

Inhaltsverzeichnis

Abkürzungsverzeichnis und Symbole	1
1 Einleitung	3
2 Grundlagen	4
2.1 Grundlagen Graphentheorie	4
2.2 Problembeschreibung MCDPP	5
2.3 Branch and Bound Verfahren	7
3 Lösen des MCDPP mit Enumerationsverfahren	10
4 Branch and Bound Verfahren zur Lösung des MCDPP	13
4.1 Ablauf des Branch and Bound Verfahrens	13
4.2 Subgradientenverfahren und untere Schranken	17
4.3 Subinstanzen	22
4.4 Verzweigungsstrategie	23
4.5 Beschränkungskriterien und globale obere Schranke	28
4.6 Auswahlstrategie	30
4.7 Obere und untere Schranke	30
4.8 Optimalitätskriterien	32
4.9 Einstellbare Parameter und Strategien	33
5 Testinstanzen	38
5.1 Kleine Testinstanzen	38
5.2 Große Testinstanzen	41
5.3 Testinstanzen ohne optimale Lösung	45
5.4 Ergebnisse aus Testinstanzen	47
5.5 Verbesserungen des Branch and Bound Verfahrens	55
6 Zusammenfassung	56
Literaturverzeichnis	58
Abbildungsverzeichnis	59
Tabellenverzeichnis	60

Abkürzungsverzeichnis und Symbole

MCDPP	Minimum Cost Disjoint Path Problem
V	Knotenmenge im Graph
A	Kantenmenge im gerichteten Graph
E	Kantenmenge im ungerichteten Graph
$G = (V, A)$	gerichteter Graph ohne Schlingen
$G = (V, E)$	ungerichteter Graph ohne Schlingen/ Basis-Graph des MCDPP
$P(s, t)$	einfacher s-t-Weg auf Graph
T	Terminals
D	Demands
$H = (T, D)$	Demand-Graph
$H' = (T', D')$	Anangepasster Demand-Graph
V^d	Knotenmenge eines Weges eines eingebetteten Demands d
E^d	Kantenmenge eines Weges eines eingebetteten Demands d
ZL	Menge aller zulässigen Lösungen des MCDPP
X	eine zulässige Lösung des MCDPP
X^*	eine optimale Lösung des MCDPP
$z(X)$	Gesamtkosten der zulässigen Lösung X
c	Kantenkostenfunktion eines Graphen
S	Subinstanz des Branch and Bound Verfahrens
S'	verzweigte Subinstanz des Branch and Bound Verfahrens
W	Menge der verzweigbaren Subinstanzen
U	Globale untere Schranke des Branch and Bound Verfahrens
O	Globale obere Schranke des Branch and Bound Verfahrens
$ZK(i)$	Knotenzustand eines Knotens $i \in V$ in $G = (V, E)$
MZ	Menge aller Kombinationen aus der Zuordnung aller Knotenzustände und zugehörigen angepassten Demand-Graphen $H'(T', D')$ in $G = (V, E)$
MZ'	Menge aller zulässigen Lösungen aus MZ
$B(S)$	Unterbaum einer Subinstanz S im Branch and Bound Verfahren
VZK	Verzweigungsknoten
S_{Start}	Gesamtinstanz
MS	Menge verzweigter Subinstanzen
W_W	Liste der Warteschlange W
O_W	Globale obere Schranke der Warteschlange W
X_W	Zulässige Lösung der Warteschlange W
U_S	Untere Schranke der Subinstanz S
Z_S	Zustand der Subinstanz S
L_S	Menge aller Knoten mit Knotenzustand 0 in Subinstanz S
D_S	Demands des angepassten Demand-Graphen von Subinstanz S gespeichert als Vektor von Knotenpaaren

λ	Lagrange-Vektor der Lagrange-Relaxation
c'	Um λ angepasste Kantenkostenfunktion eines Graphen
ω	Vektor ω des Subgradientenverfahrens
μ	Vektor μ des Subgradientenverfahrens
ζ	Subgradient ζ des Subgradientenverfahrens
α	Schrittweite des Subgradientenverfahrens
θ	Parameter der Schrittweite α
MV	Menge an Konfliktknoten
X_S^*	eine optimale Lösung der Subinstanz S
X_S	eine zulässige Lösung der Subinstanz S
ZL_S	Menge aller zulässigen Lösungen der Subinstanz S
ZL'	Menge gefundener zulässiger Lösungen während des Subgradientenverfahrens
S^*	Subinstanz, die durch Verzweigung zu einer optimalen Lösung führt
$\Delta U(S, S', \lambda)$	Differenz unterer Schranken von zu verzweigender Subinstanz S und verzweigter Subinstanz S' bei λ
λ_{Option}	Einstellbarer Parameter des Branch and Bound Verfahrens
VZK_{Option}	Einstellbarer Parameter des Branch and Bound Verfahrens

1 Einleitung

Schnelle und effiziente Algorithmen sind bei der Lösung anwendungsorientierter mathematischer Probleme von großer Bedeutung. Ein anwendungsrelevantes kombinatorisches Problem ist die minimale knotendisjunkte Einbettung in Graphen auch Minimum Cost Disjoint Paths Problem (MCDPP) genannt. Sie wird in der Netzwerkoptimierung verwendet. Ein Beispiel ist die Einbettung eines kostengünstigen VPNs in einem Telekommunikationsnetz.

Das MCDPP beschreibt folgendes Problem: Für einen gegebenen Graphen mit nicht negativen Kantenkosten und einer bestimmten Menge von Knotenpaaren sollen Wege auf dem Graphen zwischen den Knotenpaaren gefunden werden. Diese Wege dürfen sich nicht schneiden. Ziel ist es die zulässige Lösung zu finden, die den kostengünstigste Gesamtweg besitzt. Es ist zur Zeit kein Algorithmus bekannt, der dieses Problem in polynomialer Zeit löst.

Es handelt sich beim MCDPP um ein ganzzahliges lineares Optimierungsproblem. Dieses Problem kann mit einem Solver für ganzzahlige lineare Optimierungsprobleme gelöst werden. Das kann jedoch mit einem großen Zeitaufwand verbunden sein. Ein anderer Ansatz ist die Verwendung einer Heuristik. Jedoch besitzt man bei einer Heuristik häufig nicht die Garantie, dass sich die gefundene Lösung nahe an der optimalen Lösung befindet. Ein weiterer Ansatz ist das Branch-and-Bound Verfahren. Gute an das Problem angepasste Branch and Bound Verfahren können den Suchraum für die optimale Lösung sehr gut einschränken und damit eine optimale Lösung mit geringen Zeitaufwand finden. In dieser Arbeit wird ein Branch and Bound Verfahren erstellt und verwendet, das für das Lösen von Instanzen des MCDPP verwendet werden kann.

2 Grundlagen

2.1 Grundlagen Graphentheorie

Dieser Abschnitt basiert hauptsächlich auf der Quelle [1]. Ein gerichteter Graph ist wie folgt definiert:

Definition 2.1. [1] Es sei V eine endliche Menge beliebiger Objekte und $A \subseteq V \times V$ eine beliebige Relation auf V . Der gerichtete Graph G wird definiert als das Paar $G = (V, A)$. V heißt seine Knotenmenge und A seine Kantenmenge.

Ein ungerichteter Graph ist ein spezieller Fall eines gerichteten Graphen mit symmetrischer Relation. Die Kantenmenge ungerichteter Graphen wird mit E beschrieben und der ungerichtete Graph mit $G = (V, E)$ abgekürzt. Jeder ungerichtete Graph kann als gerichteter Graph beschrieben werden.

Graphen können graphisch dargestellt werden. Im Fall des ungerichteten Graphen werden die Knoten als Punkte und die Kanten als verbindende Linien zwischen zwei Knoten beschrieben. Sind zwei Knoten durch eine Kante verbunden, so sind diese Knoten jeweils zueinander benachbart. Führt eine Kante von einem Knoten in denselben Knoten, so wird dies als Schlinge bezeichnet. In dieser Arbeit werden ausschließlich mit ungerichteten Graphen ohne Schlingen verwendet.

Auf $G = (V, E)$ können Wege wie folgt definiert werden:

Definition 2.2. [1] Sei $G = (V, E)$ ein ungerichteter Graph. Ein einfacher $s-t$ -Weg $P(s, t) \subseteq G$ ist ein Teilgraph, der aus einer Folge von paarweise verschiedenen, in der gegebenen Reihenfolge jeweils benachbarten Knoten besteht. Die Knoten s und t markieren Anfang und Ende dieser Folge.

In dieser Arbeit werden immer einfache Wege betrachtet.

2.2 Problembeschreibung MCDPP

Dieser Abschnitt basiert hauptsächlich auf der Quelle [2]. Zur Beschreibung des MCDPP werden zwei Graphen verwendet. Der Basis-Graph ist ein ungerichteter Graph, der mit $G = (V, E)$ abgekürzt wird. Der Demand-Graph ist ein ungerichteter Graph, der mit $H = (T, D)$ abgekürzt wird und die Eigenschaft $T \subseteq V$ besitzt. Die Knoten des Demand-Graphen werden Terminals und die Kanten des Demand-Graphen werden Demands genannt. Das MCDPP wird wie folgt beschrieben werden:

Definition 2.3. [2] Gegeben sei ein Basis-Graph $G = (V, E)$ mit der Kantenkostenfunktion $c : E \rightarrow \mathbb{Q}^+$ und ein Demand-Graph $H = (T, D)$. Für jeden Demand $d = \{s, t\}$ soll eine Einbettung in G als einfacher s-t-Weg $P(d) = (V^d, E^d)$ gefunden werden, sodass alle Wege gegenseitig knotendisjunkt sind und die gesamten Kosten aller gebrauchten Kanten minimal sind:

$$\min \sum_{d \in D} \sum_{e \in E^d} c(e)$$

$$\text{sodass } V^d \cap V^{d'} \subseteq d \cap d' \text{ mit } d \neq d'$$

Die Existenz einer optimalen Lösung ist nicht garantiert. Zulässige Lösungen des MCDPP werden durch knotendisjunkte Wege beschrieben, die für die Einbettung der Demands in $G = (V, E)$ verwendet werden. Auf Grund der Knotendisjunktheit dieser Wege können zulässige Lösungen mit einer Menge von Kanten $X \subseteq E$ von $G = (V, E)$ beschrieben werden, die den Wegkanten aller eingebetteten Demands entsprechen. Die Eigenschaft der globalen Optimalität von optimalen Lösungen des MCDPP wird wie folgt beschrieben:

Definition 2.4. Die Menge aller zulässigen Lösungen des MCDPP wird mit ZL beschrieben. Die Gesamtkosten $z(X)$ einer zulässigen Lösung $X \in ZL$ betragen:

$$z(X) = \sum_{e \in X} c(e) \tag{2.1}$$

Eine optimale Lösung X^* mit Kosten $z(X^*)$ besitzt die Eigenschaft:

$$z(X^*) \leq z(X) \quad \forall X \in ZL$$

Die Kosten eines einfachen s-t-Weges eines eingebetteten Demands $d \in D$ lassen sich nach Gleichung 2.1 wie folgt berechnen:

$$z(P(d)) = \sum_{e \in E^d} c(e) \tag{2.2}$$

An folgenden Beispielgraphen werden die beiden möglichen Lösungen des MCDPP veranschaulicht:

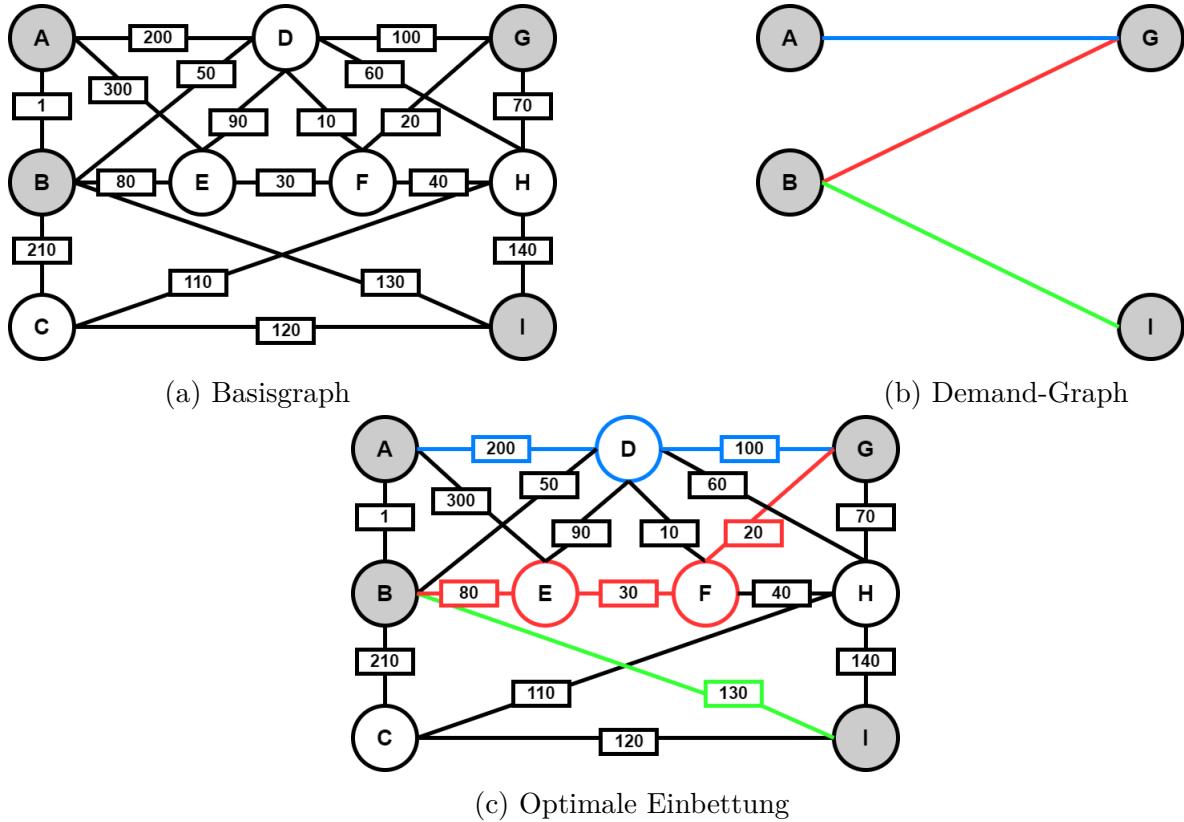


Abbildung 2.1: Testinstanz 1 dargestellt mit Basis-Graph, Demand-Graph und einer optimalen Lösung

Testinstanz 1 aus Abbildung 2.1 ist eine Beispielinstanz für das MCDPP mit einem Basis-Graphen (siehe Abbildung 2.1(a)) und einem Demand-Graphen (siehe Abbildung 2.1(b)). Die Kantenkosten c des Basis-Graphen entsprechen den Zahlen an den Kanten des Basis-Graphen. Die Terminalknoten sind die grau gefärbten Knoten A,B,G und I. Die Demands haben die Farben blau (Demand von A nach G), rot (Demand von B nach G), grün (Demand von B nach I) und sollen als Wege in den Basis-Graphen eingebettet werden. Eine optimale Lösung ist in Abbildung 2.1(c) abgebildet. Die Gesamtkosten dieser optimalen Lösung betragen 560.

Entfernt man in Testinstanz 1 die Kante zwischen Knoten A und D, die Kante zwischen Knoten B und D und die Kante zwischen Knoten C und H erhält man Testinstanz 6 (siehe Abbildung 2.2). Es ist bei dieser Testinstanz nicht möglich alle Demands als knotendisjunkte Wege einzubetten. Testinstanz 6 besitzt keine zulässige Lösung und keine optimale Lösung.

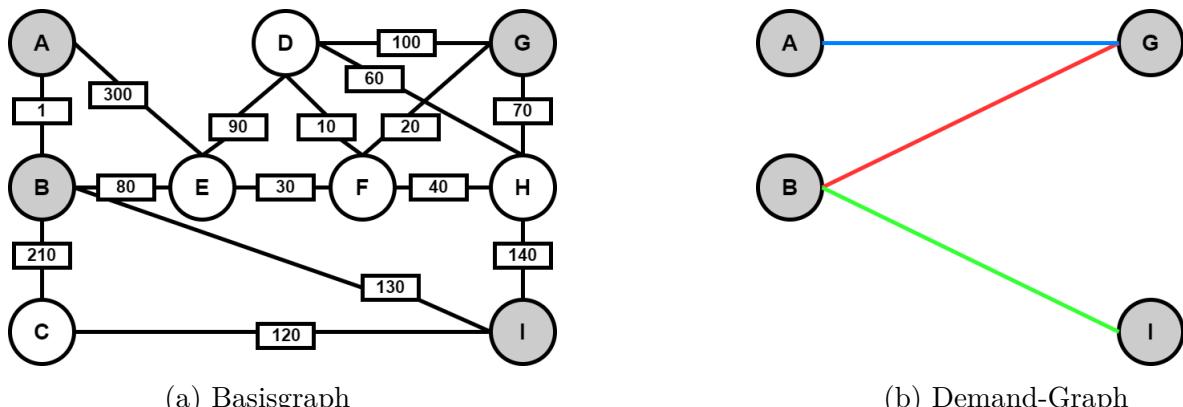


Abbildung 2.2: Testinstanz 6 mit Basis-Graphen und Demand-Graphen

2.3 Branch and Bound Verfahren

Dieser Abschnitt basiert hauptsächlich auf der Quelle [1]. Zum Lösen des MCDPP kann die Menge aller Einbettungen der Demands als Wege gebildet werden. Aus dieser Menge kann die Menge aller zulässigen Lösungen bestimmt werden. Aus der Menge der zulässigen Lösungen kann eine optimale Lösung bestimmt werden. Das MCDPP kann deswegen mit einem Enumerationsverfahren gelöst werden.

Probleme, welche mit einem Enumerationsverfahren gelöst werden können, können unter bestimmten Bedingungen durch ein Branch and Bound Verfahren gelöst werden. Mit Hilfe dessen kann die Laufzeit im Vergleich zum Enumerationsverfahren verkleinert werden. Das Branch and Bound Verfahren besteht aus folgenden Komponenten:

1. Subinstanzen
2. Verzweigungsstrategien
3. Auswahlstrategie
4. Beschränkungskriterien
5. Optimalitätskriterien
6. Obere und untere Schranken

Können alle diese Komponenten für ein Enumerationsverfahren definiert werden, kann aus dem Enumerationsverfahren ein Branch and Bound Verfahren entwickelt werden. Das Branch and Bound Verfahren wird durch einen Enumerationsbaum beschrieben, der sukzessiv aufgebaut wird. In Abbildung 2.3 ist eine vereinfachender Ablauf des Branch and Bound Verfahrens dargestellt.

Subinstanzen sind Objekte, die dem Knoten des Enumerationsbaumes entsprechen. Sie können beispielsweise ganze lineare Programme oder Teile einer Lösung des Problems speichern. Das Branch and Bound Verfahren startet am Anfang mit einer Subinstanz, die Gesamtinstanz (siehe einzelner schwarzer Knoten links in Abbildung 2.3) genannt wird.

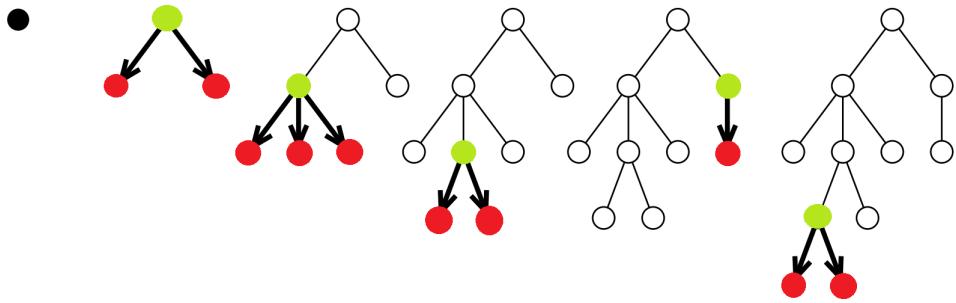


Abbildung 2.3: Schematische Darstellung des Branch and Bound Verfahrens [1]

Subinstanzen können verzweigt werden. Wenn eine Subinstanz S verzweigt wird, entstehen an der Stelle von S im Enumerationsbaum Kindknoten. Diese Kindknoten entsprechen den aus S verzweigten Subinstanzen S' . Wie eine Subinstanz S in seine verzweigten Subinstanzen S' verzweigt wird, wird in der Verzweigungsstrategie beschrieben. In Abbildung 2.3 entsprechen die roten Kindknoten den verzweigten Subinstanzen.

Jede Verzweigung entspricht einer Iteration des Branch and Bound Verfahrens. Während jeder Iteration verändert sich der Baum des Branch and Bound Verfahrens durch die Verzweigung. In Abbildung 2.3 sind 5 Iterationen des Branch and Bound Verfahrens dargestellt, in denen sich der Baum des Branch and Bound Verfahrens durch die Verzweigung verändert hat. Die verzweigbaren Subinstanzen werden in einer Menge W gespeichert. Aus dieser Menge W wird die nächste Subinstanz zur Verzweigung gewählt. Welche der Subinstanzen in W für die Verzweigung gewählt wird, wird in der Auswahlstrategie beschrieben. Die ausgewählten Subinstanzen sind in Abbildung 2.3 als grüne Elternknoten dargestellt.

Es ist möglich, dass eine Subinstanz S nicht mehr weiterverzweigt werden kann. S ist dann nicht verzweigbar und wird aus der Menge W entfernt, ohne dass S weiterverzweigt wird. Unter welchen Bedingungen eine Subinstanz nicht mehr weiterverzweigt werden kann, wird durch die Beschränzungskriterien beschrieben.

Das Branch and Bound Verfahren endet, sobald keine Subinstanz weiterverzweigt werden kann und $W = \emptyset$ ist. Das Verfahren kann vorher beendet werden, wenn eine Subinstanz gefunden wird, die das Optimalitätskriterium erfüllt. Ist das Optimalitätskriterium erfüllt, ist die optimale Lösung des Problems gefunden und das Verfahren endet vorzeitig. Die Existenz eines Optimalitätskriteriums ist nicht garantiert.

Die Gesamtkosten der optimalen Lösung X^* kann mit einer globalen unteren Schranke U und einer globalen oberen Schranke O beschränkt werden. Es gilt:

$$U \leq z(X^*) \leq O \quad (2.3)$$

Wenn beispielsweise eine zulässige Lösung X des MCDPP bekannt ist, bilden deren Gesamtkosten $z(X)$ eine globale obere Schranke für $z(X^*)$ und die Kosten 0 eine globale untere Schranke.

Verzweigungsstrategien, Auswahlstrategien, Beschränzungskriterien und Optimalitätskriterien können sich auf lokale obere und lokale untere Schranken beziehen. Diese lokalen Schranken beziehen sich auf den entstehenden Unterbaum $B(S)$ einer verzweigbaren Subinstanz

S. Sie ordnen S eine Zahl $obereSchranke(S)$ oder $untereSchranke(S)$ zu. Der Wert der Gesamtkosten $z(X)$ einer zulässige Lösung X , die im Unterbaum einer verzweigbaren Subinstanz S entsteht, kann die lokalen Schranken nicht überschreiten. Es gilt:

$$z(X) \leq obereSchranke(S) \text{ mit } X \text{ ist zulässige Lösung und entsteht in } B(S) \quad (2.4)$$

$$z(X) \geq untereSchranke(S) \text{ mit } X \text{ ist zulässige Lösung und entsteht in } B(S) \quad (2.5)$$

Befindet sich keine zulässige Lösung im Unterbaum von S , können nach der Verzweigung die lokalen Schranken unbegrenzt ansteigen oder sinken. Mit Hilfe lokaler Schranken kann gezielter nach der optimalen Lösung des Problems gesucht werden. Demzufolge ist der Suchraum des Branch and Bound Verfahrens kleiner als der Suchraum des Enumerationsverfahrens. Dies kann zu kürzeren Laufzeiten des Branch and Bound Verfahrens im Vergleich zum Enumerationsverfahren führen.

3 Lösen des MCDPP mit Enumerationsverfahren

Das Branch and Bound Verfahren für das MCDPP wird aus dem Enumerationsverfahren hergeleitet. Jeder Demand von $H = (T, D)$ soll als Weg zwischen seinen Terminals in $G = (V, E)$ eingebettet werden. Sind alle eingebetteten Wege knotendisjunkt, entspricht dies einer zulässigen Lösung des MCDPP. Für jede beliebige zulässige Lösung kann jedem Nicht-Terminal-Knoten $i \in V \setminus T$ ein Knotenzustand $ZK(i) \in \{0, 1, \dots, |D|\}$ zugeordnet werden. Der Knotenzustand hat folgende Bedeutung:

1. $ZK(i) = 0 \Leftrightarrow$ Knoten $i \in V \setminus T$ befindet sich auf keinem eingebetteten Weg
2. $ZK(i) = j \wedge j \in \{1, \dots, |D|\} \Leftrightarrow$ Knoten $i \in V \setminus T$ befindet sich auf eingebettetem Weg des Demands j

An Testinstanz 1 wird das Prinzip des Knotenzustands in Abbildung 3.1 veranschaulicht:

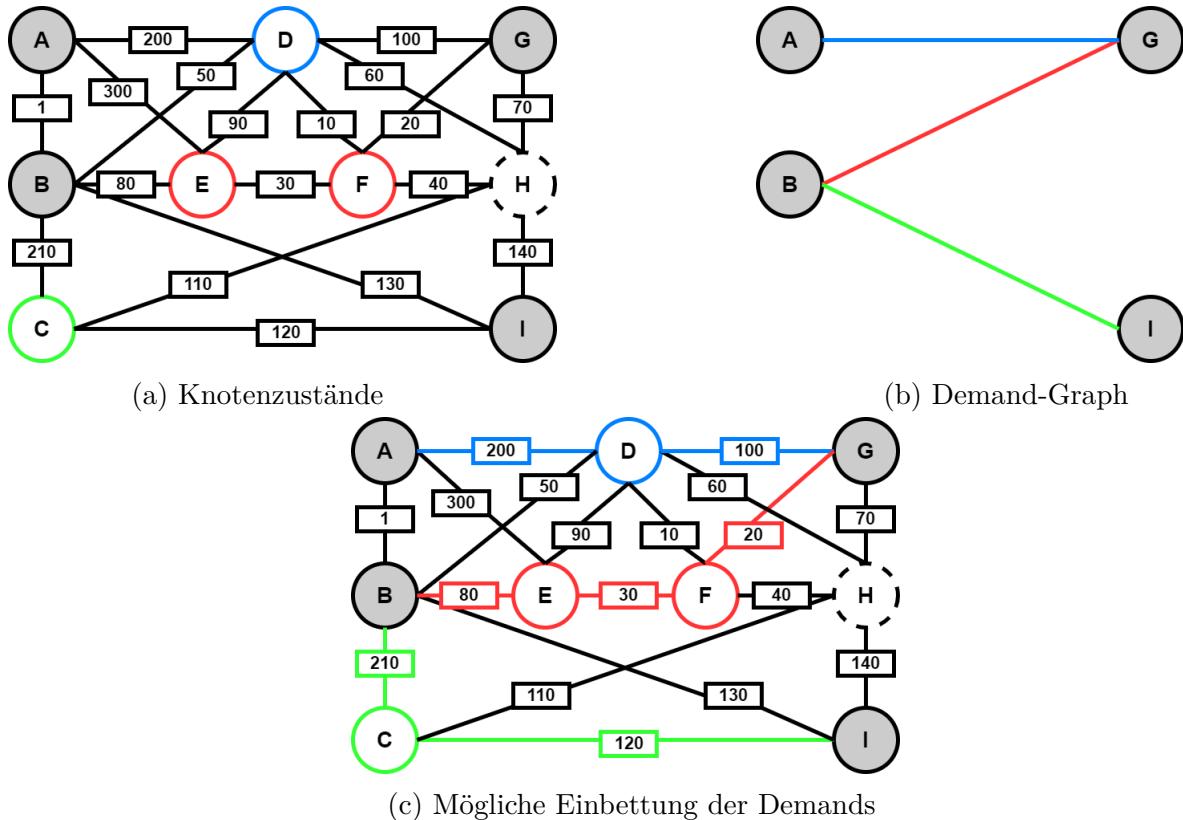


Abbildung 3.1: Testinstanz 1 mit gesetzten Knotenzuständen

In der Abbildung 3.1(a) wird in Testinstanz 1 für alle Nicht-Terminalknoten des Basis-Graphen Knotenzustände gesetzt. Die Terminalknoten besitzen per Definition keinen Knotenzustand. Wenn Demand blau Demand 1, rot Demand 2, grün Demand 3 und schwarz gestrichelt kein Demand entspricht, ergeben sich folgende Knotenzustände: $ZK(H) = 0$, $ZK(D) = 1$, $ZK(E) = ZK(F) = 2$ und $ZK(C) = 3$. In Abbildung 3.1(c) ist eine mögliche Einbettung der Demands als Wege mit diesen gesetzten Knotenzuständen dargestellt.

Wegen der Knotendisjunktheit zulässiger Lösungen, ist der Knotenzustand eindeutig. Terminalknoten können mehreren Demands zugeordnet werden und werden deswegen von dieser Definition ausgenommen. Der Startknoten und Endknoten jedes Weges eines eingebetteten Demands ist durch die Terminals gegeben. Jeder zulässigen Lösung kann mit $G = (V, E)$ und $H = (T, D)$ Knotenzustände $ZK(i)$ mit $i \in V \setminus T$ zugeordnet werden. Jedoch kann $G = (V, E)$, $H = (T, D)$ und $ZK(i)$ mit $i \in V \setminus T$ nicht jede zulässige Lösung eindeutig beschreiben. In folgendem Beispiel ist dies veranschaulicht:

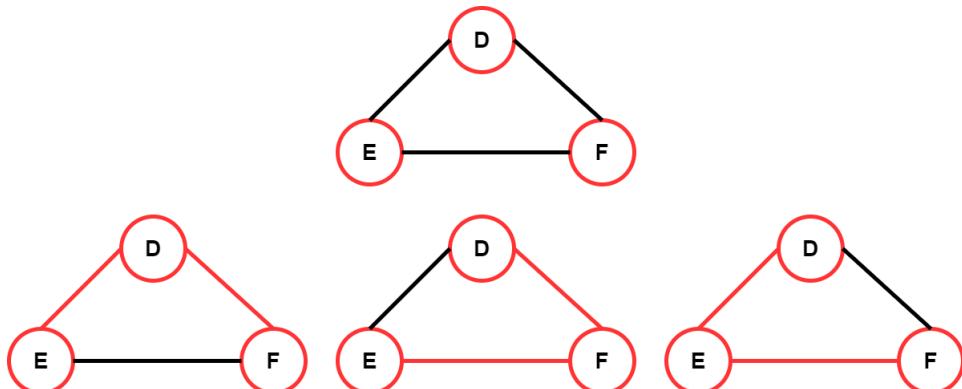
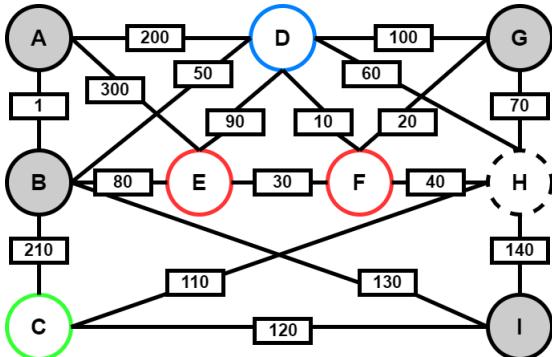
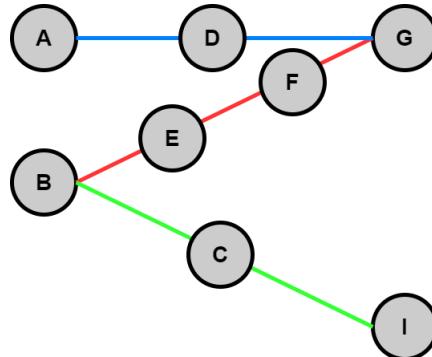


Abbildung 3.2: Mögliche Wege von 3 benachbarten Knoten mit gleichem Knotenzustand

In Abbildung 3.2 sind 3 benachbarte Knoten mit gleichem Knotenzustand dargestellt. Sind die Knotenzustände in Testinstanz 1 für D, E und F gesetzt wie in Abbildung 3.2 und alle anderen Knotenzustände auf 0 gesetzt, dann ist die Einbettung des roten Demands als Weg mehrdeutig. Es gibt 3 mögliche Wege. Deswegen muss die Reihenfolge der Knoten im Weg berücksichtigt werden. Um Knotenzustände und die Reihenfolge der Knoten zu berücksichtigen, können die Knoten mit Knotenzuständen im Demand-Graph als Zwischenterminals eingefügt werden. Dieser Graph wird angepasster Demand-Graph $H' = (T', D')$ genannt. In Abbildung 3.3(b) ist ein angepasster Demand-Graph für die Knotenzustände in Abbildung 3.3(a) angegeben. Dieser angepasste Demand-Graph führt eindeutig zu der Einbettung in Abbildung 3.1(c).



(a) Knotenzustände



(b) Angepasster Demand-Graph

Abbildung 3.3: Testinstanz 1 mit gesetzten Knotenzuständen und angepassten Demand-Graph

Jede zulässige Lösung kann eindeutig einem $G = (V, E)$ und $H' = (T', D')$ und $ZK(i)$ mit $i \in V \setminus T$ zugeordnet werden. Jedoch führt nicht jede beliebige Kombination aus der Zuordnung aller Knotenzustände und zugehörigen $H' = (T', D')$ zu einer zulässigen Lösung. Wenn MZ die Menge aller möglichen Kombinationen aus der Zuordnung aller Knotenzustände und zugehörigen $H' = (T', D')$ entspricht, dann wird die Menge aller zulässigen Lösungen durch eine Teilmenge $MZ' \subseteq MZ$ beschrieben.

Im Enumerationsverfahren werden alle Elemente aus MZ durchgegangen und die Menge der zulässigen Lösungen MZ' bestimmt. Aus MZ' wird die zulässige Lösung mit den kleinsten Gesamtkosten bestimmt, um eine optimale Lösung des MCDPP zu erhalten.

Der Aufwand für dieses Verfahren lässt sich nach oben abschätzen. Es existieren insgesamt $(|D| + 1)^{|V \setminus T|}$ mögliche Zuordnungen von Knotenzuständen. Für eine Zuordnung der Knotenzustände, können in $H' = (T', D')$ alle Reihenfolgen der Knoten $i \in V^d$ in jedem Demand berücksichtigt werden. Damit existieren $\prod_{d \in D} |V^d \setminus T'|!$ mögliche angepasste Demand-Graphen für eine Zuordnung der Knotenzustände. Das wird abgeschätzt mit:

$$\prod_{d \in D} |V^d \setminus T|! \leq \prod_{d \in D} |V \setminus T|! = (|V \setminus T|!)^{|D|}$$

Somit kann $|MZ|$ nach oben abgeschätzt werden mit:

$$|MZ| \leq (|D| + 1)^{|V \setminus T|} (|V \setminus T|!)^{|D|} \quad (3.1)$$

Mit $\beta = \max(|D| + 1, |V \setminus T|)$ gilt:

$$|MZ| \leq \beta^\beta (\beta!)^\beta \quad (3.2)$$

Der Aufwand für dieses Verfahren ist nach oben abgeschätzt super-exponentiell $O((\beta!)^\beta)$ und kann damit einen sehr großen Zeitaufwand verursachen.

4 Branch and Bound Verfahren zur Lösung des MCDPP

4.1 Ablauf des Branch and Bound Verfahrens

Im Branch and Bound Verfahren werden die Knoten des Basis-Graphen sukzessiv mit Knotenzuständen belegt und jede Subinstanz besitzt einen angepassten Demand-Graph. Im Gegensatz zum reinen Enumerationsverfahren, in dem alle Knotenzustände gesetzt sind, gibt es im Branch and Bound Verfahren Knoten mit und ohne gesetzten Knotenzustand. In jeder Verzweigung wird ein Verzweigungsknoten VZK ausgewählt, dessen Knotenzustand gesetzt werden soll. Die daraus resultierenden verzweigten Subinstanzen besitzen jeweils für VZK einen anderen Knotenzustand und verschiedene angepasste Demand-Graphen. Die verzweigbaren Subinstanzen werden in einer Menge W gespeichert. Es werden solange Subinstanzen aus W ausgewählt und verzweigt, bis W leer wird. Am Ende erhält man die Gesamtkosten $z(X^*)$ und die Kanten aller Wege der eingebetteten Demands X^* einer optimalen Lösung der Instanz des MCDPP, sofern sie existiert. Jede der $|MZ|$ Elemente des reinen Enumerationsverfahrens in Kapitel 3 ist unterschiedlich, weshalb es sich bei dem daraus resultierenden Branch and Bound Verfahren um ein knotendisjunktes Branching handelt.

Der Ablauf des Branch and Bound Verfahrens wird in Algorithm 1 beschrieben. Für die Instanz des MCDPP mit dem Basis-Graph $G = (V, E)$ und dem Demand-Graph $H = (T, D)$ soll eine optimale Lösung gefunden werden. Als Output des Branch and Bound Verfahrens erhält man eine optimale Lösung X^* und deren Gesamtkosten $z(X^*)$. Sollte keine optimale Lösung existieren ist $X^* = \emptyset$ und $z(X^*) = -1$.

Am Anfang wird eine globale obere Schranke O für die Instanz des MCDPP bestimmt (Zeile 2 Algorithm 1). Diese globale obere Schranke wird in jeder Iteration des Branch and Bound Verfahrens aktualisiert (siehe Zeile 18 Algorithm 1). Die Menge aller verzweigbaren Subinstanzen W wird leer initialisiert und die globale obere Schranke gespeichert (Zeile 4-5 Algorithm 1). In der Gesamtinstanz S_{Start} des Branch and Bound Verfahrens ist kein Knotenzustand gesetzt. Sie ist die erste Subinstanz, die zu W hinzugefügt wird, wenn S_{Start} verzweigbar ist (Zeile 7-10 Algorithm 1). Unter welchen Bedingungen eine Subinstanz verzweigbar ist, wird in Kapitel 4.4 erklärt. Bei der Bildung von S_{Start} können zulässige Lösungen entstehen. X' ist die gefundene zulässige Lösung mit den kleinsten Gesamtkosten. X' wird verwendet, um die globale obere Schranke von W zu aktualisieren (siehe Zeile 11 Algorithm 1). Sollte S_{Start} nicht verzweigbar sein, wird eine optimale Lösung gefunden oder die Nicht-Existenz einer optimalen Lösung festgestellt und das Branch and Bound Verfahren endet.

Es werden solange Iterationen des Branch and Bound Verfahren ausgeführt, bis keine Subinstanz zum Weiterverzweigen existiert und W leer ist. Die zu verzweigende Subinstanz S wird aus W ausgewählt und entfernt (siehe Zeile 14 Algorithm 1). Anschließend wird der

Verzweigungsknoten bestimmt, an dem S verzweigt werden soll (siehe Zeile 16 Algorithm 1). Danach findet die Verzweigung von S statt. Es entstehen dabei eine Menge von verzweigten Subinstanzen MS und eine zulässige Lösung X'' (siehe Zeile 19 Algorithm 1). X'' ist die zulässige Lösung, die bei der Bildung aller verzweigten Subinstanzen gefunden wird und von allen gefundenen zulässigen Lösungen die geringsten Gesamtkosten besitzt. Alle S' aus MS werden W hinzugefügt und die obere Schranke von W mit X'' und deren Gesamtkosten aktualisiert (Zeile 20-23 Algorithm 1). Ist W leer geworden, wurde eine optimale Lösung X^* gefunden oder das MCDPP besitzt keine Lösung (Zeile 26-30 Algorithm 1).

W ist die Menge aller verzweigbaren Subinstanzen, aus welcher die nächste Subinstanz zum Verzweigen ausgewählt wird. W wird als Prioritätswarteschlange implementiert.

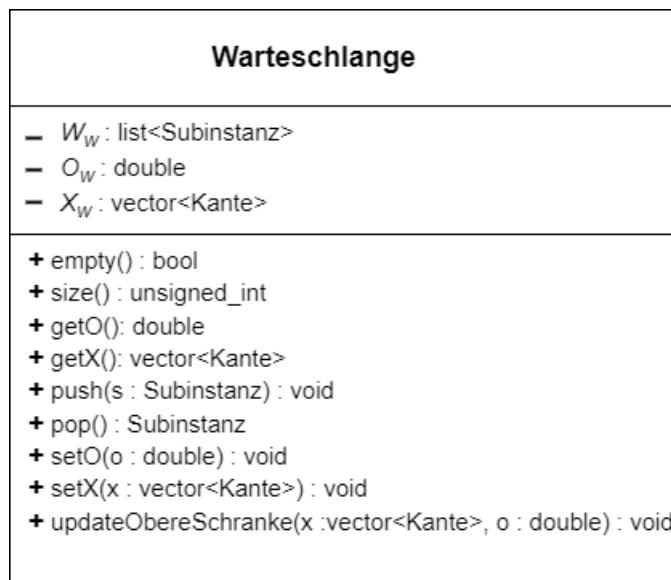


Abbildung 4.1: Menge aller verzweigbaren Subinstanzen W dargestellt als Prioritätswarteschlange in einem UML Diagramm

Die Warteschlange W besitzt als Attribut eine Liste W_W aus Subinstanzen, bei denen die Subinstanzen S nach der Größe ihrer unteren Schranke U_S aufsteigend sortiert sind. Es wird die kleinste gefundene obere Schranke im Attribut O_W gespeichert. O_W entspricht der globalen oberen Schranke der Instanz des MCDPP. Führt eine gefundene zulässige Lösung zu einem Update von O_W , werden die Wegkanten aller Wege der eingebetteten Demands dieser zulässigen Lösung im Attribut X_W als ein Vektor von Kanten gespeichert.

Zugriff auf O_W und X_W hat man über Getter `getO` und `getX` und Setter `setO` und `setX`. `empty()` gibt aus, ob W_W leer ist und `size()` gibt die Größe von W_W an.

Über die Methode `pop` wird das erste Element in W_W ausgewählt und aus W_W entfernt. Die Methode `pop` wird zum Auswählen der Subinstanz benutzt, die weiterverzweigt werden soll (siehe Zeile 14 in Algorithm 1).

Algorithm 1 branchAndBound($G = (V, E)$, $H = (T, D)$)

Input: Basis-Graph $G = (V, E)$, Demand-Graph $H = (T, D)$

Output: Optimale Lösung X^* und seine Gesamtkosten $z(X^*)$

```

1:  $\triangleright$  Globale obere Schranke bilden
2:  $O = \text{obereSchranke}(G, H)$ 

3:  $\triangleright$  Initialisiere  $W$ 
4:  $W = \emptyset$ 
5:  $W.\text{setO}(O)$ 

6:  $\triangleright$  Initialisiere Gesamtinstanz  $S_{start}$ 
7:  $(S_{start}, X') = \text{instanzBilden}(G, H, \emptyset, \emptyset, \emptyset, O)$ 
8: if instanzVerzweigbar( $S_{start}, O$ ) then
9:      $W.\text{push}(S_{start})$ 
10: end if
11:  $W.\text{updateObereSchranke}(X', z(X'))$ 

12: while  $W \neq \emptyset$  do

13:      $\triangleright$  Subinstanz für Verzweigung auswählen
14:      $S = W.\text{pop}()$ 

15:      $\triangleright$  Knoten VZK wählen, dessen Knotenzustand bei Verzweigung gesetzt wird
16:     Bilde Verzweigungsknoten VZK

17:      $\triangleright$   $S$  verzweigen
18:      $O = W.\text{getO}()$ 
19:      $(MS, X'') = \text{instanzVerzweigen}(G, H, S, VZK, O)$ 
20:     for  $S' \in MS$  do
21:          $W.\text{push}(S')$ 
22:     end for
23:      $W.\text{updateObereSchranke}(X'', z(X''))$ 

24: end while

25:  $\triangleright$  Optimale Lösung initialisieren, wenn sie existiert
26:  $X^* = \emptyset$ 
27:  $z(X^*) = -1$ 
28: if  $! W.\text{getX}().\text{empty}()$  then
29:      $X^* = W.\text{getX}()$ 
30:      $z(X^*) = W.\text{getO}()$ 
31: end if

32: return  $X^*, z(X^*)$ 

```

Algorithm 2 $\text{push}(S)$

Input: Subinstanz S

Output: Veränderung von W_W

- 1: Füge S aufsteigend sortiert nach U_S in W_W ein
-

Subinstanzen können über die Methode push (siehe Algorithm 2) zur Warteschlange hinzugefügt werden und damit W_W verändern. Die Subinstanz S wird über die Methode push nach dem Wert ihrer unteren Schranke U_S aufsteigend in W_W eingefügt. Auf Grund des knotendisjunkten Branchings existieren keine Subinstanzen, die als identisch interpretiert werden können. Eine weitere Überprüfung der Subinstanz bei der Aufnahme in die Warteschlange ist deswegen nicht notwendig.

Algorithm 3 $\text{updateObereSchranke}(X', z(X'))$

Input: zulässige Lösung X' , Gesamtkosten der zulässigen Lösung $z(X')$

Output: Veränderung von O_W , X_W und W_W

- 1: **if** $z(X') < O_W \wedge !X'.empty()$ **then**
 - 2: setO($z(X')$)
 - 3: setX(X')
 - 4: Entferne alle Subinstanzen S mit unterer Schranke U_S aus W_W mit $U_S \geq z(X')$
 - 5: **end if**
 - 6: **if** $z(X') == O_W \wedge X_W.empty() \wedge !X'.empty()$ **then**
 - 7: setX(X')
 - 8: **end if**
-

Die Methode updateObereSchranke (siehe Algorithm 3) führt ein Update von O_W , X_W und W_W durch. Sind die Gesamtkosten $z(X')$ einer zulässigen Lösung X' kleiner als die globale obere Schranke O_W , bildet $z(X')$ eine neue globale obere Schranke. Tritt dieser Fall ein, wird O_W auf $z(X')$ und X_W auf X' gesetzt. Weiterhin werden alle Subinstanzen S aus W_W entfernt, deren lokale untere Schranke U_S größer gleich der neuen globalen oberen Schranke $z(X')$ sind (Zeile 1-5 in Algorithm 3). Die entfernten Subinstanzen sind suboptimal (siehe Kapitel 4.5) und können deswegen aus W_W entfernt werden. Ein Spezialfall kann auftreten, wenn eine zulässige Lösung mit $z(X') = O_W$ für das Update der oberen Schranke verwendet wird und X_W leer ist. In diesem Fall besitzt eine zulässige Lösung X' die Gesamtkosten O_W und X_W kann auf X' gesetzt werden.

4.2 Subgradientenverfahren und untere Schranken

Das Subgradientenverfahren ist von zentraler Bedeutung für dieses Branch and Bound Verfahren. Es wird hauptsächlich verwendet, um untere Schranken für Instanzen des MCDPP zu bestimmen. Eine untere Schranke der Instanz eines MCDPP wird mit Hilfe der Lagrange Relaxation ermittelt. Die genaue Beschreibung des Subgradientenverfahrens und der Lagrange-Relaxation ist nicht Teil dieser Bachelorarbeit. Beides wird in [2] ausführlich erklärt.

Untere Schranken U_S von Subinstanzen S mit Basis-Graphen $G = (V, E)$ und angepassten Demand-Graph $H' = (T', D')$ können mit der Lagrange-Relaxation bestimmt werden. Man benötigt dafür einen Lagrange-Vektor λ . λ hat $|V|$ Komponenten deren Werte reelle Zahlen größer gleich 0 sind. Jedem Knoten $j \in V$ kann eine Komponente $\lambda(j)$ von λ zugeordnet werden. Jede Kante $e = \{u, v\} \in E$ mit Kantenkosten $c(u, v)$ besitzt einen Kopfknoten $u \in V$ und Fußknoten $v \in V$. Kopfknoten und Fußknoten werden über die Kante miteinander verbunden. Die um λ angepassten Kantenkosten c' des Basis-Graphen werden nach Gleichung 4.1 bestimmt.

$$c'(u, v, \lambda) = c(u, v) + \lambda(u) + \lambda(v) \quad \forall \{u, v\} \in E \quad (4.1)$$

Der Vektor ω hat $|V|$ Komponenten und beschreibt wie viele Kanten eines Knotens maximal Teil eines Weges eines eingebetteten Demands sein können:

$$\omega(u) = \begin{cases} 2 & u \in V \setminus T' \\ d_{H'}(u) & u \in T' \end{cases} \quad (4.2)$$

$d_{H'}(u)$ ist die Anzahl aller Kanten, die ein Terminalknoten $u \in T'$ in $H' = (T', D')$ besitzt. U_S von Subinstanz S mit Basis-Graph $G = (V, E)$ und angepassten Demand-Graph $H' = (T', D')$ wird nach Gleichung 4.3 berechnet.

$$U_S(\lambda) = \sum_{d \in D'} z(\text{kürzesterWeg}(G, d, \lambda)) - \sum_{u \in V} \omega(u) \lambda(u) \quad (4.3)$$

In der Funktion kürzesterWeg in Gleichung 4.3 wird der kürzeste Weg zwischen den Terminals des Demands d auf dem Basis-Graphen $G = (V, E)$ mit um λ angepassten Kantenkosten c' berechnet und dieser als einfacher s-t-Weg $P(d) = (V^d, E^d)$ zurückgegeben. Von den Gesamtkosten der kürzesten Wege (bestimmt nach Gleichung 2.2) für alle Demands wird das Skalarprodukt von ω und λ abgezogen, um die untere Schranke der Subinstanz zu generieren. Bei der Berechnung von $U_S(\lambda)$ können die entstandenen kürzesten Wege zwischen den Terminals knotendisjunkt sein und damit zulässigen Lösungen entsprechen. Es können zulässige Lösungen über die Berechnung der unteren Schranken gefunden werden.

Bei der Bildung des kürzesten Weges eines Demands d werden alle Terminalknoten aus $G = (V, A)$ gelöscht, bis auf die Terminals von d . Der Grund hierfür liegt in der Definition des MCDPP mit der Bedingung $V^d \cap V^{d'} \subseteq d \cap d'$ mit $d \neq d'$ (siehe Definition 2.3). Damit kann sich kein anderer Terminalknoten auf dem Weg eines eingebetteten Demands d befinden, außer den Terminalknoten von d . Die kürzesten Wege aller Demands können damit nur gemeinsame Nicht-Terminalknoten besitzen und gemeinsame Start- und Endterminalknoten besitzen. Die gemeinsamen Nicht-Terminalknoten der kürzesten Wege aller Demands werden

Konfliktknoten genannt. Existieren keine Konfliktknoten, ist eine zulässige Lösung gefunden worden. Ist es nicht möglich für mindestens einen Demand einen Weg zwischen seinen Terminals zu bilden, so existiert keine Lösung für die Subinstanz S . Wie häufig ein Knoten Teil eines kürzesten Weges für einen Demand ist, wird im Vektor μ beschrieben.

$$\mu(u) = \sum_{d \in D'} (u \in V^d) \quad \forall u \in V \quad (4.4)$$

In Gleichung 4.4 ist $(u \in V^d)$ ein boolscher Ausdruck, welcher beschreibt, ob der Knoten u in der Knotenmenge V^d des kürzesten Weges $P(d)$ eines eingebetteten Demands $d \in D'$ enthalten ist.

Der Subgradient ζ wird aus der Differenz von μ und ω gebildet nach Gleichung 4.5:

$$\zeta = \mu - \omega \quad (4.5)$$

Die komplementäre Schlupfbedingung [2] ist eine hinreichende Bedingung, welche eine Aussage über die Optimalität einer zulässigen Lösung trifft, die für eine untere Schranke $U_S(\lambda)$ gefunden wurde.

Definition 4.1. *Entsprechen die kürzesten Wege aller eingebetteten Demands des angepassten Demand-Graphen einer Subinstanz S für eine untere Schranke $U_S(\lambda)$ einer zulässigen Lösung der Instanz des MCDPP und ist das Skalarprodukt aus λ und ζ gleich 0, dann entsprechen alle Kanten X der kürzesten Wege der optimalen Lösung X_S^* der Subinstanz*

In Algorithm 4 werden die Menge an Konfliktknoten MV , die Kanten der kürzesten Wege aller eingebetteten Demands und der Vektor μ für einen gegebenen Basis-Graphen, angepassten Demand-Graphen und Lagrange-Vektor λ bestimmt.

Die Kantenkosten des Basis-Graphen werden wie in Gleichung 4.1 um den Parameter λ angepasst (Zeile 1 in Algorithm 4). Jeder Demand d von $H' = (T', D')$ soll als kürzester Weg auf $G = (V, E)$ mit um λ angepassten Kantenkosten eingebettet werden. Die Wegkanten dieser kürzesten Wege werden in X gespeichert (Zeile 7 und 13-15 in Algorithm 4). Ist es nicht möglich für mindestens einen Demand einen Weg zwischen seinen Terminals zu bilden, so existiert keine Lösung (Zeile 8-10 Algorithm 4). μ zählt wie häufig ein Knoten $u \in V$ Teil eines kürzesten Weges für einen eingebetteten Demand ist (Zeile 18 in Algorithm 4). Entsteht ein Konfliktknoten wird dieser in MV gespeichert (Zeile 19-21 Algorithm 4).

Algorithm 4 wegFinden($G = (V, E)$, $H = (T, D)$, λ)

Input: Basis-Graph $G = (V, E)$, angepasster Demand-Graph $H' = (T', D')$, Lagrange-Vektor λ

Output: Menge an Konfliktknoten MV , Menge aller Kanten der Wege der eingebetteten Demands X , Vektor μ aus Gleichung 4.4

```

1: Verändern der Kantenkosten  $c$  von  $G$  zu  $c'$  mit Parameter  $\lambda$  nach Gleichung 4.1
2:  $\triangleright$  Initialisiere  $MV$ ,  $X$  und  $\mu$ 
3:  $X = \emptyset, MV = \emptyset, \mu = \mathbf{0}$ 

4:  $\triangleright$  Finde kürzeste Wege für Demands auf  $G$  mit angepassten Kantenkosten
5: for  $d$  in  $D'$  do
6:    $\triangleright$  Bestimme kürzesten Weg, wenn er existiert
7:    $P(d) = (V^d, E^d) =$  kürzesterWeg( $G, d, \lambda$ )
8:   if  $\exists P(d)$  then
9:      $X = \emptyset$ 
10:    break
11:   end if

12:   $\triangleright$  Hinzufügen der Kanten des kürzesten Weges
13:  for  $e \in E^d$  do
14:     $X.\text{insert}(e)$ 
15:  end for

16:   $\triangleright$  Berechnung von  $\mu$  und Bestimmung von Konfliktknoten
17:  for  $i \in V^d$  do
18:     $\mu(i) = \mu(i) + 1$ 
19:    if  $\mu(i) > 1$  then
20:       $MV.\text{insert}(i)$ 
21:    end if
22:  end for
23: end for

24: return  $MV, X, \mu$ 

```

Der Zustand $Z(\lambda)$ einer unteren Schranke $U_S(\lambda)$ beschreibt die Optimalität der gefundenen Lösung X beim Parameter λ und wird wie folgt beschrieben:

1. $X = \emptyset \Leftrightarrow Z(\lambda) = \text{UNZULÄSSIG}$
2. (Definition 4.1 ist erfüllt) oder (alle Knotenzustände sind gesetzt und $X \neq \emptyset \Leftrightarrow Z(\lambda) = \text{OPTIMAL}$)
3. $Z(\lambda) \neq \text{OPTIMAL} \wedge Z(\lambda) \neq \text{UNZULÄSSIG} \Leftrightarrow Z(\lambda) = \text{KANDIDAT}$

Ist $X = \emptyset$ existiert für diese Subinstanz keine Lösung. Dann ist $Z(\lambda) = \text{UNZULÄSSIG}$. Nach Definition 4.1 kann hinreichend bestimmt werden, ob es sich bei der Einbettung aller Demands X um eine optimale Lösung der Subinstanz handelt. Ist dies der Fall, ist $Z(\lambda) = \text{OPTIMAL}$. Sollten alle Knotenzustände gesetzt sein und es existiert eine zulässige Lösung, ist diese die einzige zulässige Lösung und damit eine optimale Lösung, wodurch $Z(\lambda) = \text{OPTIMAL}$. Ist $Z(\lambda)$ weder OPTIMAL noch UNZULÄSSIG , so ist der Zustand KANDIDAT .

Unterschiedliche Lagrange-Vektoren λ führen bei der Lagrange-Relaxation zu unterschiedlichen unteren Schranken $U_S(\lambda)$. Es wird die größtmögliche untere Schranke für eine Subinstanz gesucht. Dies erfolgt nach dem Subgradientenverfahren. Das Subgradientenverfahren wird für das Branch and Bound Verfahren angepasst und wird in Algorithm 5 dargestellt. n_{sub} gibt die Anzahl der Iterationen des Subgradientenverfahrens an, in denen untere Schranken gesucht werden. λ_{Start} wird verwendet, um die untere Schranke in der ersten Iteration zu berechnen (Zeile 3-4 Algorithm 5). In jeder Iteration des Subgradientenverfahrens werden andere $\lambda^{(k)}$ und damit andere Wege für die eingebetteten Demands und andere untere Schranken $U'(\lambda^{(k)})$ gebildet. Die unteren Schranken $U'(\lambda^{(k)})$ werden über die Lagrange Relaxation ermittelt (Zeile 9 Algorithm 5). Ziel ist die größte untere Schranke U zu finden, die während n_{sub} Iterationen entsteht (Zeile 11 Algorithm 5). U gehört zu den Output-Werten des Subgradientenverfahrens SubgradOpt.

Die Kanten der kürzesten Wege der eingebetteten Demands $X^{(k)}$, die Menge der Konfliktknoten $MV^{(k)}$ und der Vektor $\mu^{(k)}$ werden mit wegFinden in jeder Iteration des Subgradientenverfahrens bestimmt (siehe Zeile 7 Algorithm 5). Ist $|MV^{(k)}| = 0$, wurde eine zulässige Lösung gefunden, die durch ihre Kantenmenge $X^{(k)}$ eindeutig beschrieben ist. Diese zulässige Lösung wird der Menge aller zulässigen Lösungen ZL' hinzugefügt (Zeile 13-15 Algorithm 5). ZL' gehört zu den Output-Werten von SubgradOpt.

In jeder Iteration nimmt $\lambda^{(k)}$ andere Werte an. Dies passiert mit Hilfe der Schrittweite $\alpha^{(k)}$ und dem Subgradienten $\zeta^{(k)}$ (Zeile 21-23 Algorithm 5). Die Schrittweite $\alpha^{(k)}$ ist eine positive reelle Zahl. Wie $\alpha^{(k)}$ gewählt wird, wird in Kapitel 4.9 beschrieben.

Der Zustand der unteren Schranke $Z(\lambda^{(k)})$ kann sich mit wechselnden $\lambda^{(k)}$ verändern. Wenn $Z(\lambda)$ bestimmte Werte annimmt, kann das Subgradientenverfahren vorzeitig abgebrochen werden. Wird $Z(\lambda^{(k)})$ UNZULÄSSIG oder OPTIMAL wird der Zustand Z auf diesen Wert gesetzt und das Subgradientenverfahren wird beendet. Das Subgradientenverfahren wird ebenfalls beendet, sobald eine untere Schranke ermittelt wird, die größer ist, als die globale obere Schranke O (Zeile 17-19 Algorithm 5). Die Subinstanz S ist dann suboptimal. Suboptimalität von Subinstanzen wird in Kapitel 4.5 beschrieben. Passiert keiner dieser Fälle während der n_{sub} Iterationen, bleibt $Z = \text{KANDIDAT}$ (siehe Zeile 2 Algorithm 5). Der Zustand Z gehört ebenfalls zu den Output-Werten von SubgradOpt.

Algorithm 5 SubgradOpt($G = (V, E)$, $H' = (T', D')$, O)

Input: Basis-Graph $G = (V, E)$, angepasster Demand-Graph $H' = (T', D')$, globale obere Schranke O

Output: Untere Schranke U , Zustand Z der unteren Schranke, Menge aller zulässigen gefundenen Lösungen ZL'

```

1:  $\triangleright$  Initialisiere Output-Werte
2:  $U = -\infty$ ,  $Z = \text{KANDIDAT}$ ,  $ZL' = \emptyset$ 

3: Anzahl Iterationen  $n_{sub}$  festlegen
4: Start-Lagrange-Vektor  $\lambda_{Start}$  festlegen und  $\lambda = \lambda_{Start}$  setzen
5: Bilde  $\omega$  nach Gleichung 4.2

6: for  $k = 1$  to  $n_{sub}$  do

7:    $(MV^{(k)}, X^{(k)}, \mu^{(k)}) = \text{wegFinden}(G, H', \lambda^{(k)})$ 
8:   Bestimme Zustand  $Z(\lambda^{(k)})$ 
9:    $U'(\lambda^{(k)}) = \sum_{d \in D'} z(\text{kürzesterWeg}(G, d, \lambda^{(k)})) - \sum_{u \in V} \omega(u) \lambda^{(k)}(u)$ 

10:   $\triangleright$  Update der größten unteren Schranke
11:   $U = \max(U, U'(\lambda^{(k)}))$ 

12:   $\triangleright$  Sammeln zulässiger Lösungen
13:  if  $|MV^{(k)}| == 0$  then
14:     $ZL'.insert(X^{(k)})$ 
15:  end if

16:   $\triangleright$  Abbruch bei optimaler, unzulässiger oder suboptimaler Lösung
17:  if  $Z(\lambda^{(k)}) == \text{OPTIMAL} \vee Z(\lambda^{(k)}) == \text{UNZULAESSIG} \vee U'(\lambda^{(k)}) \geq O$  then
18:     $Z = Z(\lambda^{(k)})$ 
19:    break
20:  end if

21:   $\triangleright \lambda$  für die nächste Iteration
22:   $\zeta^{(k)} = \mu^{(k)} - \omega$ 
23:   $\lambda^{(k+1)} = \lambda^{(k)} + \alpha^{(k)} \cdot \zeta^{(k)}$ 
24: end for

25: return  $U, Z, ZL'$ 

```

4.3 Subinstanzen

Während des Branch and Bound Verfahrens werden Knotenzustände gesetzt. Jede Subinstanz S speichert einen angepassten Demand-Graphen, welche diejenigen Knoten als Zwischenterminals beinhaltet, deren Knotenzustände gesetzt sind. Der angepasste Demand-Graph der Subinstanz wird in D_S gespeichert. An folgendem Beispiel wird dies veranschaulicht:

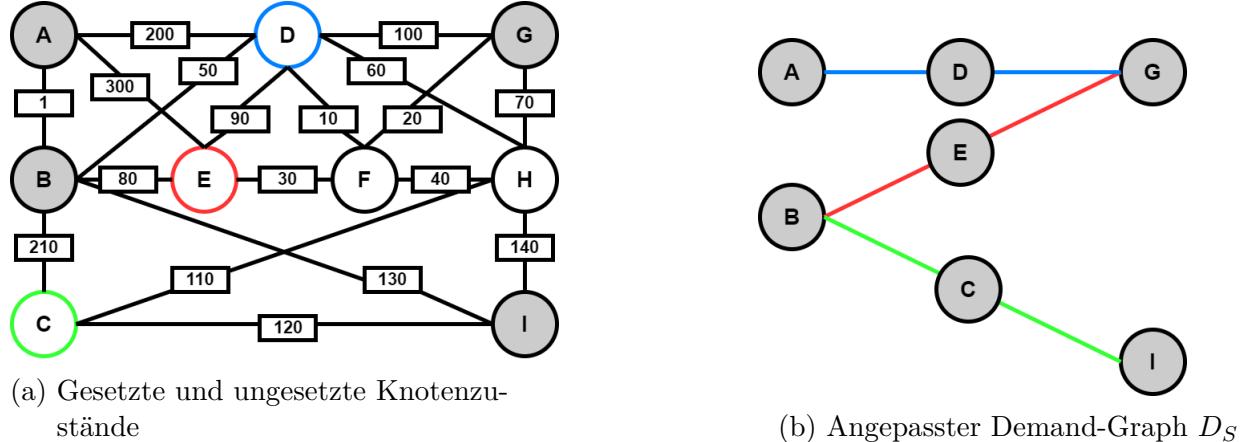


Abbildung 4.2: Testinstanz 1 mit angepasstem Demand-Graph D_S , gesetzten und ungesetzten Knotenzuständen

In Abbildung 4.2(a) sind die Knotenzustände von D,E und C bestimmten Demands (blau, rot, grün) über ihre Knotenzustände zugeordnet. Der Knotenzustand von F und H wurde noch nicht gesetzt. Die für die Demands zugeordneten Knotenzustände sind als Zwischenterminals zwischen den eigentlichen Demands des Demand-Graphen gesetzt (siehe Abbildung 4.2 (b)). Es entstehen dadurch in jeder Subinstanz zusätzliche Demands und Terminals. Der Knotenzustand kann 0 sein und damit keinem Demand zugeordnet werden. Die Menge L_S aller Knoten mit Knotenzustand 0 wird separat in jeder Subinstanz gespeichert. Für jede Subinstanz wird ein Subgradientenverfahren (siehe Kapitel 4.2) durchgeführt, bei dem der angepasste Demand-Graph D_S entspricht und beim Basis-Graph die Knoten aus L_S gelöscht werden.

Der Zustand der Subinstanz Z_S entspricht dem Zustand der unteren Schranke aus dem Subgradientenverfahren und kann deswegen 3 mögliche Zustände annehmen: UNZULÄSSIG, KANDIDAT, OPTIMAL. Ist der Zustand einer Subinstanz UNZULÄSSIG, ist es nicht möglich für diese Subinstanz eine zulässige Lösung zu finden. Ist der Zustand der Subinstanz KANDIDAT, kann die Subinstanz weiterverzweigt werden. Ist der Zustand der Subinstanz OPTIMAL, wurde für diese Subinstanz eine optimale Lösung gefunden.

Die untere Schranke U , welche während des Subgradientenverfahrens ermittelt wird, wird als U_S in der Subinstanz gespeichert. U_S ist eine untere Schranke für eine optimale Lösung der Subinstanz. Es ist deswegen nicht garantiert, dass U_S eine untere Schranke der Instanz des MCDPP ist.

Die Subinstanz kann dann als Klasse wie folgt implementiert werden:

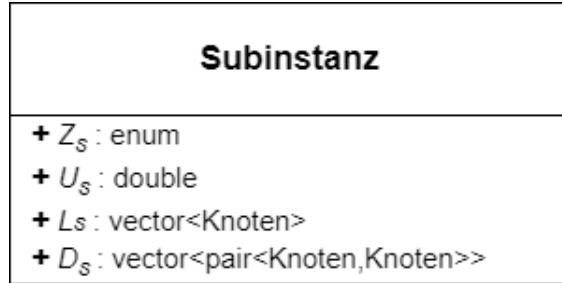


Abbildung 4.3: UML Diagramm der Subinstanz

Die Klasse Subinstanz besteht nur aus Attributen. Im enum Z_s kann einer der 3 Zustände (UNZULÄSSIG, KANDIDAT, OPTIMAL) abgespeichert werden. Die untere Schranke der Subinstanz wird als double Wert in U_s gespeichert. L_s speichert als Vektor alle Knoten mit Knotenzustand 0. D_s speichert die Demands $d \in D'$ angepassten Demand-Gruppen $H' = (T', D')$ als Vektor von Knotenpaaren der Terminals $t \in T'$. D_s speichert alle Informationen von $H' = (T', D')$, die für die Berechnungen des Branch and Bound Verfahren benötigt werden.

4.4 Verzweigungsstrategie

Eine Subinstanz S ist verzweigbar, wenn $Z_s = \text{KANDIDAT}$ entspricht und S nicht suboptimal ist. Dies wird in der Funktion instanzVerzweigbar geprüft (siehe Algorithm 6).

Algorithm 6 instanzVerzweigbar(S, O)

Input: Subinstanz S , globale obere Schranke O

Output: boolscher Ausdruck, ob Subinstanz S verzweigbar ist

1: **return** $S.Z_s == \text{KANDIDAT} \wedge S.U_s < O$

Bei der Verzweigung entstehen neue Subinstanzen durch Setzen des Knotenzustandes eines noch nicht gesetzten Knotens. Dieser wird Verzweigungsknoten VZK genannt. Bei der Verzweigung wird eine Subinstanz S aus W ausgewählt und entfernt. Dann wird aus dieser Subinstanz der Verzweigungsknoten bestimmt (siehe Zeile 16 Algorithm 1). Es gibt viele unterschiedliche Strategien diesen Knoten zu wählen, von denen Einige im Kapitel 4.9 beschrieben werden. Der Knotenzustand des Verzweigungsknotens kann $|D_s| + 1$ mögliche Zustände annehmen. Für jeden dieser Knotenzustände wird eine neue Subinstanz S' erzeugt. Dies wird in der Funktion instanzVerzweigen (siehe Zeile 19 in Algorithm 1) durchgeführt. Die Funktion instanzVerzweigen wird in Algorithm 7 definiert. Für jeden Demand d aus $S.D_s$ wird VZK als Zwischenterminal eingefügt, um die verzweigte Subinstanz S' zu bilden (siehe Zeile 5 in Algorithm 7). Ist der Knotenzustand 0, wird der Knoten keinem Demand zugeordnet (siehe Zeile 18 in Algorithm 6). Bei der Bildung der verzweigten Subinstanzen

wird die gefundene zulässige Lösung X' mit den kleinsten Gesamtkosten ermittelt (siehe Zeile 5 und 18 in Algorithm 7). Aus diesen zulässigen Lösungen wird diejenige zulässige Lösung X mit den kleinsten Gesamtkosten ermittelt (siehe Zeile 7-10 und 19-22 in Algorithm 7). In MS werden alle verzweigten Subinstanzen gespeichert, die selbst verzweigbar sind (siehe Zeil 12-14 und 23-25 in Algorithm 7).

Algorithm 7 instanzVerzweigen($G = (V, E)$, $H = (T, D)$, S , VZK , O)

Input: Basis-Graph $G = (V, E)$, Demand-Graph $H = (T, D)$, zu verzweigende Subinstanz S , Verzweigungsknoten VZK , globale obere Schranke O

Output: Menge an verzweigten Subinstanzen MS , gefundene zulässige Lösung X mit kleinsten Gesamtkosten

```

1: ▷ Initialisiere MV, X und obere Schranke Omin
2:  $MV = \emptyset, X = \emptyset, O_{min} = O$ 

3: for  $d$  in  $S.D_S$  do
4:   ▷ Bilde verzweigte Subinstanz
5:    $(S', X') = \text{instanzBilden}(G, H, S, VZK, d, O)$ 

6:   ▷ Update der zulässigen Lösung mit kleinsten Gesamtkosten
7:   if  $\neg X'.empty() \wedge z(X') < O_{min}$  then
8:      $O_{min} = z(X')$ 
9:      $X = X'$ 
10:    end if

11:   ▷ Hinzufügen verzweigbarer Subinstanzen zu MS
12:   if instanzVerzweigbar( $S'$ ,  $O$ ) then
13:      $MS.insert(S')$ 
14:   end if
15: end for

16:
17: ▷ Für Knotenzustand 0 Subinstanz bilden
18:  $(S', X') = \text{instanzBilden}(G, H, S, VZK, \emptyset, O)$ 
19: if  $\neg X'.empty() \wedge z(X') < O_{min}$  then
20:    $O_{min} = z(X')$ 
21:    $X = X'$ 
22: end if
23: if instanzVerzweigbar( $S'$ ,  $O$ ) then
24:    $MS.insert(S')$ 
25: end if

26: return  $MS, X$ 

```

Durch Einfügen des Verzweigungsknotens als Zwischenterminal für alle Demands des angepassten Demand-Graphen von S , wird jede mögliche Reihenfolge der Zwischenterminals im angepassten Demand-Graphen der verzweigten Subinstanzen berücksichtigt. Dies wird am folgenden Beispiel demonstriert:

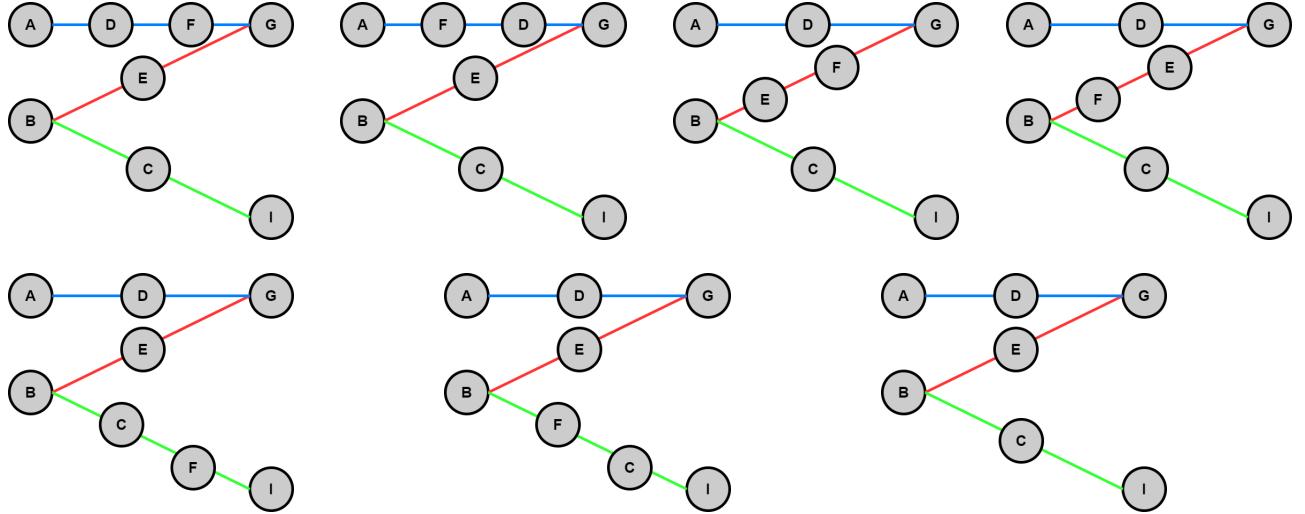


Abbildung 4.4: Alle angepassten Demand-Graphen für alle verzweigten Subinstanzen nach der Verzweigung der Subinstanz aus Abbildung 4.2 am Verzweigungsknoten F

Testinstanz 1 aus Abbildung 4.2 zeigt eine Subinstanz, in der der Knotenzustand von F und H nicht gesetzt ist. Wählt man diese Subinstanz zum Verzweigen aus und wählt als Verzweigungsknoten F, so erhält man 7 verzweigte Subinstanzen S' . Alle angepassten Demand-Graphen $D_{S'}$ der unterschiedlichen verzweigten Subinstanzen S' sind in Abbildung 4.4 dargestellt. Wird Knoten F dem blauen Demand zugeordnet, kann F zwischen Knoten A und D oder zwischen Knoten D und E als Zwischenterminal eingefügt werden. Es ist daraus erkennbar, dass das Zwischenterminal F in jeder möglichen Reihenfolge in die angepassten Demand-Graphen von S eingefügt wurde. Mit dieser Art der Verzweigung wird sichergestellt, dass jeder der $|MZ|$ möglichen Kombinationen aus Zuordnung aller Knotenzustände und zugehörigen angepassten Demand-Graphen aus dem Enumerationsverfahren in Kapitel 3 mit dem Branch and Bound Verfahren gebildet werden kann.

Subinstanzen entstehen aus der Verzweigung einer anderen Subinstanz. Die einzige Ausnahme bildet die Gesamtinstanz S_{Start} , welche gleichzeitig der Instanz des MCDPP entspricht. S_{Start} wird mit der Funktion instanzBilden (siehe Zeile 7 Algorithm 1) erzeugt. Jede verzweigte Subinstanz wird ebenfalls mit instanzBilden erzeugt (siehe Zeile 5 und 18 in Algorithm 7). instanzBilden wird in Algorithm 8 dargestellt.

Die Gesamtinstanz S_{Start} entspricht der Instanz des MCDPP und wird aus keiner Verzweigung gebildet. S_{Start} besitzt keine gelöschten Knoten und D_S entspricht dem Demand-Graphen (siehe Zeile 3-5 in Algorithm 8).

Wenn der Verzweigungsknoten als Zwischenterminal für einen Demand d eingefügt werden soll, so befindet sich d in $S.D_S$. Es wird der Startknoten s und der Endknoten e von d bestimmt. Es werden alle Demands von S übernommen und für der Verzweigungsknoten als Zwischenterminal eingefügt. Um das Zwischenterminal einzufügen, wird der Demand d ge-

löscht und stattdessen die Demands $\{s, VZK\}$ und $\{VZK, e\}$ hinzugefügt (siehe Zeile 6-12 in Algorithm 8). L_S wird von S übernommen.

Soll der Knotenzustand von VZK zu keinem Demand gehören, tritt der else-Fall ein (Zeile 13-16 in Algorithm 8). In der neuen Subinstanz S' werden alle Knoten aus $S \setminus L_S$ übernommen und zusätzlich VZK hinzugefügt. D_S wird von S übernommen.

Um U_S und Z_S für S' zu bestimmen muss das Subgradientenverfahren mit veränderten Basis-Graphen und angepassten Demand-Graphen durchgeführt werden. Deswegen werden in Zeile 18 und 19 alle Knoten aus $S' \setminus L_S$ im Basis-Graphen gelöscht und $S' \setminus D_S$ als angepasster Demand-Graph $H' = (T', D')$ gebildet. Mit diesen beiden Graphen wird das Subgradientenverfahren durchgeführt, um U_S , Z_S für die neue Subinstanz zu setzen (siehe Zeile 21-23 in Algorithm 8).

Mit Hilfe des Subgradientenverfahrens können zulässige Lösungen gefunden werden, die in der Menge $ZL'_{S'}$ gespeichert werden (siehe Zeile 21 in Algorithm 8). Aus $ZL'_{S'}$ wird die zulässige Lösung X' mit den kleinsten Gesamtkosten ermittelt (siehe Zeile 25-31 in Algorithm 8).

Das Setzen des Knotenzustandes von VZK entspricht einer zusätzlichen Bedingung, die die zulässigen Lösungen der verzweigten Subinstanz erfüllen müssen. Die verzweigten Subinstanzen S' besitzen eine Bedingung mehr als die zu verzweigende Subinstanz S , aus der sie entstanden sind. Für die Menge $ZL_{S'}$ der zulässigen Lösungen der Subinstanz S' und die Menge ZL_S der zulässigen Lösungen der Subinstanz S gilt:

$$ZL_{S'} \subseteq ZL_S$$

Für eine optimale Lösung X_S^* der Subinstanz S und eine optimale Lösung $X_{S'}^*$ der Subinstanz S' gilt somit:

$$z(X_{S'}^*) \geq z(X_S^*) \geq z(X^*) \quad (4.6)$$

Aus Gleichung 4.6 folgt: Wenn $z(X_{S'}^*) > z(X^*)$ ist, können die untere Schranken U_S von Subinstanzen S die Gesamtkosten einer optimalen Lösung der Instanz des MCDPP übersteigen. Wenn S' aus der Verzweigung von S am VZK für den Demand $d = \{s, e\}$ oder durch Löschen des VZK entstanden ist, gilt für die Differenz der unteren Schranken von S und S' bei gleichen Lagrange-Vektor λ :

$$\Delta U(S, S', \lambda) = U_{S'}(\lambda) - U_S(\lambda) = \begin{cases} \sum_{d \in D'} z(\text{kürzesterWeg}(G', d, \lambda)) & d = \emptyset \\ - \sum_{d \in D'} z(\text{kürzesterWeg}(G, d, \lambda)) \\ z(\text{kürzesterWeg}(G, \{s, VZK\}, \lambda)) & d = \{s, e\} \\ + z(\text{kürzesterWeg}(G, \{VZK, e\}, \lambda)) \\ - z(\text{kürzesterWeg}(G, d, \lambda)) \end{cases} \quad (4.7)$$

Im Fall des gelöschten VZK ($d = \emptyset$) in Gleichung 4.7 entspricht G' dem Basis-Graphen G mit gelöschtem Knoten VZK . Wegen der zusätzlichen Bedingung des gelöschten Knotens VZK in G' , ist für diesen Fall $\Delta U(S, S', \lambda) \geq 0$. Wegen der optimalen Substruktur kürzester Wege (siehe [1]) ist $\Delta U \geq 0$ für den Fall $d = \{s, e\}$. Es gilt:

$$\Delta U(S, S', \lambda) \geq 0 \quad (4.8)$$

Algorithm 8 instanzBilden($G = (V, E)$, $H = (T, D)$, S , VZK , d , O)

Input: Basis-Graph $G = (V, E)$, Demand-Graph $H = (T, D)$, zu verzweigende Subinstanz S , Verzweigungsknoten VZK , Demand für den Verzweigungsknoten d , globale obere Schranke O

Output: Verzweigte Subinstanz S' , gefundene zulässige Lösung X mit kleinsten Gesamtkosten

```

1: Erzeuge neue Subinstanz  $S'$ 

2:  $\triangleright$  Subinstanz verzweigen
3: if ( $S == \emptyset$ ) then
4:    $S'.L_S = \emptyset$ 
5:   Alle Demands aus  $H$  in  $S'.D_S$  speichern
6: else if  $d$  in  $S.D_S$  then
7:   Startknoten  $s$  und Endknoten  $e$  aus  $d$  bestimmen
8:    $S'.D_S = S.D_S$ 
9:    $S'.D_S.erase(d)$ 
10:   $S'.insert(\{s, VZK\})$ 
11:   $S'.insert(\{VZK, e\})$ 
12:   $S'.L_S = S.L_S$ 
13: else
14:    $S'.L_S = S.L_S$ 
15:    $S'.L_S.insert(VZK)$ 
16:    $S'.D_S = S.D_S$ 
17: end if

18: Lösche alle Knoten aus  $S'.L_S$  in  $G$ 
19: Erstelle aus  $S'.D_S$  angepassten Demand-Graph  $H' = (T', D')$ 

20:  $\triangleright$  Subgradientenverfahren durchführen
21:  $(U_{S'}, Z_{S'}, ZL'_{S'}) = \text{SubgradOpt}(G, H', O)$ 

22:  $\triangleright$  Zuweisung der Werte aus Subgradientenverfahren an Subinstanz
23:  $S'.Z_S = Z_{S'}$ ,  $S'.U_S = U_{S'}$ 

24:  $\triangleright$  zulässige Lösung mit kleinsten Gesamtkosten bestimmen
25:  $O_{min} = \infty$ ,  $X = \emptyset$ 
26: for  $X' \in ZL'_{S'}$  do
27:   if  $z(X') < O_{min}$  then
28:      $O_{min} = z(X')$ 
29:      $X = X'$ 
30:   end if
31: end for

32: return  $S', X$ 

```

4.5 Beschränkungskriterien und globale obere Schranke

Eine zulässige Lösung X_{max} mit den größten Gesamtkosten $z(X_{max})$ bildet eine globale obere Schranke der Instanz des MCDPP. Maximal können die Wege der eingebetteten Demands von X_{max} alle Nicht-Terminalknoten beinhalten. Auf Grund der Knotendisjunktheit einer zulässigen Lösung kann jede Kante nur Teil eines Weges sein und in X_{max} nur einmal vorkommen. Je größer die Kantenkosten einer Kante sind, die Teil eines Weges ist, desto größer werden die Gesamtkosten des Weges. Wenn ein Nicht Terminalknoten Teil eines Weges ist, sind 2 unterschiedliche Kanten dieses Knotens ebenfalls Teil des Weges. Die 2 Kanten mit den größten Kantenkosten eines Nicht Terminalknotens, erhöhen die Kosten des Weges am meisten und werden in einer Menge K gespeichert. Ein Terminal ist immer Teil so vieler Wege, wie Demands, die von ihm ausgehen. Nach dem gleichen Argument können Anzahl der Demands viele Kanten des Terminalknotens mit den größten Kantenkosten der Menge K hinzugefügt werden. Für die Kanten in K wurde die Bedingung der Zusammenhängigkeit der Wege vernachlässigt. Es gilt:

$$z(X_{max}) \leq z(K) \quad (4.9)$$

Aus Gleichung 4.9 folgt eine weitere globale obere Schranke O des Branch and Bound Verfahrens, die in Gleichung 4.10 dargestellt wird:

$$O = z(K) \quad (4.10)$$

O kann einfach berechnet werden und ist die globale obere Schranke mit der das Branch and Bound Verfahren startet (siehe Zeile 2 in Algorithm 1). Diese globale obere Schranke wird mit der Funktion obereSchranke erzeugt, die in Algorithm 9 dargestellt wird.

In Algorithm 9 werden alle Knoten des Basis-Graphen durchgegangen. Wie viele Kanten für jeden Knoten gesucht werden, wird in ω aus Gleichung 4.2 beschrieben. In jeder Iteration wird die Menge $GK(i)$ aller Kanten des Knotens i gebildet und nach den Kantenkosten absteigend sortiert (siehe Zeile 5-6 in Algorithm 9). Die Kante k mit den größten Kantenkosten ist das erste Element in $GK(i)$. k wird aus GK entfernt und zu K hinzugefügt (siehe Zeile 8-9 in Algorithm 9). Dies geschieht $\omega(i)$ Male. Auf diese Weise werden $\omega(i)$ viele Kanten des Knotens i mit den größten Kantenkosten gefunden und in K hinzugefügt. K ist eine Menge und besitzt ausschließlich unterschiedliche Kanten. Am Ende können die Gesamtkosten $z(K)$ berechnet werden, um die globale obere Schranke zu ermitteln.

Algorithm 9 obereSchranke($G = (V, E)$, $H = (T, D)$)

Input: Basis-Graph $G = (V, E)$, Demand-Graph $H = (T, D)$

Output: Globale obere Schranke O

- 1: $O = \infty$
 - 2: $K = \emptyset$
 - 3: Bilde ω nach Gleichung 4.2
 - 4: **for** Knoten i in G **do**
 - 5: Bilde Liste $GK(i)$ aller Kanten von i
 - 6: Sortiere Kanten in $GK(i)$ nach Kantenkosten absteigend
 - 7: **for** $j = 1$ to $\omega(i)$ **do**
 - 8: $k = GK(i).pop()$
 - 9: $K.insert(k)$
 - 10: **end for**
 - 11: **end for**
 - 12: $O = z(K)$
 - 13: **return** O
-

An Testinstanz 1 kann die globale obere Schranke O dargestellt werden:

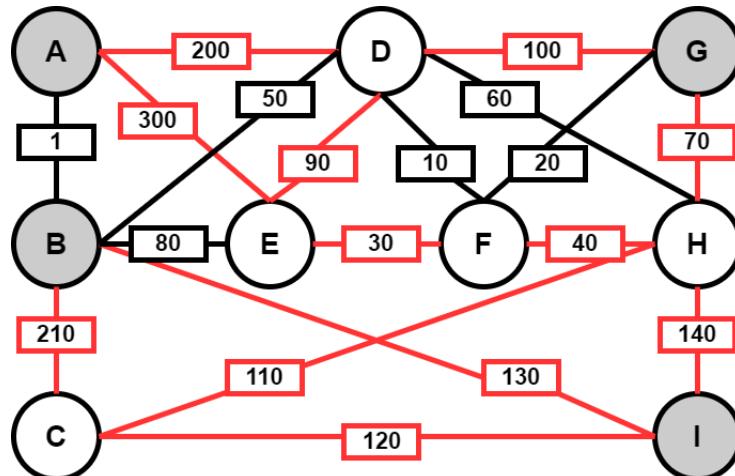


Abbildung 4.5: Darstellung der Kantenmenge K für Testinstanz 1

In Abbildung 4.5 wurde die Kantenmenge K für Testinstanz 1 gebildet und als rote Kanten in dargestellt. $z(K)$ beträgt in diesem Fall 1540.

Während des Branch and Bound Verfahrens entstehen im Subgradientenverfahren zulässige Lösungen. Diese verändern den Wert der globalen oberen Schranke, wenn die Gesamtkosten der zulässigen Lösung kleiner sind, als der Wert der globalen oberen Schranke. Dieses Update der globalen oberen Schranke wurde in Kapitel 4.1 erklärt.

Besitzt eine Subinstanz S eine untere Schranke U_S , die größer gleich der globalen oberen

Schranke ist, ist diese Subinstanz suboptimal. Suboptimale Subinstanzen können nach Verzweigung nur zulässige Lösungen X mit $z(X) \geq U_S \geq O$ erzeugen. Somit führt die Verzweigung dieser Subinstanzen nicht zu einer optimalen Lösung. Suboptimale Subinstanzen werden nicht weiterverzweigt und aus W entfernt oder nicht in W hinzugefügt.

Subinstanzen mit $Z_S = \text{UNZULÄSSIG}$ können keine zulässigen Lösungen erzeugen. Die Verzweigung dieser Subinstanzen führt wieder zu Subinstanzen mit $Z_S = \text{UNZULÄSSIG}$. Demzufolge führt die Verzweigung dieser Subinstanzen nicht zu einer optimalen Lösung der Instanz des MCDPP. Diese Subinstanzen werden nicht weiterverzweigt und nicht in W aufgenommen.

Ist $Z_S = \text{OPTIMAL}$, wurde während des Subgradientenverfahrens eine optimale Lösung X_S^* dieser Subinstanz S gefunden. Diese Lösung entspricht einer zulässigen Lösung der Instanz des MCDPP und ist damit in ZL' (siehe Algorithm 5 Zeile 13-15) enthalten. X_S^* wird somit beim Update der globalen oberen Schranke (siehe Zeile 11 und 23 in Algorithm 1) berücksichtigt und kann die globale obere Schranke der Instanz des MCDPP verändern. Eine Verzweigung dieser Subinstanz kann nur zu zulässigen Lösungen führen, deren Gesamtkosten größer gleich der $z(X_S^*)$ sind (siehe Gleichung 4.6). Alle verzweigten Subinstanzen S' sind dadurch suboptimal. Demzufolge müssen Subinstanzen mit Zustand $Z_S = \text{OPTIMAL}$ nicht weiterverzweigt werden. Sie werden nicht in W aufgenommen.

4.6 Auswahlstrategie

W ist die Menge aller verzweigbaren Subinstanzen und aus ihr wird die nächste Subinstanz zum Verzweigen ausgewählt. Für dieses Branch and Bound Verfahren wird immer die Subinstanz S mit dem kleinsten Wert U_S von allen Subinstanzen in W ausgewählt. Die Subinstanz, die weiterverzweigt werden soll, entspricht somit dem ersten Element der Prioritätswarteschlange W und kann über `pop` ausgewählt werden (siehe Zeile 14 in Algorithm 1). Die kleinste untere Schranke aller Subinstanzen in W bildet eine globale untere Schranke (siehe Kapitel 4.7). Wird die globale untere Schranke der Instanz des MCDPP größer gleich der globalen oberen Schranke der Instanz des MCDPP, endet das Branch and Bound Verfahren. Ziel dieser Auswahlstrategie ist es durch Verzweigung die globale untere Schranke des Branch and Bound Verfahrens zu erhöhen, weshalb die erste Subinstanz in der Warteschlange zum Weiterverzweigen ausgewählt wird.

4.7 Obere und untere Schranke

Für die lokale untere Schranke gilt Gleichung 2.5. Durch Gleichung 4.6 und der Erzeugung unterer Schranken mit Hilfe der Lagrange Relaxation ist Gleichung 2.5 gewährleistet. Lokale untere Schranken einer Subinstanz werden durch das Subgradientenverfahren ermittelt.

Eine lokale obere Schranke nach Gleichung 2.4 wird nicht benötigt, weil sie den Baum des Branch and Bound Verfahrens nicht beschränkt.

Für die globale obere Schranke gilt Gleichung 2.3. O berechnet nach Gleichung 4.10 ist eine globale obere Schranke. Weiterhin sind die Gesamtkosten jeder zulässigen Lösung, die während des Branch and Bound Verfahrens gefunden werden, eine globale obere Schranke. Globale obere Schranken beschränken die Gesamtkosten einer optimalen Lösung nach oben

und werden zur Beschränkung des Baums des Branch and Bound Verfahrens verwendet (siehe Suboptimalität in Kapitel 4.5).

Die untere Schranke U_S der Subinstanz S wird mit dem Subgradientenverfahren berechnet. S besitzt eine optimale Lösung X_S^* und zulässige Lösungen X_S . Es gilt:

$$z(X_S) \geq z(X_S^*) \geq U_S \quad (4.11)$$

$$X_S, X_S^* \in ZL \quad (4.12)$$

$$z(X_S^*) \geq z(X^*) \quad (4.13)$$

Aus Gleichung 4.13 und 4.11 und der Tatsache, dass suboptimale Lösungen aus W entfernt werden folgt:

$$U_S < z(X^*) \wedge Z_S = \text{KANDIDAT} \Rightarrow S \in W \quad (4.14)$$

Keine zulässige Lösung besitzt kleinere Gesamtkosten als $z(X^*)$. Verzweigbare Subinstanzen mit $U_S < z(X^*)$ können deswegen nicht aus W entfernt werden. Nach Gleichung 4.14 sind alle im Branch and Bound Verfahren gefundenen verzweigbaren Subinstanzen mit $U_S < z(X^*)$ in W enthalten.

Bei der Verzweigung wird jeder Knotenzustand und jeder angepasste Demand-Graph berücksichtigt. Es ist deshalb möglich, jede der $|M_Z|$ Lösungen des Enumerationsverfahrens aus Kapitel 3 zu bilden. Ist in der Menge der zulässigen Lösungen ZL_S einer Subinstanz eine optimale Lösung enthalten, so gilt wegen Gleichung 4.11:

$$X^* \in ZL_S \Rightarrow U_S \leq z(X^*) \quad (4.15)$$

Bei der Verzweigung der Gesamtinstanz S_{start} wird jeder Knotenzustand des Verzweigungsknotens und zugehöriger angepasster Demand-Graph berücksichtigt, sodass die Subinstanz S^* , die durch Verzweigung zu einer optimalen Lösung führen wird, in W enthalten ist. Wegen Gleichung 4.15 und 4.14 entstehen nach der Verzweigung von S^* Subinstanzen, die die Gleichung 4.14 erfüllen. In W sind deswegen solange Elemente enthalten, bis eine optimale Lösung gefunden wird. Dies wird in Gleichung 4.16 beschrieben.

$$\exists X^* \wedge O_W > z(X^*) \Rightarrow W_W \neq \emptyset \quad (4.16)$$

Bis eine optimale Lösung gefunden wird, existiert mindestens eine Subinstanz S mit $U_S < z(X^*)$ in W . Die kleinste untere Schranke aller Subinstanzen in W ist deswegen eine globale untere Schranke der Instanz des MCDPP.

Das Branch and Bound Verfahren endet, sobald W leer ist. Existiert für die Instanz des MCDPP keine optimale Lösung, so können die unteren Schranken der Subinstanzen unendlich groß werden. Die unteren Schranken der Subinstanzen können deswegen die globale obere Schranke O (Gleichung 4.10) überschreiten und werden suboptimal oder besitzen $Z_S = \text{UNZULÄSSIG}$. In beiden Fällen werden diese Subinstanzen nicht mehr weiterverzweigt und die Größe von W wird kleiner, bis $W = \emptyset$ ist. Während des gesamten Branch and Bound Verfahrens fand kein Update der oberen Schranke statt, weil keine zulässige Lösung gefunden werden kann. Dementsprechend ist $X_W = \emptyset$ und man erfährt aus dem Branch and Bound Verfahren, das diese Instanz des MCDPP keine Lösung besitzt. Die untere Schranke einer Subinstanz wird durch das Subgradientenverfahren bestimmt, wodurch nicht garantiert

ist, untere Schranken zu finden, die größer sind als O . Spätestens wenn alle Knotenzustände des Basis-Graphen gesetzt sind, ist der Zustand der Subinstanz UNZULÄSSIG und die Subinstanz ist nicht mehr verzweigbar.

Existiert für die Instanz des MCDPP eine optimale Lösung, so befindet sich in W mindestens eine Subinstanz, bis eine optimale Lösung gefunden wird (siehe Gleichung 4.16). Die zulässigen Lösungen einschließlich optimaler Lösungen werden durch das Subgradientenverfahren bestimmt. Das Finden zulässiger Lösungen wird durch das Subgradientenverfahren nicht garantiert. Spätestens wenn alle Knotenzustände des Basis-Graphen gesetzt sind, ist der Zustand der Subinstanz OPTIMAL oder UNZULÄSSIG. Ist der Zustand OPTIMAL, wurde eine zulässige Lösung gefunden, die zu einem Update der globalen oberen Schranke führen kann. Es können mit dem Branch and Bound Verfahren jede der $|MZ|$ Lösungen des Enumerationsverfahrens aus Kapitel 3 gebildet werden, wodurch eine optimale Lösung spätestens dann gefunden wird, sobald alle $|MZ|$ Subinstanzen ermittelt wurden. Für das Update der globalen oberen Schranke wird immer die zulässige Lösung mit den kleinsten Gesamtkosten gewählt, wodurch garantiert wird, eine optimale Lösung der Instanz des MCDPP zu finden. Wird eine optimale Lösung gefunden, findet ein Update der globalen oberen Schranke statt. Die neu entstehenden Subinstanzen können wegen Gleichung 4.13 durch Verzweigung nur suboptimal werden oder $Z_S = \text{UNZULÄSSIG}$ oder $Z_S = \text{OPTIMAL}$ besitzen. Dies passiert spätestens, wenn in einer Subinstanz alle Knotenzustände gesetzt sind. Die durch Verzweigung entstehenden Subinstanzen werden damit nicht mehr verzweigbar und werden nicht in W aufgenommen. Die Größe von W wird stetig kleiner, bis $W = \emptyset$ und das Branch and Bound Verfahren endet. Eine optimale Lösung der Instanz des MCDPP wurde in X_W gespeichert und vom Branch and Bound Verfahren zurückgegeben.

4.8 Optimalitätskriterien

Für das MCDPP existiert kein Optimalitätskriterium, welches zum vorzeitigen Abbruch des Branch and Bound Verfahrens führen könnte. Der Beweis des Findens optimaler Lösungen mit Hilfe des Branch and Bound Verfahrens wurde in Kapitel 4.7 gezeigt.

In folgendem Spezialfall existiert ein hinreichendes Optimalitätskriterium, wodurch das Branch and Bound Verfahren vorzeitig abgebrochen werden kann. Ist der Zustand der Gesamtinstanz S_{Start} OPTIMAL, so gilt Definition 4.1 und eine optimale Lösung für S_{Start} wurde gefunden. S_{Start} ist die Instanz des MCDPP, wodurch die gefundene Lösung gleichzeitig eine optimale Lösung der Instanz des MCDPP ist. Das Branch and Bound Verfahren kann damit vorzeitig abgebrochen werden. Dies wird in Zeile 7-11 in Algorithm 1 berücksichtigt. Wurde für S_{Start} eine optimale Lösung gefunden, ist die S_{Start} nicht verzweigbar. Das Update der globalen oberen Schranke mit der optimalen Lösung findet statt und es ist $W = \emptyset$. Das Branch and Bound Verfahren endet mit der gefundenen optimalen Lösung. Ist der Zustand der Gesamtinstanz $Z_S = \text{UNZULÄSSIG}$ oder ist die Gesamtinstanz suboptimal, so kann sie nicht zu W hinzugefügt werden. W und X sind leer und es findet kein Update der globalen oberen Schranke statt. Das Branch and Bound Verfahren bestimmt damit die Nicht-Existenz einer optimalen Lösung.

4.9 Einstellbare Parameter und Strategien

Die in C++ implementierte Version des Branch and Bound Verfahrens besitzt einstellbare Parameter, die in diesem Kapitel beschrieben werden.

Es gibt viele unterschiedliche Möglichkeiten die Schrittweite α des Subgradientenverfahrens zu realisieren. Hier wird die newtonsche Schrittweite (siehe [2]) verwendet:

$$\alpha^{(k)} = \theta^{(k)} \frac{\bar{L} - U(\lambda^{(k)})}{\|\zeta^{(k)}\|^2} \quad (4.17)$$

$$\theta^{(k)} = \begin{cases} 2 & k = 1 \\ \frac{\theta^{(k-1)}}{2} & \text{mod}(k, n_{iter_half}) = 0 \wedge \max_{j \in [1, k]} U(\lambda^{(j)}) = \max_{j \in [1, k-n]} U(\lambda^{(j)}) \\ \theta^{(k-1)} & \text{sonst} \end{cases} \quad (4.18)$$

$\theta^{(k)}$ aus Gleichung 4.18 ist eine Folge von reellen Werten, die mit dem Wert 2 startet. Sollte innerhalb von n_{iter_half} Iterationen sich der Wert der größten gefundenen unteren Schranke nicht verändern, wird $\theta^{(k)}$ halbiert. Tritt dies nicht ein, bleibt der Wert von $\theta^{(k)}$ gleich. In Gleichung 4.17 ist \bar{L} eine beliebige obere Schranke der Instanz des MCDPP. Diese Schrittweite $\alpha^{(k)}$ wurde heuristisch aus dem Newton-Verfahren hergeleitet (siehe [2]).

Es ergeben sich folgende einstellbare Parameter für das Subgradientenverfahren, die für das Branch and Bound Verfahren ermittelt werden müssen:

- n_{sub} : Anzahl der Iterationen des Subgradientenverfahrens
- n_{iter_half} : Anzahl Iterationen für Update von θ_k
- \bar{L} : obere Schranke der Instanz des MCDPP
- λ_{Start} : Lagrange-Vektor mit dem Subgradientenverfahren startet

Für die Gesamtinstanz ist $\lambda_{start} = \mathbf{0}$, wodurch mit den Basis-Graphen ohne angepasste Kantenkosten gestartet wird. Für die verzweigten Subinstanzen kann λ_{start} beliebige Werte annehmen. Im Rahmen der Bachelorarbeit werden 3 unterschiedliche Varianten gewählt, um λ_{start} zu ermitteln. Dies wird mit dem Parameter λ_{Option} wie folgt beschrieben:

1. $\lambda_{Option} = \lambda_NULL \Leftrightarrow \lambda_{Start} = \mathbf{0}$
2. $\lambda_{Option} = \lambda_MAX \Leftrightarrow \lambda \in P$ mit $P = \{\lambda^{(k)} \wedge k \in [1, n_{sub}] \mid U(\lambda^{(k)}) = U\}$
3. $\lambda_{Option} = \lambda_KONFLIKT \Leftrightarrow \lambda \in P'$ mit

$$P' = \{\lambda^{(k)} \wedge k \in [1, n_{sub}] \mid MK^{(k)} = \min_{k \in [1, n_{sub}]} |MK^{(k)}|\}$$

Ist $\lambda_{Option} = \lambda_NULL$ startet bei jeder verzweigten Subinstanz das Subgradientenverfahren mit $\lambda_{Start} = \mathbf{0}$. Die Kantenkosten des Basis-Graphen werden somit in der ersten Iteration nicht durch den Parameter λ verändert.

Ist $\lambda_{Option} = \lambda_MAX$ startet bei jeder verzweigten Subinstanz S' das Subgradientenverfahren mit dem λ , welches beim Subgradientenverfahren der zu verzweigenden Subinstanz S zur größten gefundenen unteren Schranke U geführt hat. Wegen Gleichung 4.8 wird die untere

Schranke von S' größer gleich der unteren Schranke von S sein. Dies hat zur Folge, dass die globale untere Schranke nach jeder Verzweigung nur ansteigen oder gleich bleiben kann. Sollte P mehr als 1 Element besitzen, wird das $\lambda^{(k)}$ mit dem kleinsten Wert für k gewählt. Diese Wahl ist beliebig.

Ist $\lambda_{Option} = \lambda_KONFLIKT$ startet bei jeder verzweigten Subinstanz S' das Subgradientenverfahren mit dem λ , welches beim Subgradientenverfahren der zu verzweigenden Subinstanz S zur minimalen Anzahl an Konfliktknoten $|MV^{(k)}|$ geführt hat. Es wird vermutet, mit dieser Strategie schnell zulässige Lösungen zu finden, weil die Lösung mit der geringsten Zahl an Konfliktknoten einer zulässigen Lösung sehr nahe kommt. Besitzt P' mehr als ein Element, wird das $\lambda^{(k)}$ gewählt, welches der gefundenen zulässigen Lösung mit den geringsten Gesamtkosten entspricht. Werden keine zulässigen Lösungen im Subgradientenverfahren ermittelt, wird das $\lambda^{(k)}$ gewählt, welches von allen Elementen in P' die größte untere Schranke erzeugt hat. Der Grund hierfür ist derselbe wie bei $\lambda_{Option} = \lambda_MAX$. Existieren mehrere Elemente in P' , welche die größte untere Schranke erzeugt haben, wird das $\lambda^{(k)}$ mit dem kleinsten Wert für k gewählt. Diese Wahl ist beliebig.

Ist $\lambda_{Option} = \lambda_KONFLIKT$ oder $\lambda_{Option} = \lambda_KONFLIKT$ kann λ_{Start} für jede Subinstanz unterschiedlich sein. Der Lagrange-Vektor λ für die unterschiedlichen λ_{Option} wird deswegen als zusätzliches Attribut λ_S in jeder Subinstanz gespeichert.

\bar{L} wird mit der Funktion IUpperBilden in Algorithm 10 erzeugt. Sollte die Summe aller Kantenkosten KS größer sein als die Summe aus globaler obere Schranke O aus Gleichung 4.10 und minimalen Kantenkosten a_{min} , dann ist \bar{L} die Summe aller Kantenkosten. Andernfalls ist \bar{L} die Summe aus O und a_{min} (siehe Zeile 3 in Algorithm 10). Es gilt somit: $\bar{L} > O$. Die newtonscche Schrittweite hängt von der Differenz $\bar{L} - U(\lambda^{(k)})$ ab. Wird diese Differenz nahezu Null, ändert sich $\lambda^{(k)}$ im Subgradientenverfahren nur sehr gering. Besitzt das MCDPP keine Lösung, können dadurch Subinstanzen entstehen, deren untere Schranke sich langsam \bar{L} nähern. Ist $\bar{L} > O$ werden suboptimale Subinstanzen schneller erkannt.

Algorithm 10 IUpperBilden($G = (V, E)$, O)

Input: Basis-Graph $G = (V, E)$, globale obere Schranke O nach Gleichung 4.10

Output: Parameter des Subgradientenverfahrens \bar{L}

- 1: Bestimme minimales Kantenkosten a_{min}
 - 2: $KS = \sum_{e \in E} c(e)$
 - 3: $\bar{L} = \max(KS, O + a_{min})$
 - 4: **return** \bar{L}
-

Mit steigender Anzahl an Iterationen n_{sub} für das Subgradientenverfahren können höhere untere Schranken und mehr zulässige Lösungen gefunden werden. Jedoch steigt damit der Zeitaufwand. Deswegen wird n_{sub} vom Benutzer festgelegt.

n_{iter_half} dient der Aktualisierung von $\theta^{(k)}$ und damit auch der Aktualisierung der Schrittweite. Die Wahl von n_{iter_half} ist dem Benutzer überlassen.

Ein Verzweigungsknoten VZK besitzt die Eigenschaft kein Terminalknoten zu sein und einen unbesetzten Knotenzustand zu besitzen. Existiert kein solcher Knoten für eine Subinstanz, sind alle Knotenzustände im Basis-Graphen gesetzt. Diese Subinstanz besitzt dann

den Zustand OPTIMAL oder UNZULÄSSIG. Jede Subinstanz in W besitzt den Zustand KANDIDAT, wodurch mindestens ein Verzweigungsknoten bestimmt werden kann. Theoretisch kann ein beliebiger VZK gewählt werden, der die vorher genannten Eigenschaften eines Verzweigungsknotens erfüllt. Ziel ist es durch Wahl des VZK die unteren Schranken der verzweigten Subinstanzen zu erhöhen. λ_{Start} wird wegen λ_{Option} für S' durch S erzeugt. Es ergibt sich somit nach Gleichung 4.8 eine Vergleichsmöglichkeit der unteren Schranken von S und S' bei λ_{Start} . Bei den folgenden 3 Verzweigungsstrategien, die durch VZK_{Option} eingestellt werden, wird versucht $\Delta U(S, S', \lambda_{Start})$ zu erhöhen. Dadurch können je nach λ_{Option} höhere untere Schranken für S' erzeugt werden im Vergleich zu unteren Schranken für S .

1. $VZK_{Option} = \text{MAX_FLUSS}$
2. $VZK_{Option} = \text{WEG}$
3. $VZK_{Option} = \text{KONFLIKTE}$

Tritt der erste Fall aus Gleichung 4.8 ein, wird VZK im Basis-Graphen gelöscht. Befindet sich VZK auf keinem der kürzesten Wege, die beim Subgradientenverfahren für S bei λ_{Start} entstehen, ist $\Delta U(S, S', \lambda_{Start}) = 0$. Um $\Delta U(S, S', \lambda_{Start})$ erhöhen zu können werden deswegen nur VZK ausgewählt, die sich auf den kürzesten Wegen befinden, die beim Subgradientenverfahren für S bei λ_{Start} entstehen.

Ist $VZK_{Option} = \text{MAX_FLUSS}$, ist VZK der Knoten u mit dem größten Wert von $\mu(u)$ für das Subgradientenverfahrens von S bei λ_{Start} . Es wird folglich der Konfliktknoten gewählt, welcher am häufigsten Teil eines kürzesten Weges für einen Demand ist. Ist es nicht möglich solch einen VZK auf diese Weise zu bestimmen, ist VZK der erste Nicht-Terminalknoten mit ungesetztem Knotenzustand, der beim Durchlauf aller Knoten des Basis-Graphen gefunden wird.

Ist $VZK_{Option} = \text{WEG}$, wird der kürzeste Weg eines eingebetteten Demands gewählt, der beim Subgradientenverfahren von S bei λ_{Start} entsteht. Betrachtet man diesen Weg als Knotenfolge mit m Knoten, so ist der mittlere Knoten dieses Weges an der Position $\lfloor \frac{m+1}{2} \rfloor$. Dieser mittlere Knoten wird dann als Verzweigungsknoten gewählt. Sollte es nicht möglich sein, diesen Knoten zu bestimmen, ist VZK der erste Nicht-Terminalknoten mit ungesetzten Knotenzustand, der beim Durchlauf aller Knoten des Basis-Graphen gefunden wird.

Ist $VZK_{Option} = \text{KONFLIKTE}$, wird die Menge an Konfliktknoten MV , die während des Subgradientenverfahren von S bei λ_{Start} entsteht, als Menge von Kandidaten für den Verzweigungsknoten betrachtet. In Algorithm 11 ist dargestellt, wie aus der Menge MV der Knoten VZK bestimmt wird. Zuerst wird die Menge der Konfliktknoten bei λ_{Start} bestimmt (siehe Zeile 1-4 Algorithm 11). Es ist möglich, dass MV leer ist. Dieses Szenario tritt ein, wenn die Einbettung der Demands bei $S.\lambda_S$ einer zulässigen Lösung entspricht, die keine optimale Lösung der Subinstanz ist. In diesem Fall wird VZK nach $VZK_{Option} = \text{WEG}$ gewählt (Zeile 6-9 in Algorithm 11). Für jeden Knoten aus MV wird eine Verzweigung ausgeführt (siehe Zeile 12-13 in Algorithm 11). Bei der Verzweigung entspricht die Anzahl der Iterationen des Subgradientenverfahrens n'_{sub} mit $1 \leq n'_{sub} \leq n_{sub}$. Der Zeitaufwand zum Finden des Verzweigungsknotens soll dadurch begrenzt werden. Bei der Verzweigung wird die untere Schranke U' der verzweigten Subinstanz mit dem kleinsten Wert im Vergleich zu den anderen verzweigten Subinstanzen ermittelt. U' wird kleinste untere Schranke nach Verzweigung genannt. Von allen Knoten aus MV wird genau der Knoten als VZK gewählt,

welcher den größten Wert U' erzeugt (siehe Zeile 14-17 in Algorithm 11). Ziel ist es den Wert der lokalen unteren Schranken nach Verzweigung zu erhöhen, wodurch sich die globale untere Schranke der Instanz des MCDPP erhöhen soll. Sind alle Subinstanzen nach Verzweigung nicht verzweigbar, erzeugt dieser Verzweigungsknoten keine verzweigbaren Subinstanzen. Dieser Knoten wird bevorzugt als Verzweigungsknoten ausgewählt (Zeile 18-22 in Algorithm 11). Dies dient der Beschränkung von W .

Algorithm 11 verzweigungsKnoten($G = (V, E)$, $H = (T, D)$, S , O , n'_{sub})

Input: Basis-Graph $G = (V, E)$, Demand-Graph $H = (T, D)$, zu verzweigende Subinstanz S , globale obere Schranke O , verringerte Anzahl Iterationen des Subgradientenverfahrens n'_{sub}

Output: Verzweigungsknoten VZK

```

1: > Menge an Konfliktknoten für Wahl von VZK bestimmen
2: Lösche alle Knoten aus  $S.L_S$  in  $G$ 
3: Erstelle aus  $S.D_S$  angepassten Demand-Graph  $H' = (T', D')$ 
4:  $(MV, X, \mu) = \text{wegFinden}(G, H', S.\lambda_S)$ 

5: > Menge an Konfliktknoten ist leer
6: if  $|MV| == 0$  then
7:    $VZK$  wird nach  $VZK_{Option} = \text{WEG}$  bestimmt
8:   return  $VZK$ 
9: end if

10:  $U_{max} = -\infty$ 

11: > VZK aus MV bestimmen
12: for Knoten  $i$  in  $MV$  do

13:    $(MS, X) = \text{instanzVerzweigen}(G, H, S, i, O)$ 
14:    $U' = \min_{S' \in MS} S'.U_S$ 

15:   if  $U' > U_{max}$  then
16:      $VZK = i$ 
17:   end if
18:   if  $|MS| == 0$  then
19:      $VZK = i$ 
20:   break
21:   end if
22: end for

23: return  $VZK$ 

```

Die vom Benutzer einstellbaren Parameter sind: $G = (V, E)$, $H = (T, D)$, n_{sub} , n'_{sub} , n_{iter_half} , λ_{Option} , VZK_{Option} . Es gibt ebenfalls einen Default-Modus für beliebige Instanzen mit $G = (V, E)$, $H = (T, D)$, bei dem $n_{sub} = 200$, $n'_{sub} = 40$, $n_{iter_half} = 10$, $\lambda_{Option} = \lambda_{MAX}$, $VZK_{Option} = MAX_FLUSS$ ist.

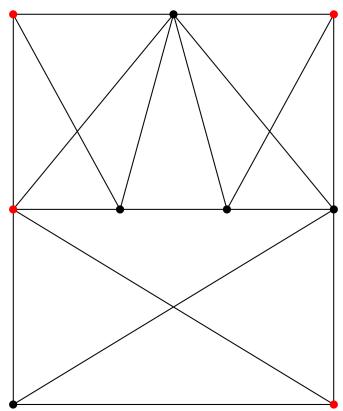
5 Testinstanzen

5.1 Kleine Testinstanzen

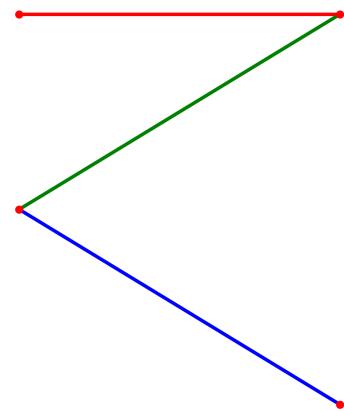
In diesem Abschnitt werden kleine Testinstanzen mit Hilfe des implementierten Branch and Bound Verfahren gelöst. Klein bedeutet, dass die Knoten- oder Kantenanzahl des Basisgraphen der Testinstanz einen Wert von 50 nicht überschreitet. In der graphischen Darstellung der Testinstanzen entsprechen die roten Knoten den Terminalknoten.

Zuerst wird Testinstanz 1 mit dem implementierten Branch and Bound Verfahren gelöst. Testinstanz 1 besitzt 9 Knoten und 18 Kanten und wird in Abbildung 5.1 dargestellt. Es sollen 3 Demands als Wege zwischen den Terminals eingebettet werden. Eine ermittelte optimale Lösung von Testinstanz 1 ist in Abbildung 5.1 dargestellt und besitzt den Wert 560. Testinstanz 2 besitzt einen Basis-Graphen mit 19 Knoten und 37 Kanten und wird in Abbildung 5.2 dargestellt. Es sollen insgesamt 4 Demands als Wege zwischen den Terminals eingebettet werden. Eine ermittelte optimale Lösung von Testinstanz 2 ist in Abbildung 5.2 dargestellt und besitzt den Wert 40.

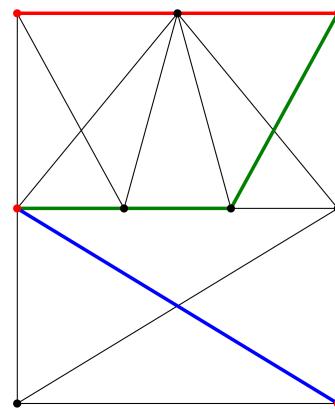
Eine optimale Lösung für Testinstanz 1 und Testinstanz 2 war gegeben und wird vom Branch and Bound Verfahren richtig erkannt.



(a) Basisgraph

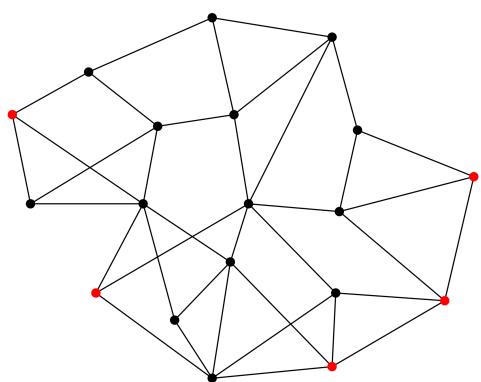


(b) Demand-Graph

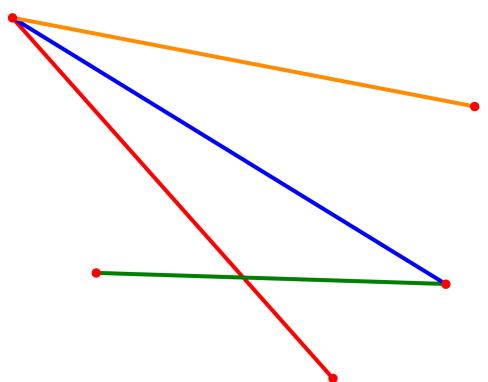


(c) Optimale Einbettung

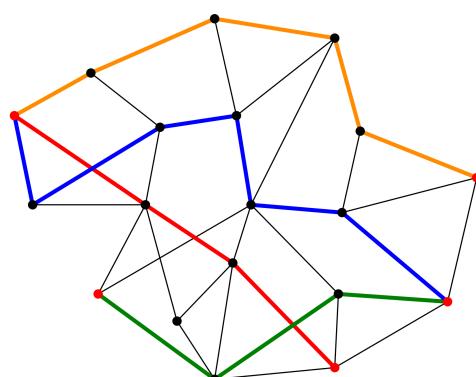
Abbildung 5.1: Testinstanz 1 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 560



(a) Basisgraph



(b) Demand-Graph



(c) Optimale Einbettung

Abbildung 5.2: Testinstanz 2 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 40

5.2 Große Testinstanzen

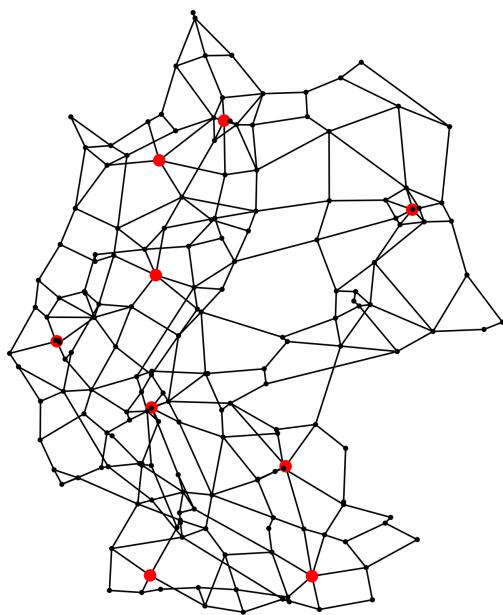
In diesem Abschnitt werden große Testinstanzen mit Hilfe des implementierten Branch and Bound Verfahren gelöst. Groß bedeutet, dass die Knoten- oder Kantenanzahl des Basisgraphen der Testinstanz immer größer als 100 ist. Diese Testinstanzen, sollen ein deutsches Telekommunikationsnetz darstellen mit unterschiedlicher Knoten- und Kantenanzahl je nach Testinstanz. In der graphischen Darstellung der Testinstanzen entsprechen die roten Knoten den Terminalknoten.

Testinstanz 3 besitzt einen Basis-Graphen mit 185 Knoten und 354 Kanten und wird in Abbildung 5.3 dargestellt. Es sollen insgesamt 19 Demands als Wege zwischen den Terminals eingebettet werden. Eine ermittelte optimale Lösung von Testinstanz 3 ist in Abbildung 5.3 dargestellt und besitzt den Wert 8462.

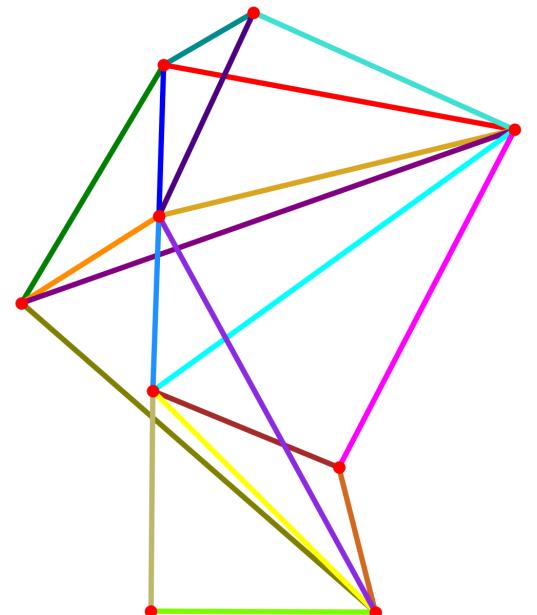
In Testinstanz 4 wurde die Knoten- und Kantenzahl von Testinstanz 3 erhöht. Testinstanz 4 besitzt einen Basis-Graphen mit 676 Knoten und 1107 Kanten und wird in Abbildung 5.4 dargestellt. Es sollen insgesamt 19 Demands als Wege zwischen den Terminals eingebettet werden. Eine ermittelte optimale Lösung von Testinstanz 4 ist in Abbildung 5.4 dargestellt und besitzt den Wert 7442.

In Testinstanz 5 wurde die Knoten- und Kantenzahl von Testinstanz 3 weiter erhöht. Testinstanz 5 besitzt einen Basis-Graphen mit 6893 Knoten und 8074 Kanten und wird in Abbildung 5.5 dargestellt. Es sollen insgesamt 20 Demands als Wege zwischen den Terminals eingebettet werden. Für diese Testinstanz existieren keine Referenzwerte für eine optimale Lösung. Eine zulässige Lösung von Testinstanz 5 ist in Abbildung 5.5 dargestellt und besitzt den Wert 8027. Der Grund für die Darstellung einer zulässigen Lösung wird in Kapitel 5.4 erläutert.

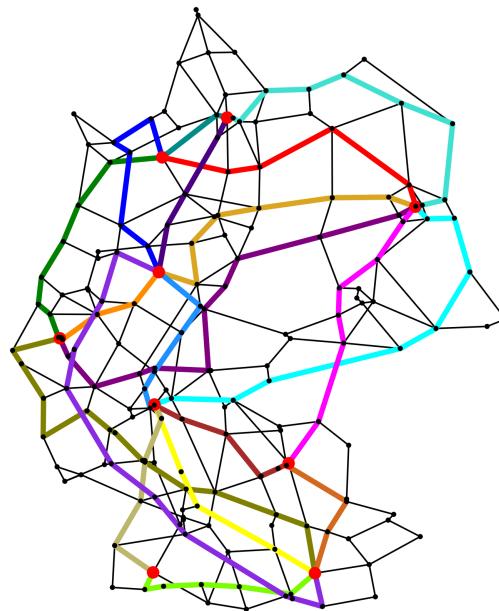
Eine optimale Lösung für Testinstanz 3 und Testinstanz 4 war gegeben und wurde vom Branch and Bound Verfahren richtig erkannt.



(a) Basisgraph

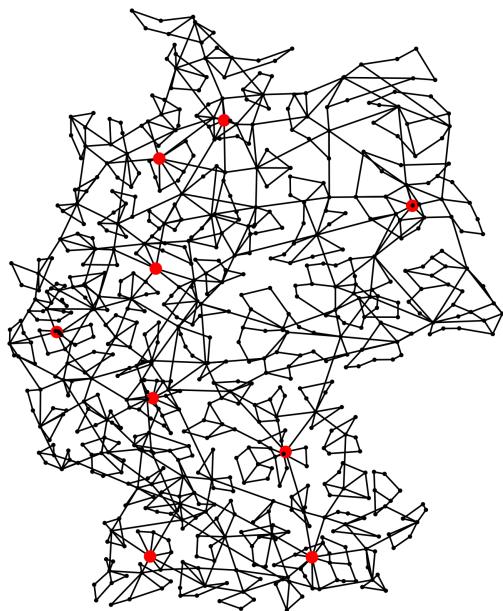


(b) Demand-Graph

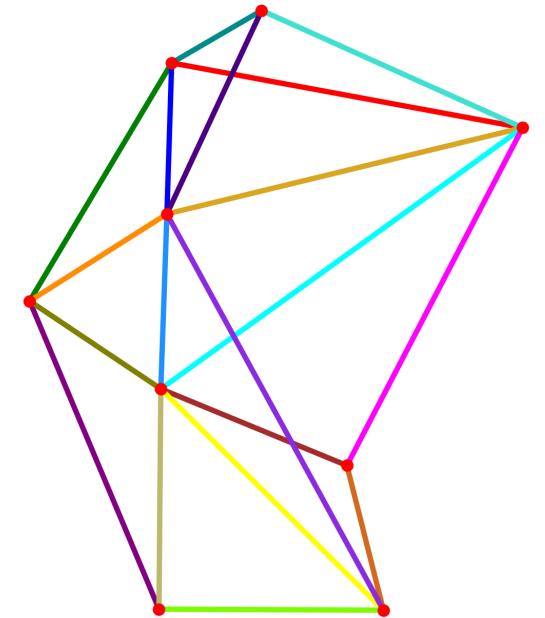


(c) Optimale Einbettung

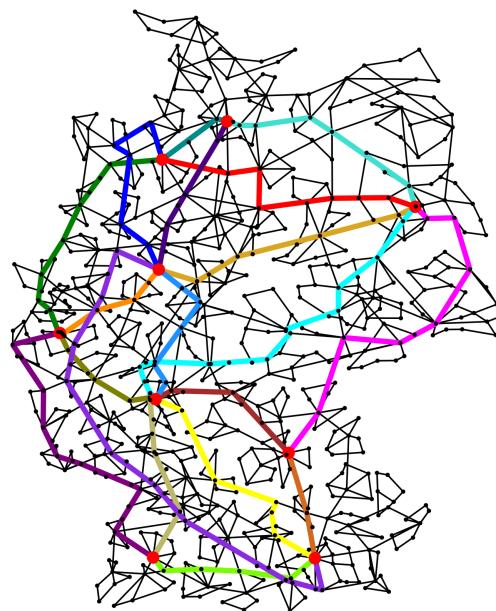
Abbildung 5.3: Testinstanz 3 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 8462



(a) Basisgraph

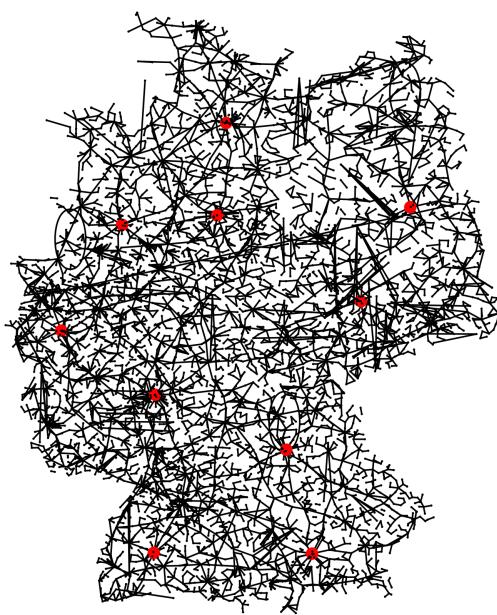


(b) Demand-Graph

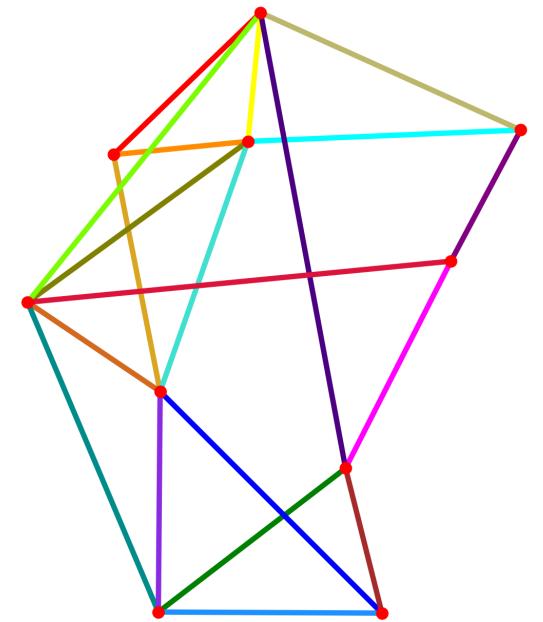


(c) Optimale Einbettung

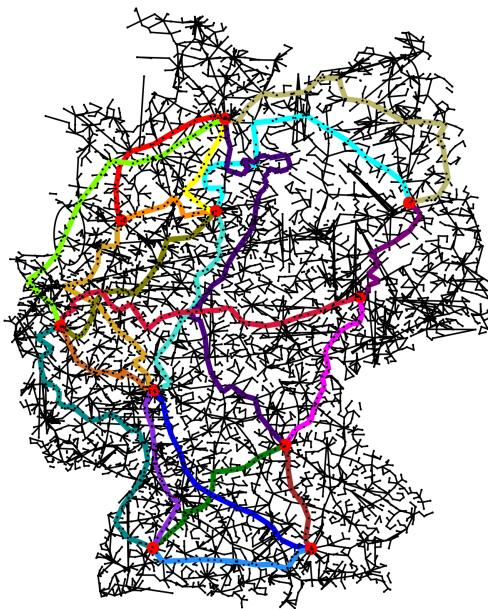
Abbildung 5.4: Testinstanz 4 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 7442



(a) Basisgraph



(b) Demand-Graph



(c) Optimale Einbettung

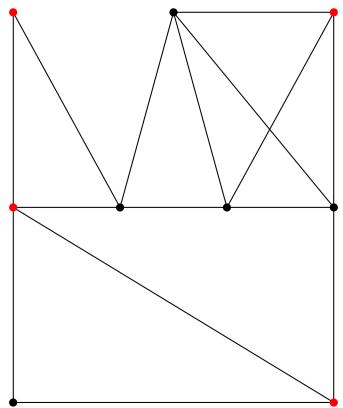
Abbildung 5.5: Testinstanz 5 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 8027. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 12, \lambda_{Option} = \lambda_NULL, VZK_{Option} = \text{KONFLIKTE}$

5.3 Testinstanzen ohne optimale Lösung

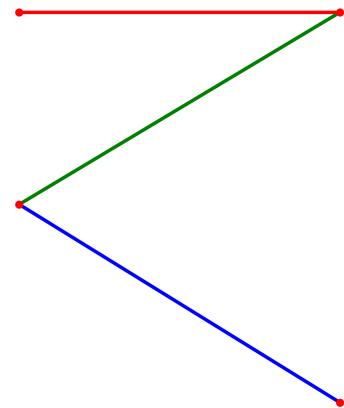
Das Branch and Bound Verfahren wird ebenfalls an Testinstanzen getestet, die keine optimale Lösung besitzen. Testinstanz 6 besitzt einen Basis-Graphen mit 9 Knoten und 15 Kanten und wird in Abbildung 5.6 dargestellt. Es sollen insgesamt 3 Demands als Wege zwischen den Terminals eingebettet werden. Die Nichtlösbarkeit von Testinstanz 6 wurde in Kapitel 2.2 beschrieben.

Testinstanz 7 besitzt ebenfalls keine optimale Lösung. Der Basisgraph entspricht dem Basis-Graphen für Testinstanz 4 und der Demandgraph entspricht dem Demand-Graph für Testinstanz 3. Demand-Graph und Basis-Graph von Testinstanz 7 werden in Abbildung 5.7 dargestellt.

Die Nichtlösbarkeit für Testinstanz 6 und Testinstanz 7 war gegeben und wurde vom Branch and Bound Verfahren richtig erkannt.

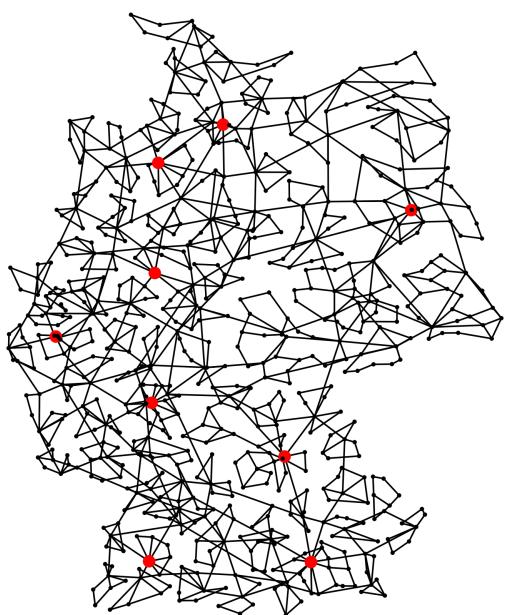


(a) Basisgraph

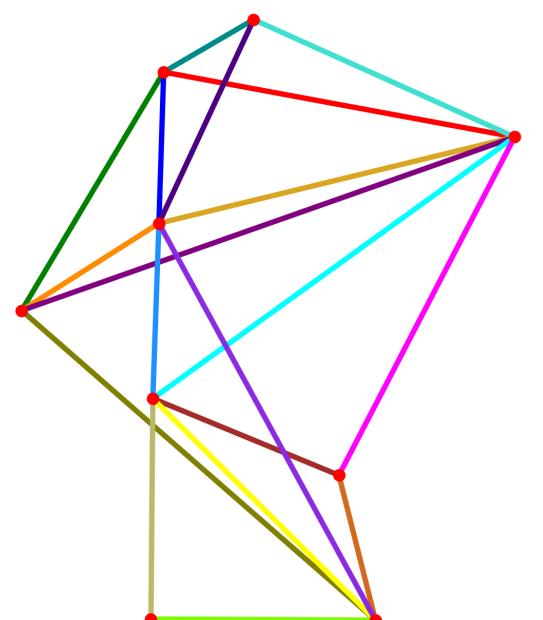


(b) Demand-Graph

Abbildung 5.6: Testinstanz 6 mit Basis-Graph, Demand-Graph und keiner optimalen Einbettung



(a) Basisgraph



(b) Demand-Graph

Abbildung 5.7: Testinstanz 7 mit Basis-Graph, Demand-Graph und keiner optimalen Einbettung

5.4 Ergebnisse aus Testinstanzen

Die Laufzeitmessungen werden auf einem Rechner mit folgenden Prozessor-Eigenschaften durchgeführt:

Prozessor Name	AMD Ryzen 5 2600 Six Core Prozessor
Taktfrequenz	3400 MHz
Anzahl Kerne	6
Anzahl logische Prozessoren	12
RAM-Größe	16 GB
Freie RAM-Größe	7,7 GB

Tabelle 5.1: Prozessor-Eigenschaften für den Rechner, welcher für Laufzeitmessungen verwendet wird

Für eine Zeitmessung werden 10 Testläufe mit den gegebenen Parametern durchgeführt und die Zeiten gemittelt. Bei Testinstanz 5 wurde für eine Zeitmessung 1 Testlauf durchgeführt, weil der Zeitaufwand des Branch and Bound Verfahrens für diese Testinstanz sehr groß ist. Während jedes Testlaufes für Testinstanz 1,2,3,4,6 und 7 konnte immer die gegebenen Gesamtkosten einer optimalen Lösung oder die fehlende Existenz einer optimalen Lösung korrekt bestimmt werden. In diesem Kapitel werden unterschiedliche Verzweigungs-Strategien miteinander verglichen und der Einfluss von n_{sub} auf die Geschwindigkeit des Branch and Bound Verfahrens untersucht.

Aus der Kombination von λ_{Option} und VZK_{Option} ergeben sich 9 unterschiedliche Strategien. Werden alle anderen Parameter konstant gehalten, können die unterschiedlichen Strategien verglichen werden. Testinstanz 1,2 und 6 sind kleine Testinstanzen. Sie dienen illustrativen Zwecken und zur Überprüfung der Korrektheit der Ergebnisse des Solvers. Die Zeiten zum Lösen dieser Testinstanzen sind kleiner als 0,001 Sekunden und sind deswegen zu kurz für sinnvolle Konkurrenztests.

In Tabelle 5.2 sind die Gesamtkosten einer optimalen Lösung und der Zeitaufwand für das Branch and Bound Verfahren für alle 9 Strategien für Testinstanz 3 bei $n_{sub} = 200$, $n'_{sub} = 40$, $n_{iter_half} = 9$ dargestellt. Alle 9 Strategien führen zu den gleichen Gesamtkosten einer optimalen Lösung. Der Zeitaufwand variiert je nach Strategie, aber ist nie größer als eine Minute. Den kürzesten Zeitaufwand zum Finden einer optimalen Lösung erhält man in 5,6 Sekunden mit der Kombination aus $\lambda_{Option} = \lambda_NULL$ und $VZK_{Option} = WEG$. Den längsten Zeitaufwand zum Finden einer optimalen Lösung erhält man in 54,5 Sekunden mit der Kombination aus $\lambda_{Option} = \lambda_MAX$ und $VZK_{Option} = KONFLIKTE$.

λ_{Option}	VZK_{Option}	Gesamtkosten einer optimalen Lösung	Zeit in s
λ_NULL	MAX_FLUSS	8462	21,4
λ_NULL	WEG	8462	5,6
λ_NULL	KONFLIKTE	8462	45,5
λ_MAX	MAX_FLUSS	8462	34,1
λ_MAX	WEG	8462	10,2
λ_MAX	KONFLIKTE	8462	54,5
$\lambda_KONFLIKT$	MAX_FLUSS	8462	26,3
$\lambda_KONFLIKT$	WEG	8462	11,3
$\lambda_KONFLIKT$	KONFLIKTE	8462	27,6

Tabelle 5.2: Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 3 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 9$

In Tabelle 5.3 sind die Gesamtkosten einer optimalen Lösung und der Zeitaufwand für das Branch and Bound Verfahren für alle 9 Strategien für Testinstanz 4 bei $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 3$ dargestellt. Alle 9 Strategien führen zu den gleichen Gesamtkosten einer optimalen Lösung. Der Zeitaufwand variiert je nach Strategie, aber ist nie größer als 10 Sekunden. Die kürzeste Laufzeit beträgt 0,9 Sekunde und wird mit den Parametern $\lambda_{Option} = \lambda_KONFLIKT$ und $VZK_{Option} = WEG$ ermittelt. Die längste Laufzeit beträgt 6,7 Sekunden und wird mit den Parametern $\lambda_{Option} = \lambda_NULL$ und $VZK_{Option} = KONFLIKTE$ ermittelt.

λ_{Option}	VZK_{Option}	Gesamtkosten einer optimalen Lösung	Zeit in s
λ_NULL	MAX_FLUSS	7442	1,1
λ_NULL	WEG	7442	2,9
λ_NULL	KONFLIKTE	7442	6,7
λ_MAX	MAX_FLUSS	7442	2,0
λ_MAX	WEG	7442	3,0
λ_MAX	KONFLIKTE	7442	3,1
$\lambda_KONFLIKT$	MAX_FLUSS	7442	5,3
$\lambda_KONFLIKT$	WEG	7442	0,9
$\lambda_KONFLIKT$	KONFLIKTE	7442	1,9

Tabelle 5.3: Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 4 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 3$

Aus Tabelle 5.2 und 5.3 ist erkennbar, dass $VZK_{Option} = KONFLIKTE$ häufig zu längeren Laufzeiten führt. Bei $VZK_{Option} = KONFLIKTE$ finden mehrere Subgradientenverfahren mit n'_{sub} Iterationen statt. Dies kann längere Laufzeiten verursachen. Trotzdem können abhängig von der Instanz mit Hilfe von λ_{Option} die Laufzeiten stark variieren wie beispielsweise bei $\lambda_{Option} = \lambda_NULL$ und $\lambda_{Option} = \lambda_KONFLIKT$ bei $VZK_{Option} = KONFLIKTE$ (siehe Tabelle 5.2 und 5.3).

In Tabelle 5.4 sind die Gesamtkosten einer optimalen Lösung und der Zeitaufwand für das Branch and Bound Verfahren für alle 9 Strategien für Testinstanz 7 bei $n_{sub} = 10000, n'_{sub} = 200, n_{iter_half} = 57$ dargestellt. Der Zeitaufwand ist bei allen Strategien gleich. Für Testinstanz 7 existiert keine optimale Lösung. Die unteren Schranken solcher Testinstanzen können unendlich groß werden. Durch Ausprobieren unterschiedlicher Werte für n_{iter_half} und n_{sub} konnte für die Gesamtinstanz eine untere Schranke bestimmt werden, die die globale obere Schranke O (siehe Gleichung 4.10) überschritten hat. Das Branch and Bound Verfahren endete nach der Gesamtinstanz, wodurch keine Verzweigung stattfand und alle 9 Strategien denselben Zeitaufwand benötigen. Durch geeignete Wahl von n_{sub} und n_{iter_half} können deswegen Iterationen des Branch and Bound Verfahrens eingespart werden.

λ_{Option}	VZK_{Option}	Gesamtkosten einer optimalen Lösung	Zeit in s
λ_NULL	MAX_FLUSS	-	7,9
λ_NULL	WEG	-	7,9
λ_NULL	KONFLIKTE	-	7,9
λ_MAX	MAX_FLUSS	-	7,9
λ_MAX	WEG	-	7,9
λ_MAX	KONFLIKTE	-	7,9
$\lambda_KONFLIKT$	MAX_FLUSS	-	7,9
$\lambda_KONFLIKT$	WEG	-	7,9
$\lambda_KONFLIKT$	KONFLIKTE	-	7,9

Tabelle 5.4: Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 7 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 10000, n'_{sub} = 200, n_{iter_half} = 57$

Testinstanz 5 ist eine sehr große Testinstanz. Das Branch and Bound Verfahren konnte nicht für alle Strategien beendet werden und wurde nach 24 Stunden abgebrochen (siehe Tabelle 5.5). Bis auf die Strategie mit $\lambda_{Option} = \lambda_NULL$ und $VZK_{Option} = \text{WEG}$ konnte bei jeder Strategie mindestens eine zulässige Lösung bestimmt werden. Bei 4 der 9 Strategien konnte das Branch and Bound Verfahren beendet werden. Diese 4 Strategien führen zu unterschiedlichen Werten der Gesamtkosten einer optimalen Lösung und damit zum Widerspruch der Optimalität dieser Lösungen. Für die Strategie mit $\lambda_{Option} = \lambda\text{NULL}$ und $VZK_{Option} = \text{KONFLIKTE}$ wurde die zulässige Lösung mit den kleinsten Gesamtkosten für alle 9 Strategien mit dem Wert 8027 gefunden. Diese zulässige Lösung wird in Abbildung 5.5 dargestellt. Unterschiedliche Strategien bei dieser Testinstanz führen zu unterschiedlichen zulässigen Lösungen, jedoch nicht notwendigerweise zu einer optimalen Lösung der Testinstanz. Die Subinstanz S^* die durch Verzweigung zu einer optimalen Lösung führt, geht demzufolge im Branch and Bound Verfahren verloren. 2 mögliche Gründe sind:

1. Wenn das Branch and Bound Verfahren endet, existiert keine Garantie eine optimale Lösung gefunden zu haben
2. Numerische Fehler treten bei der Berechnung auf

Der erste Grund widerspricht dem theoretischen Ablauf des Branch and Bound Verfahrens zum Finden einer optimalen Lösung in Kapitel 4.7. Nach Gleichung 4.16 kann W nur leer werden, wenn eine optimale Lösung gefunden wird. Das Finden einer optimalen Lösung ist nach Gleichung 4.16 theoretisch sichergestellt. Weiterhin wurden bei den anderen Testinstanzen unterschiedliche Variationen der Parameter des Branch and Bound Verfahrens getestet, ohne das sich die Gesamtkosten der gefundenen optimalen Lösungen geändert haben. Dieser Grund wird ausgeschlossen.

Der zweite Grund kann ebenfalls bei Sovern für ganzzahlige lineare Optimierungsprobleme auftreten. Bei sehr großen Testinstanzen wie Testinstanz 5 können numerische Fehler wie beispielsweise Rundungsfehler auftreten. Diese können dazu führen, dass die untere Schranke der Subinstanz S^* einen größeren Wert besitzt, als eine nicht optimale zulässige Lösung X' . Beim Finden der zulässigen Lösung X' wird S^* fälschlicherweise aus W wegen Suboptimalität entfernt. Somit werden je nach Parameterwahl andere zulässige Lösungen fälschlicherweise als eine optimale Lösung deklariert. Dieser Grund erscheint plausibel, weil bei den 4 Strategien, bei denen das Branch and Bound Verfahren ohne Abbruch endete, die Gesamtkosten der gefundenen zulässigen Lösungen nicht stark voneinander abweichen.

Ein Programmierfehler bei der Implementierung des Verfahrens muss grundsätzlich bedacht werden, erscheint jedoch auf Grund der Ergebnisse für die Testinstanzen 1,2,3,4,6,7 unwahrscheinlich.

λ_{Option}	VZK_{Option}	Gesamtkosten einer optimalen Lösung	Zeit in s
λ_{NULL}	MAX_FLUSS	$7900 \leq z(X^*) \leq 8092$	86400
λ_{NULL}	WEG	$7847 \leq z(X^*) \leq 86781$	86400
λ_{NULL}	KONFLIKTE	$7903 \leq z(X^*) \leq 8027$	86400
λ_{MAX}	MAX_FLUSS	8035	7162,3
λ_{MAX}	WEG	$7929 \leq z(X^*) \leq 8056$	86400
λ_{MAX}	KONFLIKTE	8037	6928,4
$\lambda_{KONFLIKT}$	MAX_FLUSS	8032	17954
$\lambda_{KONFLIKT}$	WEG	$7904 \leq z(X^*) \leq 8047$	86400
$\lambda_{KONFLIKT}$	KONFLIKTE	8029	53775,7

Tabelle 5.5: Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 5 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200$, $n'_{sub} = 40$, $n_{iter_half} = 12$

Der Einfluss der Anzahl der Iterationen des Branch and Bound Verfahrens n_{sub} auf die Laufzeit des Branch and Bound Verfahrens wird veranschaulicht in Abbildung 5.8 für Testinstanz 3 mit $n'_{sub} = 20$, $n_{iter_half} = 9$ und in Abbildung 5.9 für Testinstanz 4 mit $n'_{sub} = 20$, $n_{iter_half} = 3$. Für alle 9 Strategien entsteht bei Testinstanz 4 ein ähnlicher Verlauf. Je größer n_{sub} ist, desto größer ist die Dauer des Branch and Bound Verfahrens. Die Ursache dafür könnte durch die Bestimmung der unteren Schranken erklärt werden. Bei der Bestimmung der unteren Schranken mit Hilfe der Lagrange-Relaxation existiert ein Maximum, welches nicht mit den Gesamtkosten einer optimalen Lösung der Instanz des MCDPP übereinstimmen muss. Bei großen n_{sub} kann die Laufzeit des Branch and Bound Verfahrens zunehmen, weil beispielsweise das Maximum der Lagrange-Relaxation früh gefunden wurde, aber trotzdem weitere Iterationen stattfinden, die zur Erhöhung der Laufzeit des Branch and

Bound Verfahrens führen.

Für die mehr als die Hälfte der 9 Strategien entsteht bei Testinstanz 3 ein ähnlicher Verlauf. Je größer n_{sub} ist, desto kleiner ist die Dauer des Branch and Bound Verfahrens. Je größer n_{sub} ist, desto mehr untere Schranken können im Subgradientenverfahren gebildet werden. Dies erhöht die Chance mehr zulässige Lösungen und höhere untere Schranken der Subinstanzen zu generieren. Das kann zu kürzeren Laufzeiten des Branch and Bound Verfahrens führen.

Welche Strategie bei einem bestimmten Wert von n_{sub} eine kleinere Laufzeit benötigt, hängt von der Instanz ab.

n_{iter_half} ist ein Parameter einer Schrittweite die heuristisch aus dem Newton-Verfahren hergeleitet wurde. Der Einfluss von n_{iter_half} auf die Laufzeit kann bei unterschiedlichen Instanzen unterschiedlich groß sein und auf Grund der heuristischen Herleitung nicht ausreichend begründet werden. Aus diesem Grund und aus zeitlichen Gründen wird keine Analyse der Laufzeit in Abhängigkeit von n_{iter_half} dargestellt.

Es ist zu beachten, dass bei $\lambda_{Option} = \lambda_{KONFLIKT}$ der Parameter λ_{Start} auf Basis der Anzahl der Konfliktknoten gewählt wird. Es ist demzufolge möglich durch bestimmte Wahl von Parametern die gefundenen unteren Schranken der Subinstanzen negativ werden zu lassen, wodurch die Beschränkung durch Suboptimalität abnimmt. Dies kann zu einem großen Zeitaufwand führen.

Ist $\lambda_{Option} = \lambda_{MAX}$ können die unteren Schranken der verzweigten Subinstanzen nicht sinken, wegen Gleichung 4.8. Demzufolge ist das beschriebene Szenario für $\lambda_{Option} = \lambda_{KONFLIKT}$ hier nicht möglich.

Ist $\lambda_{Option} = \lambda_{NULL}$, können die unteren Schranken der verzweigten Subinstanzen S' nicht unter den Wert $U_S(\lambda = 0)$ der zu verzweigenden Subinstanz S fallen. Dies entspricht den kürzesten Wegen für die Einbettung der Demands ohne Anpassung der Kantenkosten mit λ . Dieser Wert ist immer positiv. Demzufolge ist das beschriebene Szenario für $\lambda_{Option} = \lambda_{KONFLIKT}$ hier ebenfalls nicht möglich.

Der Parameter n'_{sub} ist von Bedeutung, wenn $VZK_{Option} = KONFLIKTE$ ist. n'_{sub} verringert die Anzahl von Iterationen für das Subgradientenverfahren beim Suchen des Verzweigungsknotens. Der Einfluss von n'_{sub} auf die Laufzeit des Branch and Bound Verfahrens wird veranschaulicht in Tabelle 5.6 für Testinstanz 3 mit $n_{sub} = 200, n_{iter_half} = 9, VZK_{Option} = KONFLIKTE$ und in Tabelle 5.7 für Testinstanz 4 mit $n_{sub} = 200, n_{iter_half} = 3, VZK_{Option} = KONFLIKTE$. Ist n'_{sub} klein, ist der Zeitaufwand für das Finden eines VZK klein. Das kann zu einer Verkürzung der Laufzeit führen. Dies ist in Tabelle 5.6 für $\lambda_{Option} = \lambda_{MAX}$ und $\lambda_{Option} = \lambda_{KONFLIKT}$ erkennbar. Ist n'_{sub} groß, sind die Iterationen für das Subgradientenverfahren für das Finden eines VZK größer und nähern sich n_{sub} . Es wird im Vorfeld für eine Menge von Knoten eine Verzweigung durchgeführt und der Knoten gewählt, der den größten Wert der kleinsten unteren Schranke nach der Verzweigung generiert. Das kann die globale untere Schranke von W erhöhen und kann zu kürzeren Laufzeiten führen. Dies ist in Tabelle 5.7 für $\lambda_{Option} = \lambda_{NULL}$ und $\lambda_{Option} = \lambda_{MAX}$ erkennbar.

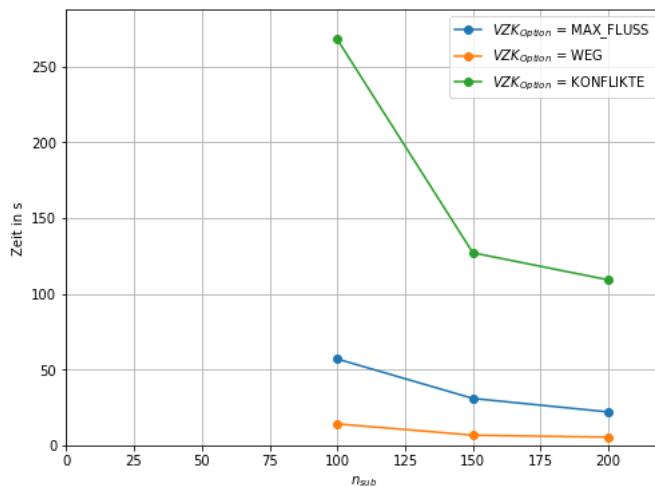
λ_{Option}	Zeit in s bei $n'_{sub} = 20$	Zeit in s bei $n'_{sub} = 40$
λ_{NULL}	109,3	45,5
λ_{MAX}	41,7	54,5
$\lambda_{KONFLIKT}$	16,1	27,6

Tabelle 5.6: Zeitaufwand für Testinstanz 3 mit unterschiedlichen Strategien. Parameter:
 $n_{sub} = 200, n_{iter_half} = 9, VZK_{Option} = \text{KONFLIKTE}$

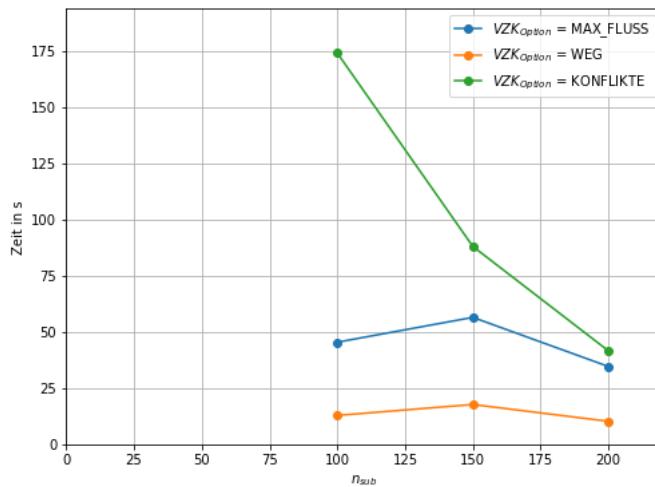
λ_{Option}	Zeit in s bei $n'_{sub} = 20$	Zeit in s bei $n'_{sub} = 40$
λ_{NULL}	8,6	6,7
λ_{MAX}	4,4	3,1
$\lambda_{KONFLIKT}$	1,3	1,9

Tabelle 5.7: Zeitaufwand für Testinstanz 4 mit unterschiedlichen Strategien. Parameter:
 $n_{sub} = 200, n_{iter_half} = 3, VZK_{Option} = \text{KONFLIKTE}$

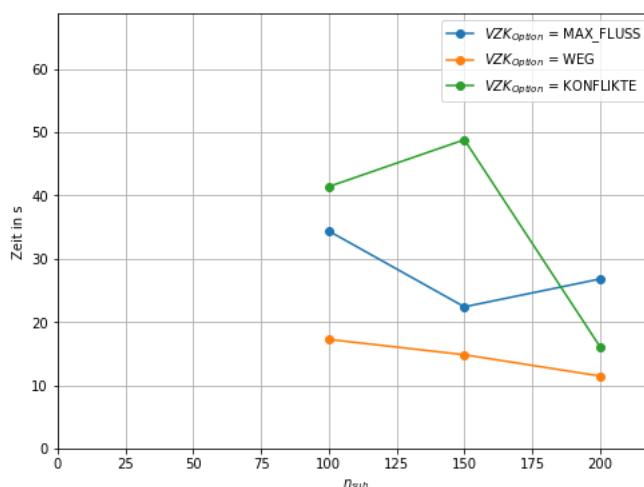
Unter den 9 Strategien ein Ranking zu erstellen, ist schwierig wegen der begrenzten Anzahl an Testinstanzen und starker Variation der Laufzeiten je nach Testinstanz. Die 4 Kombinationen aus $\lambda_{Option} = \lambda_{MAX}$ und $\lambda_{Option} = \lambda_{KONFLIKT}$ mit $VZK_{Option} = \text{KONFLIKTE}$ und $VZK_{Option} = \text{MAX_FLUSS}$ führten bei jeder Testinstanz innerhalb von 24 Stunden zu einem Ende des Branch and Bound Verfahrens ohne Abbruch des Verfahrens. $\lambda_{Option} = \lambda_{KONFLIKT}$ kann empfindlich gegenüber den anderen Parametern sein, wegen der Berechnung von λ_{Start} aus der minimalen Anzahl von Konfliktknoten während des Subgradientenverfahrens. $\lambda_{Option} = \lambda_{MAX}$ ist wegen Gleichung 4.8 gegenüber den anderen Parametern weniger empfindlich als $\lambda_{Option} = \lambda_{KONFLIKT}$. $VZK_{Option} = \text{KONFLIKTE}$ kann wegen n'_{sub} zu langen Laufzeiten führen. Auf Grund dieser Argumente wird die Verzweigungsstrategie $\lambda_{Option} = \lambda_{MAX}$ und $VZK_{Option} = \text{MAX_FLUSS}$ als stabilster und damit favorisierter Solver bestimmt. Diese Verzweigungsstrategie ist deswegen im Default-Solver des Branch and Bound Verfahrens implementiert.



(a) $\lambda_{Option} = \lambda_NULL$

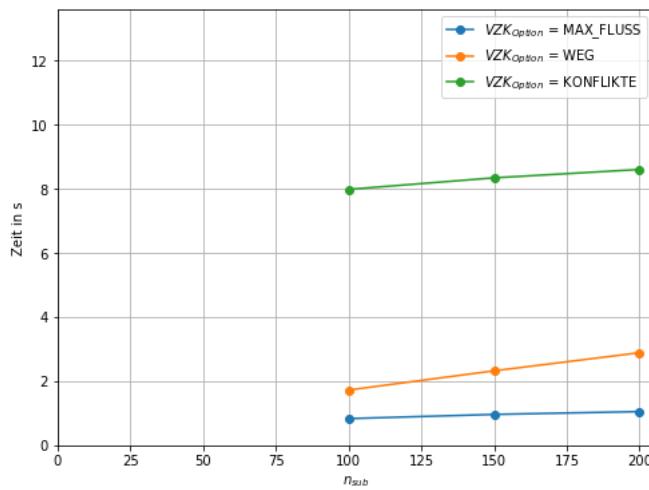


(b) $\lambda_{Option} = \lambda_MAX$

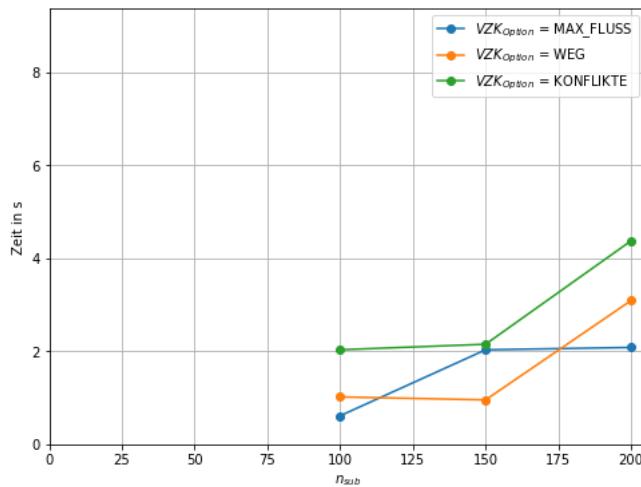


(c) $\lambda_{Option} = \lambda_KONFLIKT$

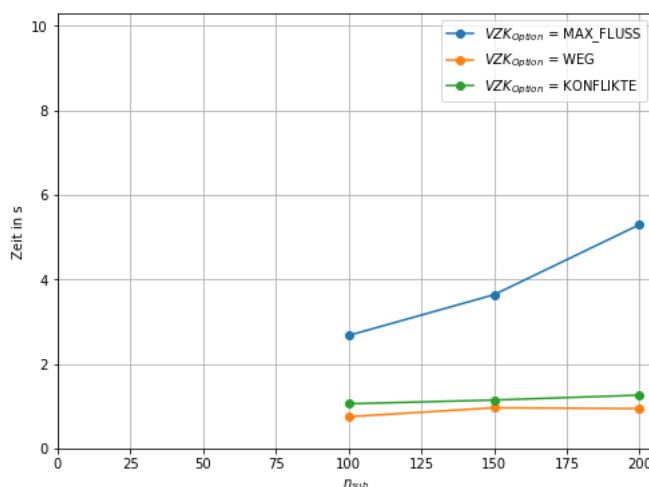
Abbildung 5.8: Laufzeit des Branch and Bound Verfahrens in Abhängigkeit von n_{sub} für Testinstanz 3. Parameter: $n'_{sub} = 20, n_{iter_half} = 9$



(a) $\lambda_{Option} = \lambda_NULL$



(b) $\lambda_{Option} = \lambda_MAX$



(c) $\lambda_{Option} = \lambda_KONFLIKT$

Abbildung 5.9: Laufzeit des Branch and Bound Verfahrens in Abhängigkeit von n_{sub} für Testinstanz 4. Parameter: $n'_{sub} = 20, n_{iter_half} = 3$

5.5 Verbesserungen des Branch and Bound Verfahrens

Im Branch and Bound Verfahren wird sehr oft das Subgradientenverfahren ausgeführt. Es wird deswegen oft die Lagrange-Relaxation zur Bestimmung unterer Schranken verwendet. Der Algorithmus für das Finden kürzester Wege wird deswegen oft aufgerufen. In diesem Branch and Bound Verfahren wird für den Algorithmus zum Finden kürzester Wege der Dijkstra Algorithmus verwendet. Der Zeitaufwand für diesen Algorithmus ist abhängig von der verwendeten Datenstruktur der Warteschlange des Dijkstra-Algorithmus. Der Aufwand für den Dijkstra-Algorithmus ist $O(|V|pop() + |E|push)$ für einen Basis-Graphen $G = (V, E)$ und stellt den Aufwand zur Einbettung eines Demands als kürzesten Weg dar. In dieser Implementierung wird eine Liste gewählt. Die Methode pop entfernt das erste Element in der Liste und hat einen konstanten Zeitaufwand $O(1)$. Bei der Methode push wird durch die gesamte Liste mit Länge n iteriert, um das Element einzufügen. Die maximale Größe der Liste ist $n = |V|$. Der Aufwand für die push-Methode beträgt $O(|V|)$. Der hier implementierte Dijkstra-Algorithmus hat einen Aufwand von $O(|V| + |A| \cdot |V|)$.

Wird ein Fibonacci Heap als Datenstruktur der Warteschlange gewählt, ist der Aufwand der Methode pop $O(\log|V|)$ und der Aufwand der Methode push $O(1)$. Es ergibt sich ein Aufwand von $O(|V|\log|V| + |A|)$. Ein Fibonacci Heap ist nach Quelle [1] die theoretisch beste bekannte Datenstruktur für die Warteschlange des Dijkstra-Algorithmus. Die Implementierung dieser Datenstruktur für den Dijkstra-Algorithmus könnte die Geschwindigkeit des Branch and Bound Verfahrens erhöhen. Jedoch verstecken sich hinter der O -Notation Konstanten, weshalb auf großen Testinstanzen die Überlegenheit dieser Datenstruktur nicht garantiert werden kann.

6 Zusammenfassung

In dieser Bachelorarbeit wurde ein Branch and Bound Verfahren zum Lösen von Instanzen des MCDPP entwickelt und in C++ implementiert. Für 6 der 7 Testinstanzen konnte eine optimale Lösung bestimmt werden oder die Nicht-Existenz einer optimalen Lösung nachgewiesen werden.

Der Zeitaufwand für das Branch and Bound Verfahren hing von der Wahl der Verfahrensparameter und der Testinstanz ab. Mit geeigneten Parametern war der Zeitaufwand für die Testinstanzen 1,2,3,4,6 und 7 immer kleiner als eine Minute. Für Testinstanz 5 war es möglich zulässige Lösungen mit kleinen Gesamtkosten zu bestimmen. Durch numerische Fehler bei großen Testinstanzen kann nicht garantiert werden, dass eine optimale Lösung gefunden wird. Deswegen kann bei großen Instanzen dieses Branch and Bound Verfahren zu unterschiedlichen Lösungen bei unterschiedlicher Wahl von Verfahrensparametern führen. Das ist zu beachten, wenn dieses Verfahren für große Testinstanzen angewendet wird.

Um numerische Fehler wie Rundungsfehler gering zu halten, können Klassen für rationale Zahlen erstellt werden, um die Genauigkeit der Berechnungen zu erhöhen. Jedoch sind die Zahlenbereiche, die solche Klassen abdecken ebenfalls begrenzt, weshalb numerische Fehler nicht ausgeschlossen werden können. Bei sehr großen Testinstanzen können numerische Fehler nicht verhindert werden.

Es wird gezeigt, dass unterschiedliche Verzweigungsstrategien in diesem Verfahren eingebaut werden können. Diese können je nach Instanz schneller zum Finden einer Lösung mit dem Branch and Bound Verfahrens führen. Für Testinstanz 5 konnte beispielsweise nur für die Kombination aus $\lambda_{Option} = \lambda_{MAX}$ und $\lambda_{Option} = \lambda_{KONFLIKT}$ mit $VZK_{Option} = MAX_FLUSS$ und $VZK_{Option} = KONFLIKTE$ eine zulässige Lösung innerhalb von 24 Stunden gefunden werden und das Branch and Bound Verfahren ohne Abbruch durchgeführt werden. Für Testinstanz 3 führten diese Kombinationen zu längeren Laufzeiten in Vergleich zu den anderen Verzweigungsstrategien. Die Geschwindigkeit des Branch and Bound Verfahrens hängt von der Verzweigungsstrategie und der Instanz ab.

Eine Aussage über die Geschwindigkeit unterschiedlicher Solver unter vergleichbaren Bedingungen wird hier über die unterschiedlichen Verzweigungsstrategien realisiert. Aus den Testläufen für die Laufzeit mit den unterschiedlichen Testinstanzen folgt, dass die Verzweigungsstrategie mit $\lambda_{Option} = \lambda_{MAX}$ und $VZK_{Option} = MAX_FLUSS$ zu den besten Ergebnissen führt. Bei dieser Verzweigungsstrategie endete das Branch and Bound Verfahren für Testinstanz 5 ohne Abbruch im Gegensatz zu 5 anderen Strategien. Das hat gezeigt, dass der Unterschied in der Laufzeit für sehr große Testinstanzen erheblich sein kann. Der Laufzeitunterschied der unterschiedlichen Verzweigungsstrategien lag bei den Testinstanzen 1,2,3,4,6,7 im Sekundenbereich. Für weitere Untersuchungen dieses Branch and Bound Verfahrens können mehr Testinstanzen und vergleichbare Solver verwendet werden, um die Geschwindigkeit des Solvers besser bewerten zu können.

Die Wahl des Parameters n_{sub} wirkt sich ebenfalls auf die Laufzeit aus. Ist n_{sub} groß, ist die Anzahl der Iterationen im Subgradientenverfahren groß und es wird die Chance erhöht

höhere untere Schranken und zulässige Lösungen zu finden. Je nachdem welcher der beiden Effekte stärker ist, führen große n_{sub} zu kleineren oder größeren Laufzeiten des Branch and Bound Verfahrens.

Wenn die Instanz keine optimale Lösung besitzt, werden nie zulässige Lösungen gefunden und das Branch and Bound Verfahren wird weniger beschränkt. Dies kann zu langen Laufzeiten führen, weshalb es wichtig ist die Verfahrensparameter geeignet zu wählen. Wie die Parameter gewählt werden, bleibt dem Benutzer überlassen. Wenn Bedingungen oder Theoreme über die Existenz optimaler Lösungen für das MCDPP existieren, können diese verwendet werden, um unnötige Iterationen des Branch and Bound Verfahrens zu vermeiden. Dies kann Gegenstand weiterer Forschung sein.

Das Subgradientenverfahren wird häufig aufgerufen, um untere Schranken zu bestimmen. Die Implementierung der kürzesten Wege Algorithmen sollte deswegen effizient sein. Die Verwendung eines Fibonacci Heap als Datenstruktur der Warteschlange des Dijkstra Algorithmus könnte die Geschwindigkeit des Branch and Bound Verfahrens erhöhen. Ob der Wechsel dieser Datenstruktur in der Praxis ebenfalls zu besseren Laufzeiten führt, bleibt nachzuweisen.

In dieser Arbeit wurde der Unterschied zwischen theoretischem Verlauf des Branch and Bound Verfahren und praktischer Anwendung des Branch and Bound Verfahrens ersichtlich. Obwohl dieses Verfahren theoretisch eine optimale Lösung findet, ist dies bei der praktischen Anwendung dieses Verfahrens nicht mehr sichergestellt. Es ist bei der praktischen Anwendung ebenfalls nicht möglich, die Optimalität der gefundenen Lösung festzustellen, weil kein Optimalitätskriterium (bis auf den Spezialfall) für das MCDPP existiert. Schließlich und endlich kann dieses Branch and Bound Verfahren verwendet werden, um zulässige Lösungen von Instanzen des MCDPP mit kleinen Gesamtkosten zu bestimmen, sofern sie existieren. Die Optimalität dieser Lösungen ist wegen möglicher numerischer Fehler von der Instanz abhängig und kann nicht garantiert werden.

Literaturverzeichnis

- [1] OELLRICH, Prof. Dr. M.: *Operations Research Vorlesungsskript*. Berliner Hochschule für Technik, 2023
- [2] HAHN, Florian: *Schrankenverfahren für sichere Subnetze*. Berlin Hochschule für Technik, 2023
- [3] BAVINDRA K. AHUJA, James B. O. Thomas L. Magnanti M. Thomas L. Magnanti: *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993

Abbildungsverzeichnis

2.1	Testinstanz 1 dargestellt mit Basis-Graph, Demand-Graph und einer optimalen Lösung	6
2.2	Testinstanz 6 mit Basis-Graphen und Demand-Graphen	7
2.3	Schematische Darstellung des Branch and Bound Verfahrens [1]	8
3.1	Testinstanz 1 mit gesetzten Knotenzuständen	10
3.2	Mögliche Wege von 3 benachbarten Knoten mit gleichem Knotenzustand . .	11
3.3	Testinstanz 1 mit gesetzten Knotenzuständen und angepassten Demand-Graph	12
4.1	Menge aller verzweigbaren Subinstanzen W dargestellt als Prioritätswarteschlange in einem UML Diagramm	14
4.2	Testinstanz 1 mit angepasstem Demand-Graph D_S , gesetzten und ungesetzten Knotenzuständen	22
4.3	UML Diagramm der Subinstanz	23
4.4	Alle angepassten Demand-Graphen für alle verzweigten Subinstanzen nach der Verzweigung der Subinstanz aus Abbildung 4.2 am Verzweigungsknoten F	25
4.5	Darstellung der Kantenmenge K für Testinstanz 1	29
5.1	Testinstanz 1 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 560	39
5.2	Testinstanz 2 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 40	40
5.3	Testinstanz 3 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 8462	42
5.4	Testinstanz 4 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 7442	43
5.5	Testinstanz 5 mit Basis-Graph, Demand-Graph und einer optimalen Einbettung mit Gesamtkosten 8027. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 12, \lambda_{Option} = \lambda_NULL, VZK_{Option} = \text{KONFLIKTE}$	44
5.6	Testinstanz 6 mit Basis-Graph, Demand-Graph und keiner optimalen Einbettung	45
5.7	Testinstanz 7 mit Basis-Graph, Demand-Graph und keiner optimalen Einbettung	46
5.8	Laufzeit des Branch and Bound Verfahrens in Abhängigkeit von n_{sub} für Testinstanz 3. Parameter: $n'_{sub} = 20, n_{iter_half} = 9$	53
5.9	Laufzeit des Branch and Bound Verfahrens in Abhängigkeit von n_{sub} für Testinstanz 4. Parameter: $n'_{sub} = 20, n_{iter_half} = 3$	54

Tabellenverzeichnis

5.1	Prozessor-Eigenschaften für den Rechner, welcher für Laufzeitmessungen verwendet wird	47
5.2	Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 3 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 9$.	48
5.3	Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 4 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 3$.	48
5.4	Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 7 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 10000, n'_{sub} = 200, n_{iter_half} = 57$	49
5.5	Gesamtkosten einer optimalen Lösung und Zeitaufwand für Testinstanz 5 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n'_{sub} = 40, n_{iter_half} = 12$.	50
5.6	Zeitaufwand für Testinstanz 3 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n_{iter_half} = 9, VZK_{Option} = \text{KONFLIKTE}$	52
5.7	Zeitaufwand für Testinstanz 4 mit unterschiedlichen Strategien. Parameter: $n_{sub} = 200, n_{iter_half} = 3, VZK_{Option} = \text{KONFLIKTE}$	52