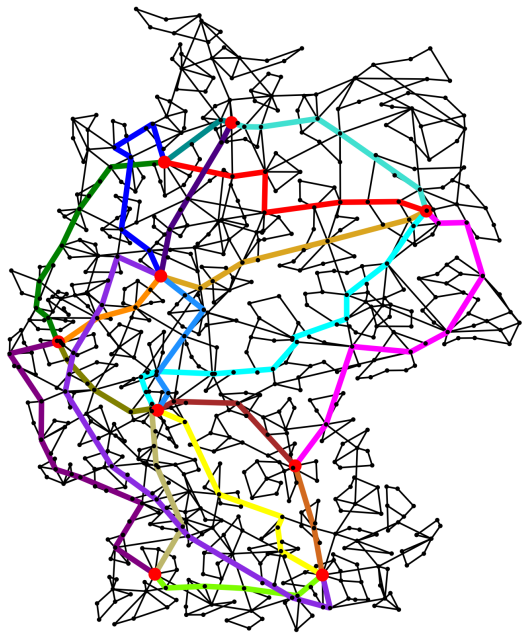
Minimale knotendiskunkte Einbettung in Graphen Entwicklung eines Branch-and-Bound Solvers

Florian Hahn

Berliner Hochschule für Technik

11.Juli 2024



Inhaltsverzeichnis

Minimum Cost Disjoint Path Problem

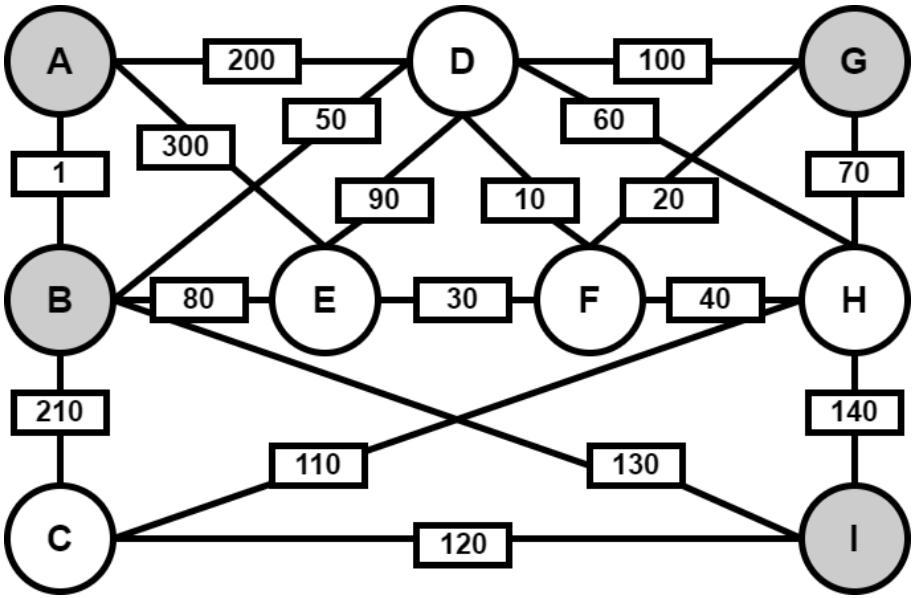
Minimum Cost Disjoint Paths Problem (MCDPP) [1]

Gegeben ist ein Basis-Graph $G=(V,E)$ mit der Kostenfunktion $c : E \to \mathbb{Q}^+$ und ein Demand-Graph $H=(T,D)$ mit $T \subseteq V$. 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:

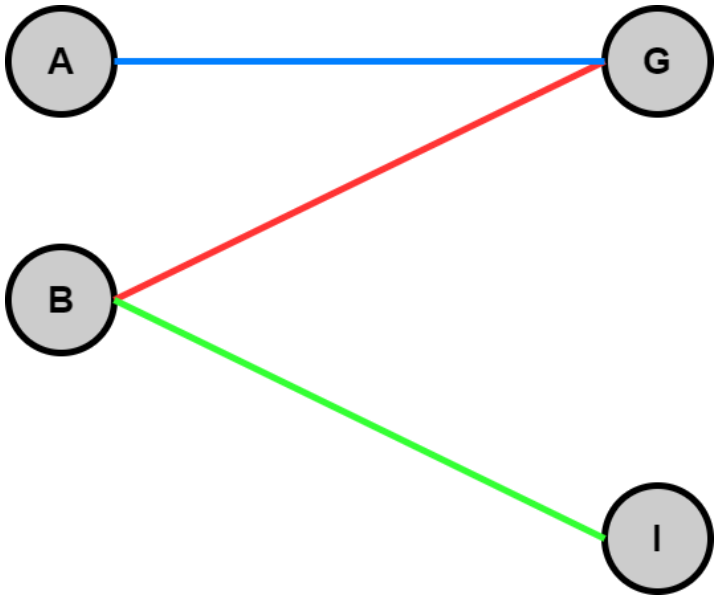
$$\min \sum_{d \in D} \sum_{e \in E^d} c(e)$$

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

Minimum Cost Disjoint Path Problem



(a) Basisgraph $G=(V,E)$



(b) Demand-Graph $H=(T,D)$

Abbildung: Testinstanz 1 dargestellt mit Basis-Graph und Demand-Graph

Minimum Cost Disjoint Path Problem

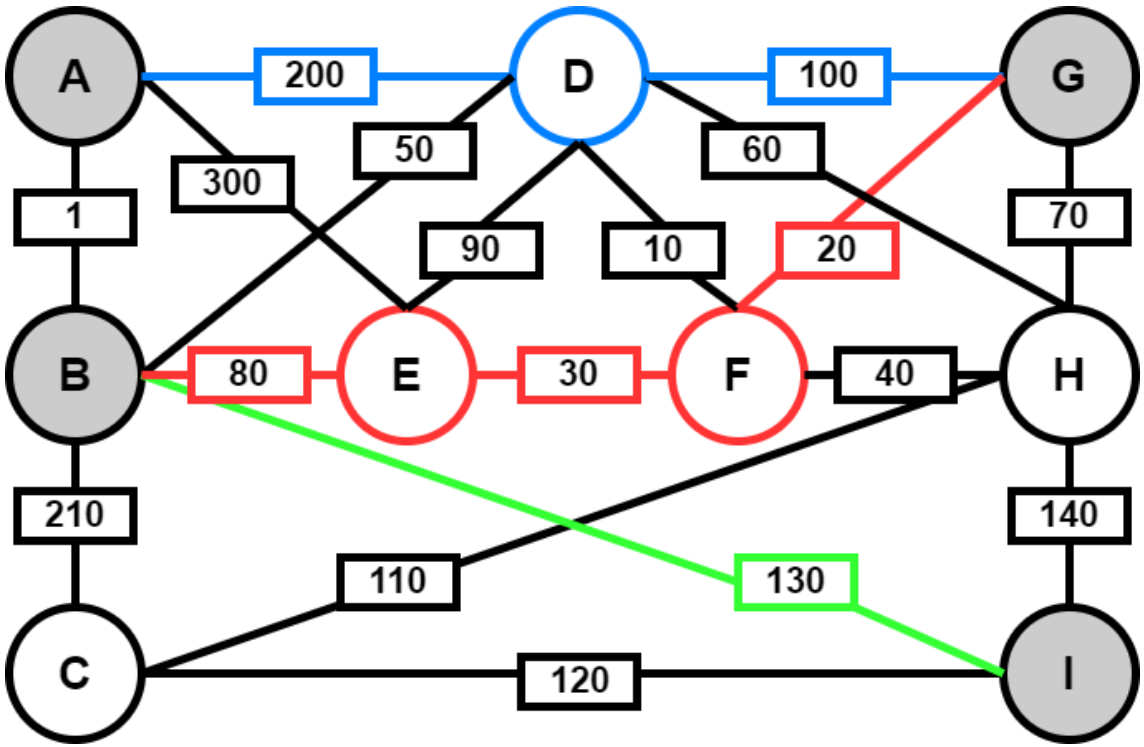
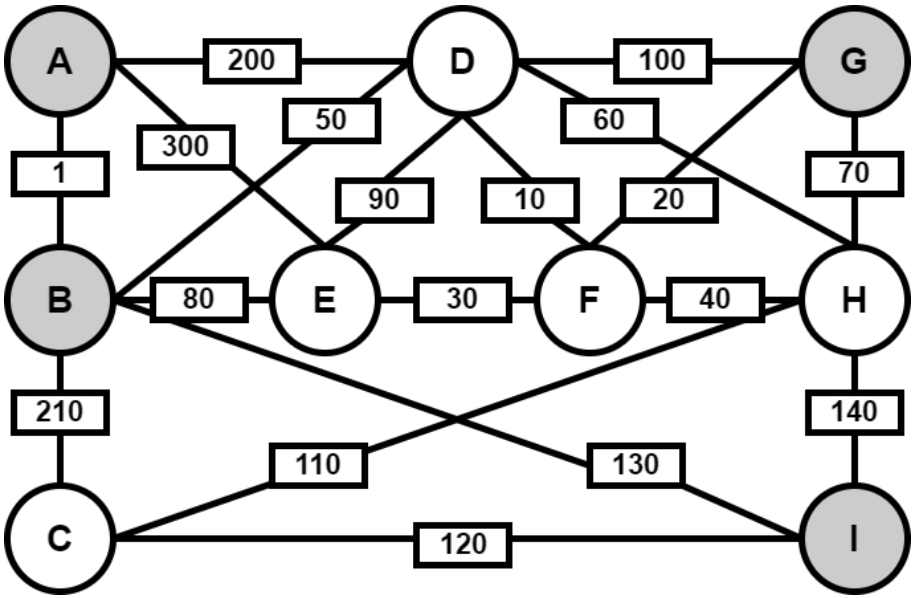


Abbildung: Eine optimale Lösung der Testinstanz 1 mit Gesamtkosten 560

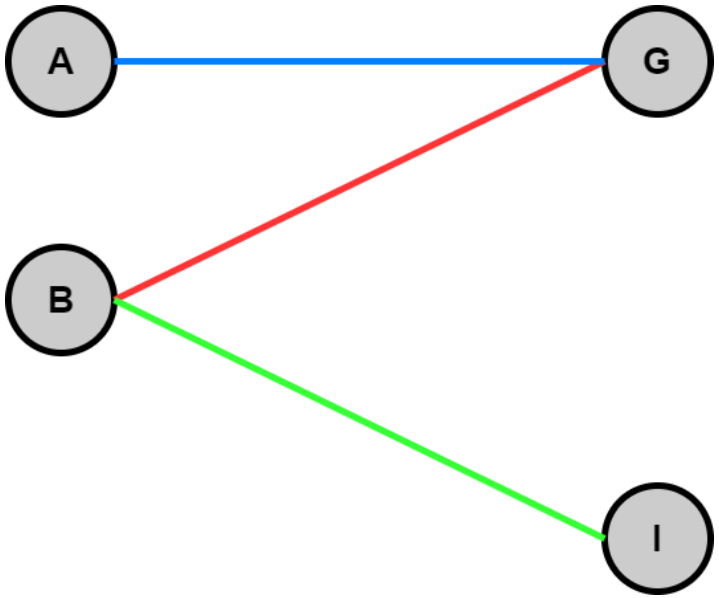
Zulässige Lösung MCDPP

- zulässige Lösung des MCDPP $\rightarrow$ Knoten entweder Teil eines Weges oder nicht
- Knotenzustand $ZK(i)$ eines Knotens $i \in V \setminus T$:
  1. $ZK(i) = 0 \Leftrightarrow$ Knoten $i \in V \setminus T$ befindet sich auf keinem eingebetteten Weg eines Demands
  2. $ZK(i) = j \ \wedge \ j \in \{1, ..., |D|\} \Leftrightarrow$ Knoten $i \in V \setminus T$ befindet sich auf eingebettetem Weg des Demands j
- zulässige Lösung = eingebettete Wege für jeden Demand = Teilgraph des Basis-Graphen
- angepasster Demand-Graph $H'(T', D')$ speichert diesen Teilgraphen durch Einfügen von Zwischenterminals im Demand-Graphen $H(T, D)$

Zulässige Lösung MCDPP



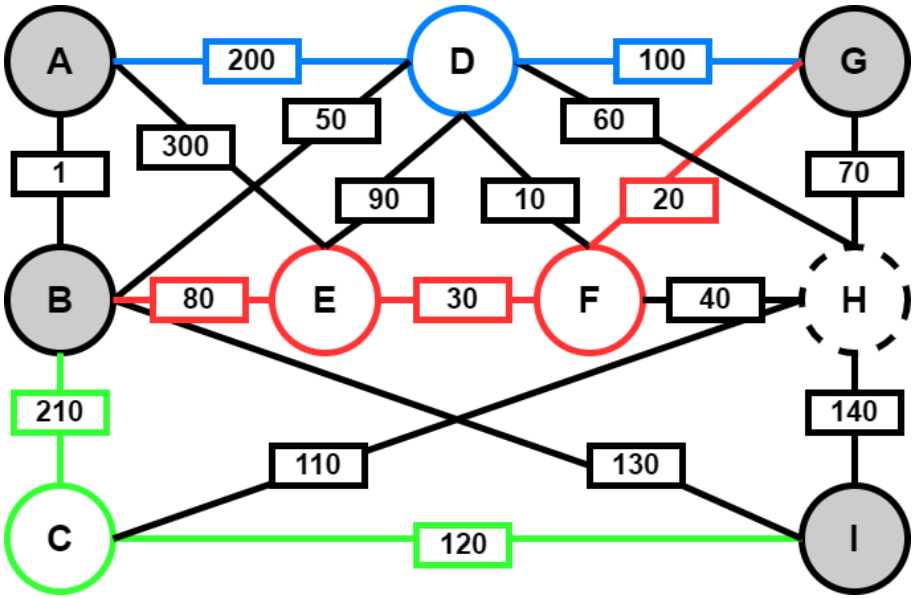
(a) Basisgraph $G=(V,E)$



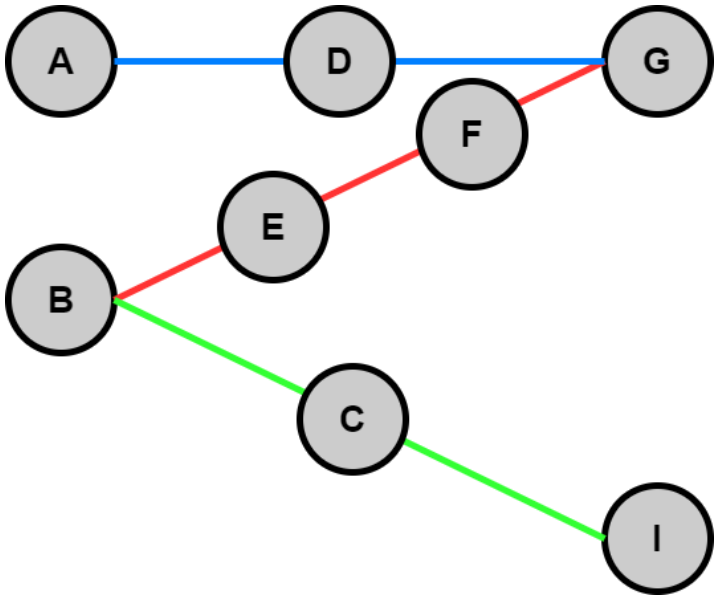
(b) Demand-Graph $H=(T,D)$

Abbildung: Testinstanz 1 dargestellt mit Basis-Graph und Demand-Graph

Zulässige Lösung MCDPP



(a) Knotenzustände und Einbettung



(b) Angepasster Demand-Graph $H'(T', D')$

Abbildung: Testinstanz 1 mit zulässiger Lösung, Knotenzuständen und angepasstem Demand-Graph $H'(T', D')$

Zulässige Lösung MCDPP

- alle Knotenzustände besetzt $\rightarrow$ zulässige oder unzulässige Lösung
- Knotenzustände im angepassten Demand-Graph gespeichert bis auf Knotenzustand 0
- Knotenzustand 0 = Löschen des Knotens im Basis-Graphen

Branch and Bound Verfahren

- Suchen eine optimale Lösung einer Instanz des MCDPP in beschränkten Suchraum
- Idee: Sukzessiv Knotenzustände setzen mit angepassten Demand-Graphen
- Aufbau:
  1. Subinstanzen
  2. Obere und untere Schranken
  3. Verzweigungsstrategie
  4. Auswahlstrategie
  5. Optimalitätskriterien
  6. Beschränkungskriterien

Branch and Bound Verfahren

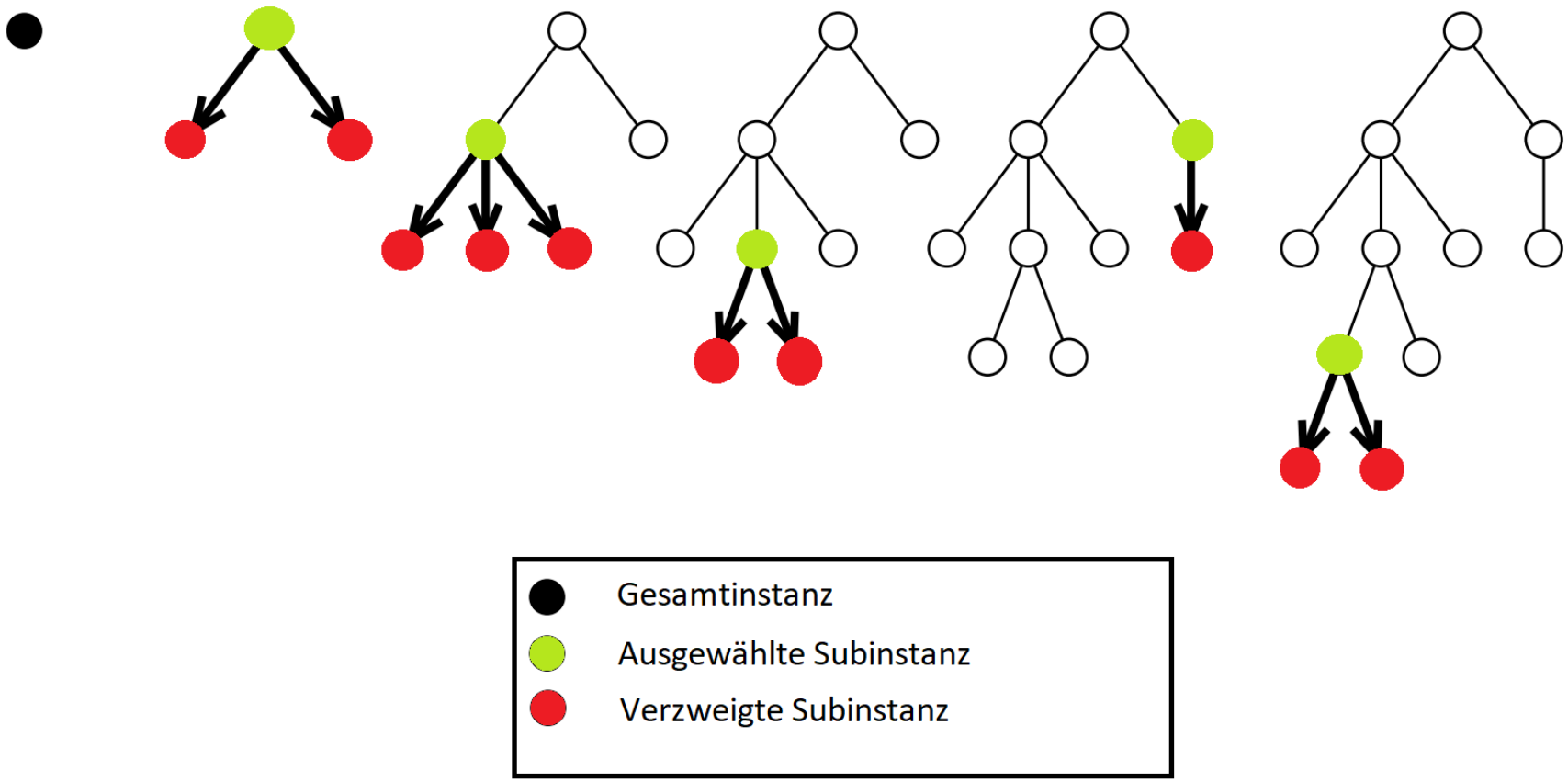
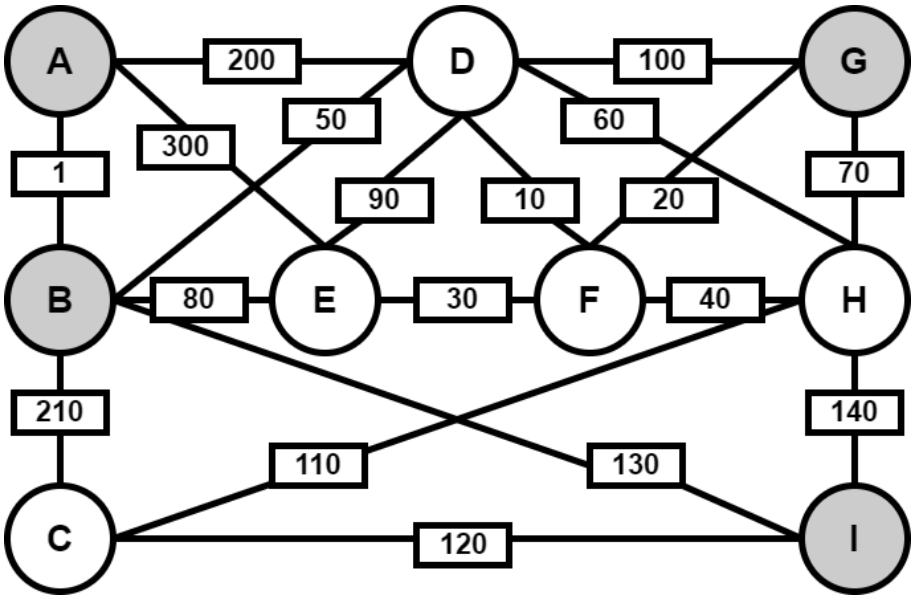
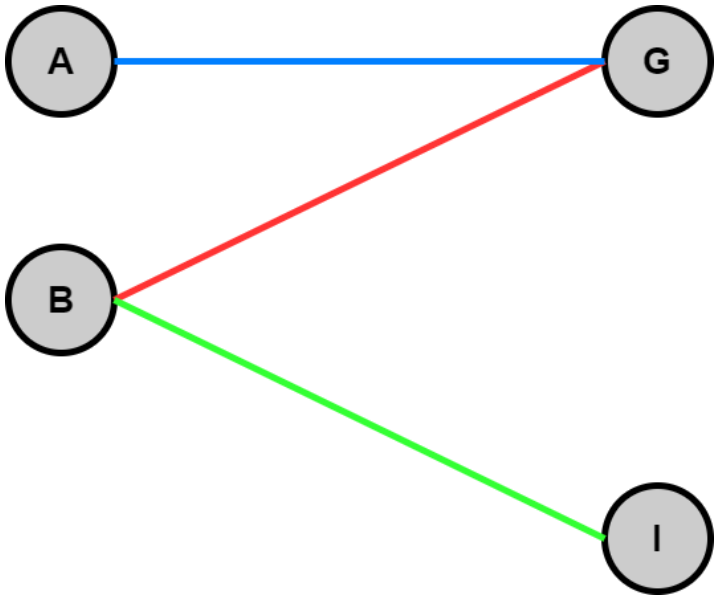


Abbildung: Schematische Darstellung des Branch and Bound Verfahren

Gesamtinstanz



(a) Basisgraph $G=(V,E)$



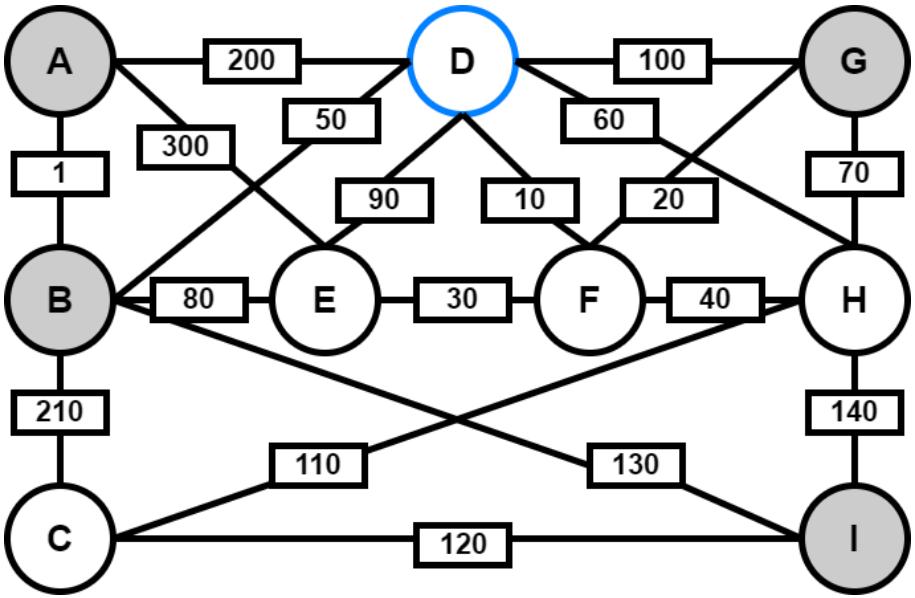
(b) Demand-Graph $H=(T,D)$

Abbildung: Testinstanz 1 dargestellt mit Basis-Graph und Demand-Graph

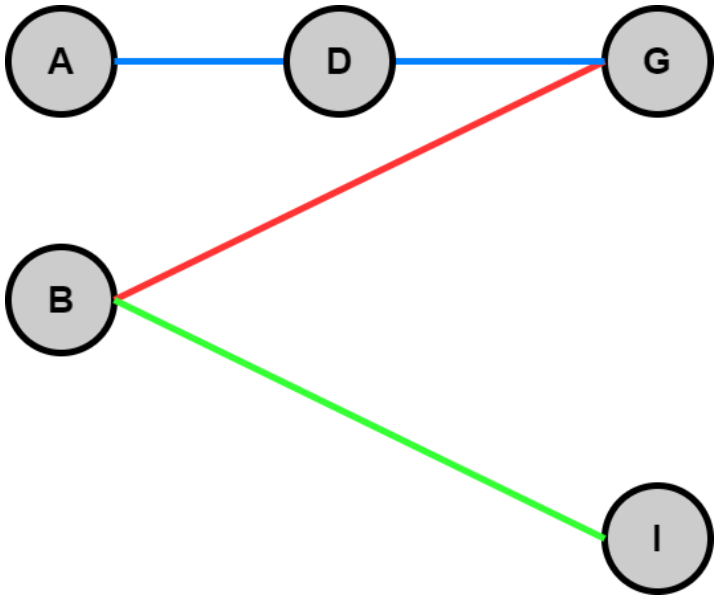
Verzweigungsstrategie

- Verzweigungsknoten VZK auswählen in Basis-Graph $G = (V, E)$
- Eigenschaften VZK:
  1. $VZK \in V \setminus T$
  2. Knotenzustand VZK ist unbesetzt
- $|D| + 1$ mögliche Knotenzustände für VZK
- $|D| + 1$ Subinstanzen, bei denen VZK als Zwischenterminal im Demand-Graphen eingefügt wird oder im Basis-Graphen gelöscht wird

Verzweigungsstrategie



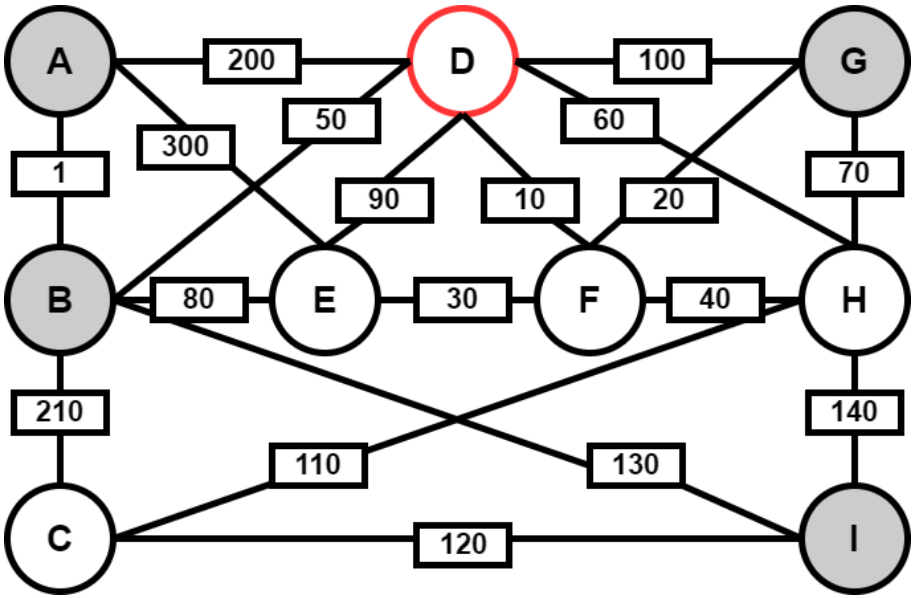
(a) Basisgraph $G=(V,E)$ mit Knotenzustand



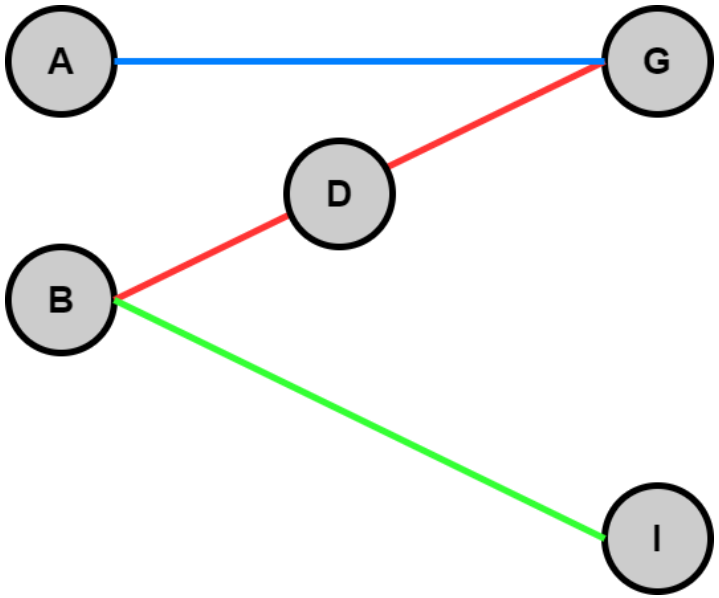
(b) Angepasster Demand-Graph $H'=(T',D')$

Abbildung: Subinstanz 1 entstanden aus Verzweigung bei D und dargestellt mit Basis-Graph, Knotenzustand und angepassten Demand-Graph

Verzweigungsstrategie



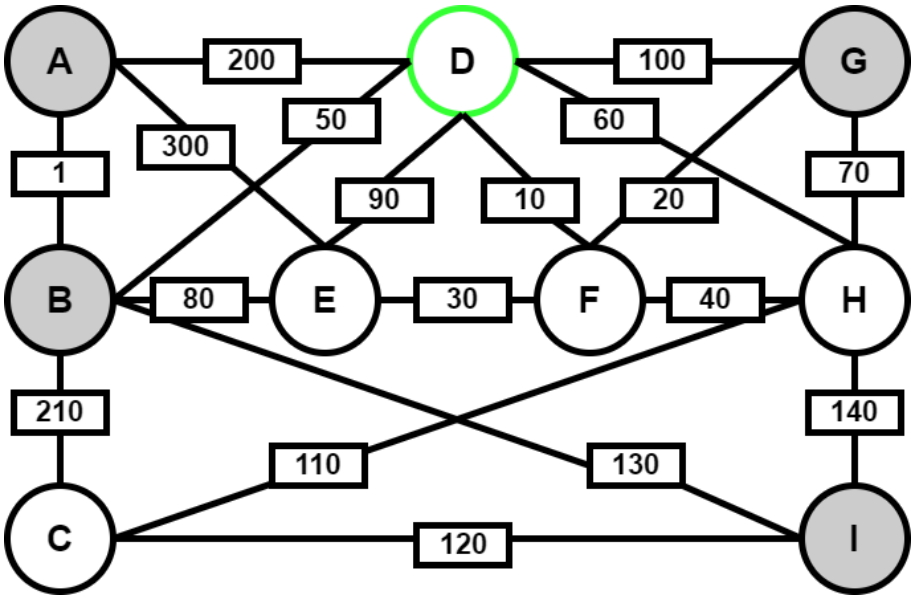
(a) Basisgraph $G=(V,E)$ mit Knotenzustand



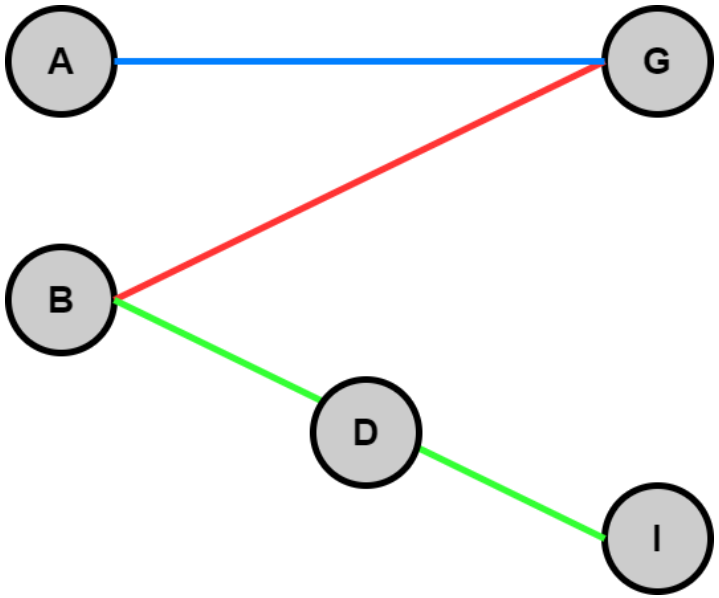
(b) Angepasster Demand-Graph $H'=(T',D')$

Abbildung: Subinstanz 2 entstanden aus Verzweigung bei D und dargestellt mit Basis-Graph, Knotenzustand und angepassten Demand-Graph

Verzweigungsstrategie



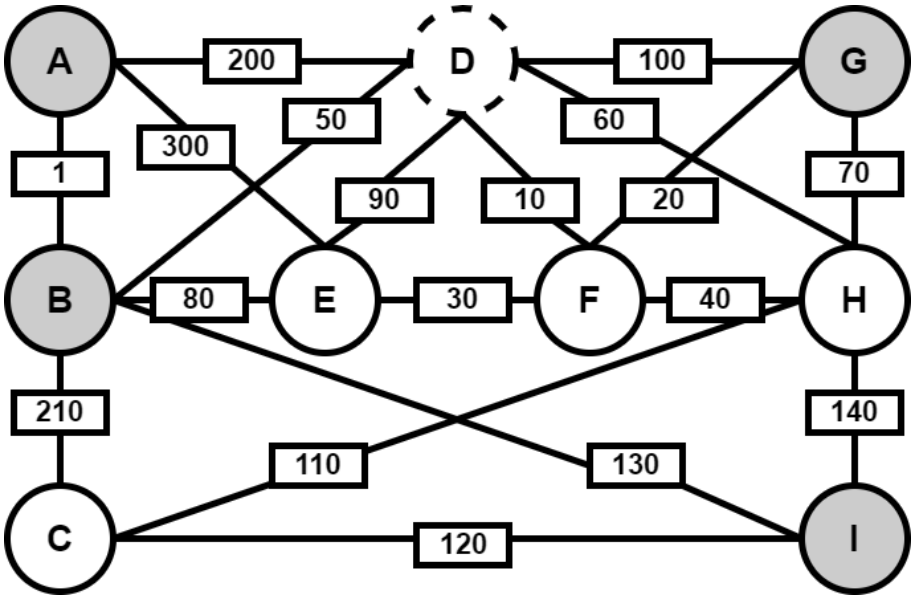
(a) Basisgraph $G=(V,E)$ mit Knotenzustand



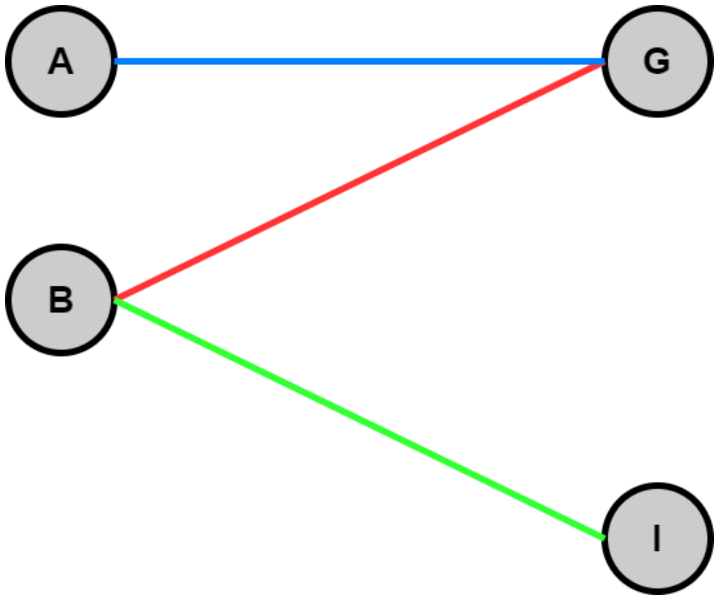
(b) Angepasster Demand-Graph $H'=(T',D')$

Abbildung: Subinstanz 3 entstanden aus Verzweigung bei D und dargestellt mit Basis-Graph, Knotenzustand und angepassten Demand-Graph

Verzweigungsstrategie



(a) Basisgraph $G=(V,E)$ mit Knotenzustand



(b) Angepasster Demand-Graph $H'=(T',D')$

Abbildung: Subinstanz 4 entstanden aus Verzweigung bei D und dargestellt mit Basis-Graph, Knotenzustand und angepassten Demand-Graph

Verzweigungsstrategie

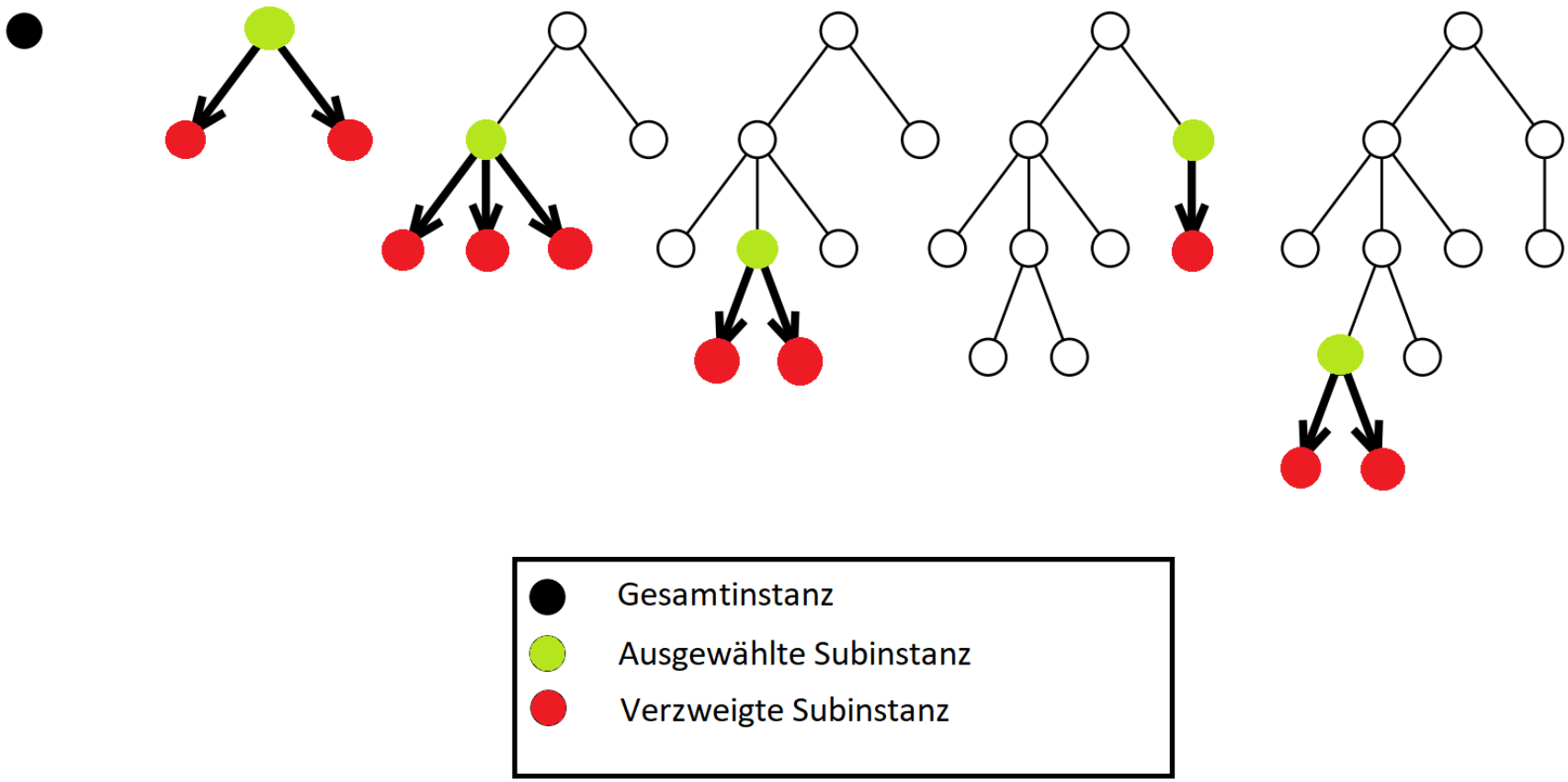


Abbildung: Schematische Darstellung des Branch and Bound Verfahren

Subinstanz

- Knotenzustand gesetzt oder ungesetzt
- Knotenzustand $= 0$ entspricht Knoten löschen
- nicht alle Knotenzustände belegt mit zugehörigen $H'(T', D') \rightarrow$ neues MCDPP mit Basis-Graph $G'(V', E')$ und Demand-Graph $H'(T', D')$, welches Subinstanz genannt wird
- Subinstanz besitzt untere Schranke $U_S \leq z(X_S^*)$
- Zustand der Subinstanz: UNZULÄSSIG, OPTIMAL, KANDIDAT

Subinstanz

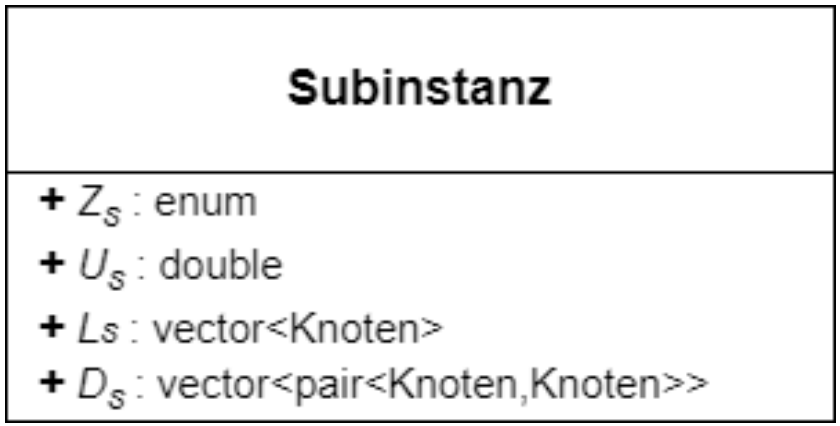


Abbildung: UML-Diagramm der Subinstanz

Auswahlstrategie

Es wird immer die Subinstanz mit dem kleinsten Wert U_S zum Verzweigen ausgewählt.

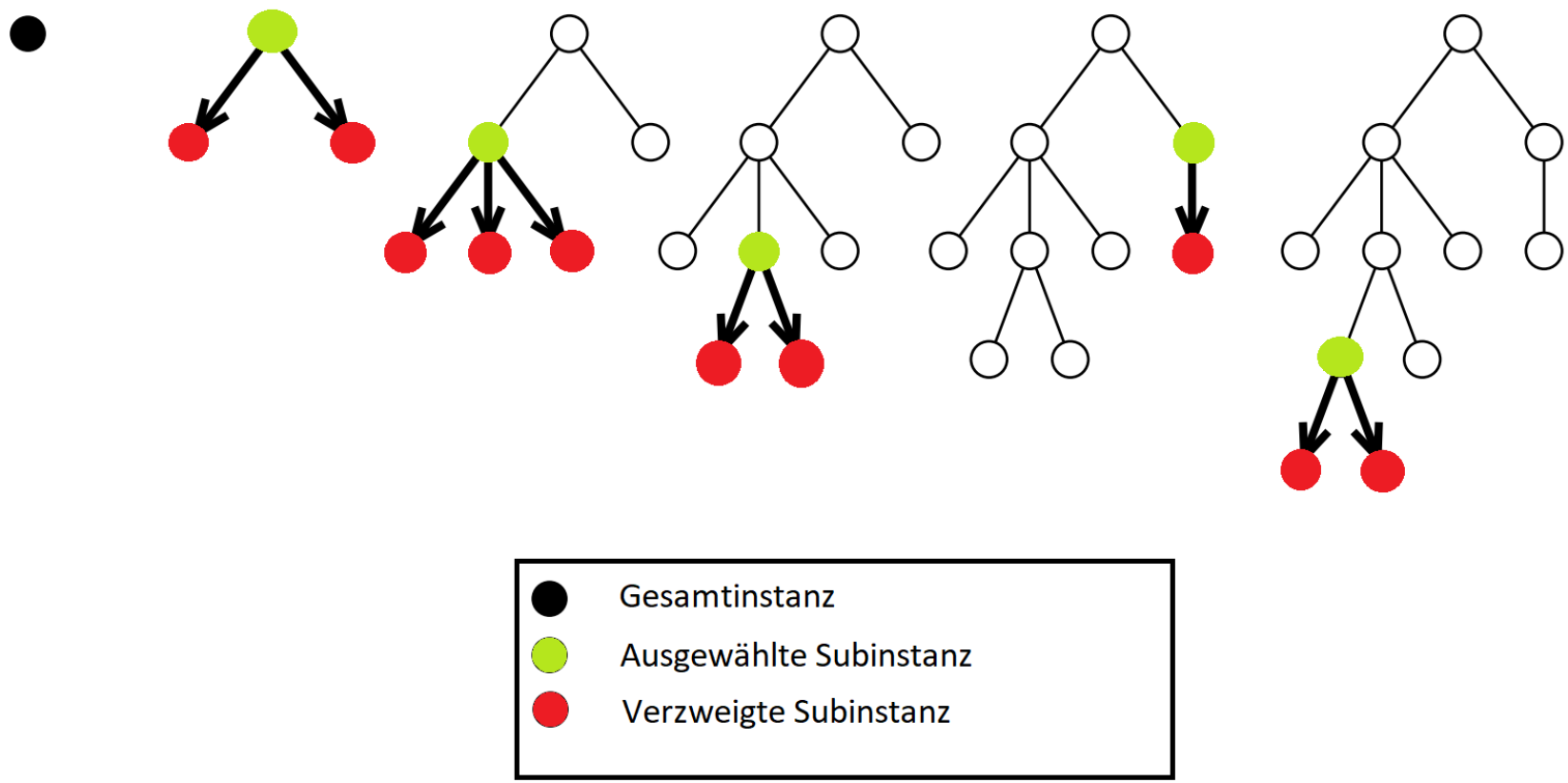


Abbildung: Schematische Darstellung des Branch and Bound Verfahren

Auswahlstrategie

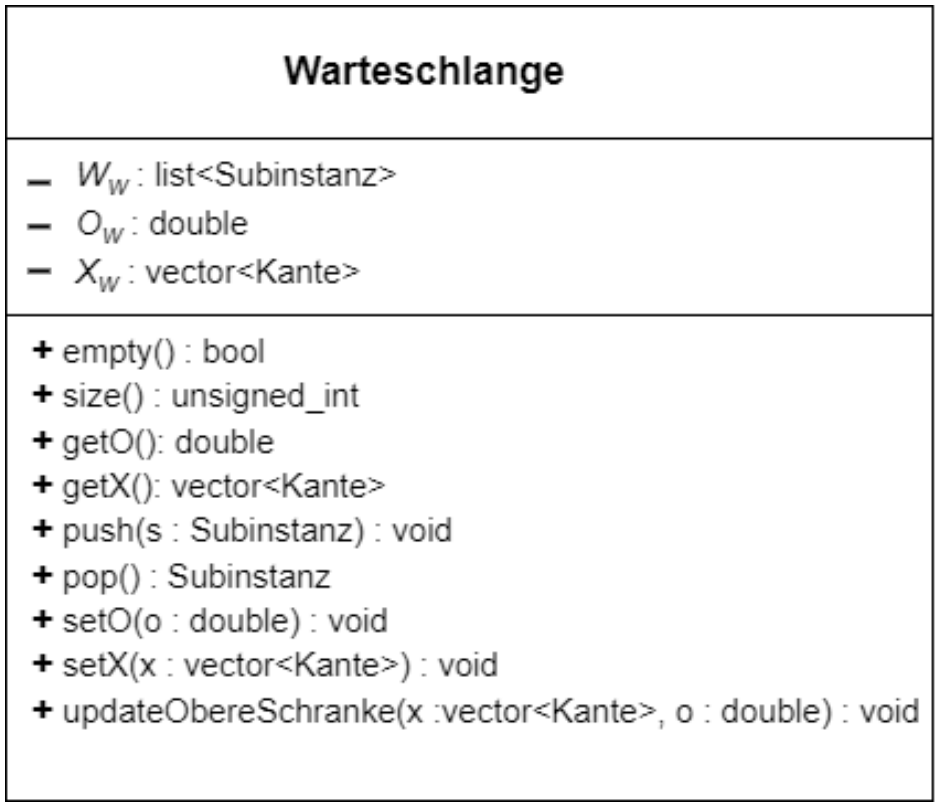


Abbildung: UML Diagramm der Menge der verzweigbaren Subinstanzen W als Warteschlange implementiert

Untere Schranke

- Lagrange-Vektor $\lambda \in (\mathbb{R}^+)^{|V|}$ mit $\mathbb{R}^+ = \{x | x \in \mathbb{R} \wedge x \geq 0\}$
- Angepasste Kosten c' des Basis-Graphen $G = (V, E)$:

$$c'(u, v, \lambda) = c(u, v) + \lambda(u) + \lambda(v) \qquad \forall \{u, v\} \in E \tag{1}$$

- Untere Schranke des MCDPP mit $G = (V, E)$ und $H = (T, D)$:

$$U(\lambda) = \sum_{d \in D} z(\text{kürzesterWeg}(G, d, \lambda)) - \sum_{u \in V} \omega(u)\lambda(u) \tag{2}$$

$$\omega(u) = \begin{cases} 2 & u \in V \setminus T \\ d_H(u) & u \in T \end{cases} \tag{3}$$

$$z(P(d)) = \sum_{e \in E^d} c'(e) \tag{4}$$

- Einbettung kürzester Wege kann zulässige Lösung sein

Untere Schranke

- große untere Schranke finden mit Subgradientenverfahren
- n_{sub} Itterationen erzeugt unterschiedliche λ
- U_S ist größte gefundene untere Schranke

$$\lambda^{(k+1)} = \lambda^{(k)} + \alpha^{(k)} \cdot \zeta^{(k)} \qquad k = 1, ..., n_{sub} \tag{5}$$

$$\zeta^{(k)} = \mu^{(k)} - \omega \tag{6}$$

$$\mu(u)^{(k)} = \sum_{d \in D} (u \in V^{d^{(k)}}) \qquad \forall u \in V \tag{7}$$

$$\alpha^{(k)} = \theta^{(k)} \frac{\overline{L} - U(\lambda^{(k)})}{||\zeta^{(k)}||^2} \tag{8}$$

$$\theta^{(k)} = \begin{cases} 2 & k = 1 \\ \frac{\theta^{(k-1)}}{2} & \text{mod}(k, n_{iter\_half}) = 0 \ \wedge \ \max\limits_{j \in [1,k]} U(\lambda^{(j)}) = \max\limits_{j \in [1,k-n]} U(\lambda^{(j)}) \\ \theta^{(k-1)} & \text{sonst} \end{cases} \tag{9}$$

Obere Schranke

- Gesamtkosten jeder gefundenen zulässigen Lösung = obere Schranke der Instanz des MCDPP
- die gefundene zulässige Lösung mit kleinsten Gesamtkosten wird gespeichert
- am Anfang keine zulässige Lösung bekannt
- Summe aller Kantenkosten = obere Schranke O des MCDPP
- O kann verkleinert werden

Optimalitätskriterium

Das MCDPP besitzt kein Optimalitätskrterium.

Komplementäre Schlupfbedingung

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 einer optimalen Lösung X_S^* der Subinstanz S.

Beschränkungskriterien

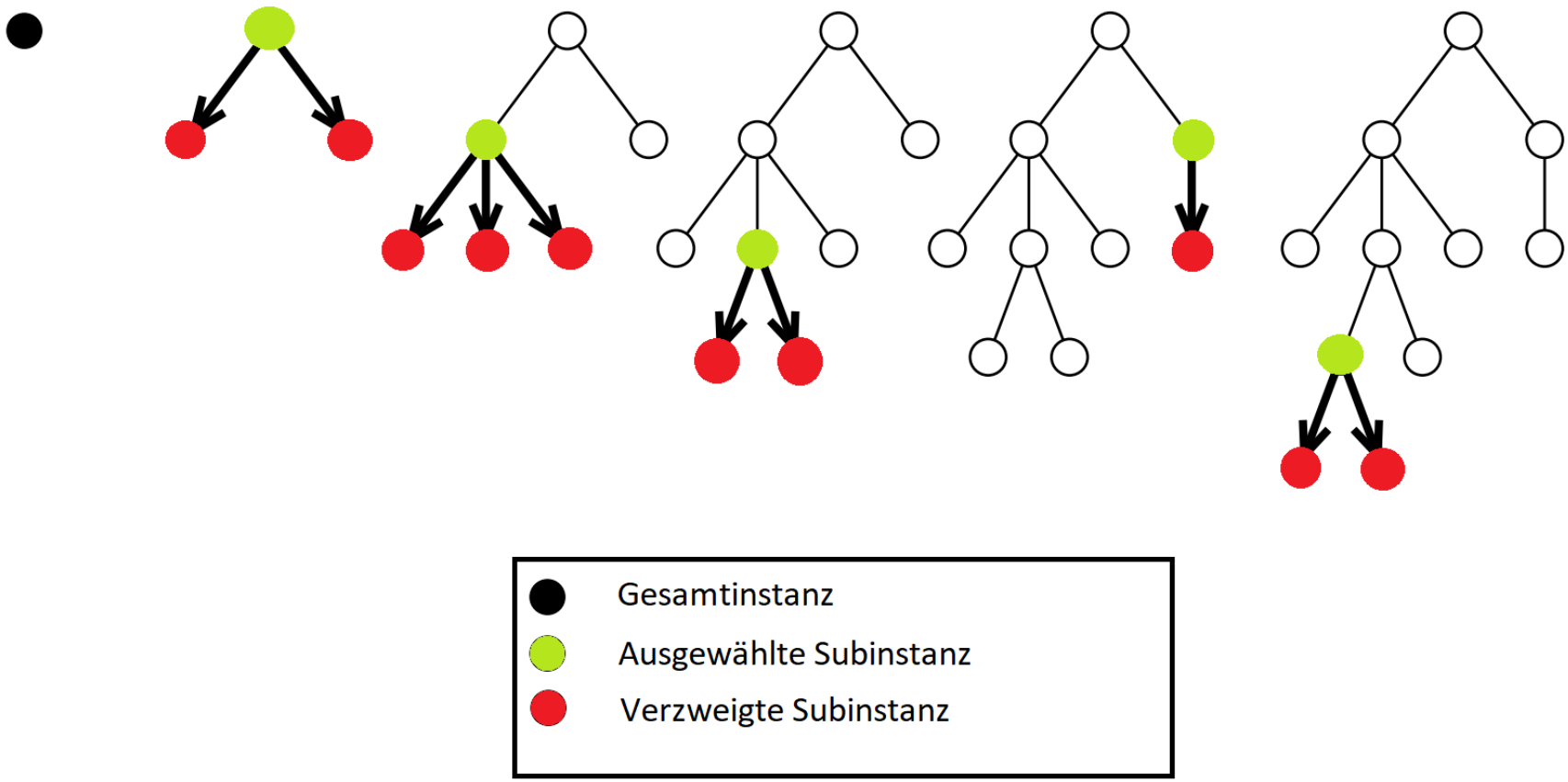


Abbildung: Schematische Darstellung des Branch and Bound Verfahren

Beschränkungskriterien

- Zustand der Subinstanz ist OPTIMAL oder UNZULÄSSIG $\rightarrow$ nicht weiter verzweigbar
- nur Subinstanzen von Zustand KANDIDAT weiter verzweigbar
- Suboptimalität $U_S \geq O \rightarrow$ nicht weiter verzweigbar
- alle Knotenzustände gesetzt $\rightarrow$ Zustand ist UNZULÄSSIG oder OPTIMAL
- existiert keine verzweigbare Subinstanz endet das Verfahren

Testergebnisse

- 7 Testinstanzen insgesamt
- 6 Testinstanzen mit Referenzlösung richtig berechnet unter 1 min
- Einstellbaren Parameter haben Einfluss auf Zeitaufwand der Solver
- Zeitaufwand ist instanzabhängig
- 7.Testinstanz sehr groß ohne Referenzlösung
- 7.Testinstanz unterschiedliche Lösungen bei Solvern mit Zeiten unter 2h (Abbruch 1 Tag)
- Numerische Fehler bei Berechnung unterer Schranke $\rightarrow$ Subinstanzen werdern verworfen (Beschränkungskriterien), die durch Verzweigung zur optimalen Lösung führen würden

Zusammenfassung

- Branch and Bound Solver theoretisch hergeleitet und praktisch in C++ implementiert
- Testinstanzen mit Referenzlösung richtig berechnet in kurzer Zeit
- Numerische Fehler bei großen Testinstanzen $\rightarrow$ keine Optimalitätsgarantie
- Laufzeit hängt von Instanz und einstellbaren Parametern ab

Quellen

[1] Florian Hahn. *Schrankenverfahren für sichere Subnetze*. Berlin Hochschule für Technik, 2023.

[2] James B. Orlin Bavindra K. Ahuja Thomas L. Magnanti. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

[3] Prof. Dr. Martin Oellrich. *Operations Research Vorlesungsskript*. Berliner Hochschule für Technik, 2023.

Fibonacci-Heap

- Dijkstra-Algorithmus $O(|V|pop + |E|push)$
- implementierte Liste für Dijkstra: $O(|V| + |E||V|)$
- Fibonacci-Heap:

$$O(pop) = O(log(V))$$

$$O(push) = O(1)$$

$$O(|V|pop + |E|push) = O(|V|log(V) + |E|)$$

- theoretisch beste bekannte Datenstruktur für Warteschlange des Dijkstra-Algorithmus

Modifizierte Kantenkosten

Modifizierte Kantenkosten

Sei $G = (V, A)$ ein gerichteter Graph mit Kantenkosten $c : A \to \mathbb{R}_0^+$ und $s \in V$. Es sei $t \in V$ ein weiterer Knoten, $dist_c(\cdot, t)$ die Distanzen aller anderen Knoten zu ihm und $c' : A \to \mathbb{R}_0^+$ die modifizierte Kantenkostenfunktion

$$c'(u, v) = c(u, v) - dist_c(u, t) + dist_c(v, t)$$

Dann gilt fur alle Knoten $u \in V$: Ein s-u-Weg ist genau dann ein kürzester bzgl. c', wenn er auch ein kürzester bzgl. c ist.

Modifizierte Kantenkosten

- Subgradientenverfahren $\rightarrow$ Kantenkosten c ändern sich mit λ
- modifizierte Kantenkosten:

$$c''(u,v) = c'(u,v) - dist_c(u,t) + dist_c(v,t)$$

$$c'(u,v) = c(u,v) + \lambda(u) + \lambda(v)$$

- kürzeste Wege bzgl c' bleiben gleich, jedoch bei Suche existiert Orientierung (siehe S. 211)
- Beweis folgt

Modifizierte Kantenkosten

$$
\begin{aligned}
c''(P(s,u)) &= \sum_{(v,w)\in P(s,u)} c''(v,w) \\
&= \sum_{(v,w)\in P(s,u)} c'(v,w) - dist_c(v,t) + dist_c(w,t) \\
&= \sum_{(v,w)\in P(s,u)} c'(v,w) + \sum_{(v,w)\in P(s,u)} dist_c(w,t) - dist_c(v,t) \\
&= c'(P(s,u)) + (dist_c(u,t) - dist_c(s,t))
\end{aligned}
$$

$\rightarrow$ $\text{argmin}_{P(s,u)} c''(P(s,u))$ und $\text{argmin}_{P(s,u)} c'(P(s,u))$ liefern die selben kürzesten Wege aber mit anderen Längenwerten

Modifizierte Kantenkosten

- Branch and Bound Verfahren $\rightarrow$ Löschen von Knoten aus Basis-Graphen möglich
- Löschen verringert Menge der Wege zwischen 2 Knoten u und v
- Beweis bleibt gleich $\rightarrow$ selbe kürzeste Wege
- erstelle am Anfang des B& B Verfahrens $|T|$ Kürzeste-Wege-Bäume für jeden Terminalknoten
  $\rightarrow$ jeder Dijkstra für einen Demand und λ im Subgradientenverfahren für jede Subinstanz kann mit c'' rechnen
  $\rightarrow$ Dijkstra findet schneller kürzesten Weg, weil alle Zwischenterminals in der Nähe des Terminals liegen

Einstellbare Parameter

- Basis-Graph: $G = (V, E)$
- Demand-Graph: $H = (T, D)$
- Anzahl Iteration Subgradientenverfahren: n_{sub}
- Update Schrittweite Subgradientenverfahren: n_{iter_half}
- Start-Lagrange-Vektor: λ_{Option}
- Wahl des Verzweigungsknotens: VZK_{Option}
- verringerte Anzahl Iteration Subgradientenverfahren: n'_{sub}

Einstellbare Parameter

1. $\lambda_{Option} = \lambda\_\text{NULL} \Leftrightarrow \lambda_{Start} = \mathbf{0}$
2. $\lambda_{Option} = \lambda\_\text{MAX} \Leftrightarrow \lambda \in P$ mit
   $P = \{\lambda^{(k)} \ \wedge \ k \in [1, n_{sub}] \mid U(\lambda^{(k)}) = \max\limits_{k \in [1, n_{sub}]} U(\lambda^{(k)})\}$
3. $\lambda_{Option} = \lambda\_\text{KONFLIKT} \Leftrightarrow \lambda \in P'$ mit
   $P' = \{\lambda^{(k)} \ \wedge \ k \in [1, n_{sub}] \mid MK^{(k)} = \min\limits_{k \in [1, n_{sub}]} |MK^{(k)}|\}$

Einstellbare Parameter

1. $VZK_{Option} = \text{MAX\_FLUSS} \Leftrightarrow$ bei λ_{Start} Knoten der am häufigsten Teil eines Weges ist
2. $VZK_{Option} = \text{WEG} \Leftrightarrow$ bei λ_{Start} mittlerer Knoten des Weges, welcher die meisten Knoten besitzt
3. $VZK_{Option} = \text{KONFLIKTE} \Leftrightarrow$ bei λ_{Start} aus Menge an Konfliktknoten den Knoten wählen, der nach Verzweigung den größten Wert der kleinsten unteren Schranke der verzweigten Subinstanzen besitzt

Relation

Eine binäre Relation R zwischen zwei Mengen A,B ist eine (beliebige) Teilmenge ihres kartesischen Produkts:

$$R \subseteq A \times B \quad \text{d.h.} \quad R = \{(x, y) \in A \times B \mid r(x, y)\}$$

mit einem definierenden zweistelligen Prädikat $r(x, y)$.

Optimale Substruktur kürzester Wege

Optimale Substruktur kürzester Wege (S.192)

Sei $G = (V, A)$ ein gerichteter Graph mit Kantenkosten $c : A \to \mathbb{R}_0^+$ und $P(s, t)$ ein kürzester s-t-Weg in G bzgl. c. Dann ist jeder Teilweg $P(u, v) \subseteq P(s, t)$ ein kürzester u-v-Weg in G bzgl. c.