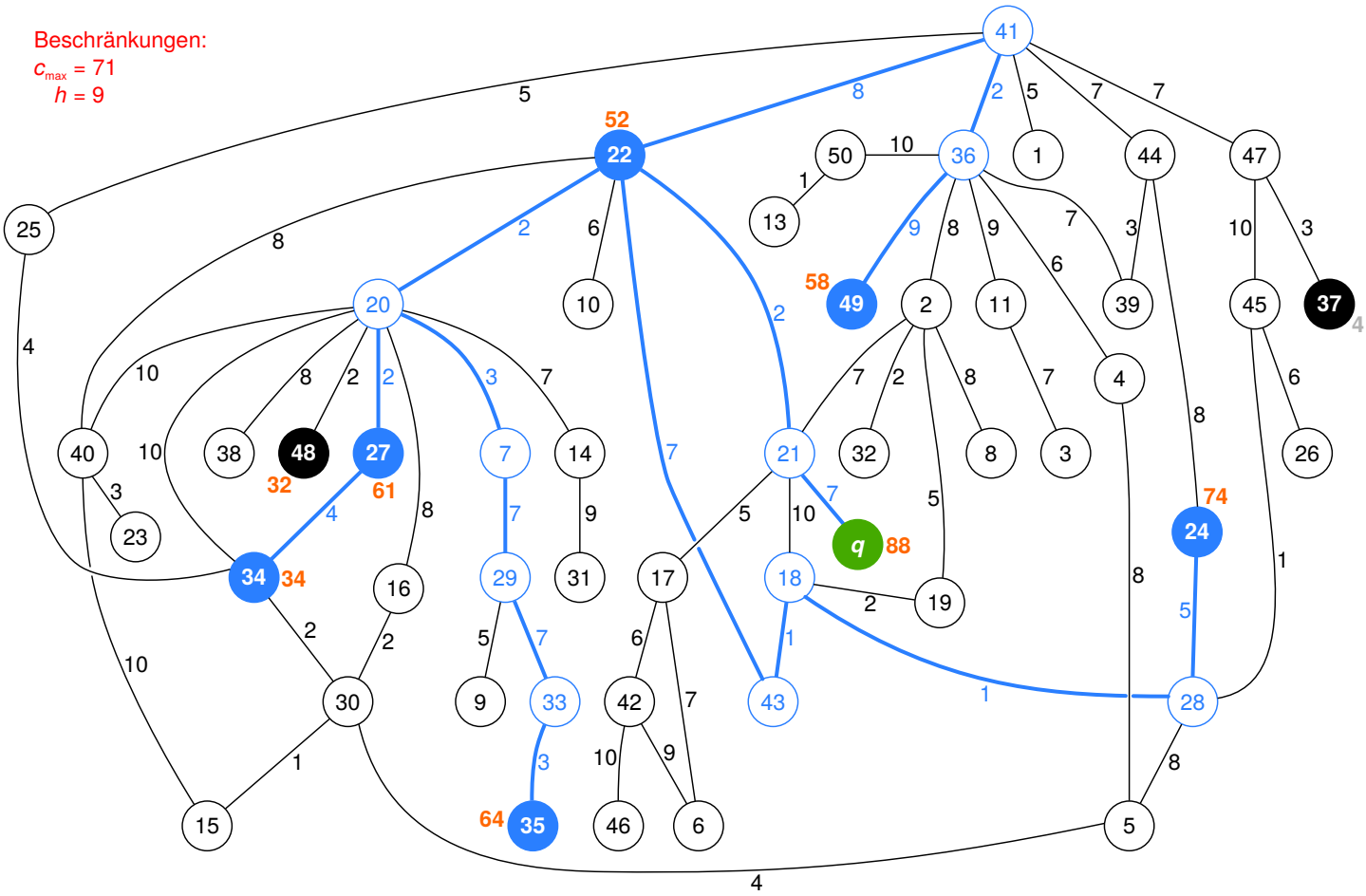
(b)

Abbildung A.2 (fortgesetzt)



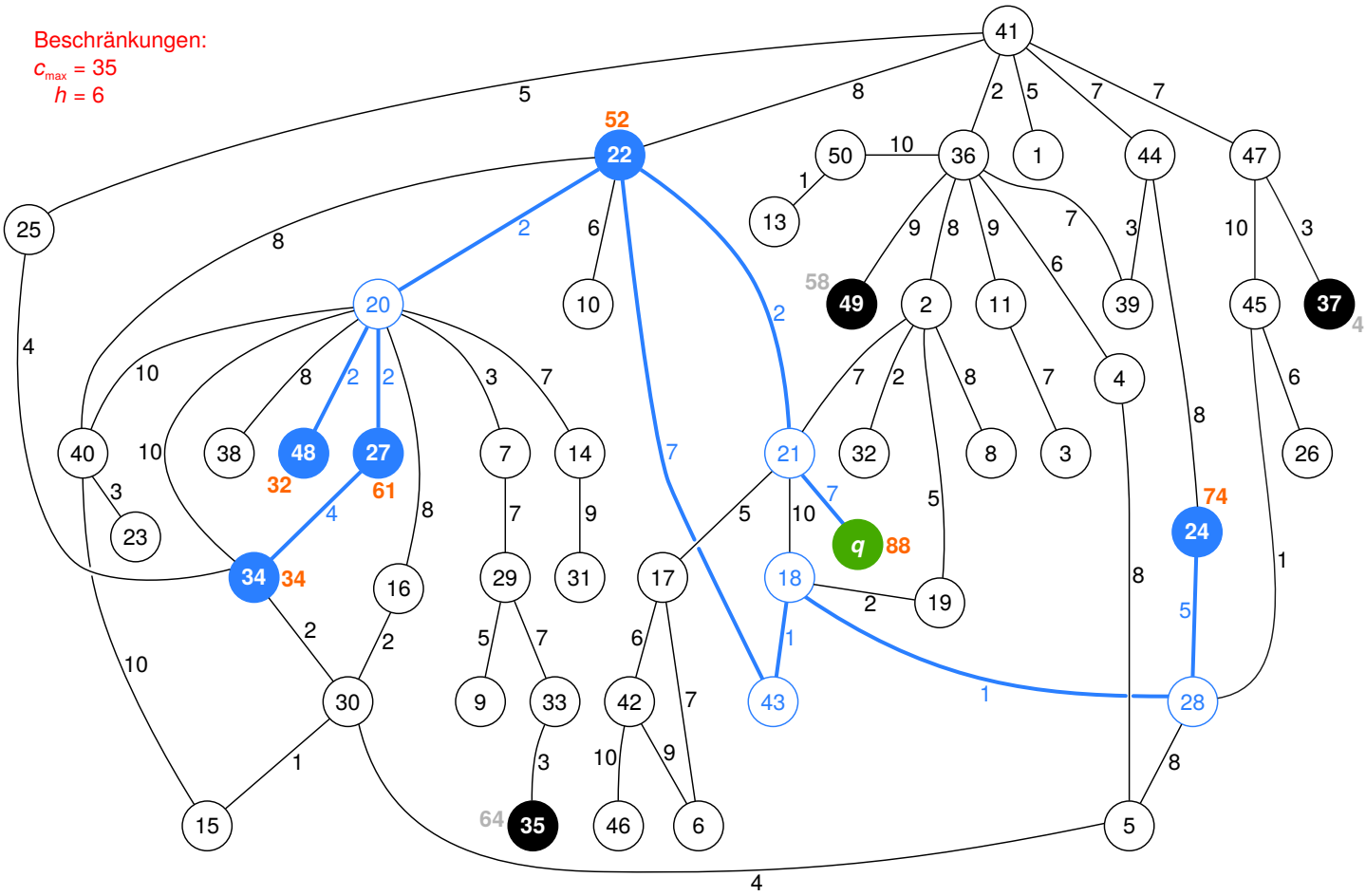
(c)

Abbildung A.2 (fortgesetzt)



(d)

Abbildung A.2 (fortgesetzt)



(e)

Abbildung A.2 (fortgesetzt)

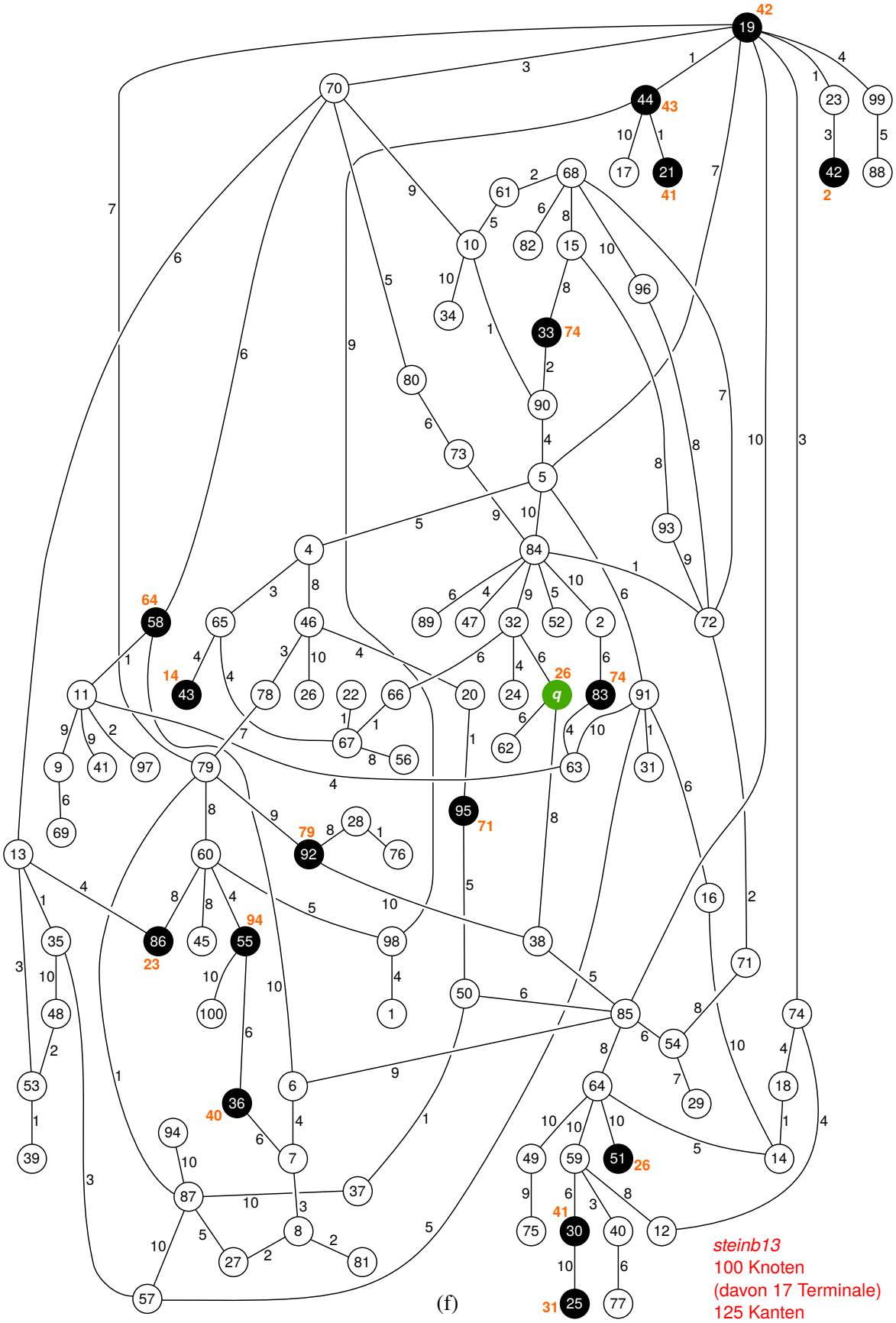


Abbildung A.2 (fortgesetzt)

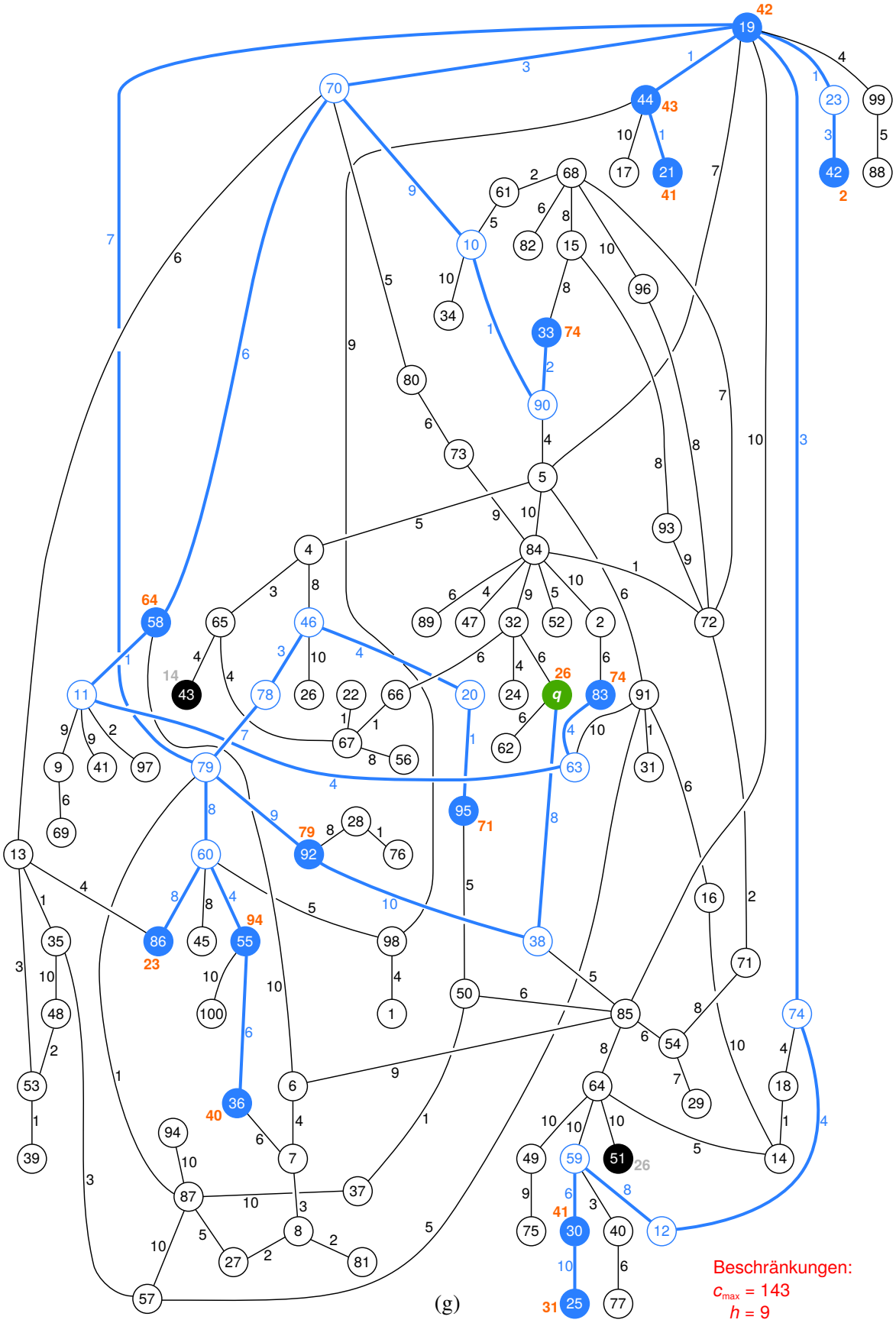


Abbildung A.2 (fortgesetzt)

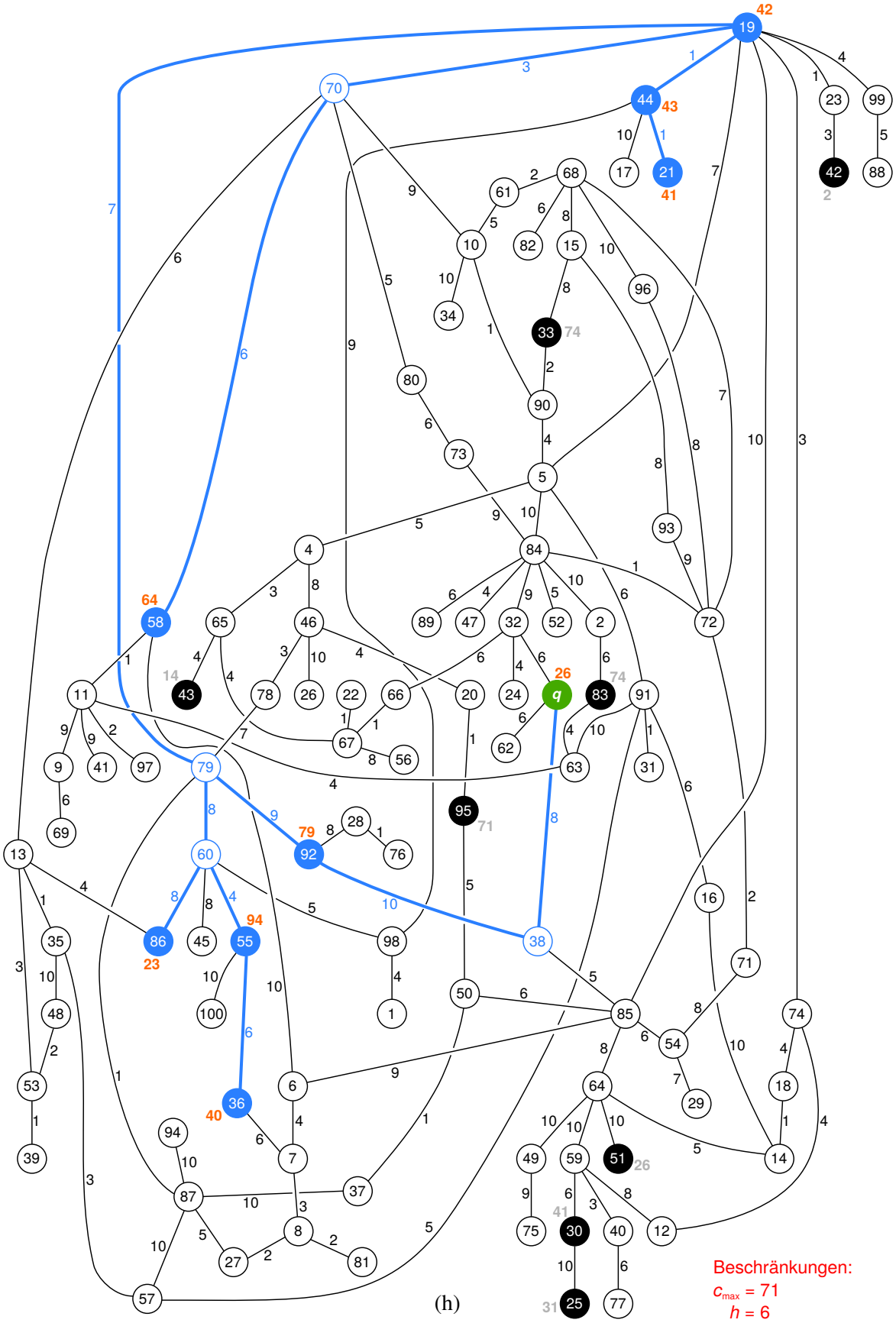


Abbildung A.2 (fortgesetzt)

Literaturverzeichnis

Agrawal, A., P. Klein und R. Ravi (1995). When trees collide: An approximation algorithm for the generalized Steiner problem on networks. In: *SIAM J. Comput.* 24.3, S. 440–456.

Aneja, Y. P. (1980). An integer linear programming approach to the Steiner problem in graphs. In: *Networks* 10.2, S. 167–178.

Applegate, D., R. Brixby *et al.* (1998). On the solution of traveling salesman problems. In: *Doc. Math.* Extra Volume 1, S. 645–656.

Banks, D. und K. Carley (1994). Metric inference for social networks. In: *J. Classif.* 11.1, S. 121–149.

Baxter, J. (1981). Local optima avoidance in depot location. In: *J. Oper. Res. Soc.* 32.9, S. 815–819.

Beasley, J. E. (1989). An SST-based algorithm for the Steiner problem in graphs. In: *Networks* 19.1, S. 1–16.

Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. In: *J. Oper. Res. Soc.* 41.11, S. 1069–1072.

Blum, C. und A. Roli (2003). Metaheuristics in combinatorial optimization. In: *ACM Comput. Surv.* 35.3, S. 268–308.

Bondy, J. A. und U. S. R. Murty (1976). Graph theory with applications. 1. Aufl. New York, Amsterdam und Oxford: North-Holland.

Brazil, M., R. L. Graham *et al.* (2014). On the history of the Euclidean Steiner tree problem. In: *Arch. Hist. Exact Sci.* 68.3, S. 327–354.

Canuto, S. A., M. G. C. Resende und C. C. Ribeiro (2001). Local search with perturbations for the prize-collecting Steiner tree problem in graphs. In: *Networks* 38.1, S. 50–58.

Chawla, S., D. Kitchin *et al.* (2002). Profit maximizing mechanisms for the extended multicasting game. Technical Report CMU-CS-02-164. Carnegie Mellon University.

Costa, A. M., J.-F. Cordeau und G. Laporte (2006). Steiner tree problems with profits. In: *INFOR* 44.2, S. 99–115.

Costa, A. M., J.-F. Cordeau und G. Laporte (2008). Fast heuristics for the Steiner tree problem with revenues, budget and hop constraints. In: *Eur. J. Oper. Res.* 190.1, S. 68–78.

Costa, A. M., J.-F. Cordeau und G. Laporte (2009). Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints. In: *Networks* 53.2, S. 141–159.

Courant, R. und H. Robbins (1941). What is mathematics? An elementary approach to ideas and methods. Oxford and New York: Oxford University Press.

Dantzig, G., R. Fulkerson und S. Johnson (1954). Solution of a large-scale traveling-salesman Problem. In: *J. Oper. Res. Soc. Am.* 2.4, S. 393–410.

Desrochers, M. und G. Laporte (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. In: *Oper. Res. Lett.* 10.1, S. 27–36.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. In: *Numer. Math.* 1.1, S. 269–271.

Dreyfus, S. E. und R. A. Wagner (1972). The Steiner problem in graphs. In: *Networks* 1.3, S. 195–207.

Du, D. Z., F. K. Hwang und J. F. Weng (1987). Steiner minimal trees for regular polygons. In: *Discrete Comput. Geom.* 2.1, S. 65–84.

Ellson, J., E. Gansner *et al.* (2002). Graphviz – Open source graph drawing tools. In: *Graph Drawing*. Hrsg. von P. Mutzel, M. Jünger und S. Leipert. Lecture Notes in Computer Science 2265. New York: Springer, S. 483–484.

Fermat, P. de (1891). Methodus ad disquirendam maximam et minimam. In: *Œuvres de Fermat*. Œuvres mathémathiques diverses – Observations sur Diophante. Hrsg. von P. Tannery und C. Henry. Bd. 1. Paris: Gauthier-Villars et fils, S. 133–179.

Floyd, R. W. (1962). Algorithm 97: Shortest path. In: *Commun. ACM* 5.6, S. 345.

Friedman, E. J. und D. C. Parkes (2002). Pricing WiFi at Starbucks – Issues in online mechanism design. URL: http://www.eecs.harvard.edu/~parkes/pubs/online.pdf (besucht am 21. 10. 2014).

Fu, Z.-H. und J.-K. Hao (2014a). Breakout local search for the Steiner tree problem with revenue, budget and hop constraints. In: *Eur. J. Oper. Res.* 232.1, S. 209–220.

Fu, Z.-H. und J.-K. Hao (2014b). Dynamic programming driven memetic search for the Steiner tree problem with revenues, budget and hop constraints. In: *IMFORMS J. Comput.* Eingereicht.

Garey, M. R., R. L. Graham und D. S. Johnson (1977). The complexity of computing Steiner minimal trees. In: *SIAM J. Appl. Math.* 32.4, S. 835–859.

Garey, M. R. und D. S. Johnson (1977). The rectilinear Steiner tree problem is *NP*-complete. In: *SIAM J. Appl. Math.* 32.4, S. 826–834.

Gergonne, J. D. (1810a). Questions proposées. Problèmes de géométrie. I. In: *Annales de mathémathiques pures et appliquées*. Hrsg. von J. D. Gergonne und J. E. Thomas-Lavernède. Bd. 1. Paris: Courcier, S. 196.

Gergonne, J. D. (1810b). Questions proposées. Problèmes de géométrie. I. In: *Annales de mathémathiques pures et appliquées*. Hrsg. von J. D. Gergonne und J. E. Thomas-Lavernède. Bd. 1. Paris: Courcier, S. 292.

Gilbert, E. N. und H. O. Pollak (1968). Steiner minimal trees. In: *SIAM J. Appl. Math.* 16.1, S. 1–29.

Glover, F. (1989). Tabu search – Part I. In: *INFORMS J. Comput.* 1.3, S. 190–206.

Glover, F. (1990). Tabu search – Part II. In: *INFORMS J. Comput.* 2.1, S. 4–32.

Goemans, M. X. (1994). The Steiner tree polytope and related polyhedra. In: *Math. Program.* 63.1–3, S. 157–182.

Gouveia, L. (1995). Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints. In: *Comput. Oper. Res.* 22.9, S. 959–970.

Grötschel, M. und R. Stephan (2014). Characterization of facets of the hop constrained chain polytope via dynamic programming. In: *Discrete Appl. Math.* 162, S. 229–246.

Gueron, S. und R. Tessler (2002). The Fermat-Steiner problem. In: *Am. Math. Mon.* 109.5, S. 443–451.

Hakimi, S. L. (1971). Steiner's problem in graphs and its implications. In: *Networks* 1.2, S. 113–133.

Hamming, R. W. (1950). Error detecting and error correcting codes. In: *Bell Syst. Tech. J.* 29.2, S. 147–160.

Hanan, M. (1966). On Steiner's problem with rectilinear distance. In: *SIAM J. Appl. Math.* 14.2, S. 255–265.

Hauptmann, M. und M. Karpinski, Hrsg. (2014). A compendium on Steiner tree problems. URL: http://theory.cs.uni-bonn.de/info5/steinerkompendium/netcompendium.html (besucht am 25. 09. 2014).

Hwang, F. K. (1992). The Steiner ratio. In: *The Steiner tree problem*. Hrsg. von F. K. Hwang, D. S. Richards und P. Winter. Annals of discrete mathematics 53. Amsterdam: Elsevier. Kap. I.3, S. 37–49.

Jarník, V. und M. Kössler (1934). O minimálních grafech obsahujících n daných bodu. In: *Časopis Pěst. Mat.* 63, S. 223–235.

Johnson, D. S., M. Minkoff und S. Phillips (2000). The prize collecting Steiner tree problem: Theory and practice. In: *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco: Society for Industrial und Applied Mathematics, S. 760–769.

Karp, R. M. (1972). Reducibility among combinatorial problems. In: *Complexity among combinatorial problems*. Hrsg. von R. E. Miller und J. W. Thatcher. New York und London: Plenum Press, S. 85–103.

Korte, B. und J. Nešetřil (2001). Vojtěch Jarník's work in combinatorial optimization. In: *Discrete Math.* 235.1–3, S. 1–17.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. In: *Proc. Am. Math. Soc.* 7.1, S. 48–50.

Levin, A. Y. (1971). Algorithm for shortest connection of a group of graph vertices. In: *Sov. Math. Dokl.* 12, S. 1477–1481.

Ljubić, I., R. Weiskircher *et al.* (2004). Solving the prize-collecting Steiner tree problem to optimality. Technical Report TR-186-1-04-01. Technische Universität Wien.

Lourenço, H. R., O. C. Martin und T. Stützle (2010). Iterated local search. In: *Handbook of metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. 2. Aufl. International Series in Operations Research & Management Science 146. New York: Springer. Kap. 11, S. 363–397.

Lucena, A. und M. G. Resende (2004). Strong lower bounds for the prize collecting Steiner problem in graphs. In: *Discrete Appl. Math.* 141.1–3, S. 277–294.

Maculan, N. (1987). The Steiner problem in graphs. In: *Surveys in combinatorial optimization*. Hrsg. von S. Martello, G. Laporte *et al.* Annals of discrete mathematics 31. Amsterdam: Elsevier. Kap. 6, S. 185–212.

Melzak, Z. A. (1961). On the problem of Steiner. In: *Canad. Math. Bull.* 4, S. 143–148.

Miller, C. E., A. W. Tucker und R. A. Zemlin (1960). Integer programming formulation of traveling salesman problems. In: *J. ACM* 7.4, S. 326–329.

Nam, N. M. (2013). The Fermat–Torricelli problem in the light of convex analysis. In: arXiv: 1302.5244.

Prim, R. C. (1957). Shortest connection networks and some generalizations. In: *Bell Syst. Tech. J.* 36.6, S. 1389–1401.

Ralphs, T. K. und M. V. Galati (2006). Decomposition in integer linear programming. In: *Integer programming. Theory and practice*. Hrsg. von J. K. Karlof. Boca Raton: Taylor & Francis. Kap. 4, S. 57–110.

Reiter, S. und G. Sherman (1965). Discrete Optimizing. In: *J. Soc. Ind. Appl. Math.* 13.3, S. 864–889.

Sinnl, M. (2011). Branch-and-price for the Steiner tree problem with revenues, budget and hop constraints. Diplomarbeit. Fakultät für Informatik der Technischen Universität Wien.

Tero, A., S. Takagi *et al.* (2010a). Rules for biologically inspired adaptive network design. In: *Science* 327.5964, S. 439–442.

Tero, A., T. Nakagaki *et al.* (2010b). A method inspired by *Physarum* for solving the Steiner problem. In: *Int. J. Unconv. Comput.* 6.2, S. 109–123.

Voss, S. (1992). Steiner's problem in graphs: heuristic methods. In: *Discrete Appl. Math.* 40.1, S. 45–72.

Voss, S. (1999). The Steiner tree problem with hop constraints. In: *Ann. Oper. Res.* 86, S. 321–345.

Warme, D. M., P. Winter und M. Zachariasen (2000). Exact algorithms for plane Steiner tree problems: A computational study. In: *Advances in Steiner trees*. Hrsg. von D.-Z. Du, J. M. Smith und J. H. Rubinstein. Combinatorial Optimization 6. Dordrecht: Springer. Kap. 6, S. 81–116.

Warshall, S. (1962). A theorem on Boolean matrices. In: *J. ACM* 9.1, S. 11–12.

Winter, P. (1992). Introduction. In: *The Steiner tree problem*. Hrsg. von F. K. Hwang, D. S. Richards und P. Winter. Annals of discrete mathematics 53. Amsterdam: Elsevier. Kap. II.1, S. 93–102.

Winter, P. (1987). Steiner problem in networks: A survey. In: *Networks* 17.2, S. 129–167.

Yang, X. S. (2011). Review of meta-heuristics and generalised evolutionary walk algorithm. In: *Int. J. Bio-Insp. Comput.* 3.2, S. 77–84.

Eidesstattliche Versicherung

Ich erkläre, dass ich die Bachelorarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich, inhaltlich oder sinngemäß entnommenen Stellen als solche den wissenschaftlichen Anforderungen entsprechend kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für Zeichnungen, Skizzen oder graphische Darstellungen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft und ausschließlich für Prüfungszwecke gespeichert wird.

3. Februar 2015

Datum Unterschrift