Karlsruher Institut für Technologie

Algorithmen für Routenplanung

14. Vorlesung, Sommersemester 2021

Jonas Sauer | 2. Juni 2021

Elektromobilität

Elektrofahrzeuge (EVs):

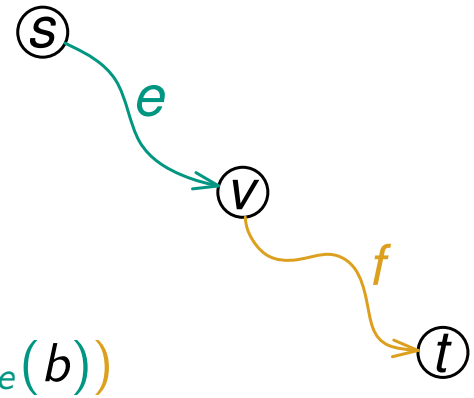
- Transportmittel der Zukunft
- Emissionsfreie Mobilität

Aber:

- Akkukapazität eingeschränkt (und damit Reichweite)
- Lange Ladezeiten, wenig öffentliche Ladestationen
- „Reichweitenangst“

$\Rightarrow$ Berücksichtigung von Energieverbrauch bei der Routenplanung

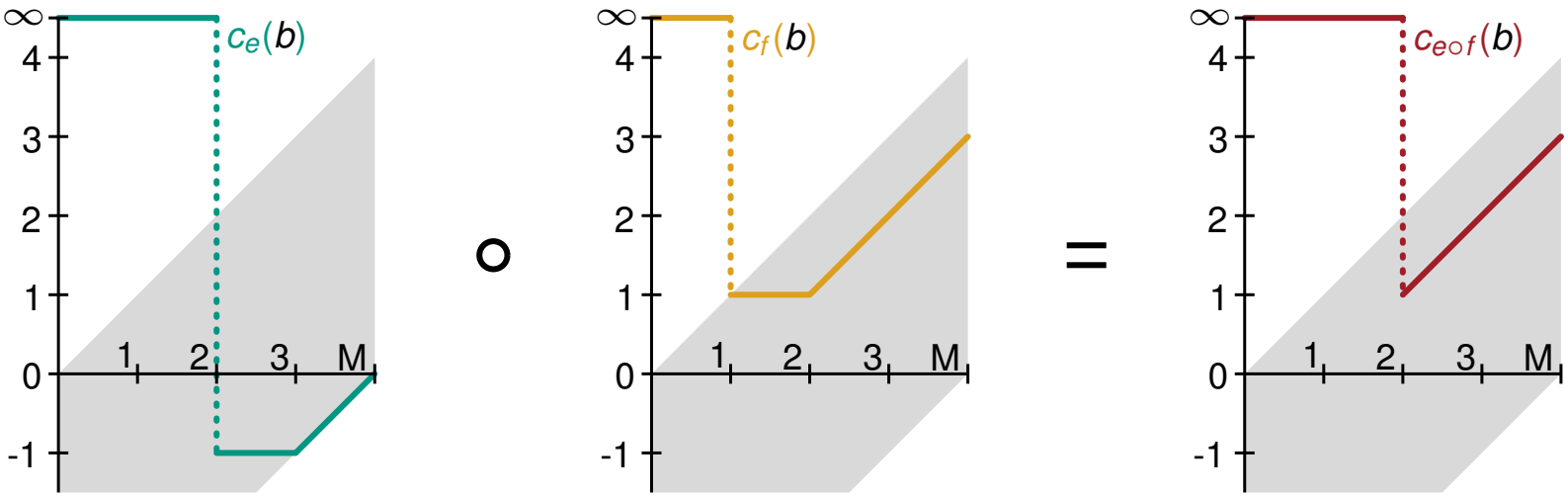
Verbrauchsfunktionen – Linking



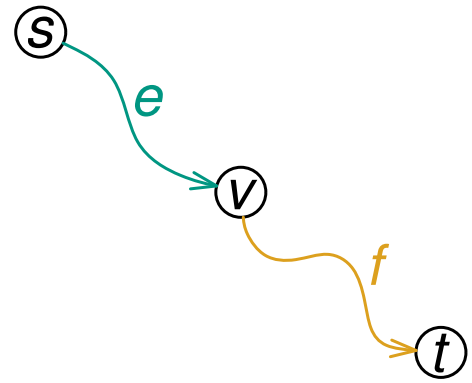
Gesucht: Verbrauch beim Traversieren von e *und* f

- Verbrauch auf e gegeben durch $c_e(b)$
- SoC nach e = SoC vor f = $b - c_e(b)$

$\Rightarrow$ Verbrauch auf e UND f: $c_{e \circ f}(b) = c_e(b) + c_f(b - c_e(b))$

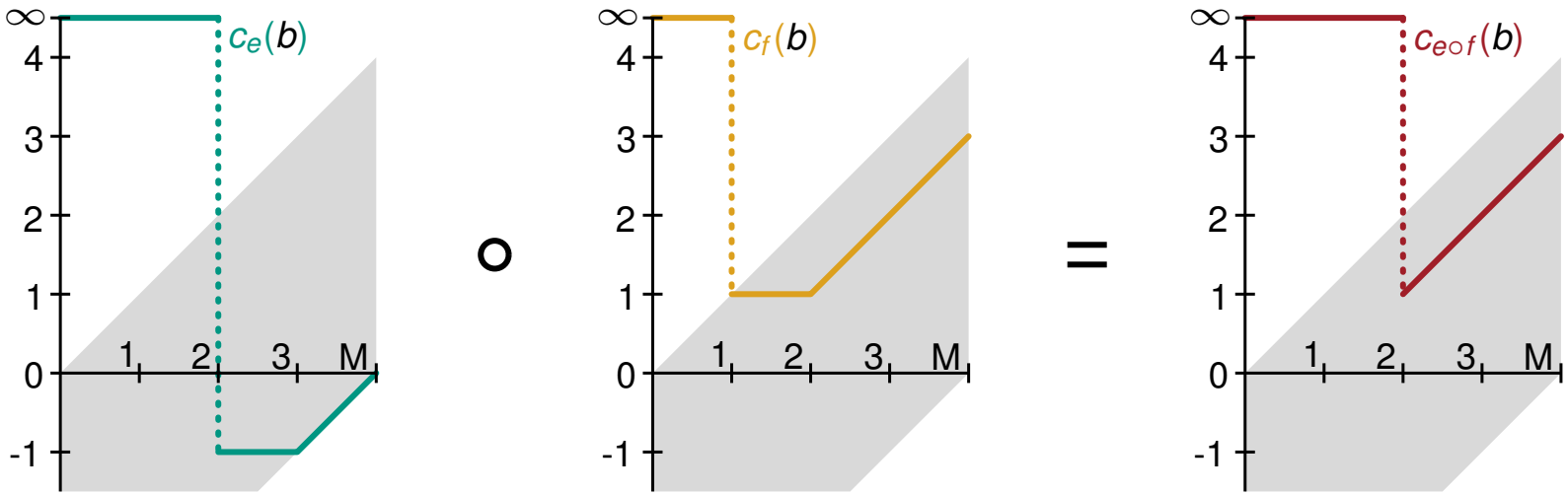


Verbrauchsfunktionen – Linking



Formal: $c_{e \circ f}(b)$ ist gegeben durch:

$$\begin{aligned} \text{minIn}_{e \circ f} &= \max[\text{minIn}_e,\ \text{minIn}_f + \text{cost}_e] \\ \text{maxOut}_{e \circ f} &= \min[\text{maxOut}_f,\ \text{maxOut}_e - \text{cost}_f] \\ \text{cost}_{e \circ f} &= \max[\text{cost}_e + \text{cost}_f,\ \text{minIn}_e - \text{maxOut}_f] \end{aligned}$$

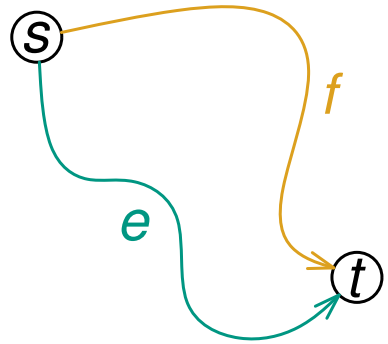


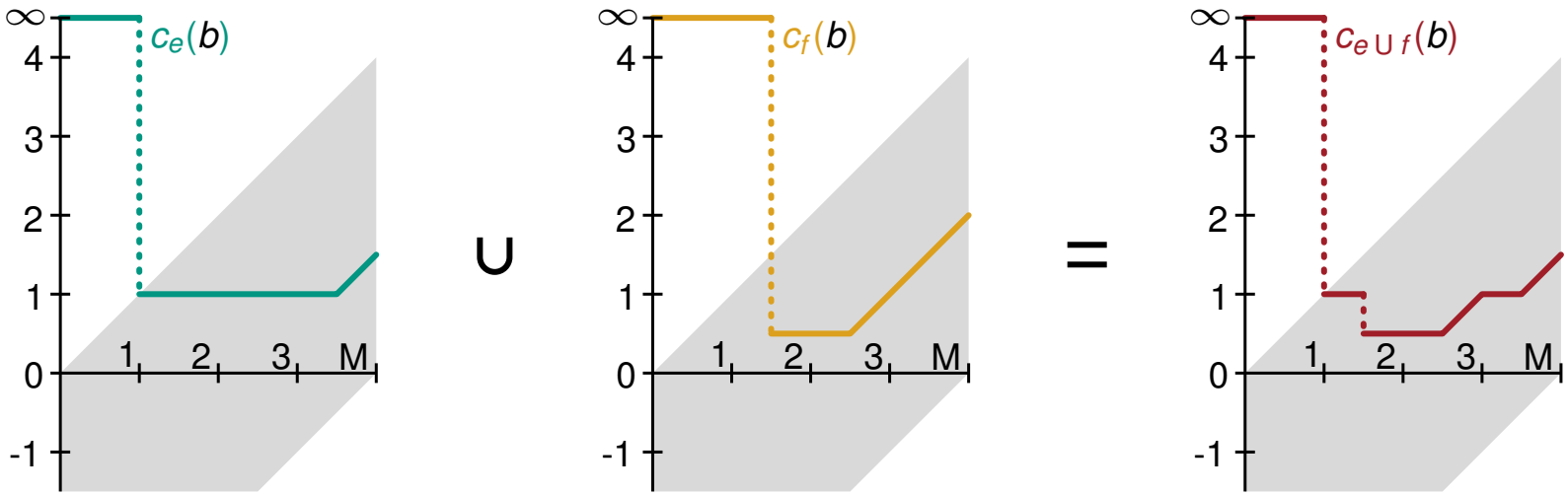
Verbrauchsfunktionen – Merging

Gesucht: Verbrauch beim Traversieren von e *oder* f

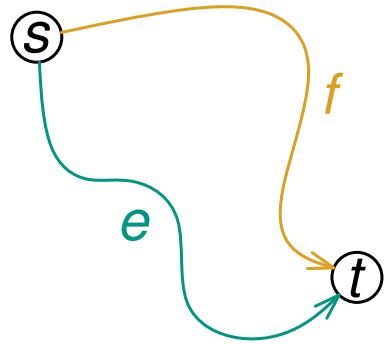
- Benutze Pfad mit geringerem Verbrauch

$\Rightarrow$ Verbrauch auf e ODER f: $c_{e \cup f}(b) = \min(c_f(b), c_e(b))$





Verbrauchsfunktionen – Merging

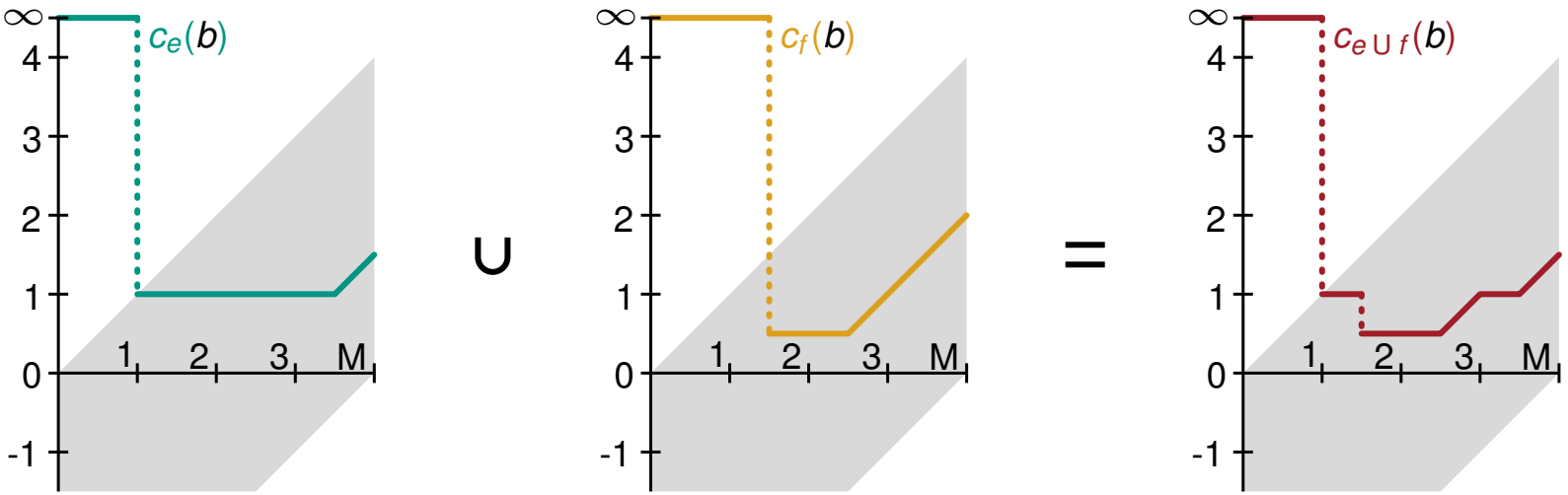


Gesucht: Verbrauch beim Traversieren von e *oder* f

- Benutze Pfad mit geringerem Verbrauch

$\Rightarrow$ Verbrauch auf e ODER f: $c_{e \cup f}(b) = \min(c_f(b), c_e(b))$

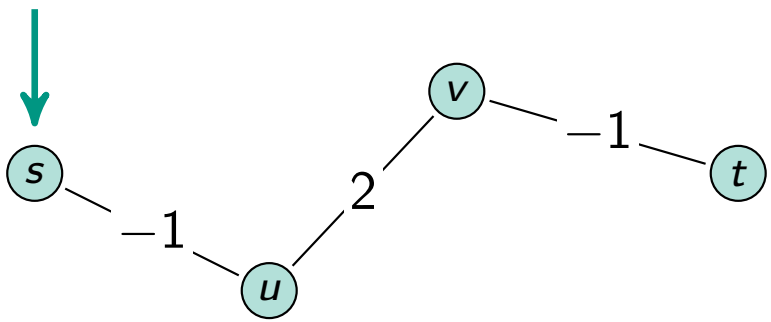
Aber: Im Allgemeinen $\mathcal{O}(m)$ Stützstellen

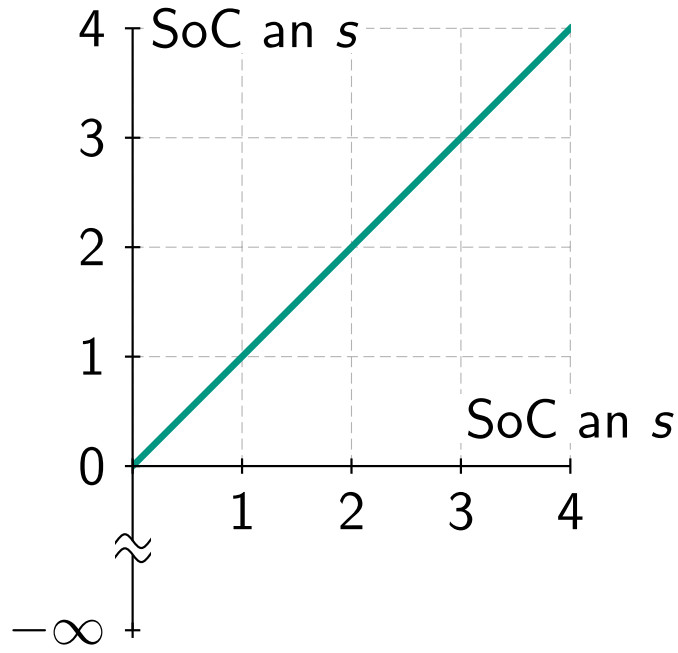


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

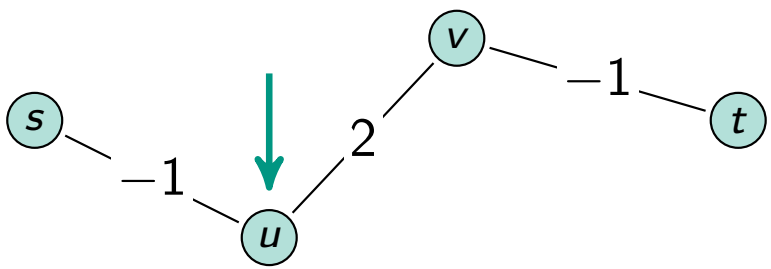


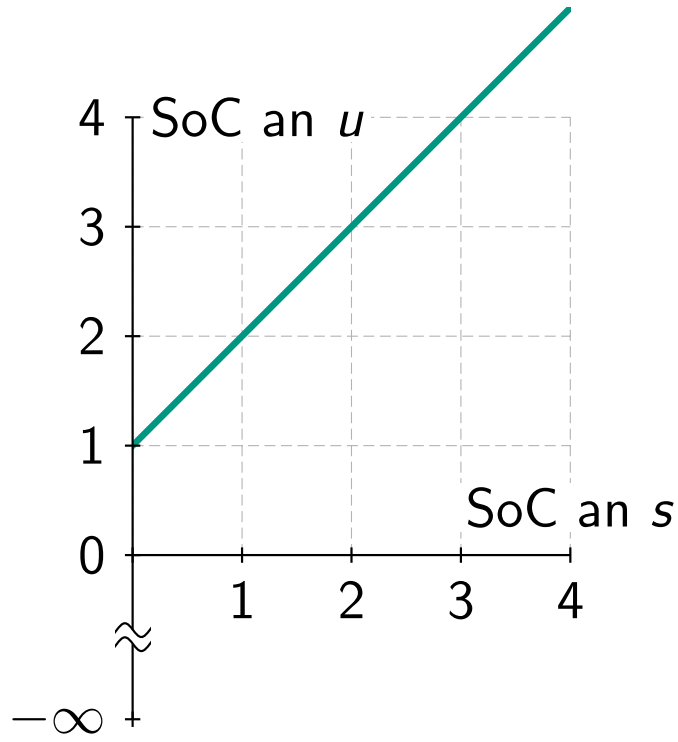


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

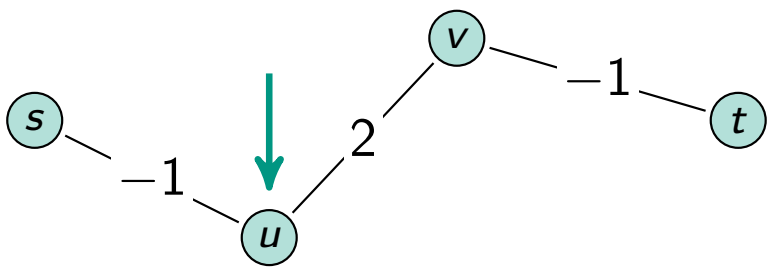


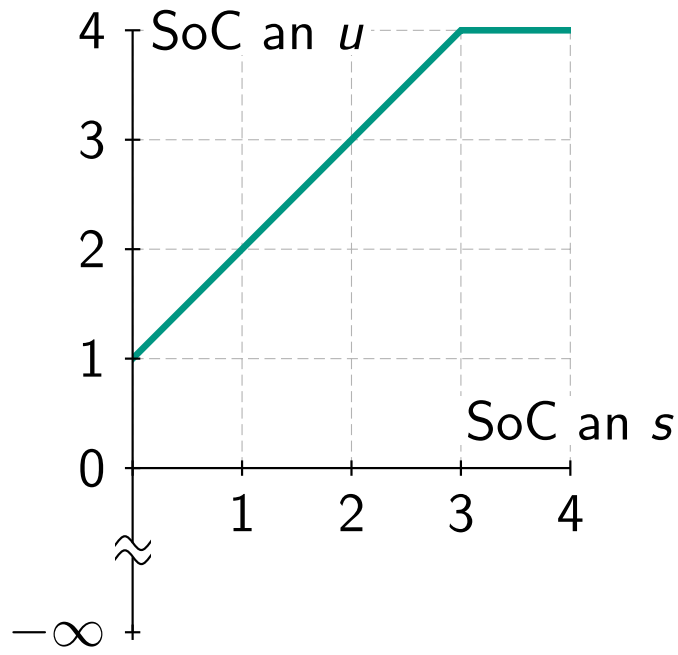


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

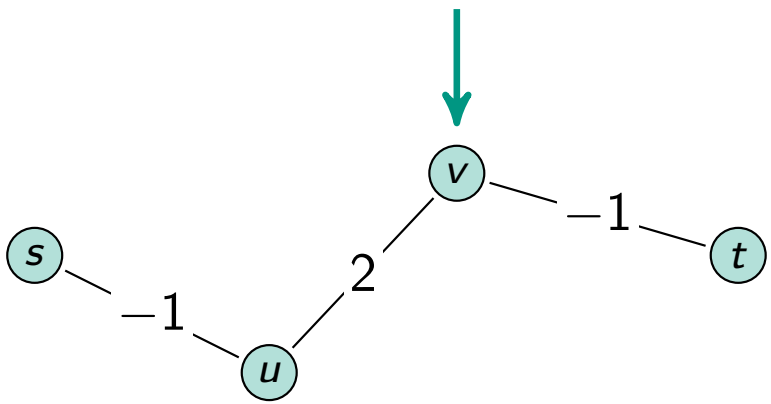


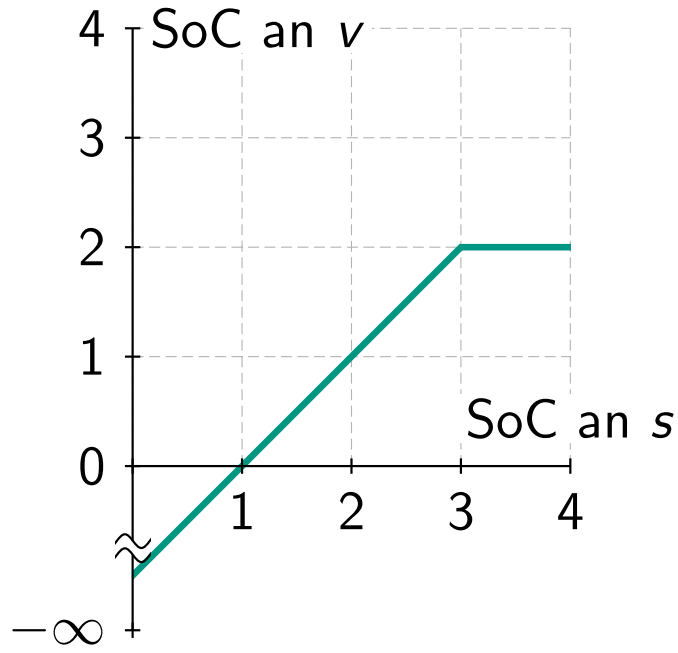


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

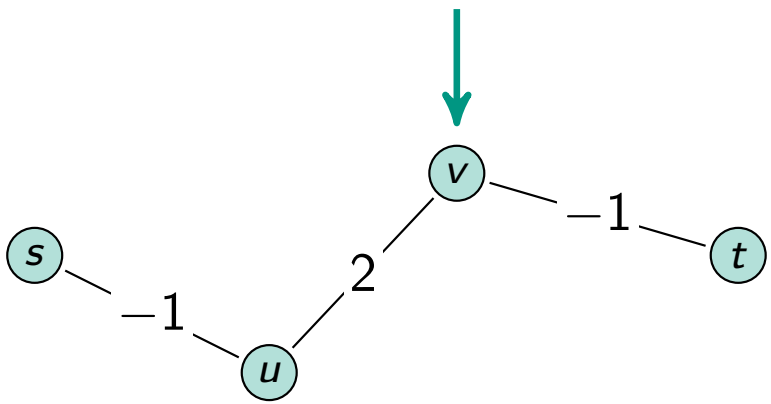


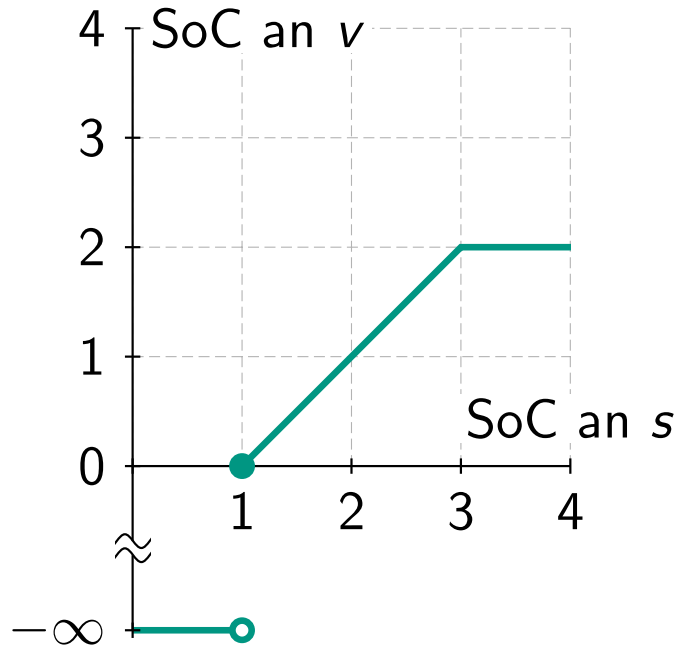


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

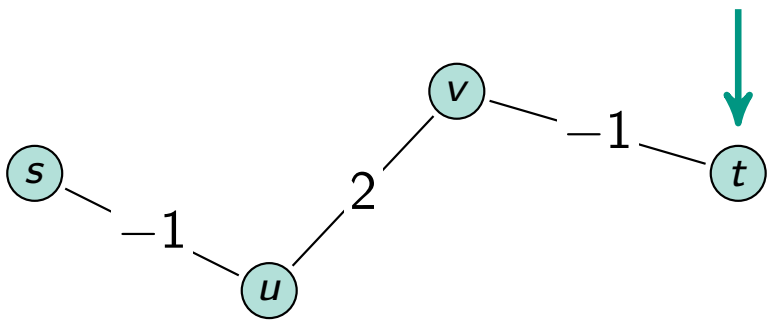


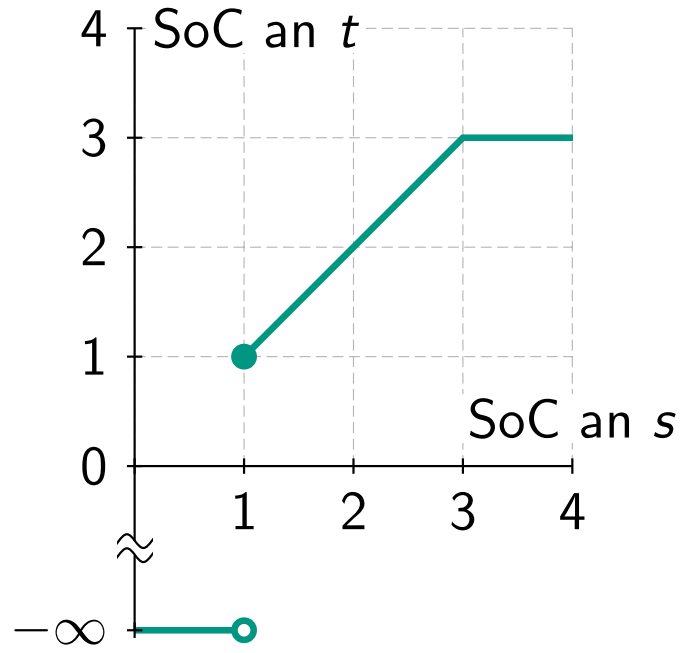


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

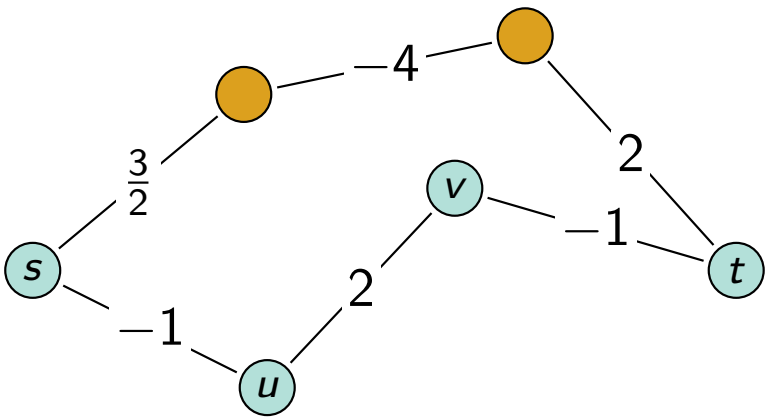


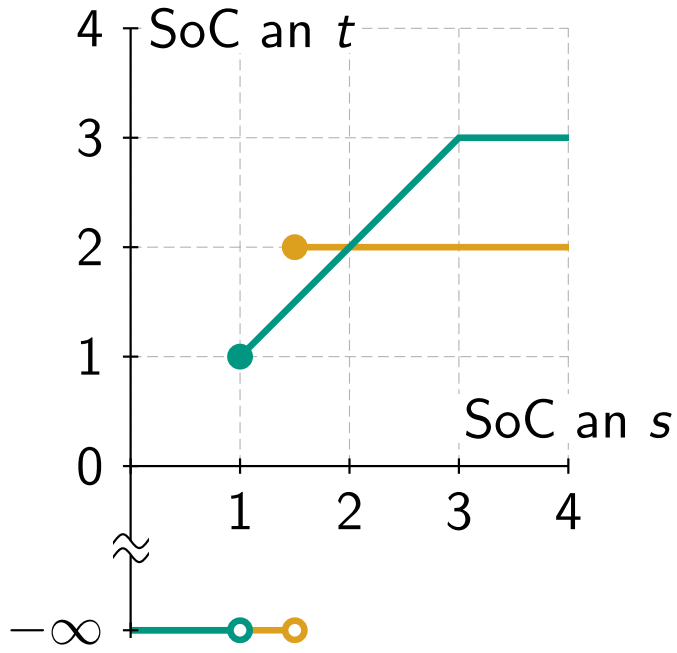


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

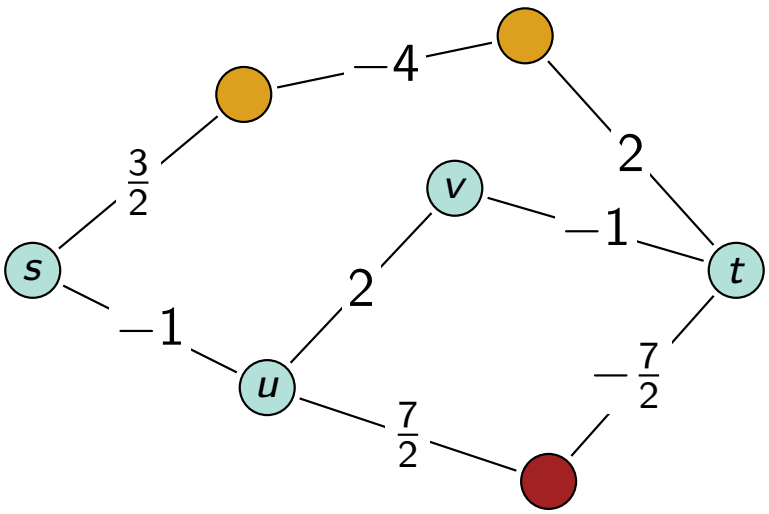


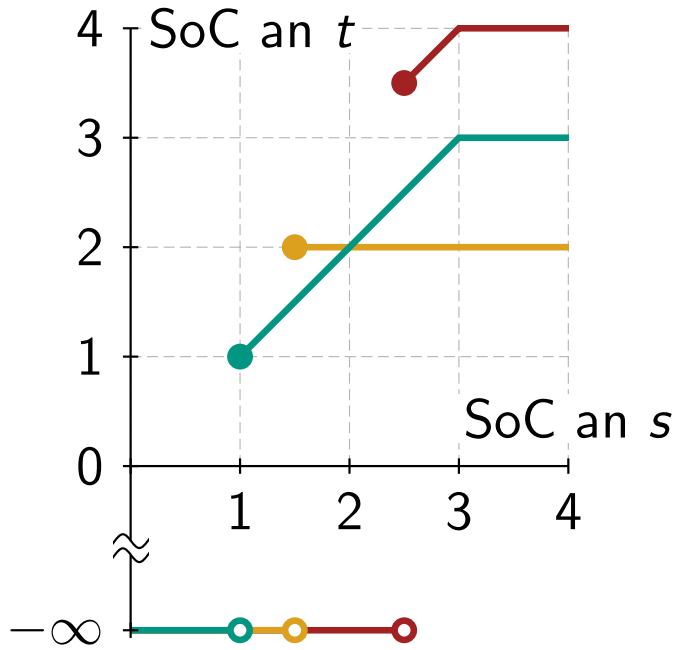


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

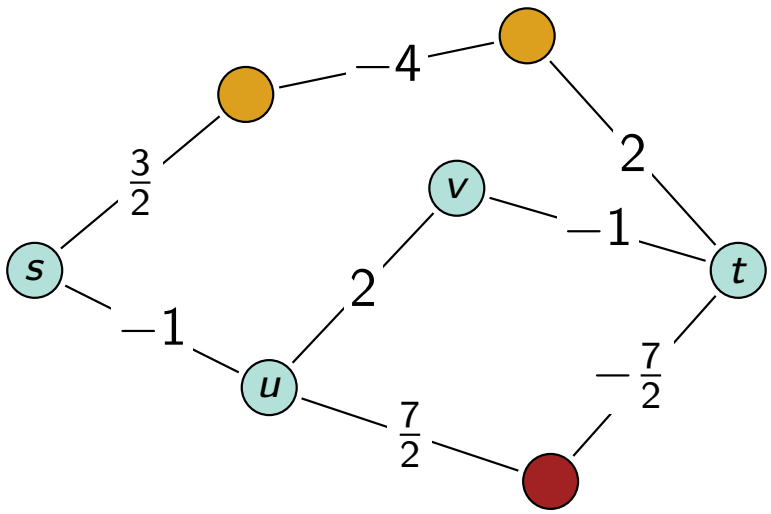


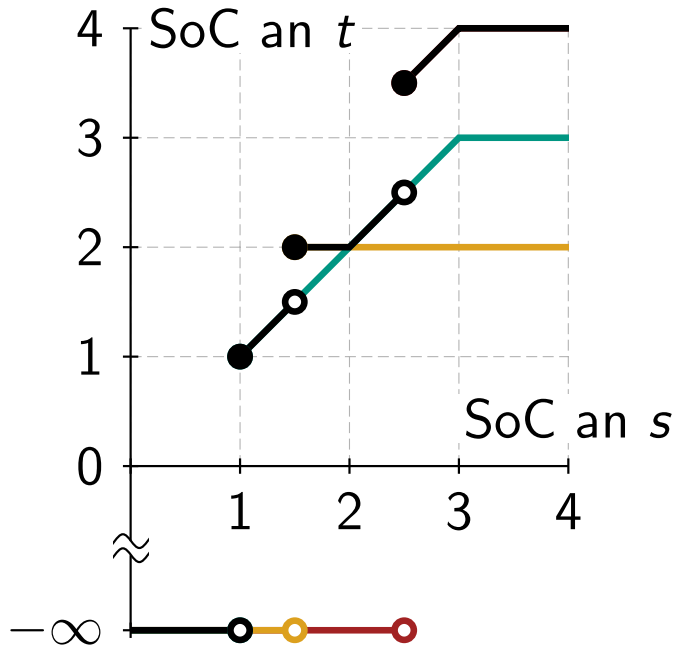


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

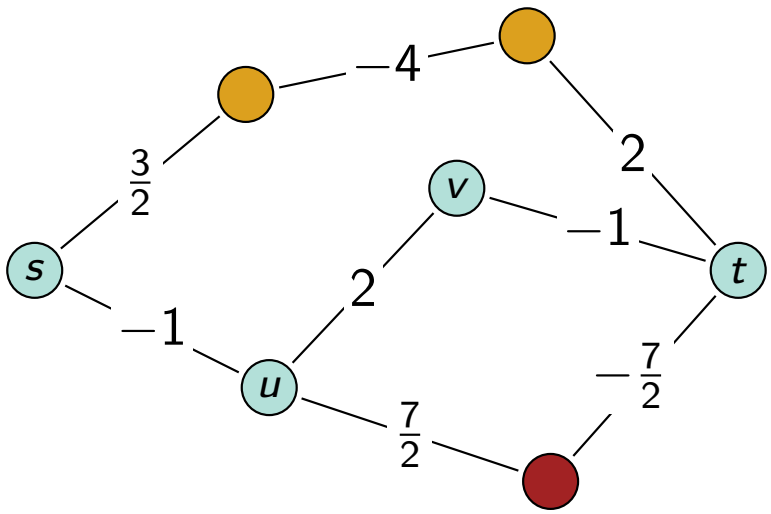


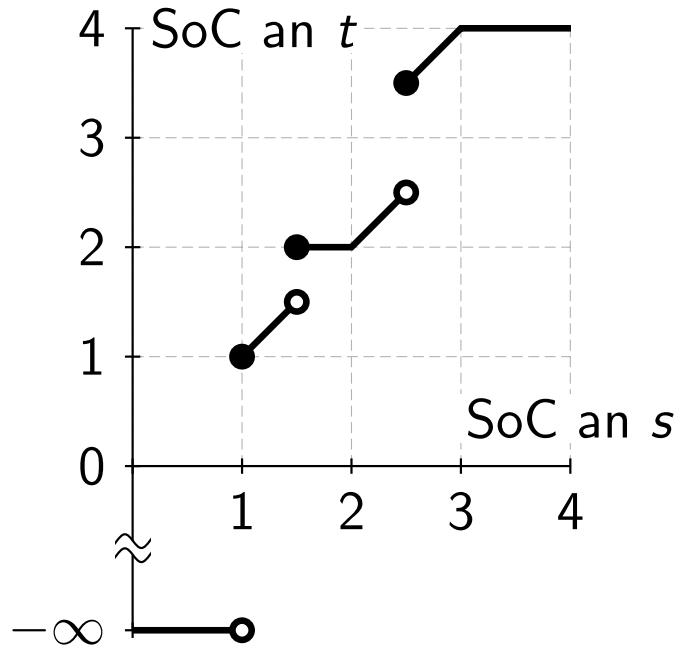


Allgemeine Profile

Profilsuche: Gegeben Start s und Ziel t, finde zulässige s-t-Pfade mit maximalem SoC an t für *alle* $b_s \in [0, M]$

Beispiel: $M = 4$

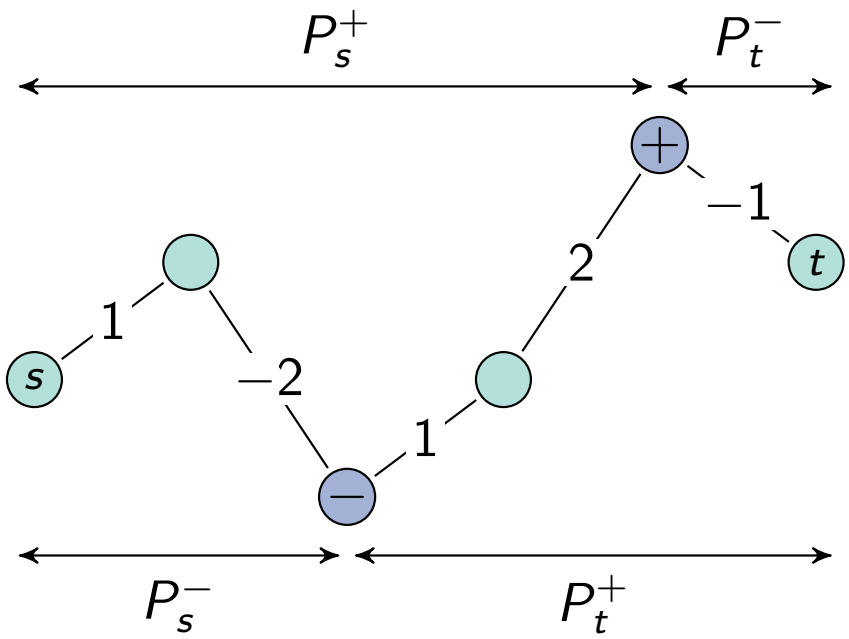




Anzahl Stützpunkte ist linear in Anzahl der Pfade
⇒ Wie viele verschiedene Pfade tragen zum Profil bei?

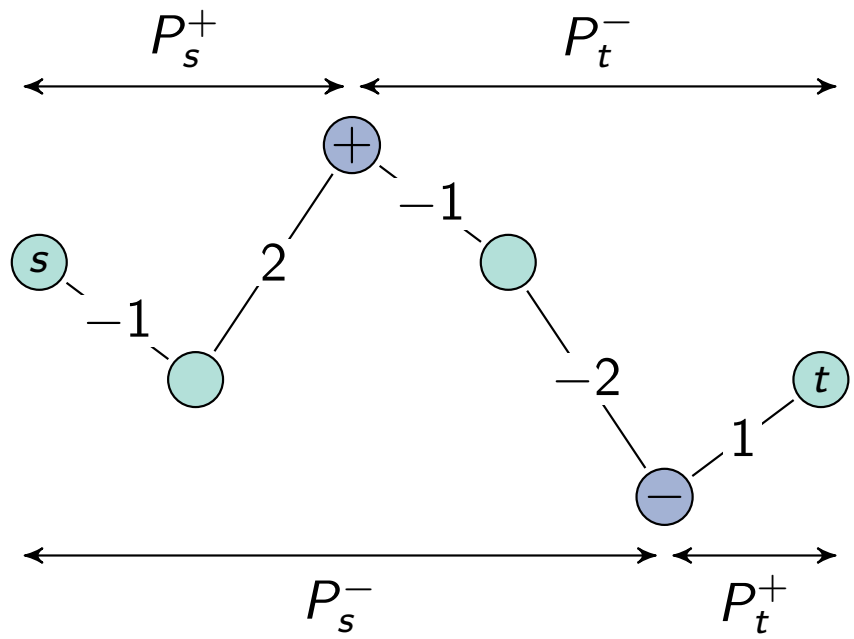
Typen von Pfaden

Tal-Berg-Pfad:



Min. Präfix endet vor max. Präfix

Berg-Tal-Pfad:



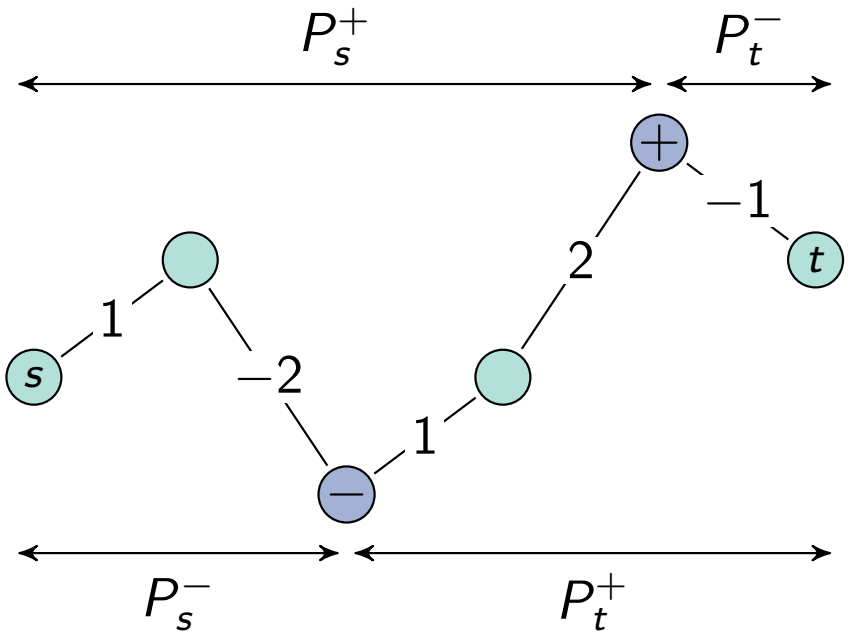
Max. Präfix endet vor min. Präfix

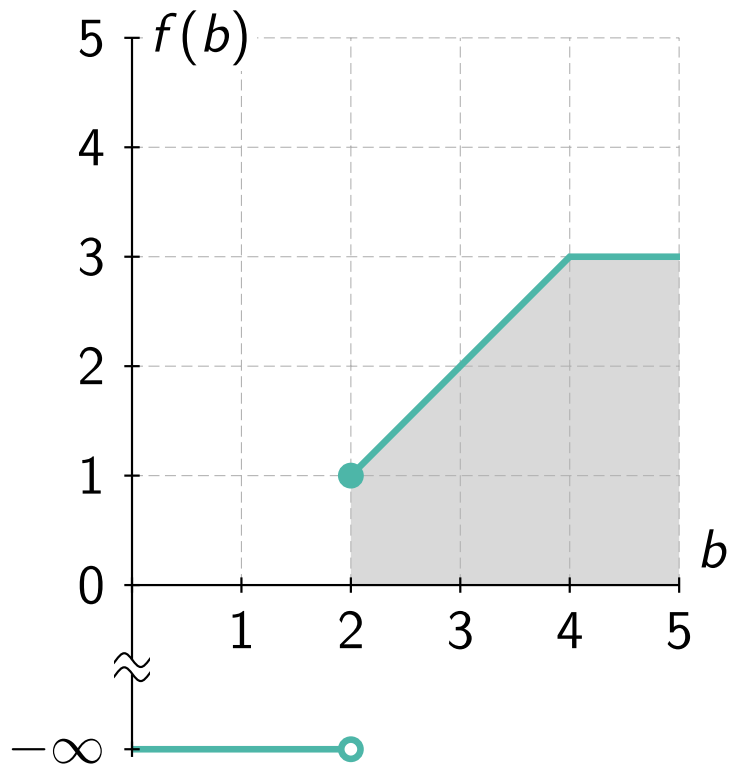
- Letzter Knoten des max. Präfix heißt Bergknoten
- Letzter Knoten des min. Präfix heißt Talknoten

Komplexität von SoC-Profilen

Betrachte beliebiges Paar von Bergknoten $\overline{v}$ und Talknoten $\underline{v}$:

Höchstens ein Tal-Berg-Pfad und ein Berg-Tal-Pfad benutzen $\overline{v}$ und $\underline{v}$ als Berg- bzw. Talknoten

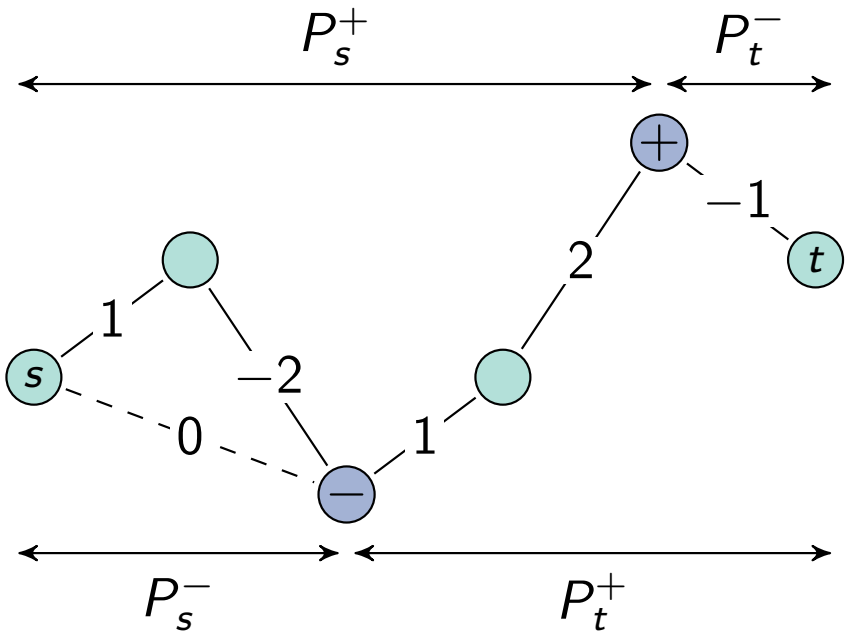


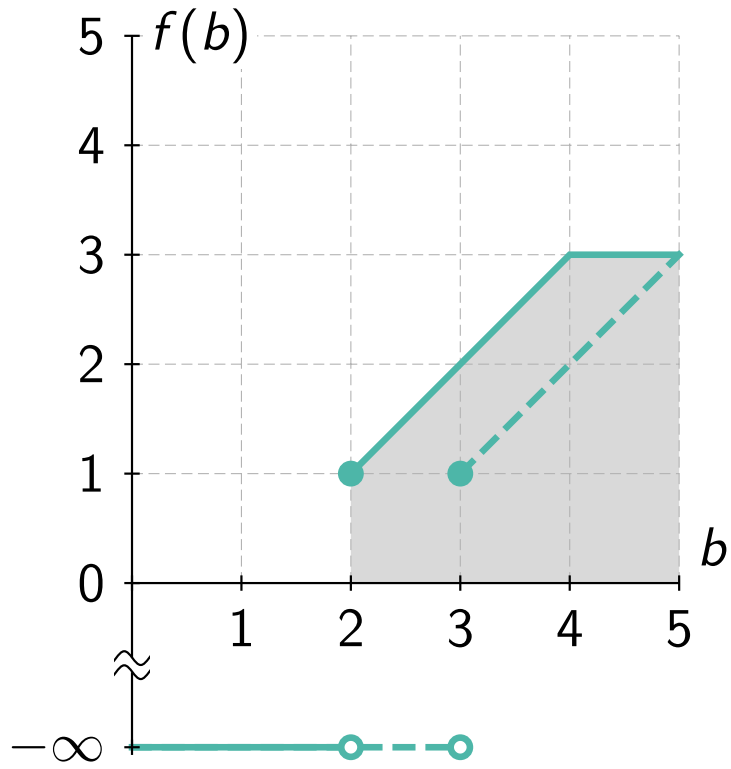


Komplexität von SoC-Profilen

Betrachte beliebiges Paar von Bergknoten $\overline{v}$ und Talknoten $\underline{v}$:

Höchstens ein Tal-Berg-Pfad und ein Berg-Tal-Pfad benutzen $\overline{v}$ und $\underline{v}$ als Berg- bzw. Talknoten

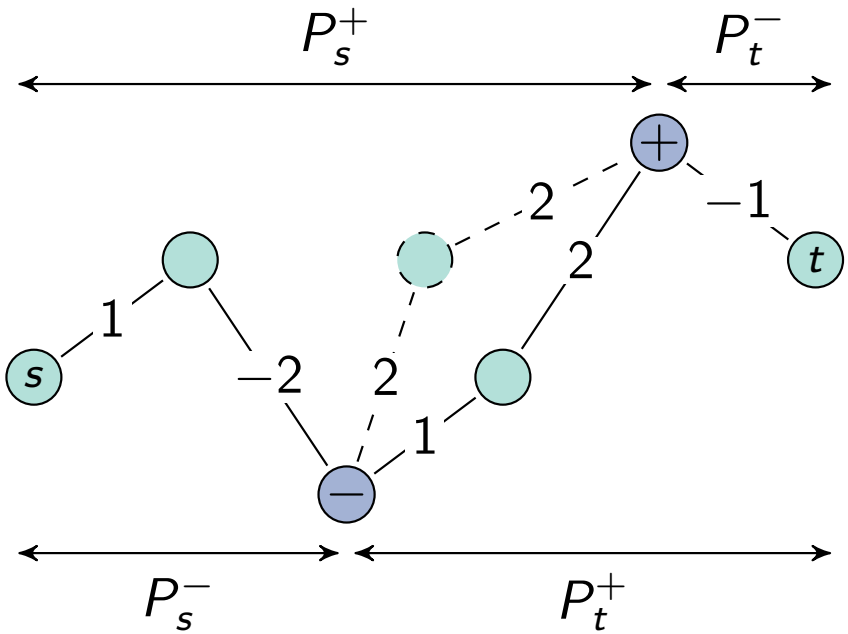


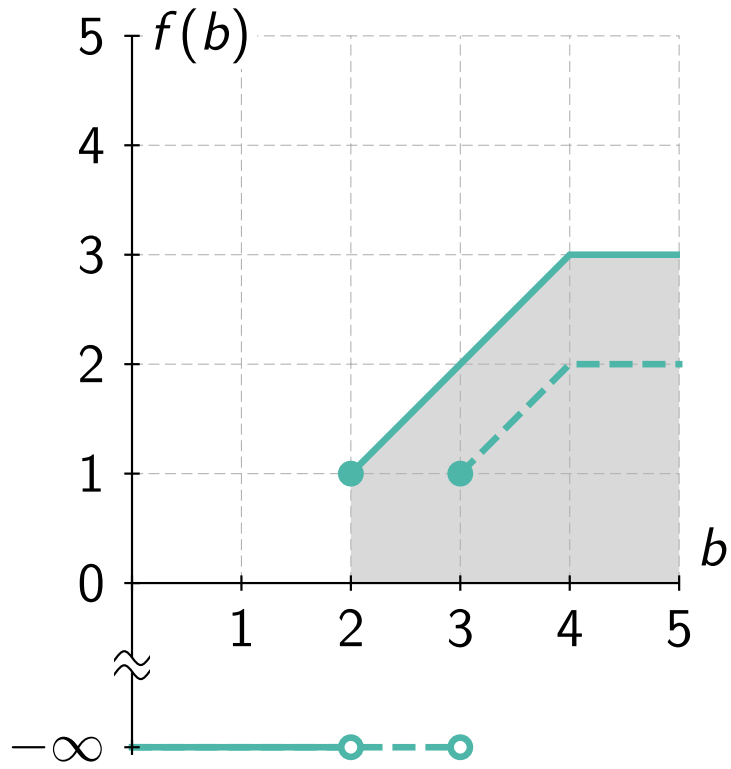


Komplexität von SoC-Profilen

Betrachte beliebiges Paar von Bergknoten $\overline{v}$ und Talknoten $\underline{v}$:

Höchstens ein Tal-Berg-Pfad und ein Berg-Tal-Pfad benutzen $\overline{v}$ und $\underline{v}$ als Berg- bzw. Talknoten

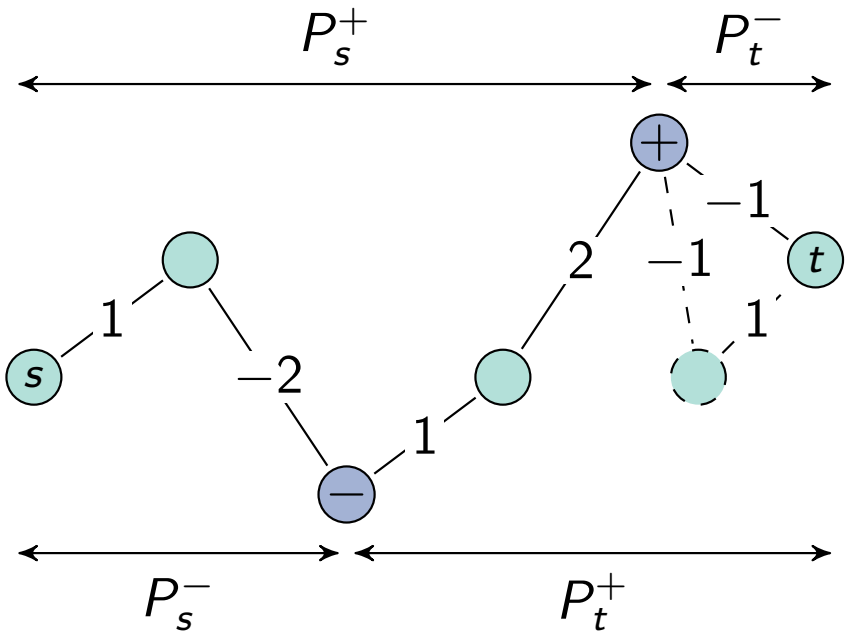


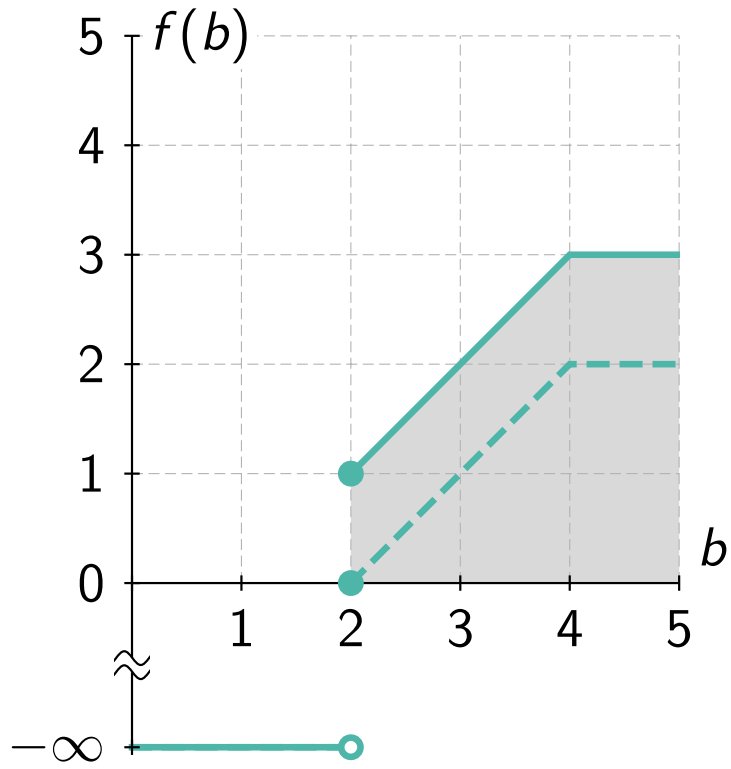


Komplexität von SoC-Profilen

Betrachte beliebiges Paar von Bergknoten $\overline{v}$ und Talknoten $\underline{v}$:

Höchstens ein Tal-Berg-Pfad und ein Berg-Tal-Pfad benutzen $\overline{v}$ und $\underline{v}$ als Berg- bzw. Talknoten





Komplexität von SoC-Profilen

Lemma

Für bel. Knoten $s, t, \underline{v}, \overline{v}$ gilt:
Das SoC-Profil für Start s und Ziel t enthält

- *höchstens einen Tal-Berg-Pfad mit Talknoten $\underline{v}$ und Bergknoten $\overline{v}$*
- *höchstens einen Berg-Tal-Pfad mit Bergknoten $\overline{v}$ und Talknoten $\underline{v}$*

$\Rightarrow$ höchstens $\mathcal{O}(|V|^2)$ Pfade (und somit Stützpunkte) im Profil

Komplexität von SoC-Profilen

Lemma

Für bel. Knoten $s, t, \underline{v}, \overline{v}$ gilt:
Das SoC-Profil für Start s und Ziel t enthält

- *höchstens einen Tal-Berg-Pfad mit Talknoten $\underline{v}$ und Bergknoten $\overline{v}$*
- *höchstens einen Berg-Tal-Pfad mit Bergknoten $\overline{v}$ und Talknoten $\underline{v}$*

$\Rightarrow$ höchstens $\mathcal{O}(|V|^2)$ Pfade (und somit Stützpunkte) im Profil

Man kann sogar zeigen:

Theorem

Die Anzahl der Pfade (und Stützpunkte) eines SoC-Profils liegt in $\mathcal{O}(|V|)$.

$\Rightarrow$ SoC-Profile haben lineare Komplexität (und sind in der Praxis klein)

MLD für energieoptimale Routen

Energieoptimale Routen

Ziel: Gegeben Start s, Ziel t und initialen Ladezustand b_s, finde zulässigen s-t-Pfad mit maximalem SoC an t

Besonderheiten: Energieverbrauch hängt von vielen Parametern ab

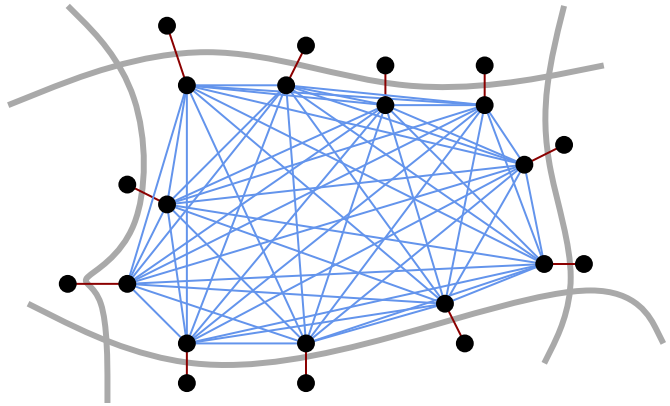
- Temperatur, Wetterbedingungen, Beladung, ...
- Abhängig von Fahrzeugtyp, Fahrstil, ...
- Verkehrslage, ...

$\Rightarrow$ Schnelle Customization wichtig

MLD für energieoptimale Routen

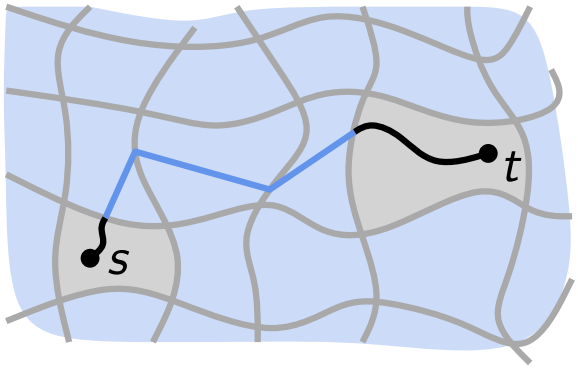
Idee MLD:

- Zerteile den Graphen in mehrere Zellen
- Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay-Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

- Start- und Zielzelle...
- ...plus Overlay-Graph
- (bidirektionaler) Dijkstra

MLD für energieoptimale Routen

1: Metrikunabhängige Vorberechnung

- Metrikunabhängig, keine Anpassung nötig

2: Metrikabhängige Vorberechnung (Customization)

- Cliquenkanten sind Profile
- Lokale Profilsuchen
- Anpassung der Datenstrukturen

3: Queries

- Knotenpotentiale für Stoppkriterium
- Varianten:
  - Unidirektionale Query
  - Bidirektionale Query mit Rückwärts-Profilsuche
  - Bidirektionale Query mit Rückwärtssuche auf Schranken
    (ähnlich wie bei zeitabhängigen Techniken)

Experimente

	CUSTOMIZING		QUERIES	
Algorithm	space [B/n]	time [s]	vertex scans	time [ms]
LC	—	—	4 471 230	709.6
LC (stop. cr.)	0.0	5.20	3 001 014	486.6
Dijkstra π_{v^*}	4.0	3.91	2 361 997	288.0
Dijkstra π_h	4.0	0.69	2 359 140	380.6
Prof. π_h	4.0	0.69	2 904 764	741.2
MLD (LC)	10.5	4.42	3 676	2.7
MLD Uni π_h	14.5	5.12	2 410	1.9
MLD BPE π_h	14.5	5.12	2 266	1.4
MLD BDB π_h	14.5	5.12	2 917	1.1

für EV mit 85kWh Akku (ca. 400 km Reichweite)

Ladestopps

Problemstellung

Bisher:

SoC-Query: Gegeben Start s, Ziel t und initialen Ladezustand b_s, finde zulässigen s-t-Pfad mit maximalem SoC an t

Immer noch sinnvoll, wenn wir Ladestopps erlauben?

Problemstellung

Bisher:

SoC-Query: Gegeben Start s, Ziel t und initialen Ladezustand b_s, finde zulässigen s-t-Pfad mit maximalem SoC an t

Immer noch sinnvoll, wenn wir Ladestopps erlauben?

- Finde Ladestation in der Nähe vom Ziel
- Route (beliebig?) zu dieser Station, lade voll auf, fahre zum Ziel

Problemstellung

Bisher:

SoC-Query: Gegeben Start s, Ziel t und initialen Ladezustand b_s, finde zulässigen s-t-Pfad mit maximalem SoC an t

Immer noch sinnvoll, wenn wir Ladestopps erlauben?

- Finde Ladestation in der Nähe vom Ziel
- Route (beliebig?) zu dieser Station, lade voll auf, fahre zum Ziel

Stattdessen:

SoC-Query: Gegeben Start s, Ziel t und initialen Ladezustand b_s, finde zulässigen s-t-Pfad, der Energieverbrauch (inklusive geladener Energie) minimiert

Analog: Maximiere $b_t - r_t$, wobei

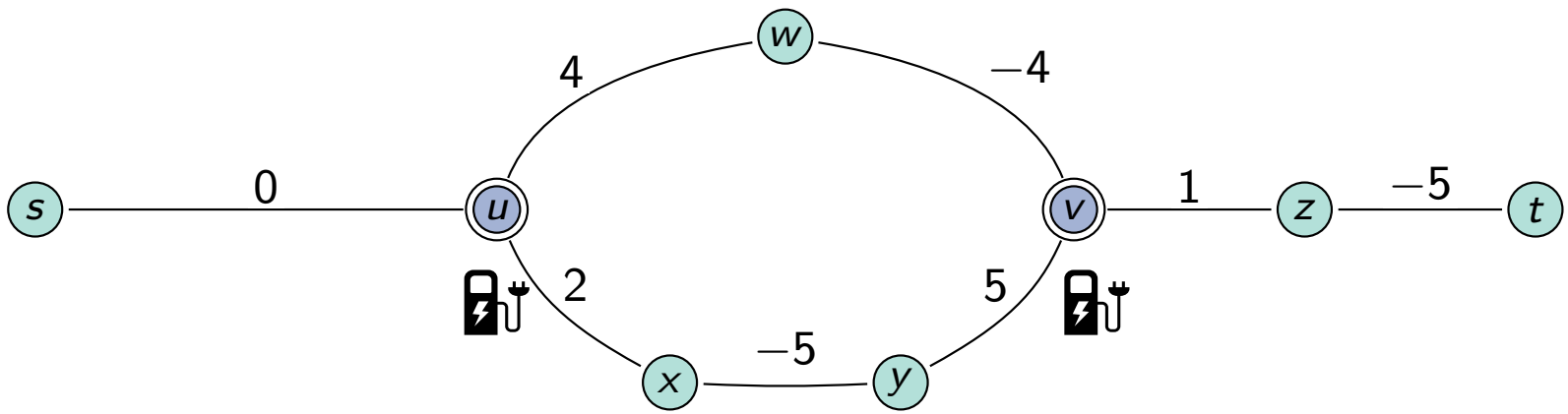
- b_t: SoC an t
- r_t: Geladene Energie, um t zu erreichen

Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$



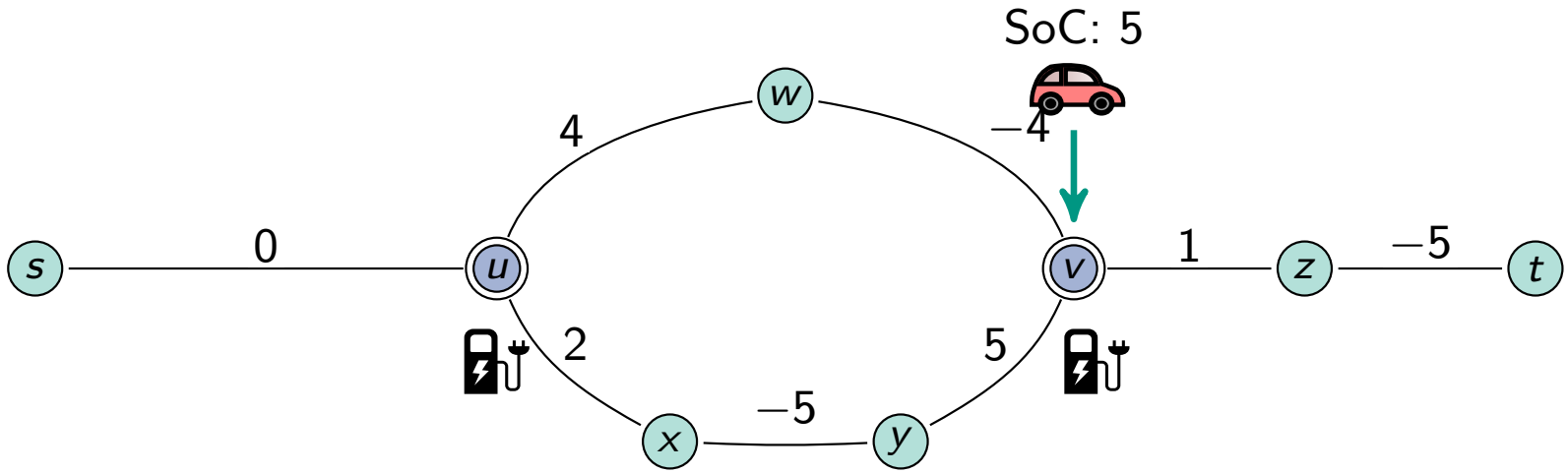
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_v - r_v = 5 - 0 = 5$



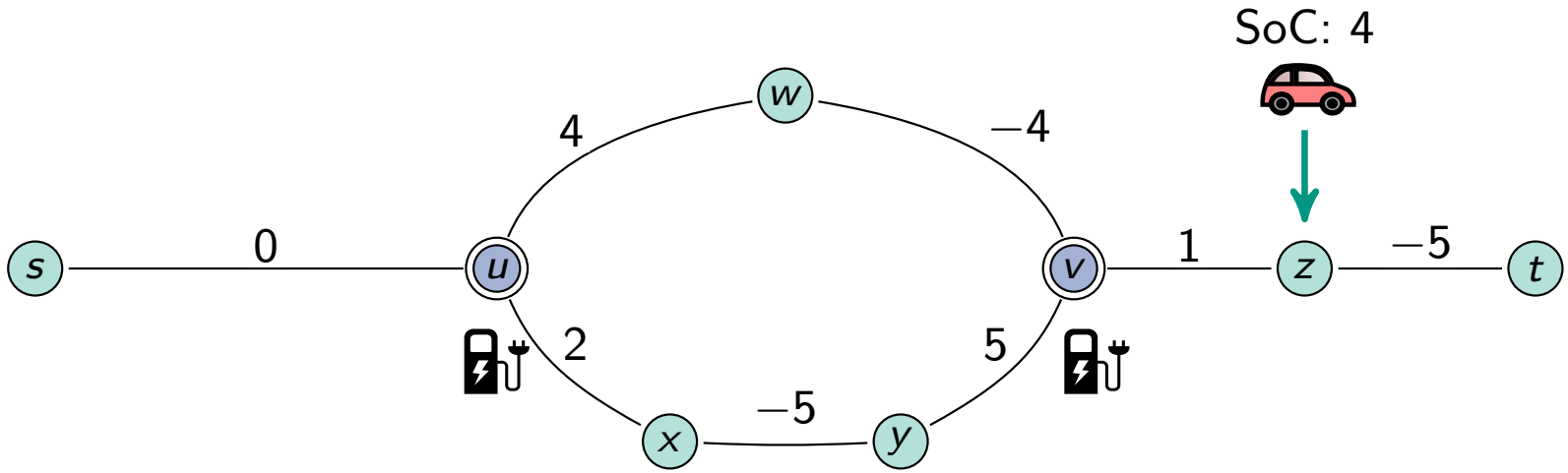
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_z - r_z = 4 - 0 = 4$



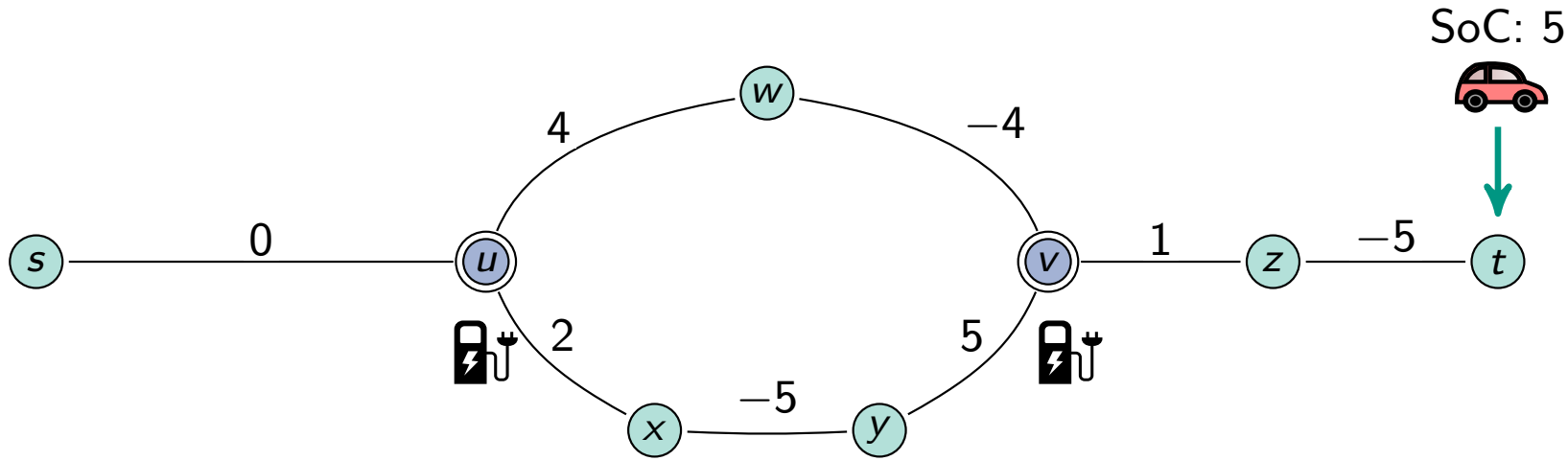
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_t - r_t = 5 - 0 = 5$



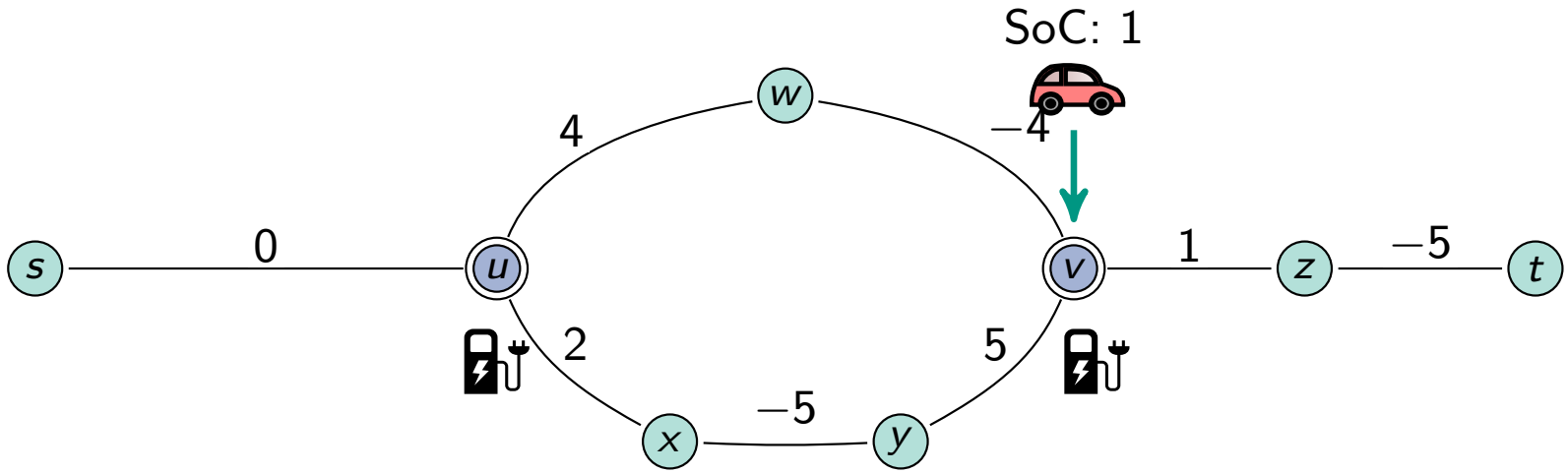
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_v - r_v = 1 - 0 = 1$



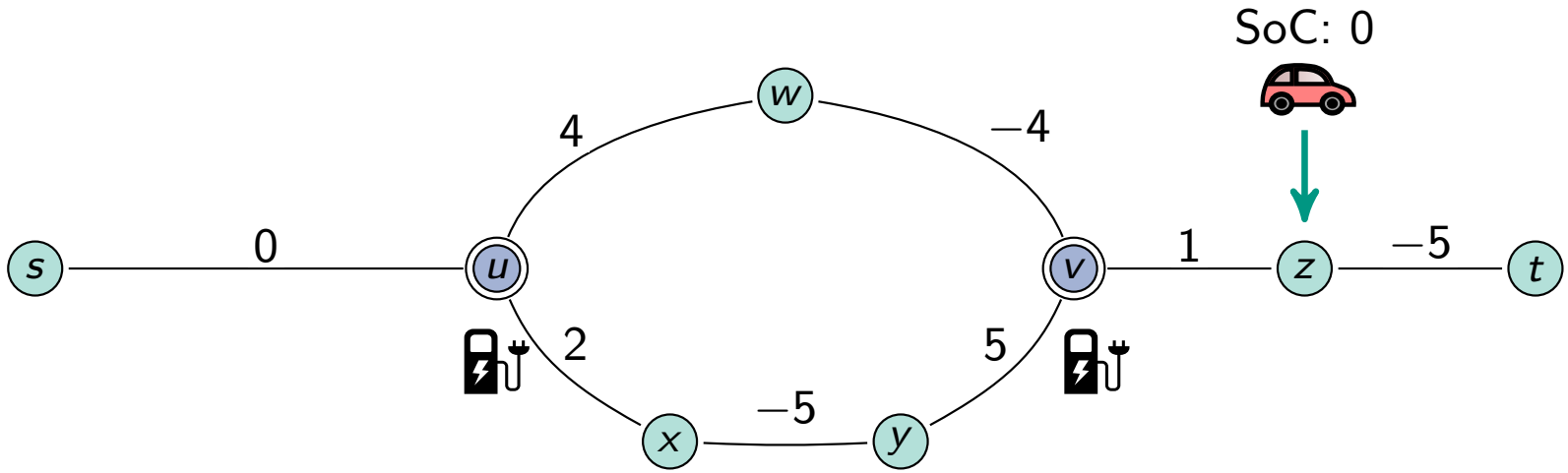
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_z - r_z = 0 - 0 = 0$



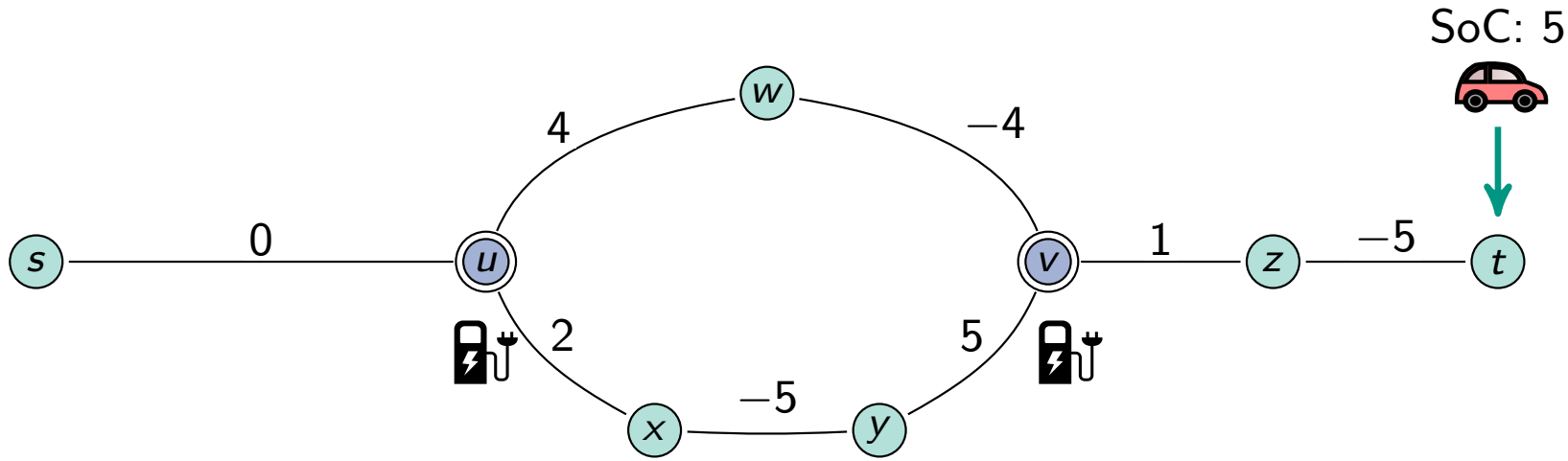
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_t - r_t = 5 - 0 = 5$



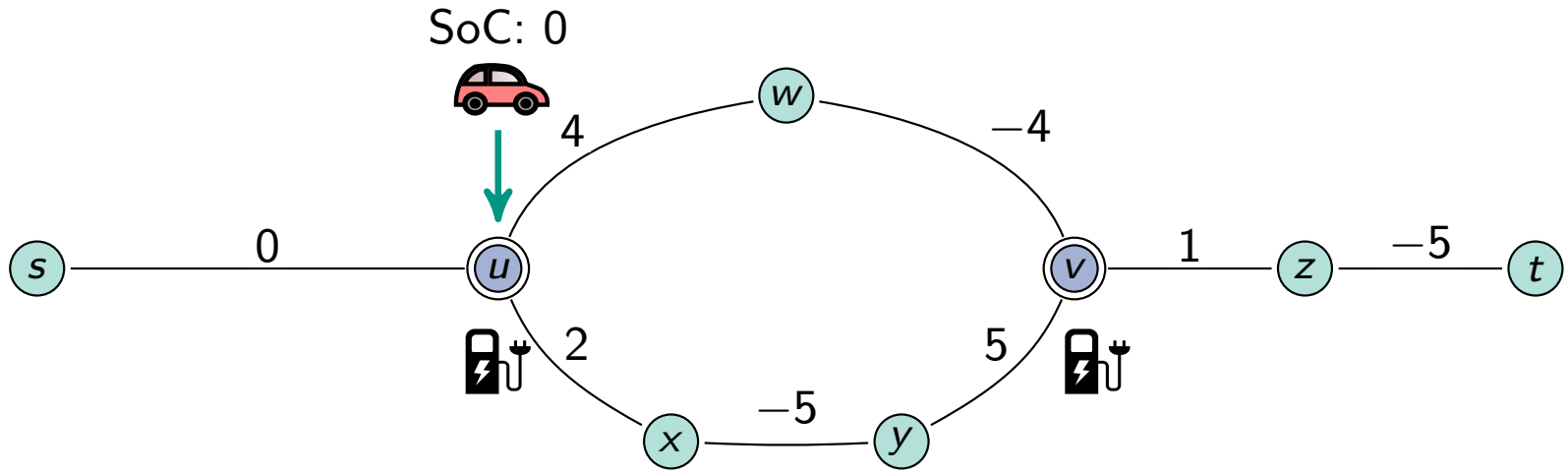
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_u - r_u = 0 - 0 = 0$



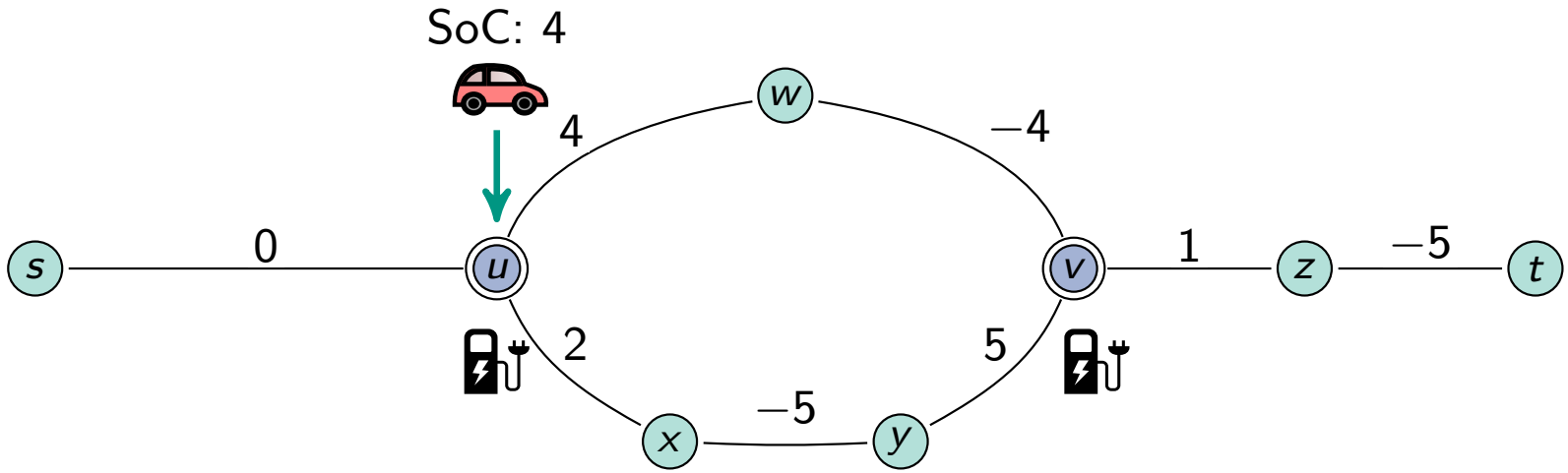
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_u - r_u = 4 - 4 = 0$



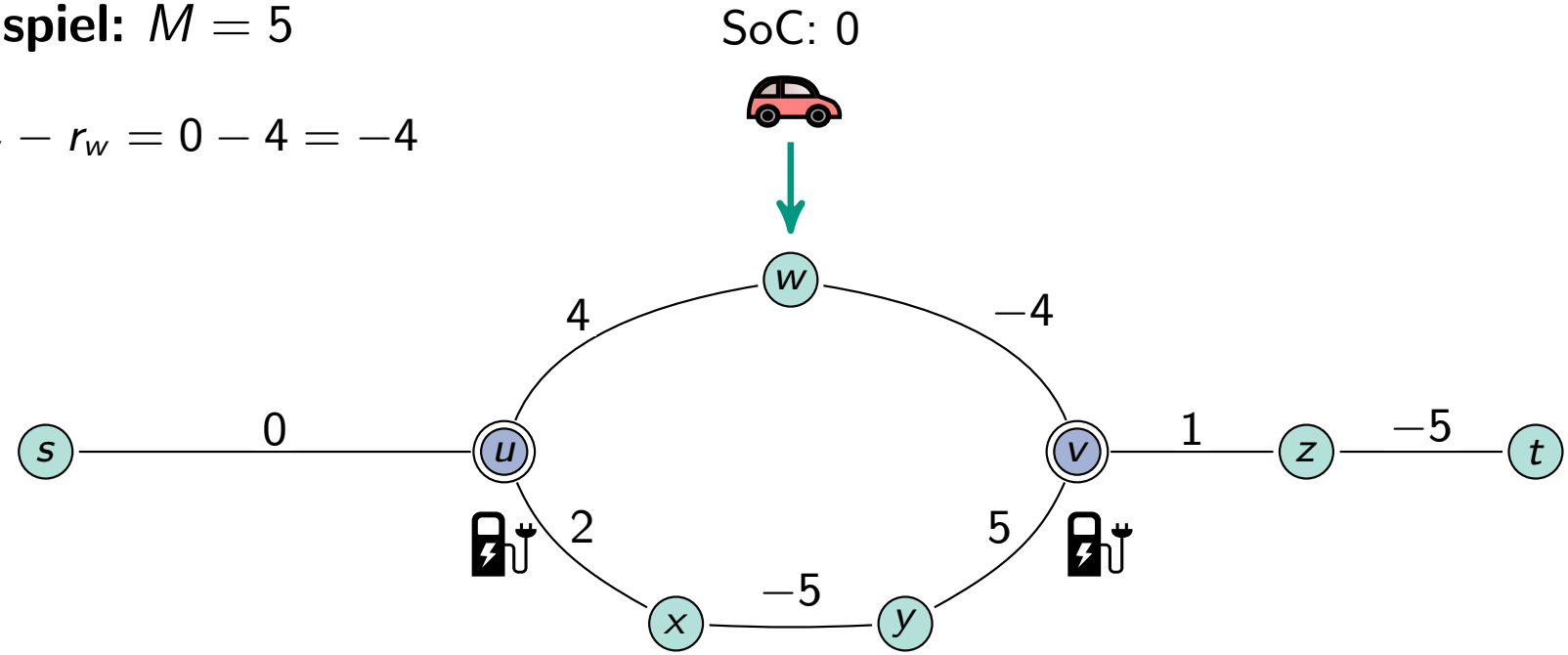
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_w - r_w = 0 - 4 = -4$



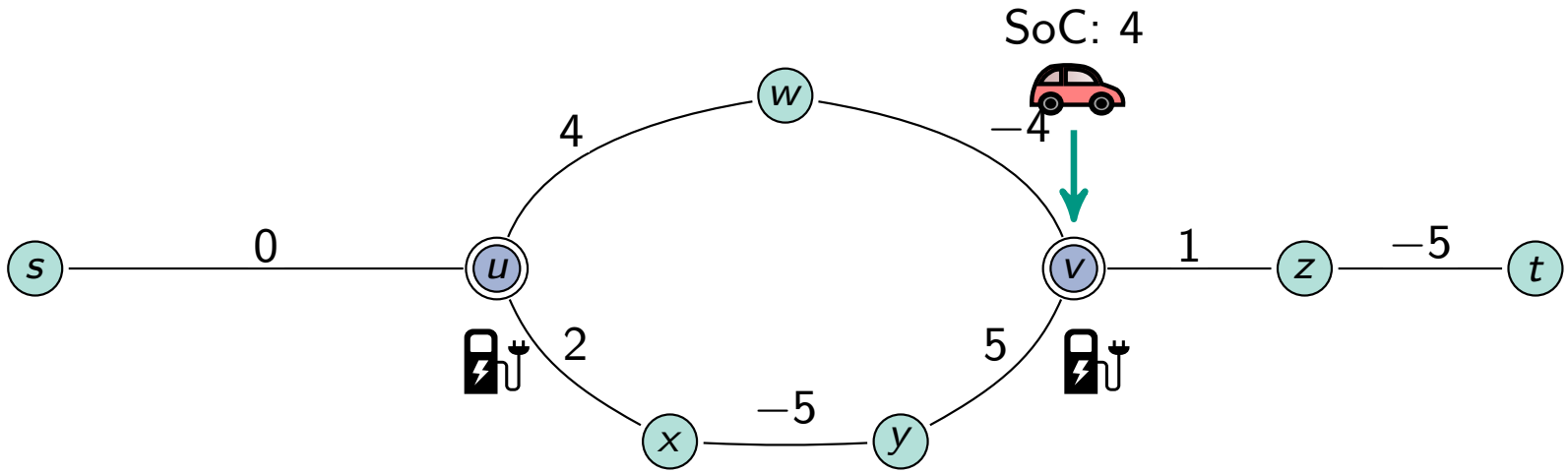
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_v - r_v = 4 - 4 = 0$



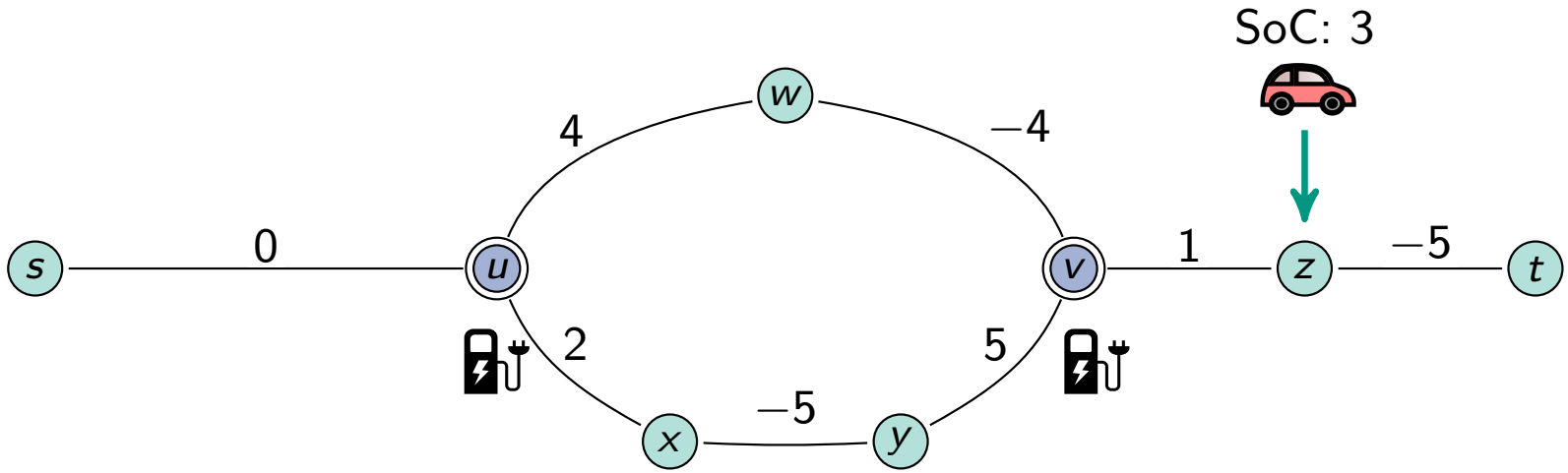
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_z - r_z = 3 - 4 = -1$



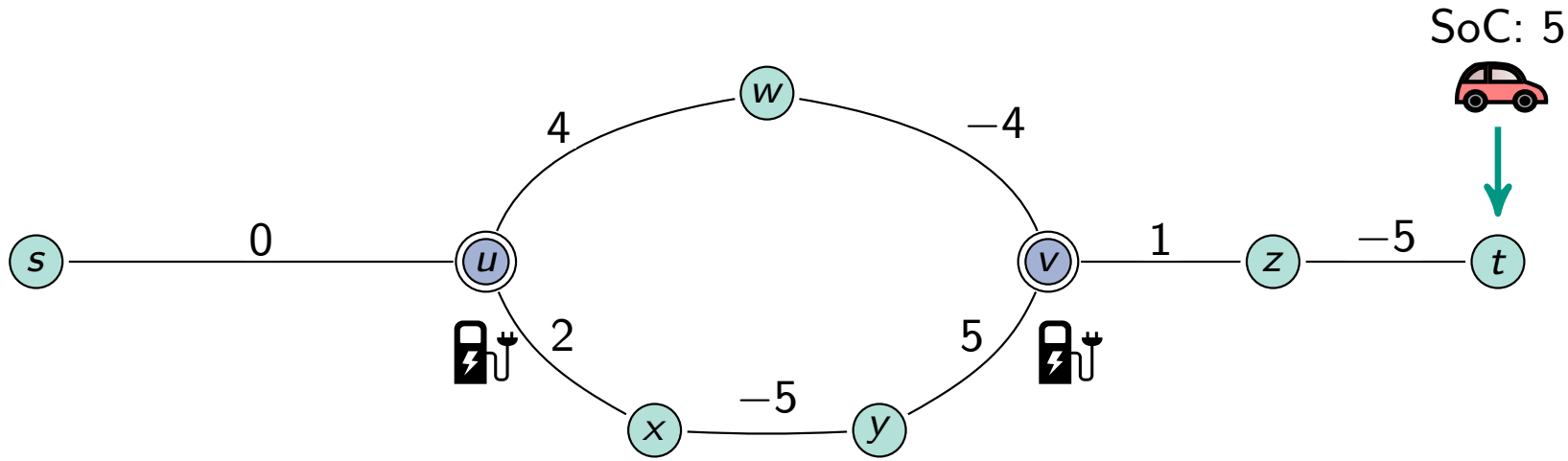
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_t - r_t = 5 - 4 = 1$



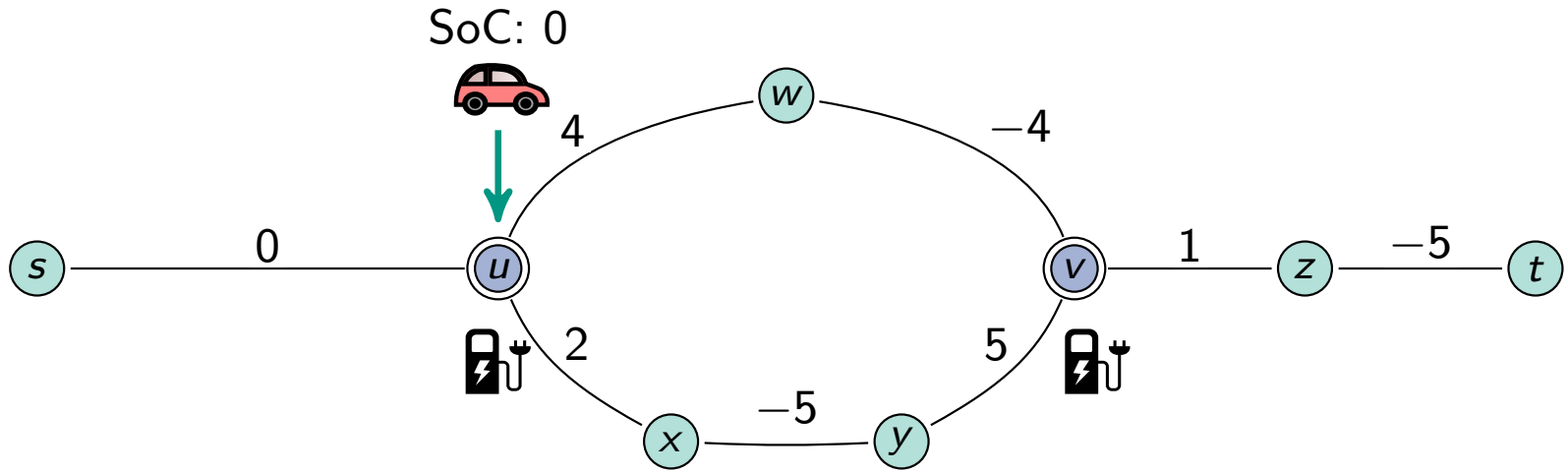
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_u - r_u = 0 - 0 = 0$



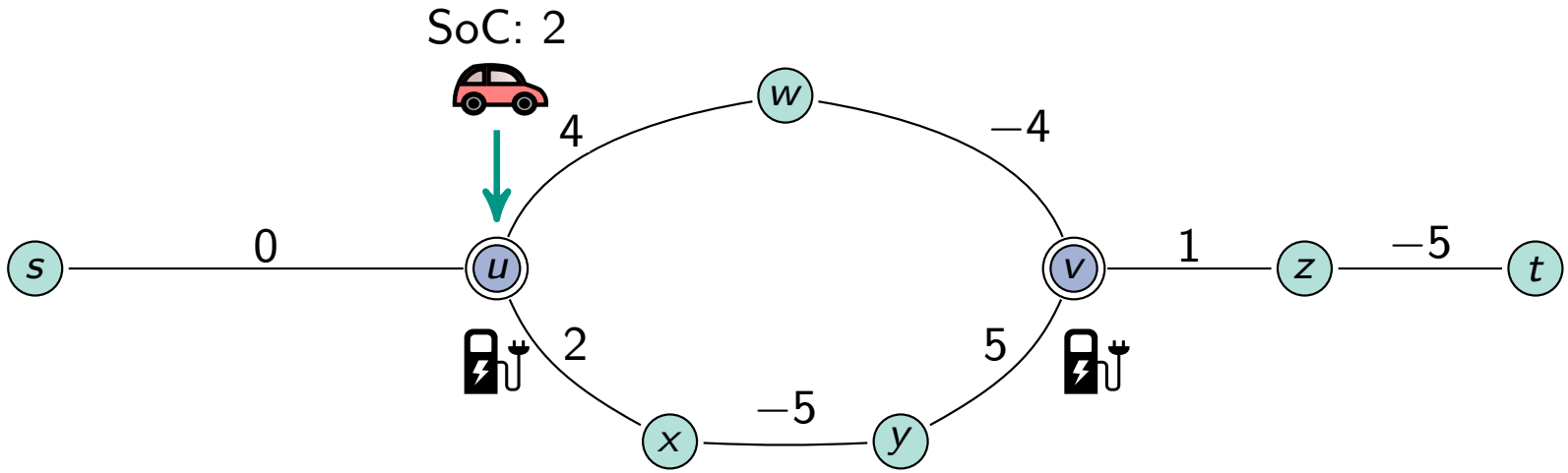
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_u - r_u = 2 - 2 = 0$



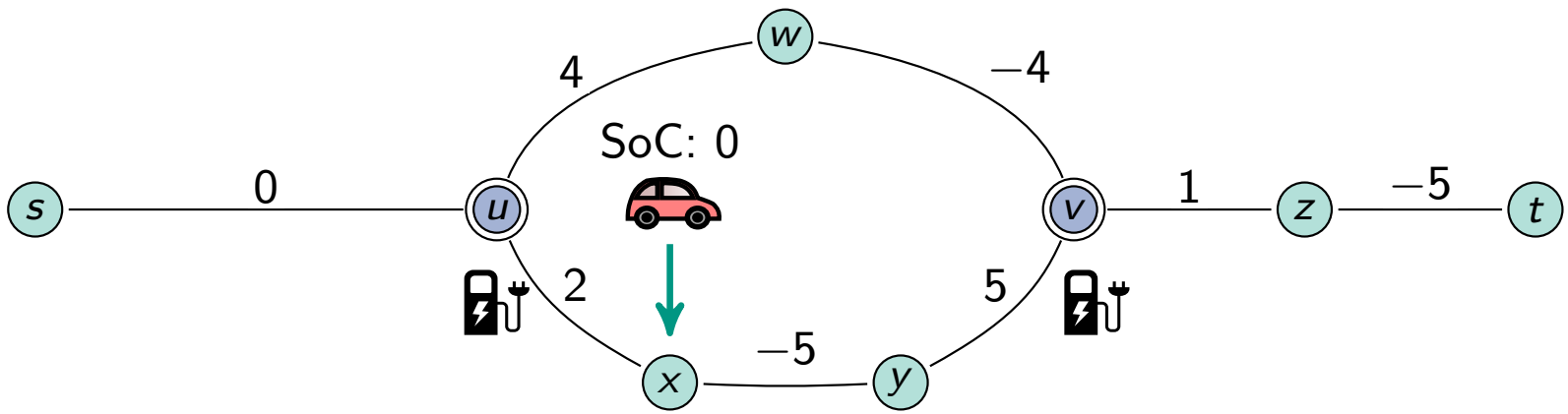
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_x - r_x = 0 - 2 = -2$



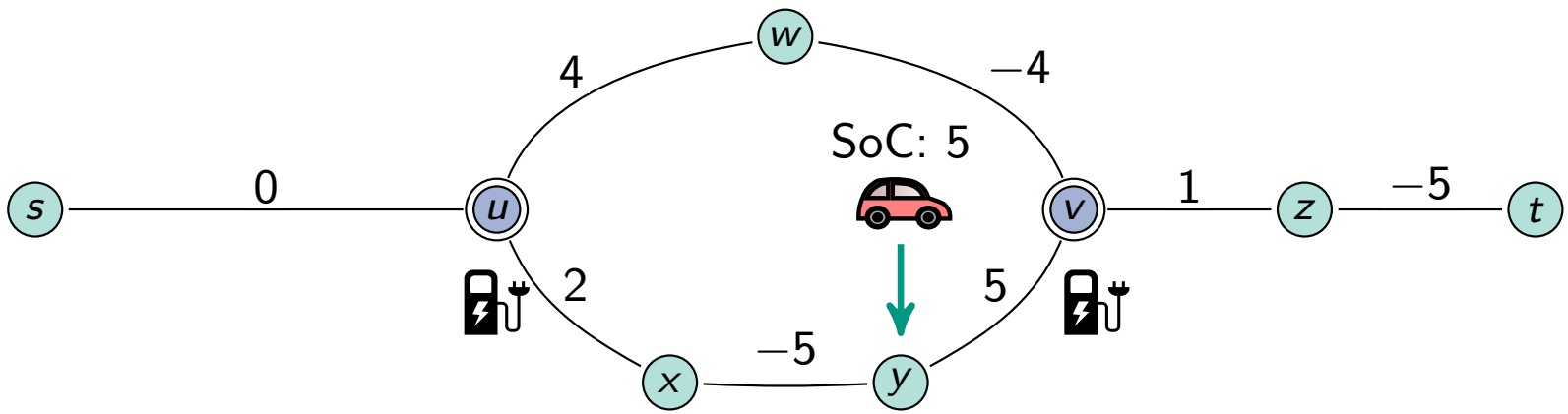
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_y - r_y = 5 - 2 = 3$



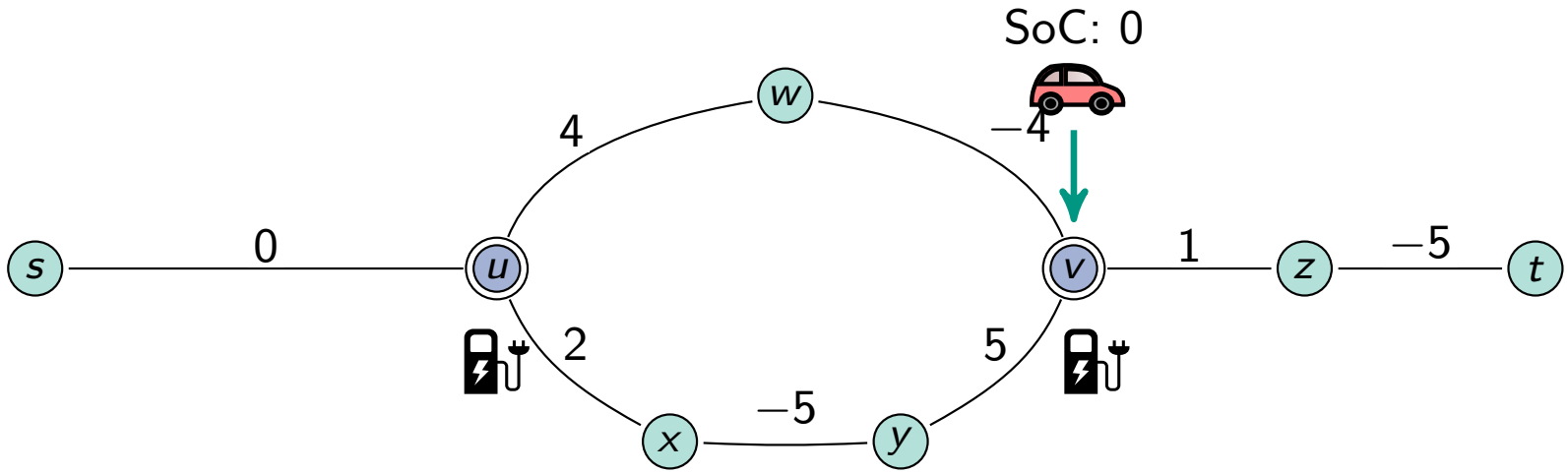
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_v - r_v = 0 - 2 = -2$



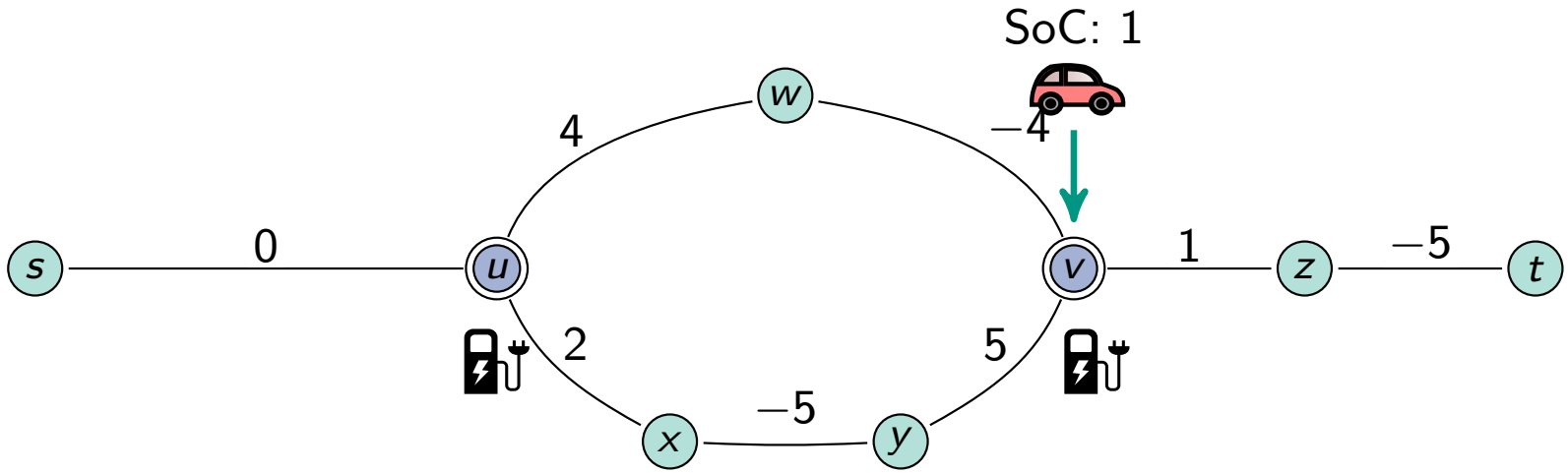
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_v - r_v = 1 - 3 = -2$



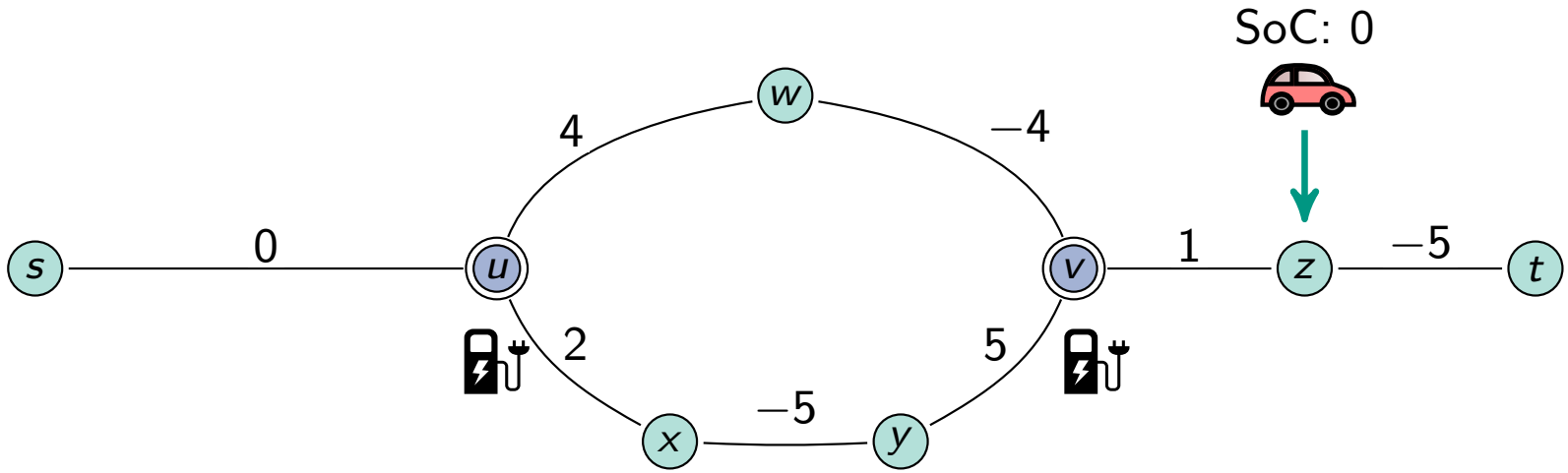
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_z - r_z = 0 - 3 = -3$



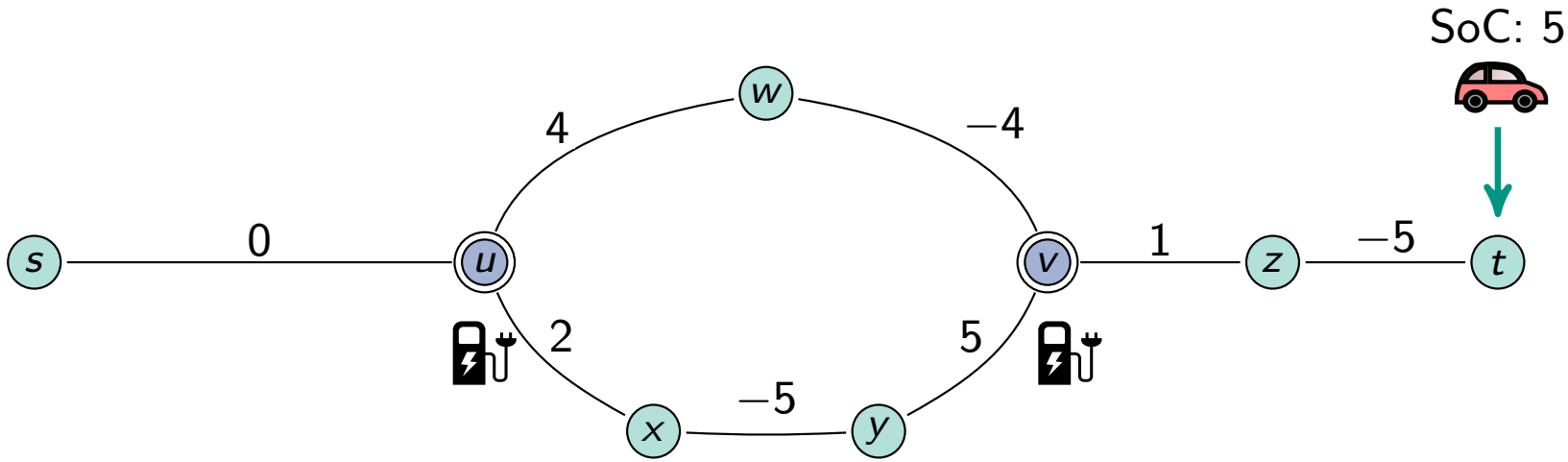
Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$

$b_t - r_t = 5 - 3 = 2$

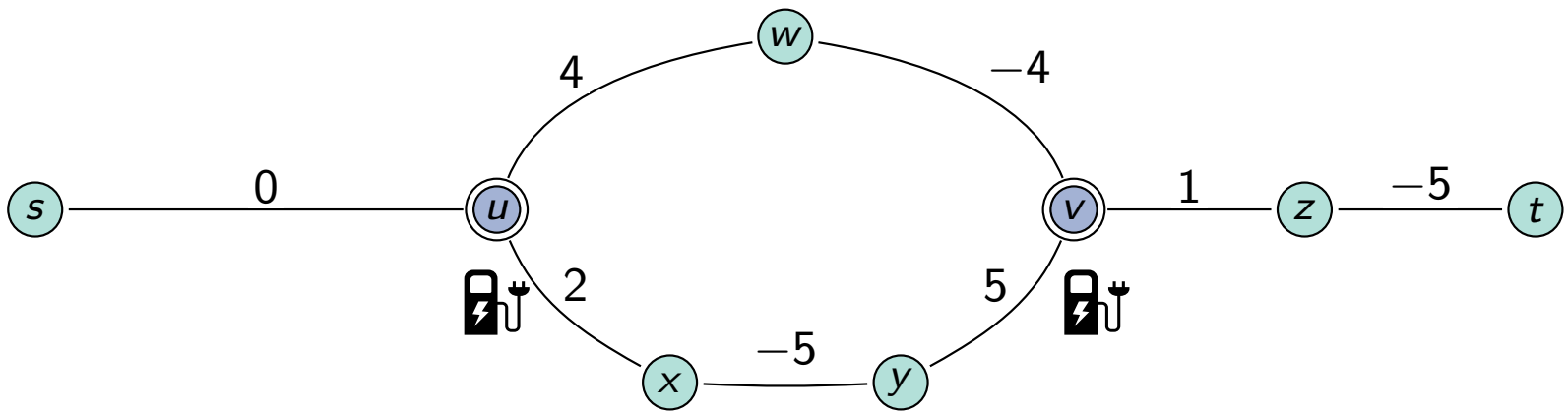


Planung von Ladestopps

Zwei Probleme:

- Volles Aufladen ist ggf. nicht optimal
- Subpfade von optimalen Pfaden ggf. selbst nicht optimal

Beispiel: $M = 5$



Für Zielknoten v**:** u–v-Pfad über w immer die optimale Wahl
Für Zielknoten t**:** u–v-Pfad über x und y ggf. besser

Polynomialzeitalgorithmus

Beobachtung: Teilpfade zwischen Ladestationen (oder von Station zum Ziel) müssen für mind. 1 initialen SoC b_s energieoptimal sein

- Berechne alle energieoptimalen Pfade zwischen Ladestationen (+Ziel)
  (linear pro Paar von Stationen)
- Berechne für jeden dieser Pfade den minimal nötigen SoC zum Befahren
- Mehr als das muss nicht geladen werden
  (man kann an der nächsten Station weiter laden bzw. ist am Ziel)

$\Rightarrow$ pro Ladestation polynomiell viele „Abfahrts-SoCs“

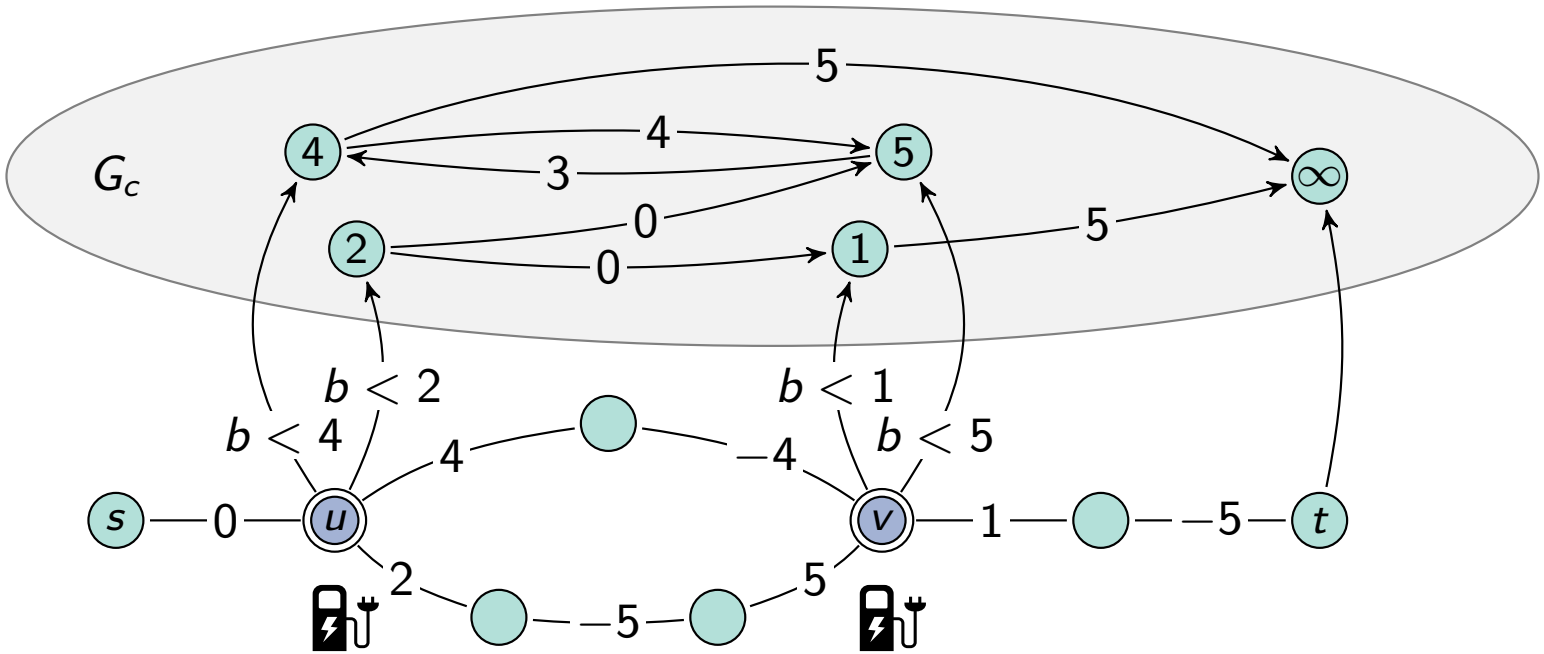
$\Rightarrow$ Konstruiere polynomiellen Suchgraph:

- Ein Knoten pro Station und Abfahrts-SoC
- Speichere Ankunfts-SoC an der Kante

Suchgraph

Für jeden Pfad im Profil zwischen Ladestationen (+Ziel):

- Berechne Ankunfts-SoC (Wert an der Kante)
- Füge Ladeknoten hinzu (Mindest-Ladewert im Knoten)

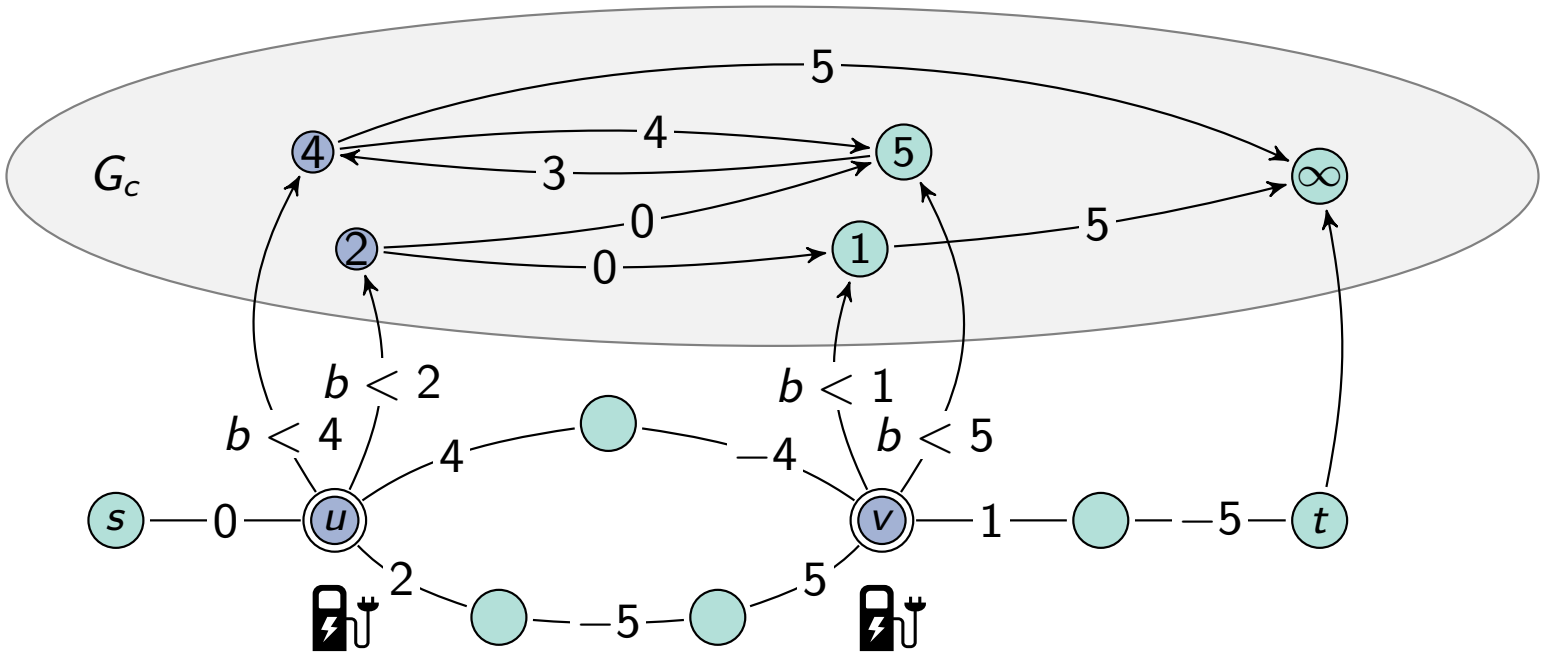


$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

Suchgraph

Für jeden Pfad im Profil zwischen Ladestationen (+Ziel):

- Berechne Ankunfts-SoC (Wert an der Kante)
- Füge Ladeknoten hinzu (Mindest-Ladewert im Knoten)

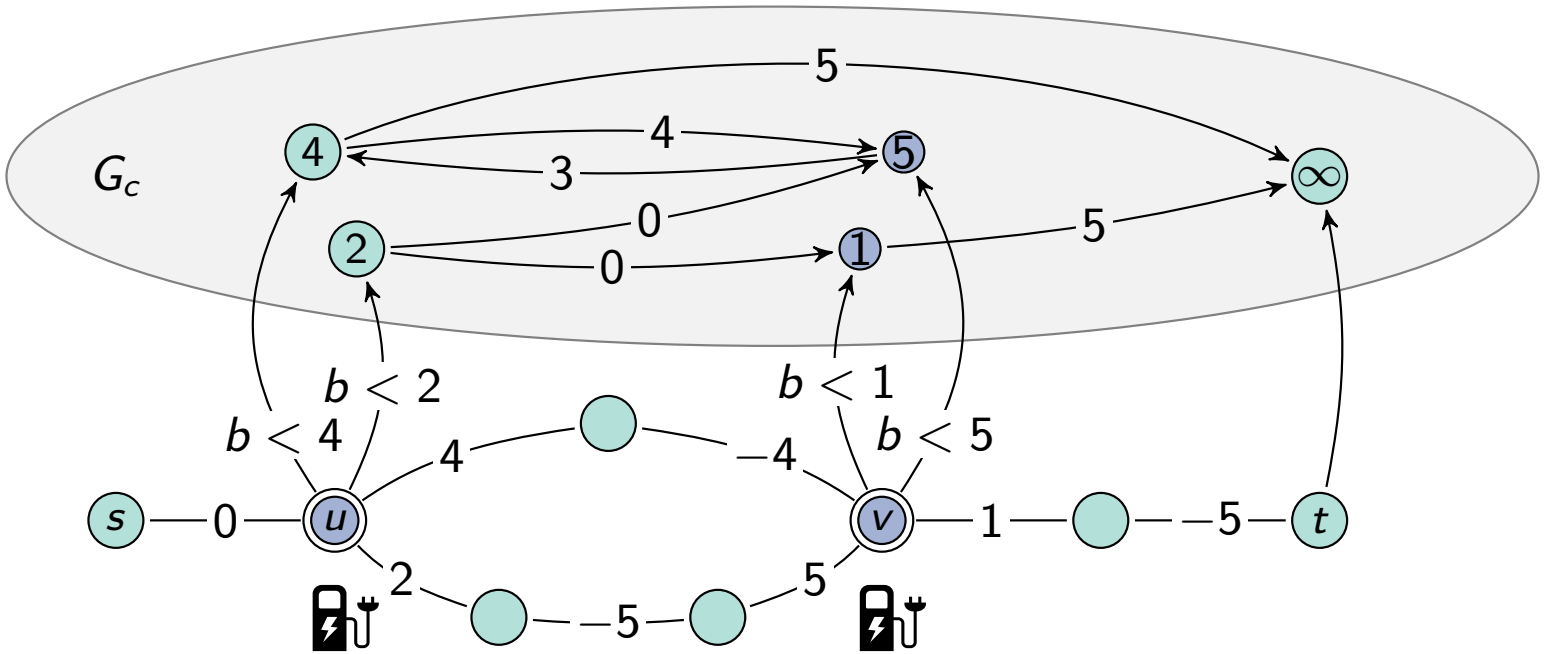


$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

Suchgraph

Für jeden Pfad im Profil zwischen Ladestationen (+Ziel):

- Berechne Ankunfts-SoC (Wert an der Kante)
- Füge Ladeknoten hinzu (Mindest-Ladewert im Knoten)

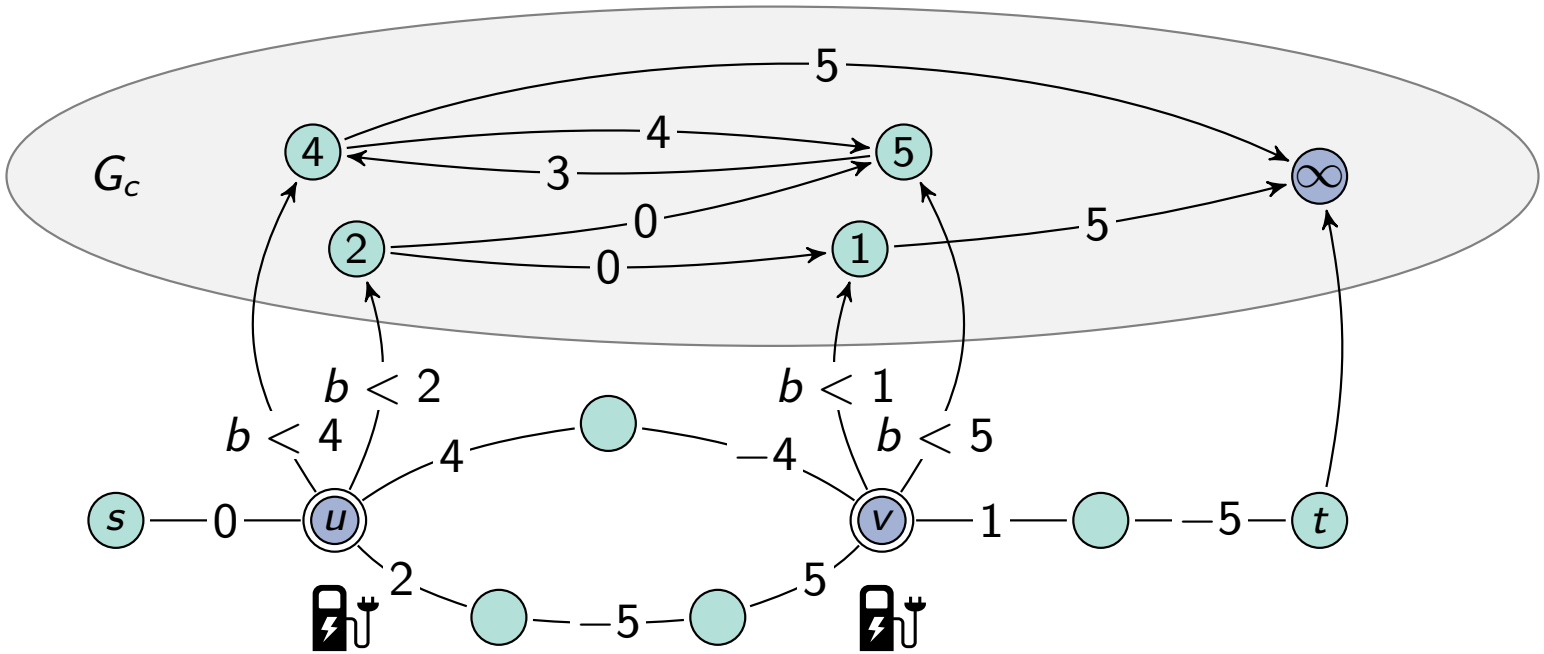


$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

Suchgraph

Für jeden Pfad im Profil zwischen Ladestationen (+Ziel):

- Berechne Ankunfts-SoC (Wert an der Kante)
- Füge Ladeknoten hinzu (Mindest-Ladewert im Knoten)

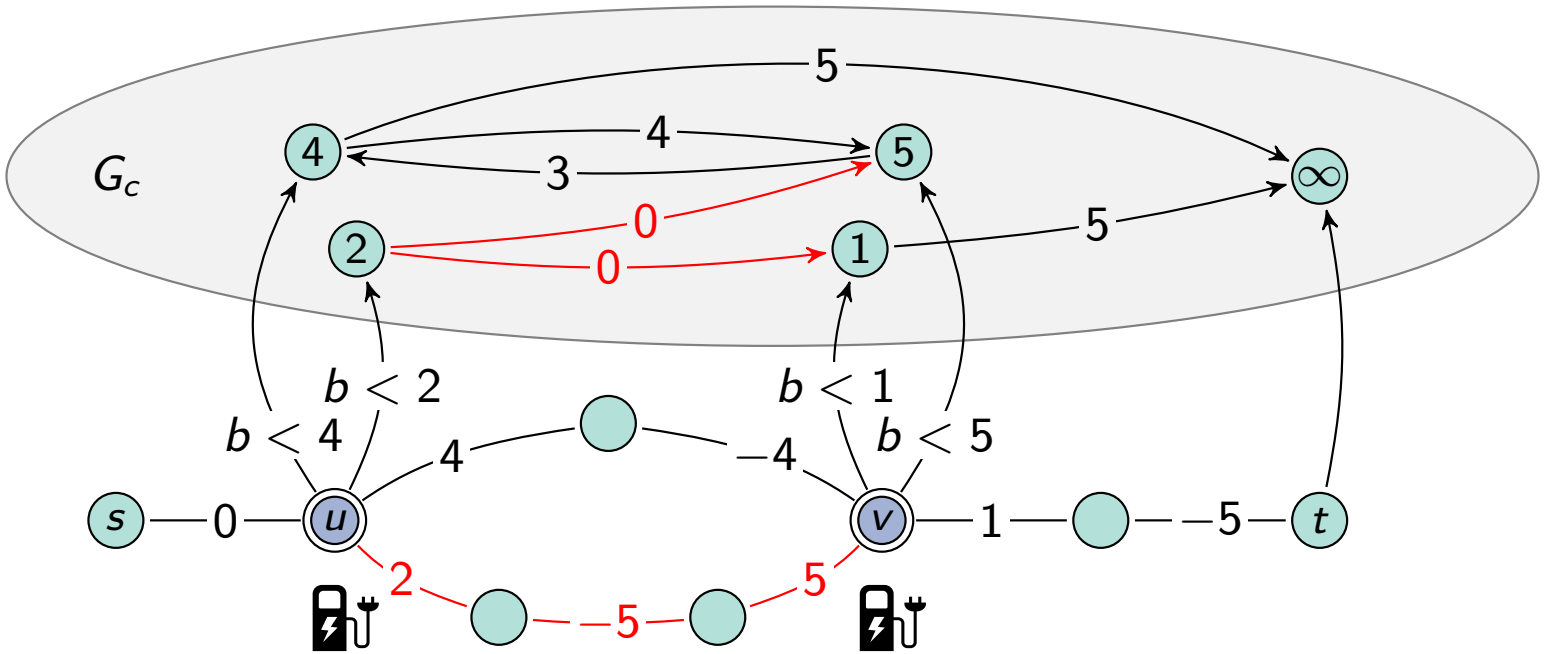


$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

Suchgraph

Für jeden Pfad im Profil zwischen Ladestationen (+Ziel):

- Berechne Ankunfts-SoC (Wert an der Kante)
- Füge Ladeknoten hinzu (Mindest-Ladewert im Knoten)

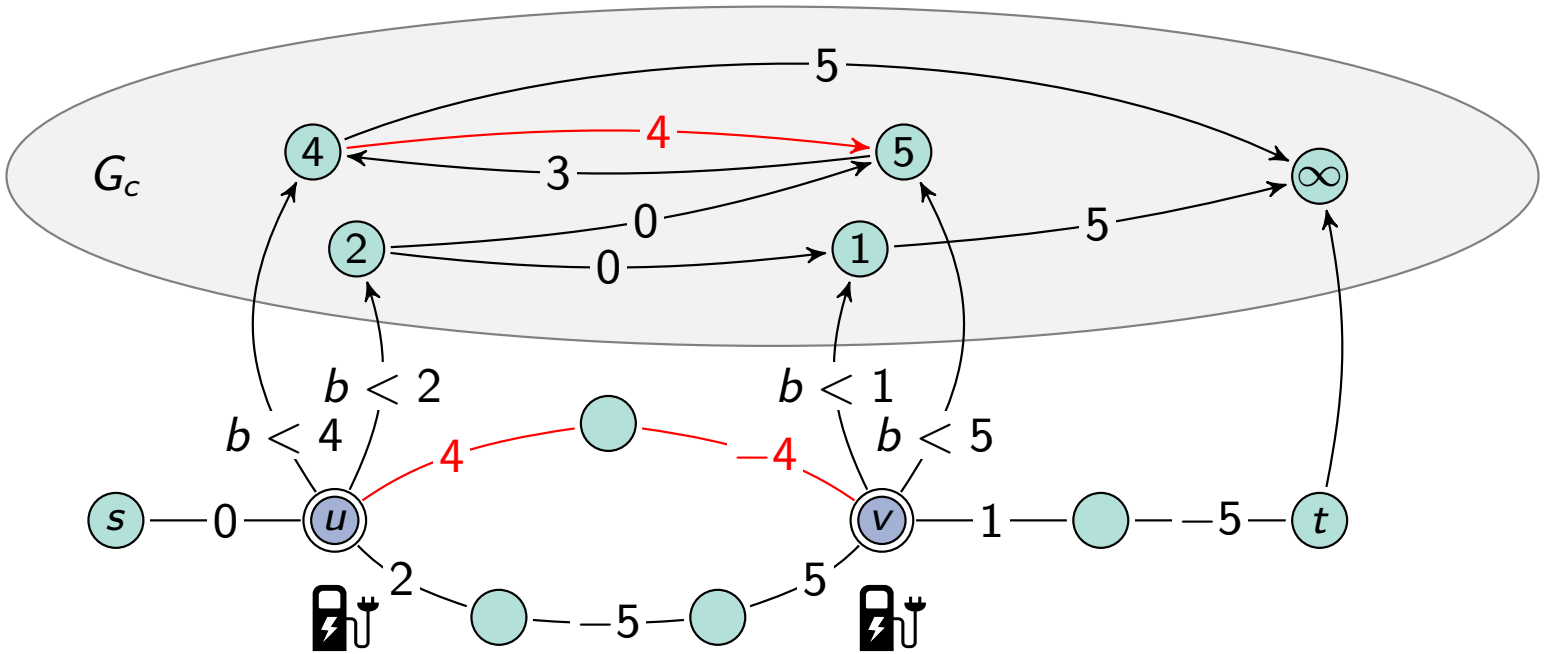


$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

Suchgraph

Für jeden Pfad im Profil zwischen Ladestationen (+Ziel):

- Berechne Ankunfts-SoC (Wert an der Kante)
- Füge Ladeknoten hinzu (Mindest-Ladewert im Knoten)

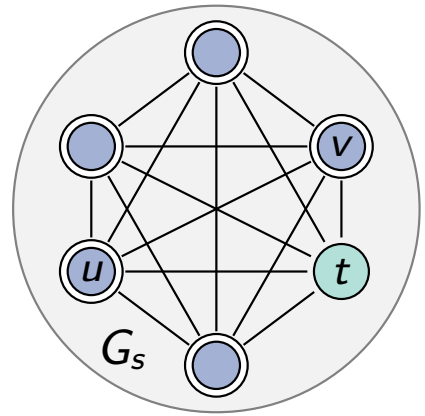


$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

Heuristische Implementierung

Suchgraph G_s:

- Knoten: Alle Ladestationen und Zielknoten t
- Clique
- An den Kanten: SoC-Profile



Traversieren von Kanten zwischen Stationen u und v:

- Lade genug Energie, um Verbrauch auf s–v-Pfad zu minimieren
- Nicht beweisbar optimal, aber (fast) immer optimal in der Praxis

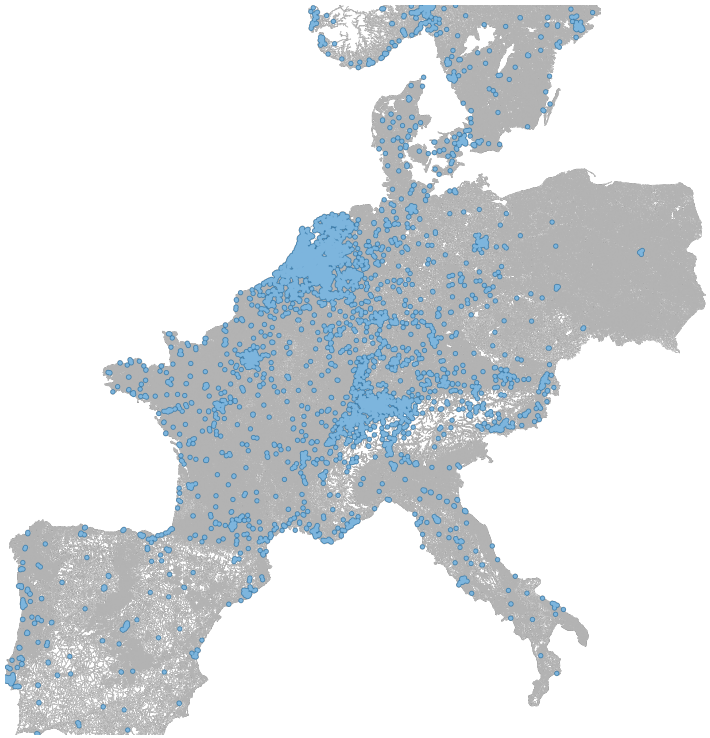
Speedup-Techniken:

- Contraction Hierarchies (CH): Partielle CH; kontrahiere keine Ladestationen
- A*-Suche: Potential-basierte Rückwärtssuche
  (ähnlich wie bei multikriterieller Suche)

Experimente

Input:

- Straßennetzwerk Europa
  (ähnlich wie DIMACS)
- 22.2M Knoten, 51.1M Kanten
- Verbrauchsdaten aus
  PHEM-Emissionsmodell (TU Graz)
- Höhendaten aus SRTM
  (Shuttle Radar Topography Mission)
- Ladestationen von ChargeMap:
  13 810 in Europa



Performance

Fahrzeugtypen:

- Peugeot iOn, 16 kWh (100–150 km)
- Künstliches Modell, 85 kWh (400–500 km, ähnl. wie Tesla)

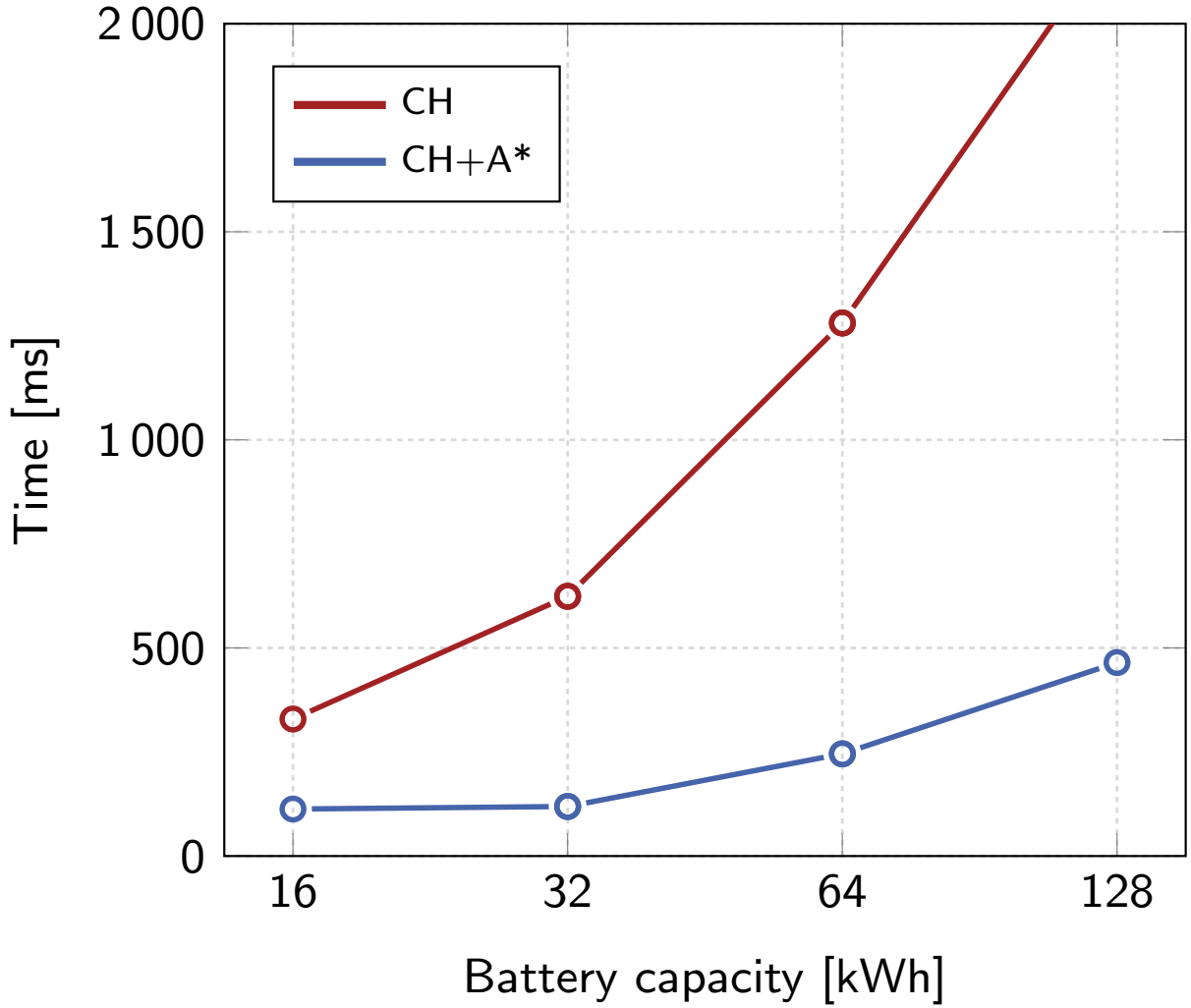
Techniques			Peugeot iOn			Artificial		
G_S	CH	A*	Prepr. [s]	# V. Sc.	Q. [ms]	Prepr. [s]	# V. Sc.	Q. [ms]
○	○	○	—	8 895k	20 161	—	11 034k	32 929
●	○	○	1 487	760k	710	15 062	7 754k	6 286
●	●	○	2 860	8k	310	3 246	20k	1 282
●	●	●	2 860	4k	128	3 246	10k	298

Ladestationsverteilungen

Algorithmus: G_s + CH + A*
Subgraph: Deutschland (4.7M vertices, 10.8M edges)

		Prepr.		Queries		
Scenario	$\|S\|$	T. [s]	$\|E_S\|$	# V. Sc.	# E. Sc.	T. [ms]
ChargeMap	1 966	549	539k	5k	126k	4.22
random-0.01	469	487	22k	2k	50k	1.30
random-0.1	4 692	583	2 263k	9k	224k	7.97
random-1.0	46 920	965	227 514k	61k	1 829k	73.46

Batteriegröße



Electric Vehicle Routing

Bisher: Energieoptimale Routen

- Energiesparendes Fahren
- Wir finden einen zulässigen Pfad, falls dieser existiert

Problem: Wir versuchen Energie zu sparen, selbst wenn:

- die Strecke sehr kurz ist
- der Akkustand mehr als ausreichend für die Strecke ist

Electric Vehicle Routing

Bisher: Energieoptimale Routen

- Energiesparendes Fahren
- Wir finden einen zulässigen Pfad, falls dieser existiert

Problem: Wir versuchen Energie zu sparen, selbst wenn:

- die Strecke sehr kurz ist
- der Akkustand mehr als ausreichend für die Strecke ist

Alternativen?

Electric Vehicle Routing

Bisher: Energieoptimale Routen

- Energiesparendes Fahren
- Wir finden einen zulässigen Pfad, falls dieser existiert

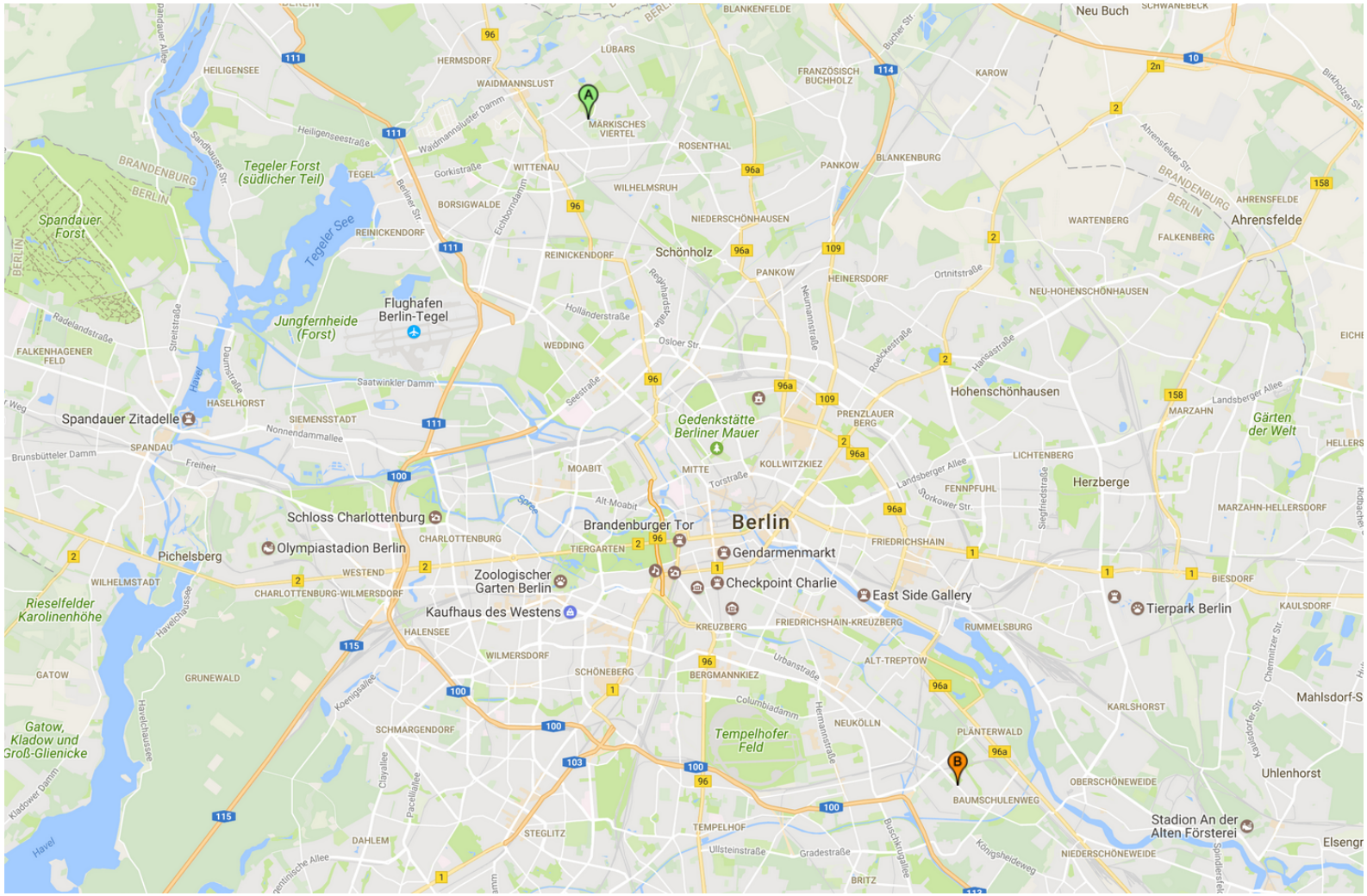
Problem: Wir versuchen Energie zu sparen, selbst wenn:

- die Strecke sehr kurz ist
- der Akkustand mehr als ausreichend für die Strecke ist

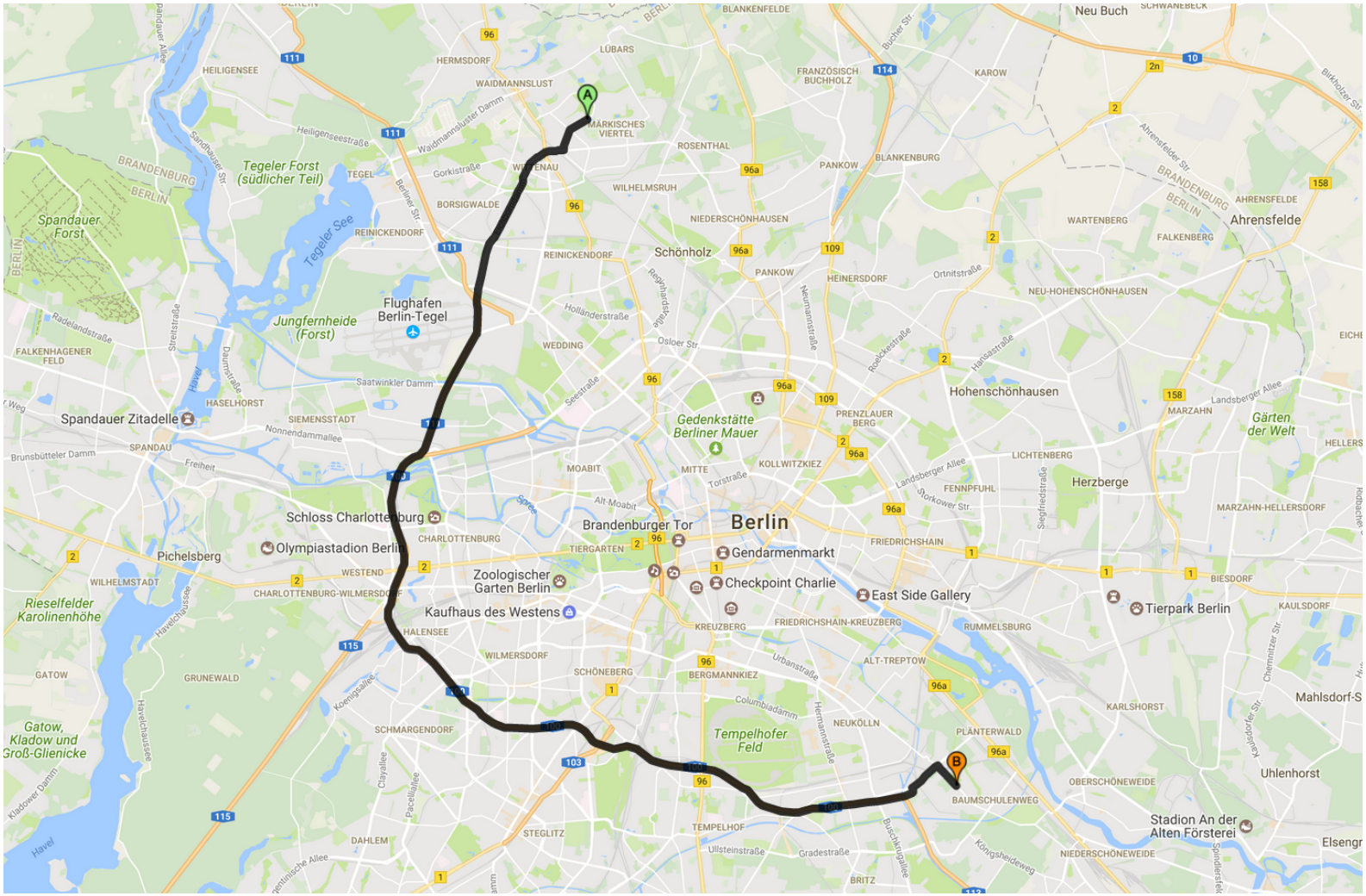
Alternativen?

- Berechne schnellste Route. Überprüfe danach, ob SoC ausreichend
- Schnellste Route mit Energieverbrauch als Nebenbedingung

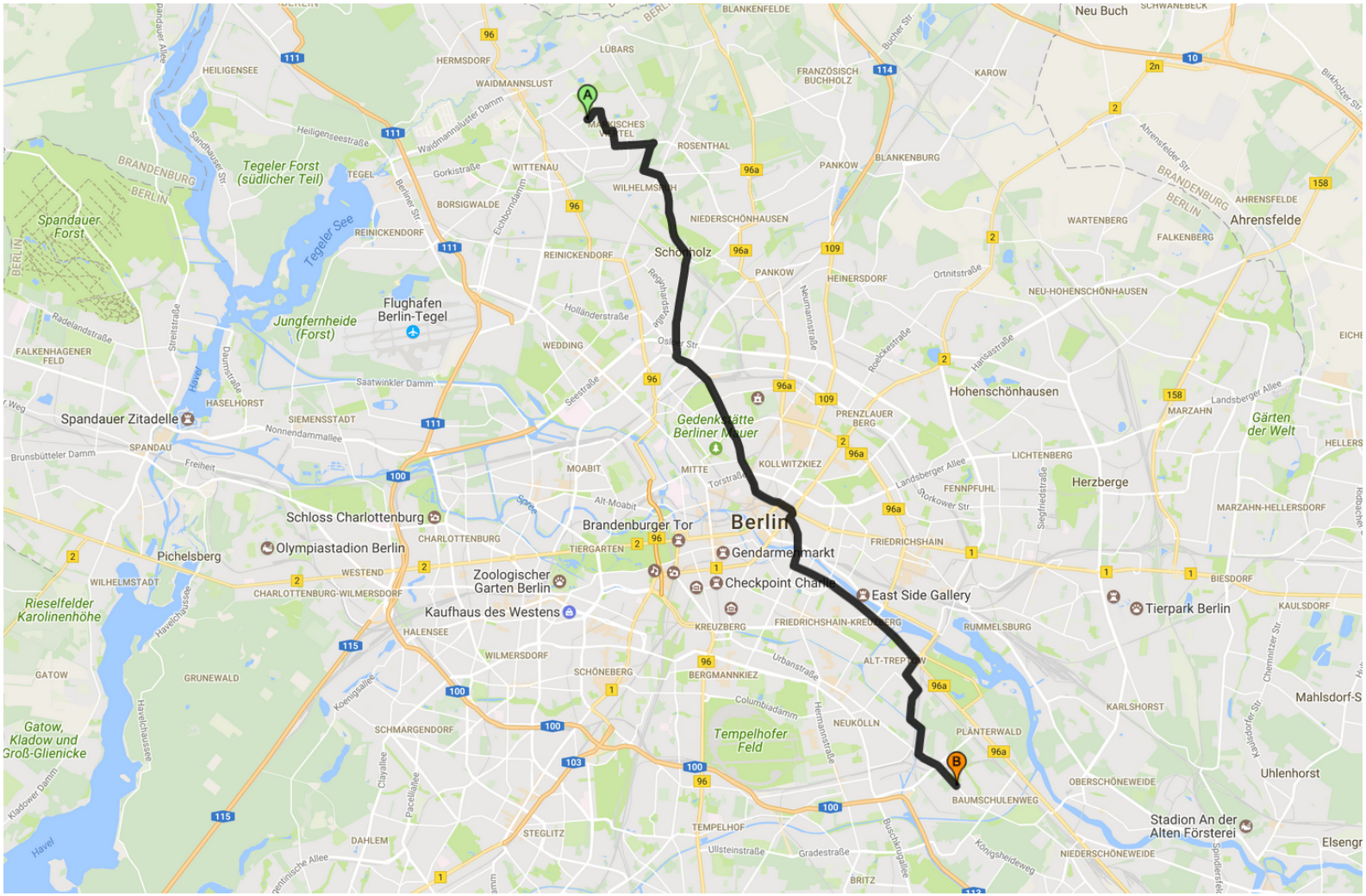
Beispielrouten



Beispielrouten



Beispielrouten



Metrikvergleich

Optimierte Metrik	Unerreichbar	Extra Energie	Extra Zeit/Dist.
Fahrzeit	60 %	62 %	63 %
Distanz	25 %	15 %	4 %

Fazit:

- Energie explizit optimieren zahlt sich aus
- Kürzeste Wege energieeffizienter als schnellste

Aber:

- Fahrzeit viel höher auf energie-optimalen Wegen
- Nur eine Metrik optimieren ist nicht zufriedenstellend

Metrikvergleich

Optimierte Metrik	Unerreichbar	Extra Energie	Extra Zeit/Dist.
Fahrzeit	60 %	62 %	63 %
Distanz	25 %	15 %	4 %

Fazit:

- Energie explizit optimieren zahlt sich aus
- Kürzeste Wege energieeffizienter als schnellste

Aber:

- Fahrzeit viel höher auf energie-optimalen Wegen
- Nur eine Metrik optimieren ist nicht zufriedenstellend

$\Rightarrow$ Finde schnellste Route mit Energieverbrauch als Nebenbedingung

Schnellste fahrbahre Route

Ziel:

- Finde schnellste Route mit Energieverbrauch als Nebenbedingung
- Zwei Metriken auf den Kanten: Fahrzeit und Energieverbrauch
- Optimiere die Fahrzeit und beschränke den Energieverbrauch
- Erweiterung des CSP-Problems

Wdh: Constrained Shortest Path

Definition: Constrained Shortest Path Problem

Gegeben: $G = (V, E)$, Länge $\ell\colon E \to \mathbb{N}_0$, Gewicht $\omega\colon E \to \mathbb{N}_0$, Start und Ziel $s, t \in V$ sowie Schranken $L, W \in \mathbb{N}_0$

Problem: Existiert ein einfacher Pfad P von s nach t in G, für den $\ell(P) \leq L$ und $\omega(P) \leq W$ gelten?

Anmerkung: Das entsprechende Optimierungsproblem lautet:

- Finde einen s-t-Pfad P mit minimalem $\ell(P)$ und $\omega(P) \leq W$

Theorem

Constrained Shortest Path Problem ist (schwach) $\mathcal{NP}$-vollständig

Wdh: Multi-Criteria Dijkstra für CSP

CSP kann mit Multi-Criteria Dijkstra (MCD) gelöst werden

- Bikriterieller Ansatz mit Metriken: Länge & Gewicht
- Nutze Constraints zum Prunen
  - Verwerfe Label mit Gewicht $> W$

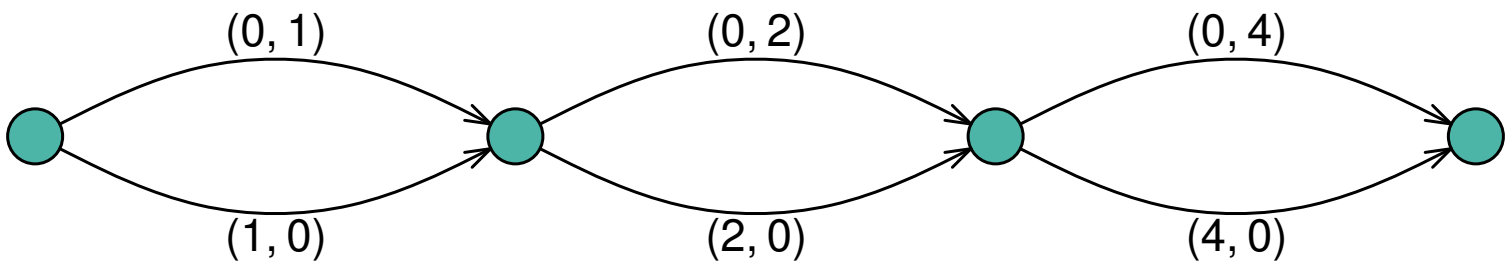
Wdh: Multi-Criteria Dijkstra für CSP

CSP kann mit Multi-Criteria Dijkstra (MCD) gelöst werden

- Bikriterieller Ansatz mit Metriken: Länge & Gewicht
- Nutze Constraints zum Prunen
  - Verwerfe Label mit Gewicht $> W$

Erinnerung:

- MCD hält pro Knoten eine Menge Pareto-optimaler Pfade
- Pareto-Mengen können exponentiell groß werden



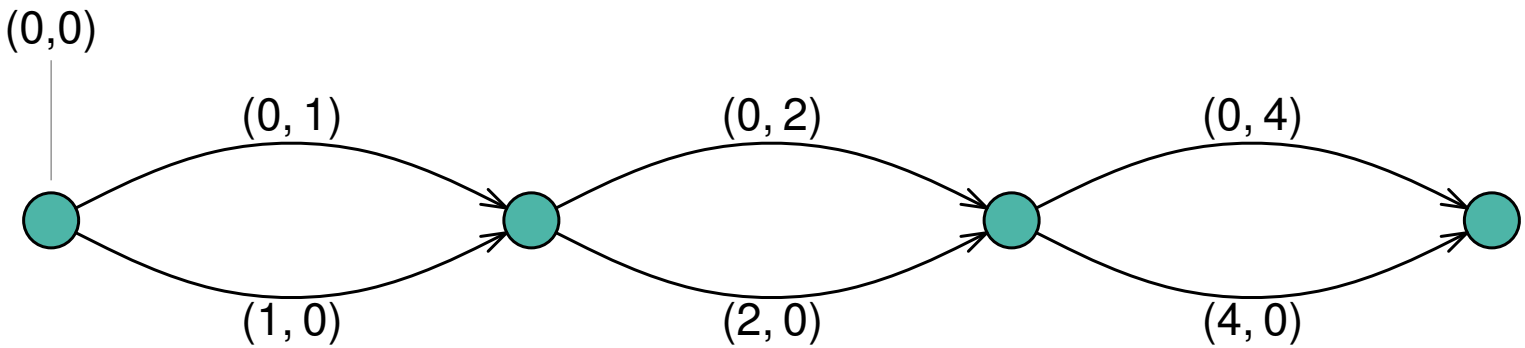
Wdh: Multi-Criteria Dijkstra für CSP

CSP kann mit Multi-Criteria Dijkstra (MCD) gelöst werden

- Bikriterieller Ansatz mit Metriken: Länge & Gewicht
- Nutze Constraints zum Prunen
  - Verwerfe Label mit Gewicht $> W$

Erinnerung:

- MCD hält pro Knoten eine Menge Pareto-optimaler Pfade
- Pareto-Mengen können exponentiell groß werden



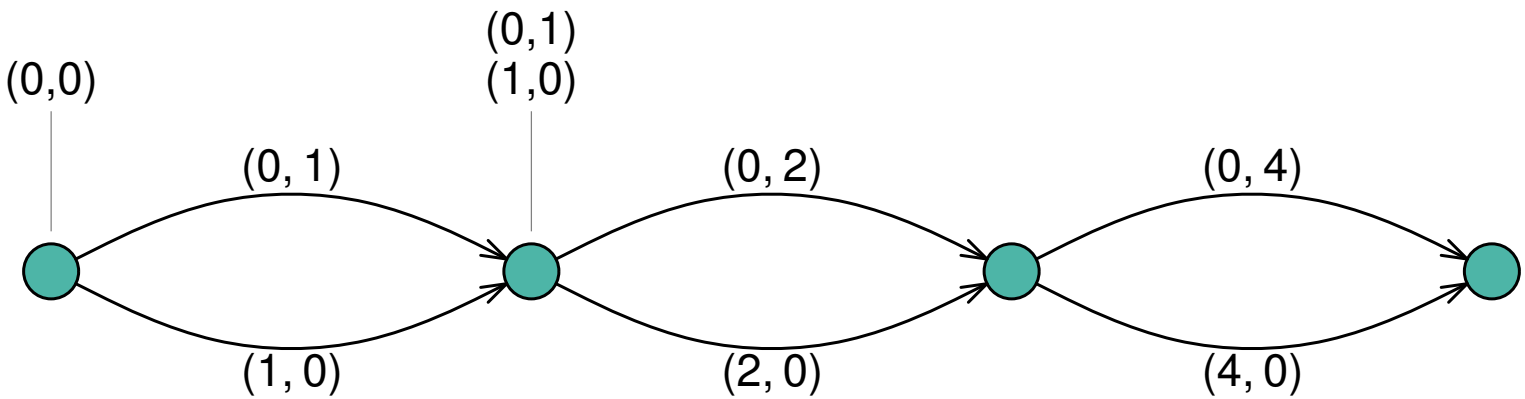
Wdh: Multi-Criteria Dijkstra für CSP

CSP kann mit Multi-Criteria Dijkstra (MCD) gelöst werden

- Bikriterieller Ansatz mit Metriken: Länge & Gewicht
- Nutze Constraints zum Prunen
  - Verwerfe Label mit Gewicht $> W$

Erinnerung:

- MCD hält pro Knoten eine Menge Pareto-optimaler Pfade
- Pareto-Mengen können exponentiell groß werden



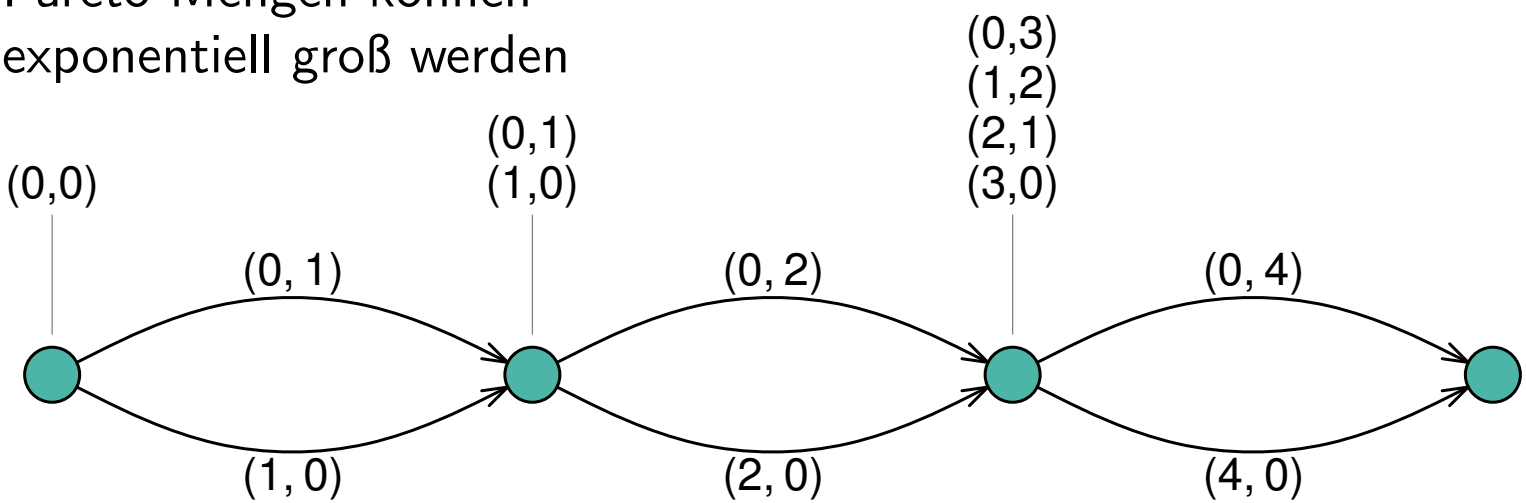
Wdh: Multi-Criteria Dijkstra für CSP

CSP kann mit Multi-Criteria Dijkstra (MCD) gelöst werden

- Bikriterieller Ansatz mit Metriken: Länge & Gewicht
- Nutze Constraints zum Prunen
  - Verwerfe Label mit Gewicht $> W$

Erinnerung:

- MCD hält pro Knoten eine Menge Pareto-optimaler Pfade
- Pareto-Mengen können exponentiell groß werden



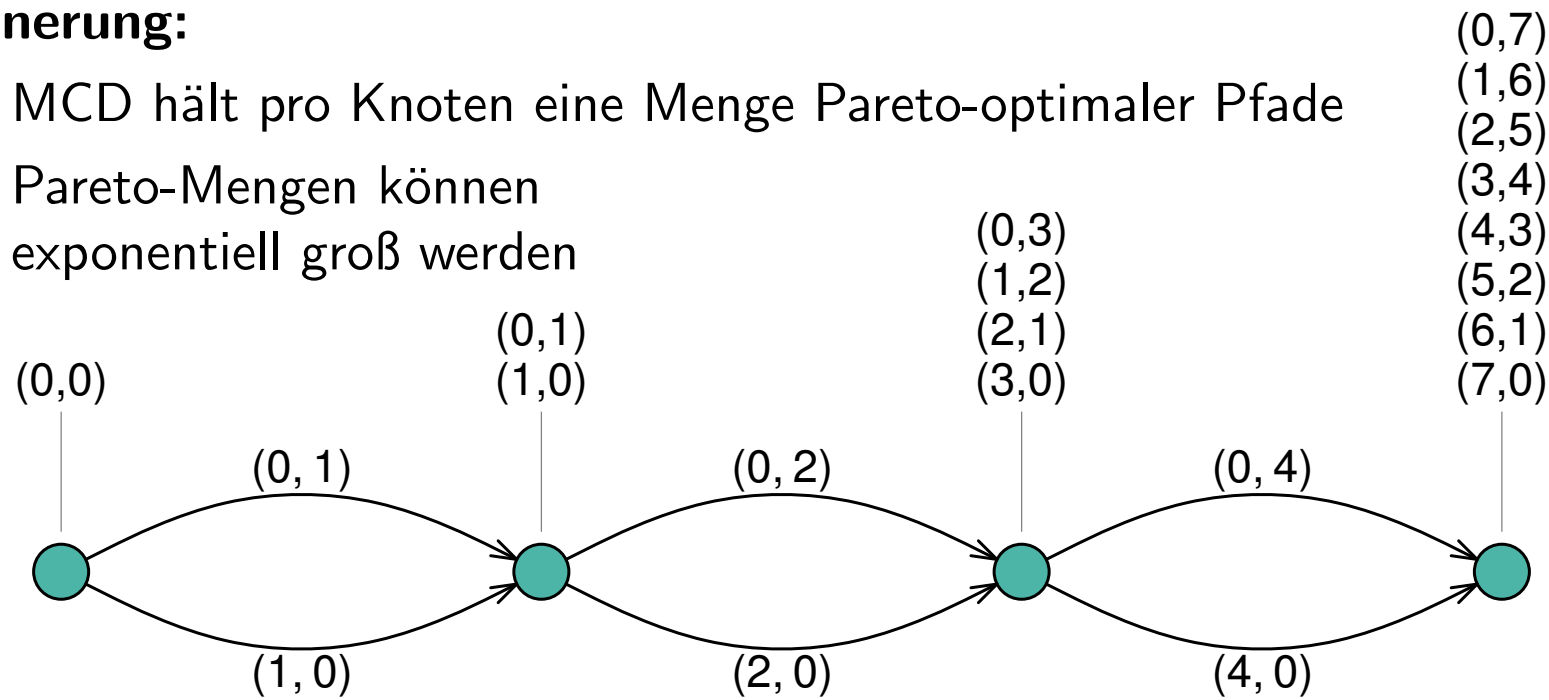
Wdh: Multi-Criteria Dijkstra für CSP

CSP kann mit Multi-Criteria Dijkstra (MCD) gelöst werden

- Bikriterieller Ansatz mit Metriken: Länge & Gewicht
- Nutze Constraints zum Prunen
  - Verwerfe Label mit Gewicht $> W$

Erinnerung:

- MCD hält pro Knoten eine Menge Pareto-optimaler Pfade
- Pareto-Mengen können exponentiell groß werden



Schnellste zulässige Route

Idee:

- Nutze gleichen Ansatz für EV-Routing
- Label sind Tupel (Fahrzeit, SoC)
- Falls SoC < 0, Pfad nicht weiter verfolgen

Schnellste zulässige Route

Idee:

- Nutze gleichen Ansatz für EV-Routing
- Label sind Tupel (Fahrzeit, SoC)
- Falls SoC < 0, Pfad nicht weiter verfolgen

Verbesserungen: Standard-Beschleunigungen von MCD übertragbar:

- Hopping Reduction
- Nur ein Label pro Knoten in Queue
- Target Pruning (Nutze max. Rekuperation $\underline{d}[t]$)
- Knotenkontraktionen (Nutze Verbrauchsfunktionen)

Schnellste zulässige Route

Idee:

- Nutze gleichen Ansatz für EV-Routing
- Label sind Tupel (Fahrzeit, SoC)
- Falls SoC < 0, Pfad nicht weiter verfolgen

Verbesserungen: Standard-Beschleunigungen von MCD übertragbar:

- Hopping Reduction
- Nur ein Label pro Knoten in Queue
- Target Pruning (Nutze max. Rekuperation $\underline{d}[t]$)
- Knotenkontraktionen (Nutze Verbrauchsfunktionen)

Beobachtung: Wir brauchen nicht alle Pareto-Optima an t:

- Sind nur an schnellster zulässiger Route interessiert
- Stoppe, sobald erstes Label an t aus Queue genommen (Queue ist nach Fahrzeit sortiert)

Problemstellungen

Mögliche (alternative) Problemstellungen:

- Finde die schnellste Route, sodass das Ziel erreichbar ist
- Finde die schnellstmögliche Route, sodass der SoC am Ziel mindestens x % ist
- Finde die schnellstmögliche Route, sodass der SoC einen bestimmten SoC niemals unterschreitet
- Finde eine Route mit minimalem Verbrauch, wobei eine vorgegebene Zeit nicht überschritten wird
- Berechne alle nichtdominierten Lösungen

Alle Probleme sind $\mathcal{NP}$-schwer
$\Rightarrow$ Auch mit Speedup-Techniken ggf. nur Laufzeiten im Sekundenbereich
$\Rightarrow$ Heuristiken

Erweiterte Szenarien

Karlsruher Institut für Technologie

Variable Geschwindigkeit:

- Bisher: Kanten mit fester Geschwindigkeit
- Idee: Erlaube langsamer zu fahren
- Ermöglicht Trade-off zwischen Fahrzeit und Energieverbrauch
- Mögliche Umsetzungen:
  - Multikanten mit verschiedenen Geschwindigkeits-/Verbrauchswerten
  - Funktionen an Kanten, die Fahrzeit auf Verbrauch abbilden

Ladestationen:

- Akku-Kapazität ist stark begrenzt (~100 km, max. 400 km)
- Lange Strecken unmöglich, selbst mit verbrauchsoptimalen Routen
- Nutzung von Ladestationen nicht zu vermeiden
- Problem: Ladestationen sind langsam und wenig verbreitet

Ladestopps

Ladestationen

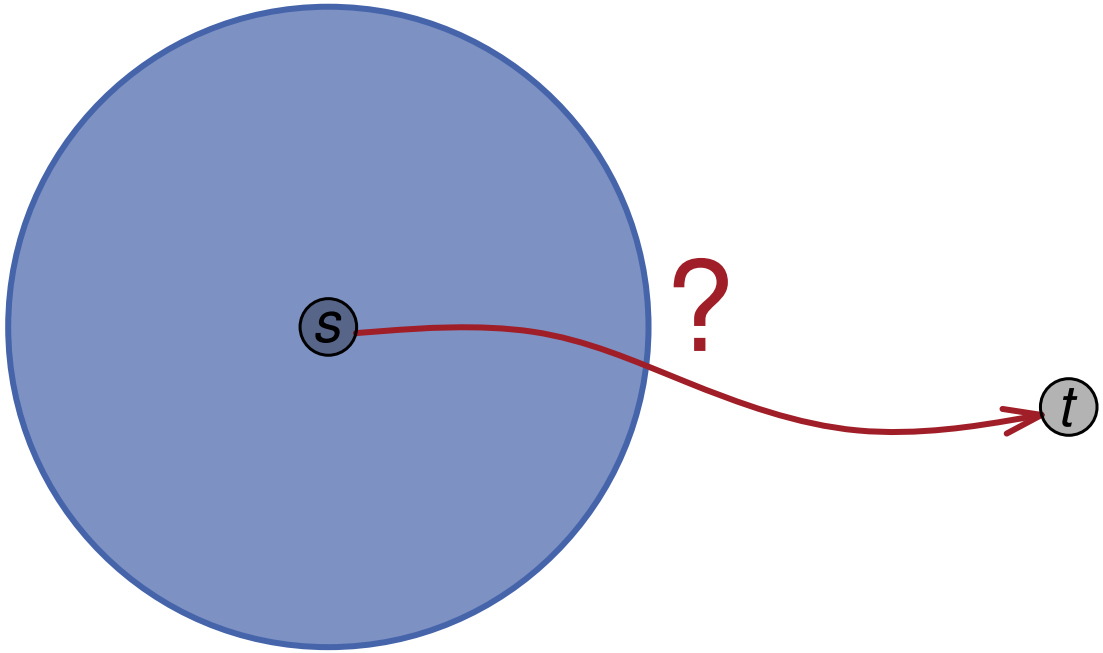
Finde schnellste Route von s nach t:

Erreichbares Gebiet

Ladestation

Ladestationen

Finde schnellste Route von s nach t:

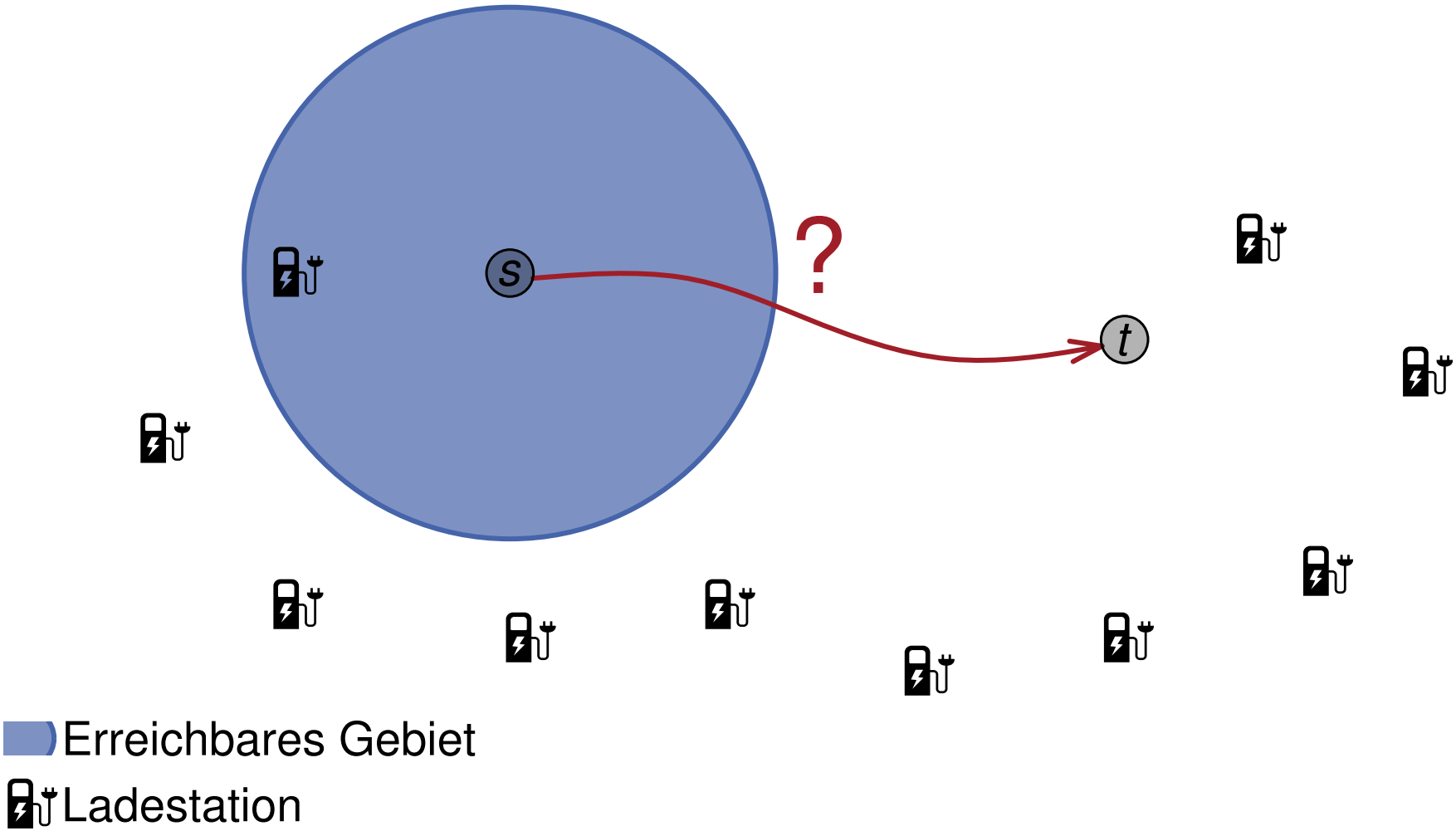


Erreichbares Gebiet

Ladestation

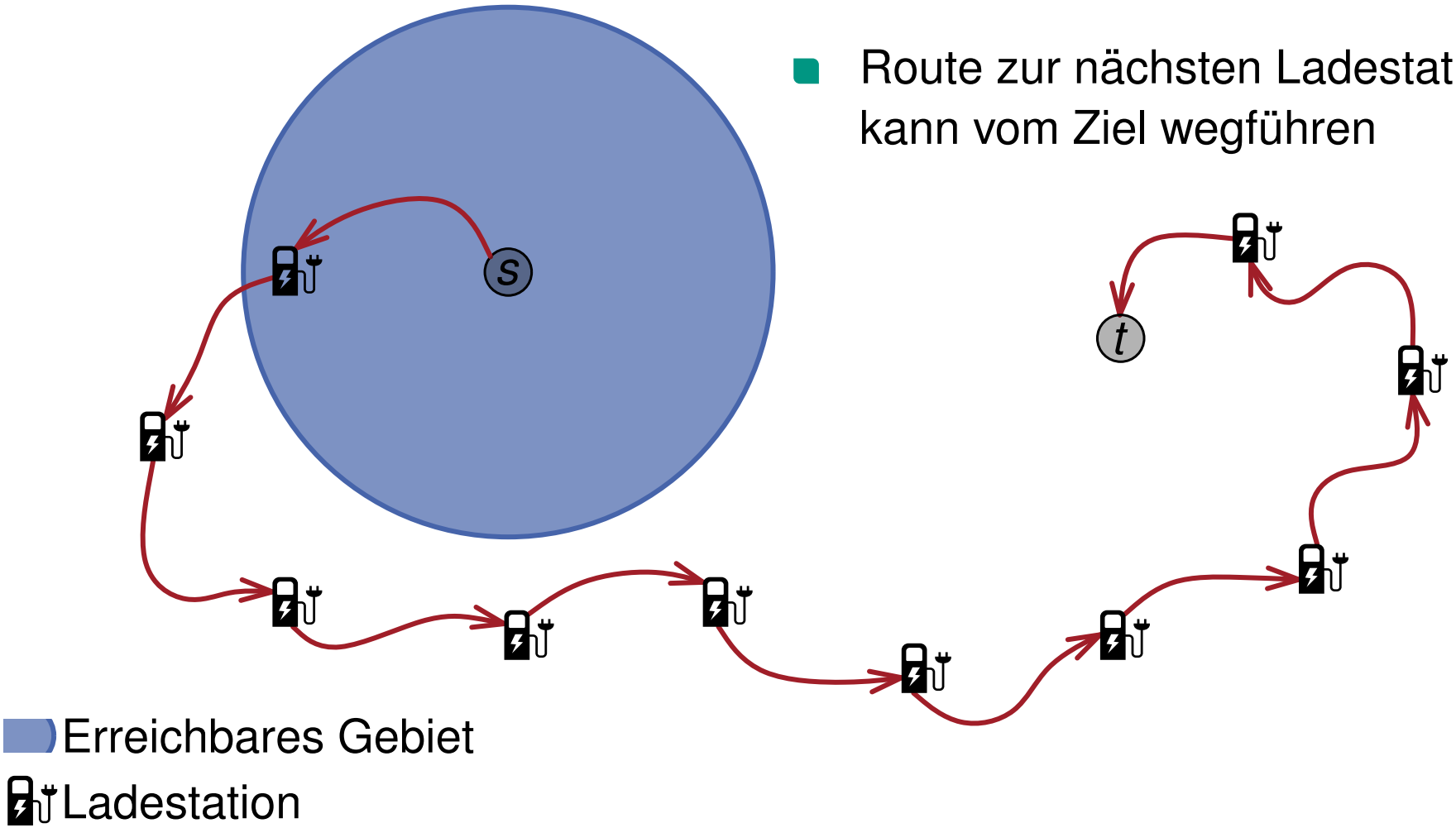
Ladestationen

Finde schnellste Route von s nach t:



Ladestationen

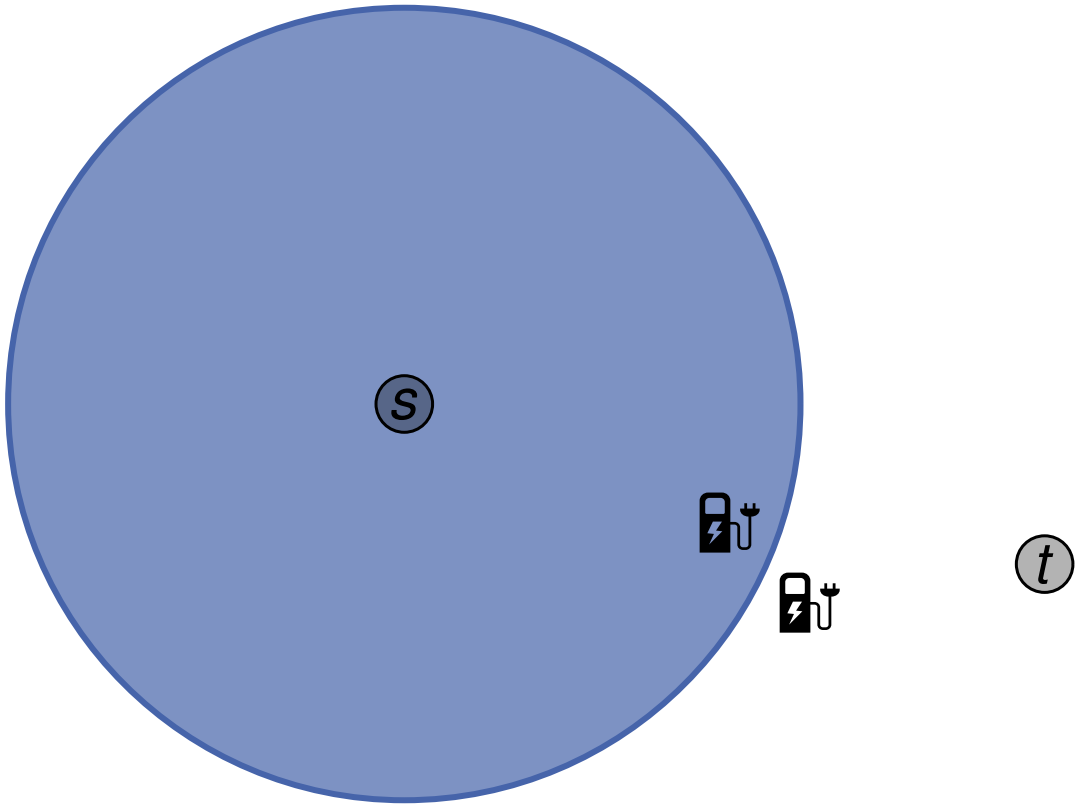
Finde schnellste Route von s nach t:



- Route zur nächsten Ladestation kann vom Ziel wegführen

Ladestationen

Finde schnellste Route von s nach t:

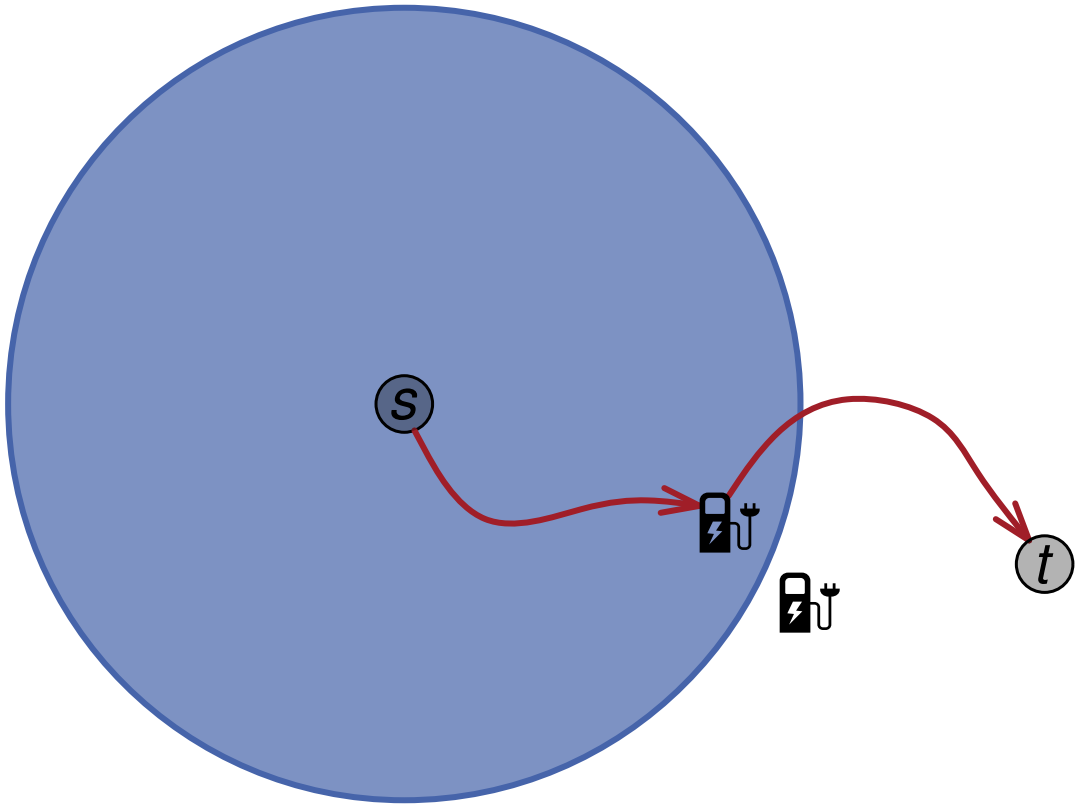


Erreichbares Gebiet

Ladestation

Ladestationen

Finde schnellste Route von s nach t:

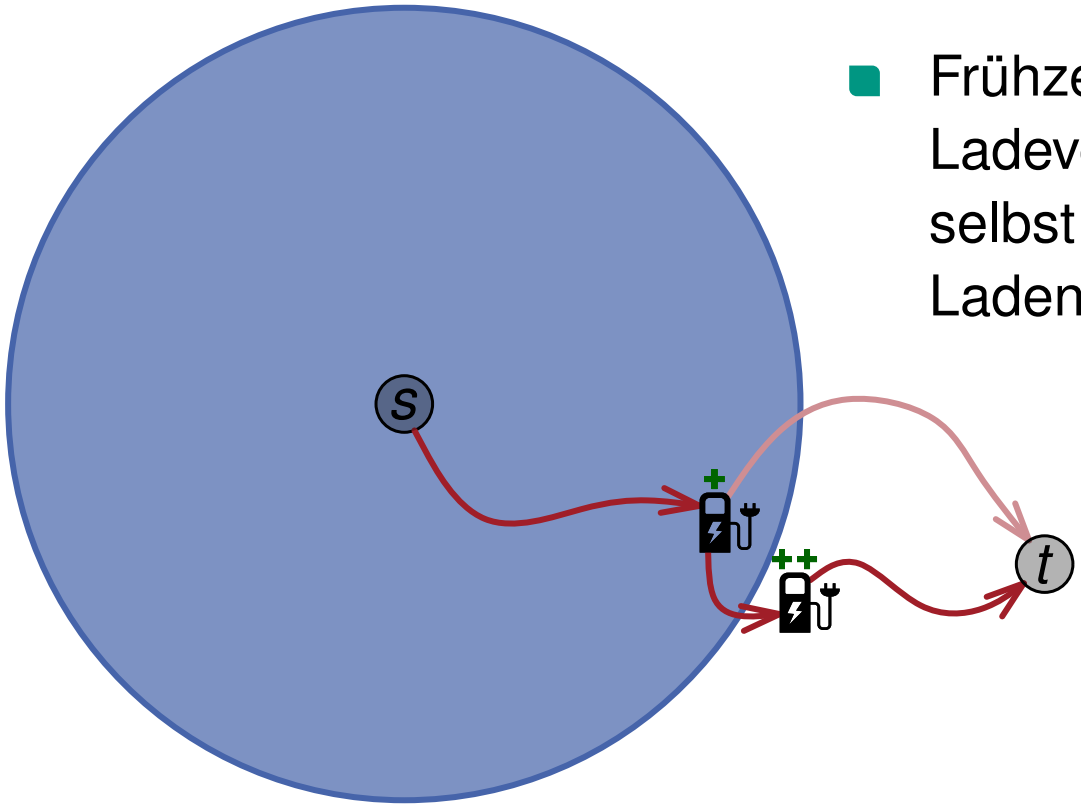


Erreichbares Gebiet

Ladestation

Ladestationen

Finde schnellste Route von s nach t:



- Frühzeitiger Abbruch des Ladevorgangs kann lohnen, selbst wenn Ziel bei weiterem Laden direkt erreichbar wäre

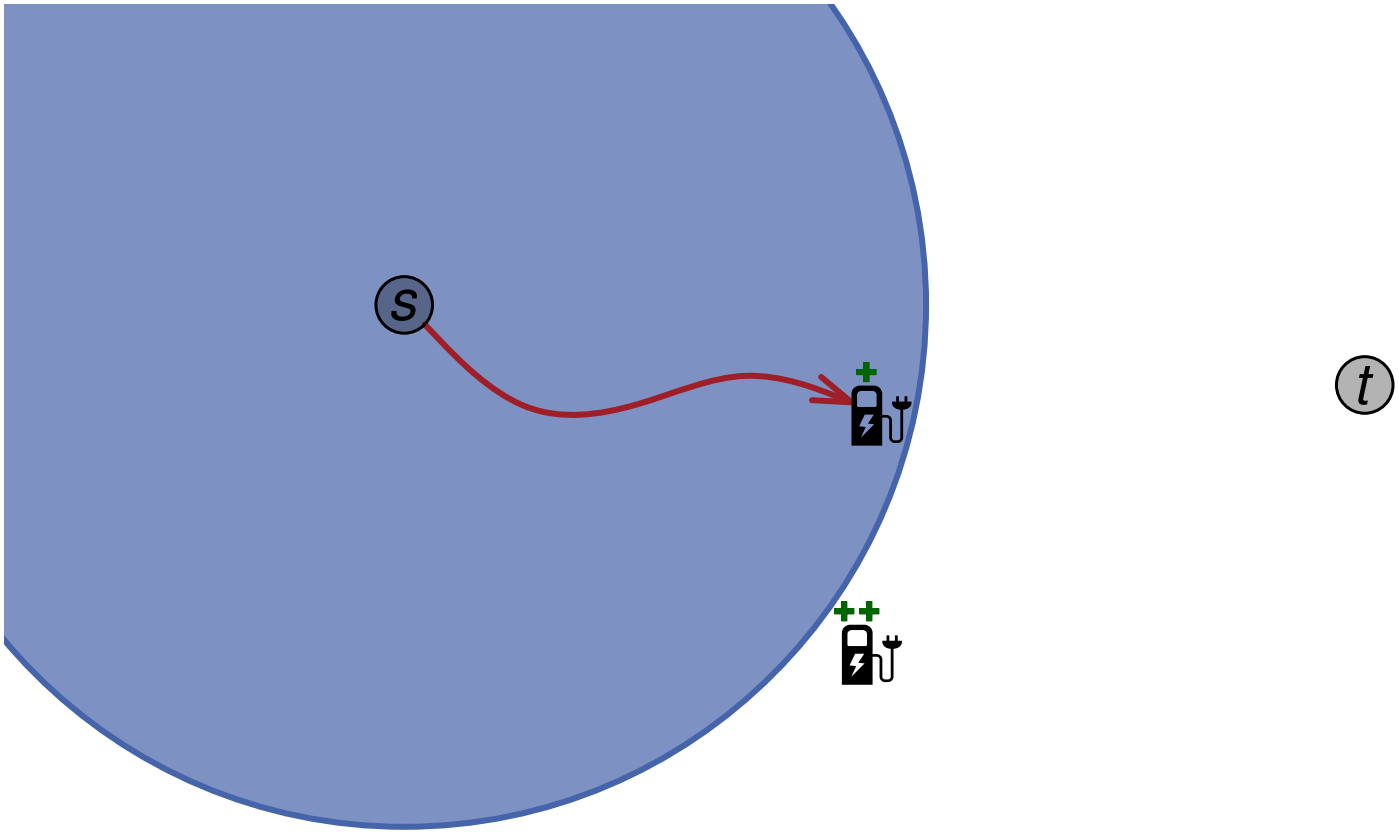
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:



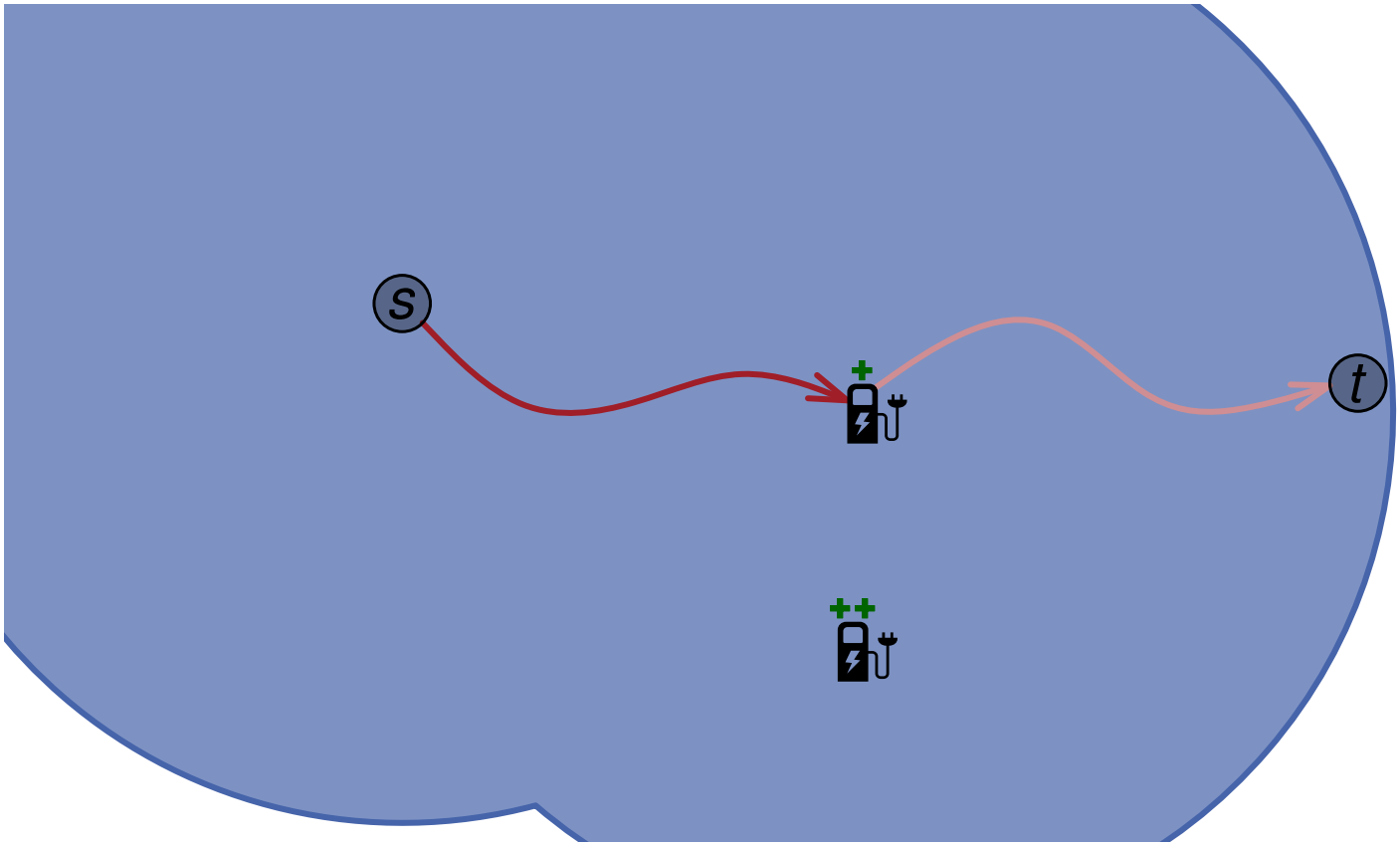
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:



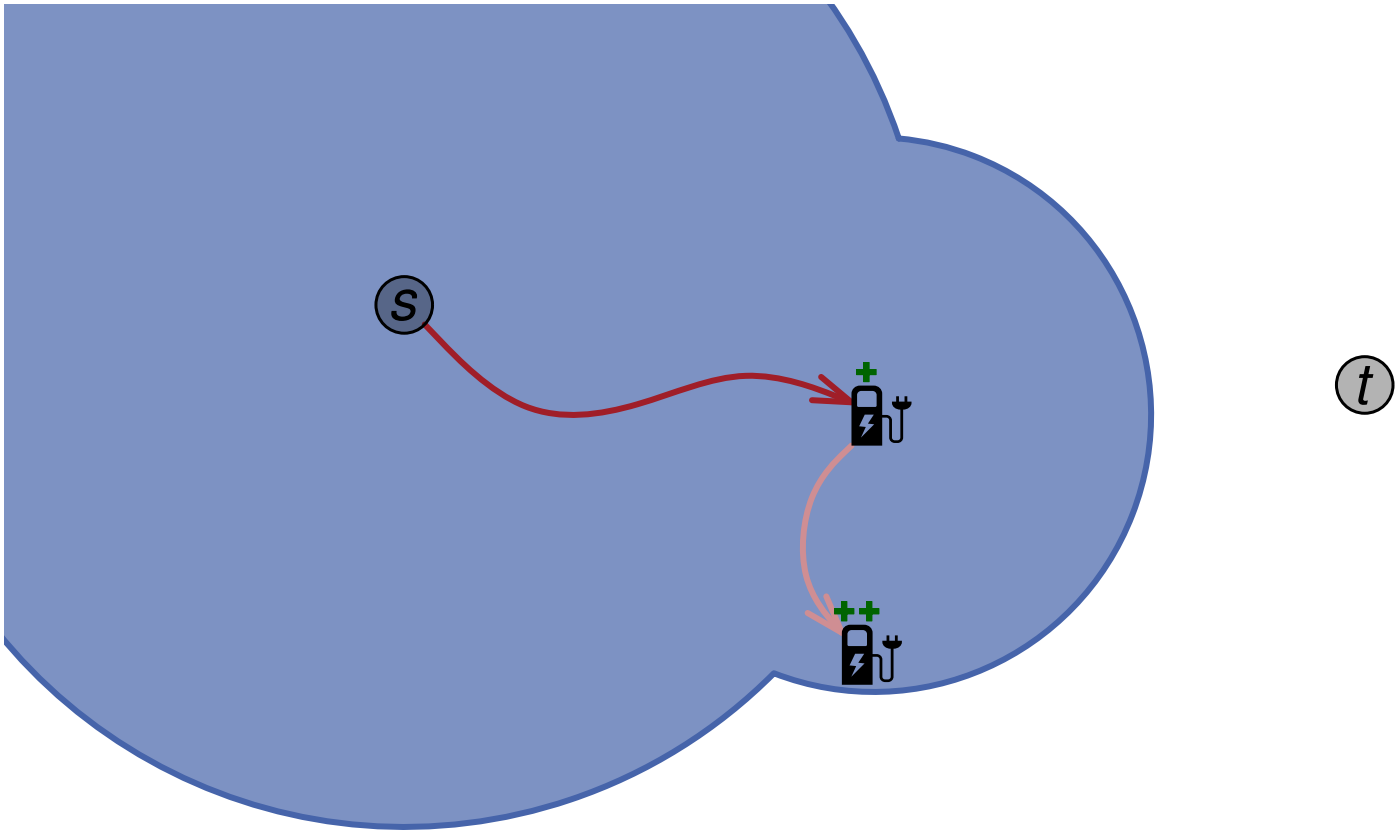
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:



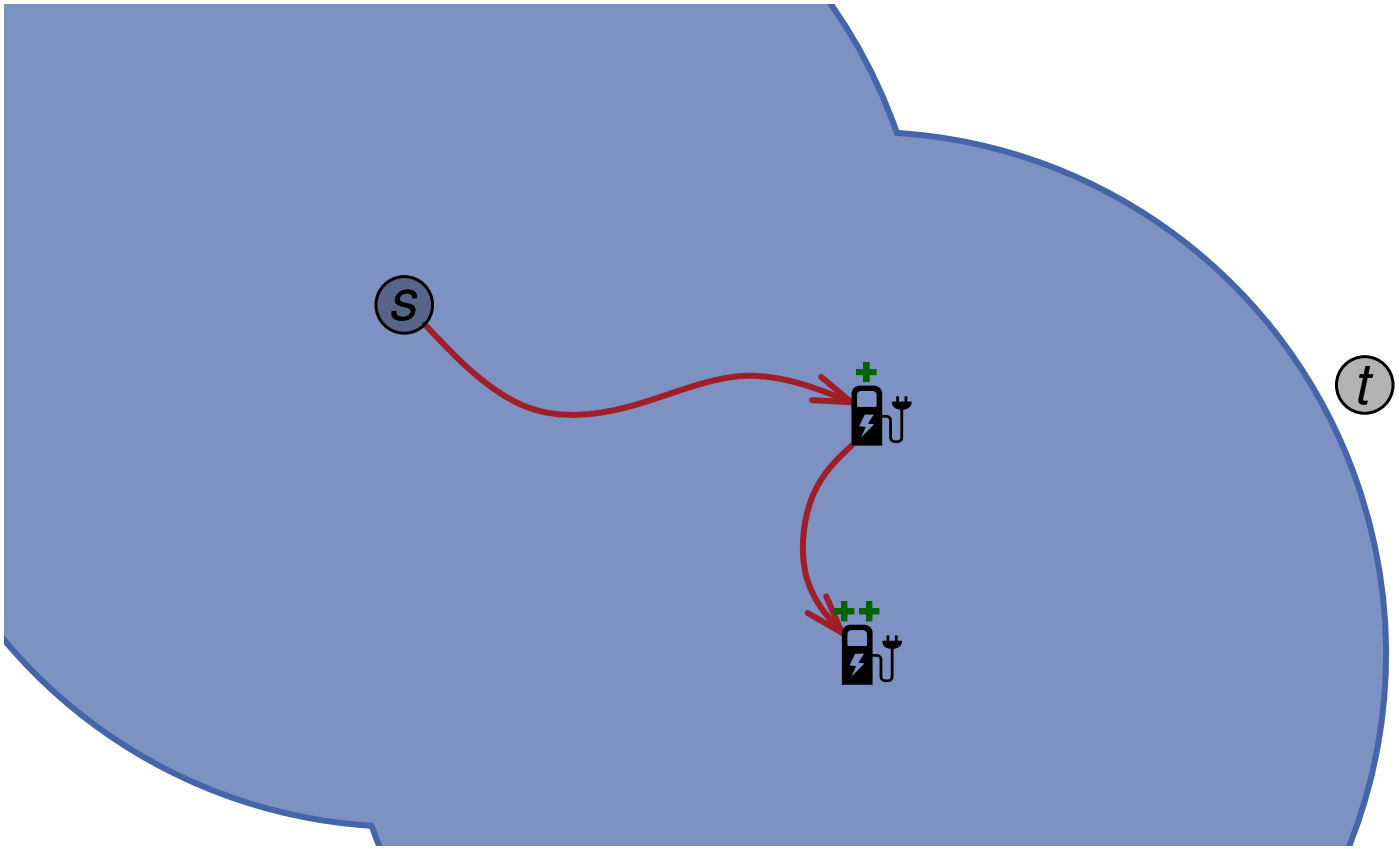
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:



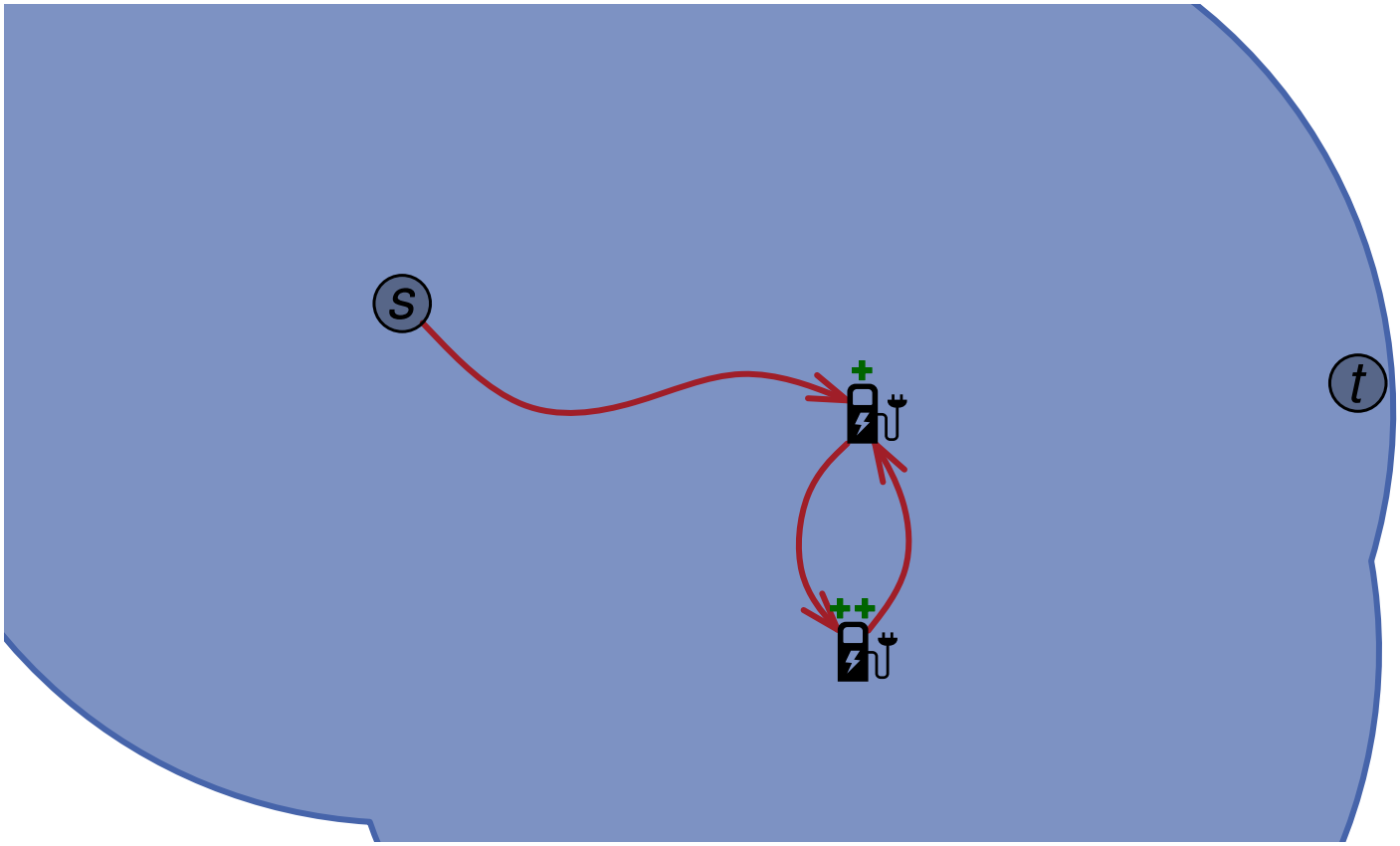
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:



Erreichbares Gebiet

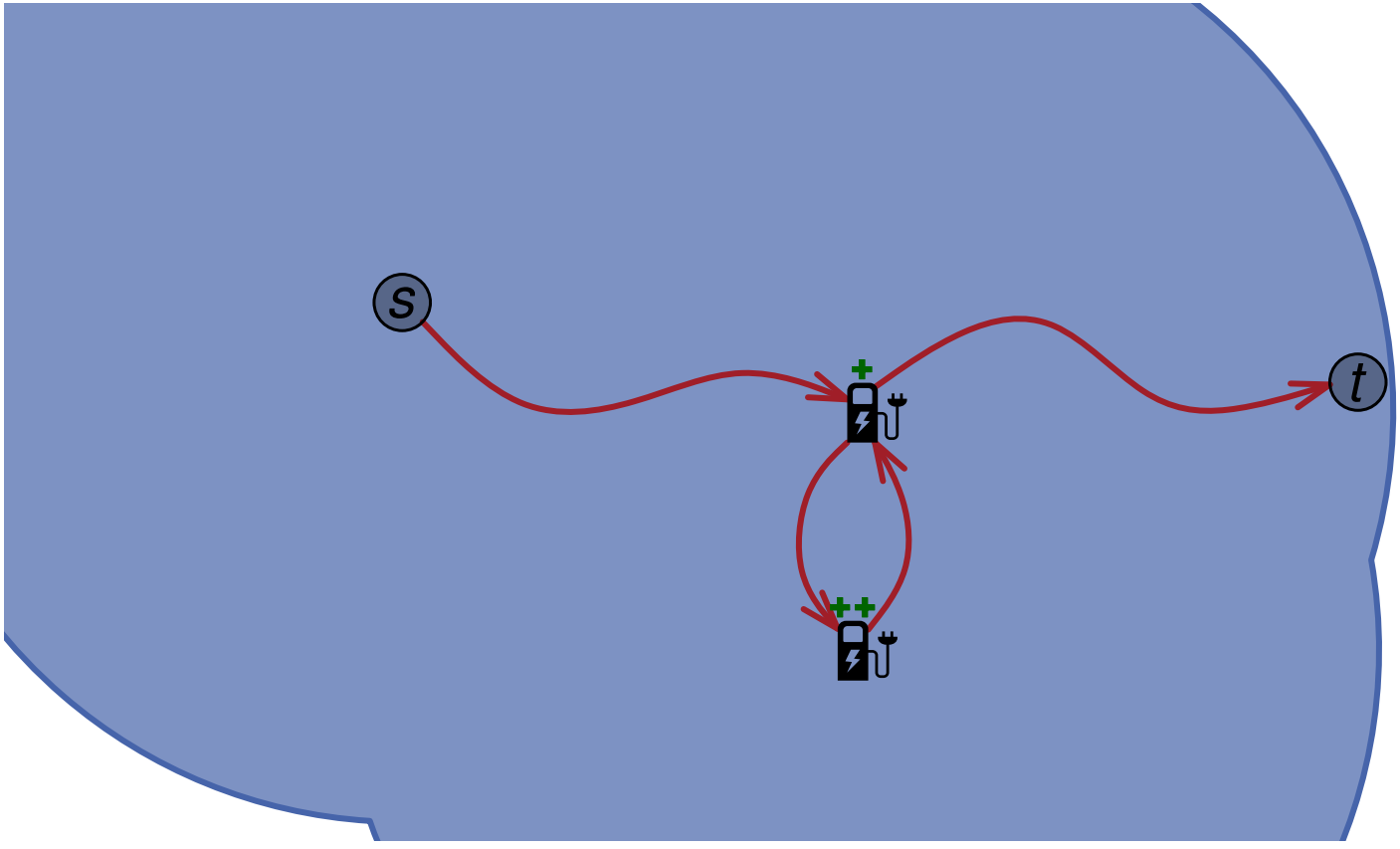
Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:

- Zyklen möglich



Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Schwierigkeiten:

- Laden dauert lange ($\Rightarrow$ lieber Energie sparen, laden vermeiden)
- Ladestationen sind selten (lohnt sich ein Umweg?)
- Laden jederzeit unterbrechbar

Ansatz:

- Ladezeiten müssen während Routenplanung berücksichtigt werden
- Optimiere Reisezeit = Fahrzeit + Ladezeit
- Teilmenge $S \subseteq V$ der Knoten sind Ladestationen
- Für jede Station eine Funktion, die das Ladeverhalten beschreibt

Ladefunktionen

Formal:

- Eine Funktion $\mathrm{cf}: [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC b_s und
  - gewünschte Ladezeit τ_c auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

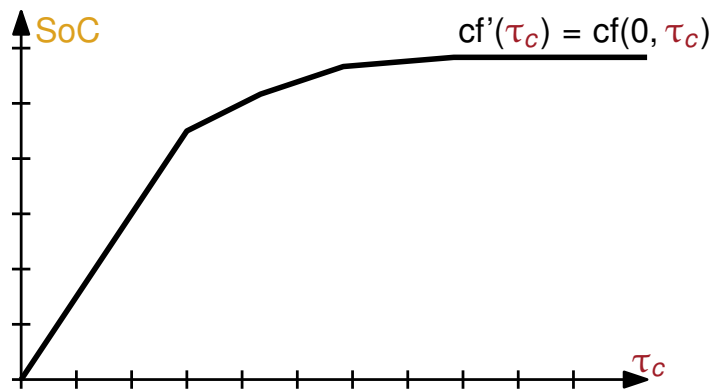
Ladefunktionen

Formal:

- Eine Funktion $\mathrm{cf}: [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC b_s und
  - gewünschte Ladezeit τ_c auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf}': \mathbb{R}_{\geq 0} \to [0, M]$
  $\mathrm{cf}(b, \tau_c) := \mathrm{cf}'(\tau_c + \mathrm{cf}'^{-1}(b))$



Ladefunktionen

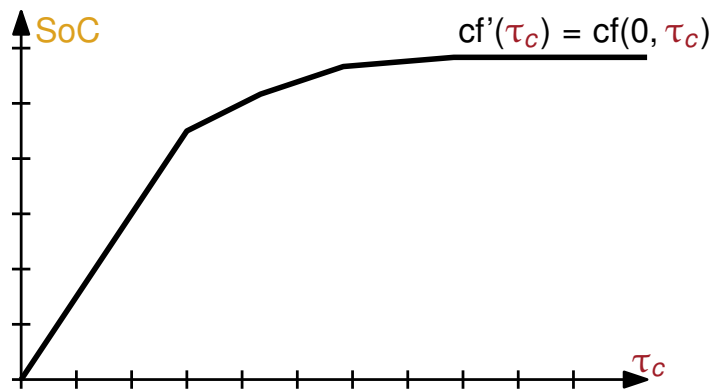
Formal:

- Eine Funktion $\text{cf} : [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC b_s und
  - gewünschte Ladezeit τ_c auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\text{cf}' : \mathbb{R}_{\geq 0} \to [0, M]$
  $\text{cf}(b, \tau_c) := \text{cf}'(\tau_c + \text{cf}'^{-1}(b))$

  $\text{cf}(3, 2)$



Ladefunktionen

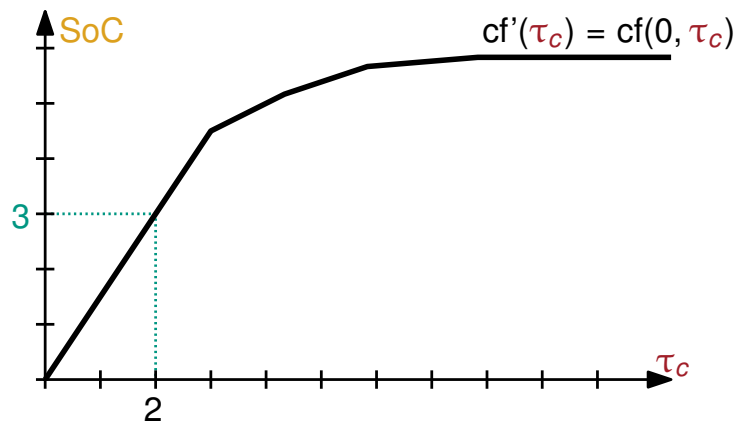
Formal:

- Eine Funktion $\mathrm{cf}: [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC b_s und
  - gewünschte Ladezeit τ_c auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf}': \mathbb{R}_{\geq 0} \to [0, M]$
  $\mathrm{cf}(b, \tau_c) := \mathrm{cf}'(\tau_c + \mathrm{cf}'^{-1}(b))$

  $\mathrm{cf}(3, 2) = \mathrm{cf}'(2 + \mathrm{cf}'^{-1}(3))$



Ladefunktionen

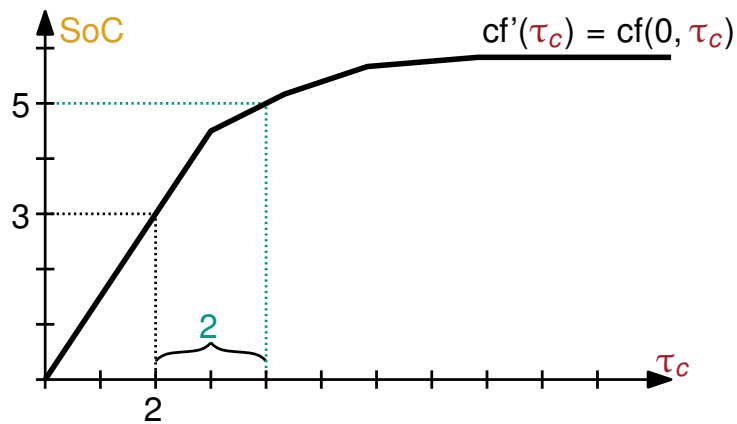
Formal:

- Eine Funktion $\mathrm{cf}: [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC b_s und
  - gewünschte Ladezeit τ_c auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf}': \mathbb{R}_{\geq 0} \to [0, M]$
  $\mathrm{cf}(b, \tau_c) := \mathrm{cf}'(\tau_c + \mathrm{cf}'^{-1}(b))$

$$\mathrm{cf}(3, 2) = \mathrm{cf}'(2 + \mathrm{cf}'^{-1}(3))$$
$$= \mathrm{cf}'(2 + 2)$$



Ladefunktionen

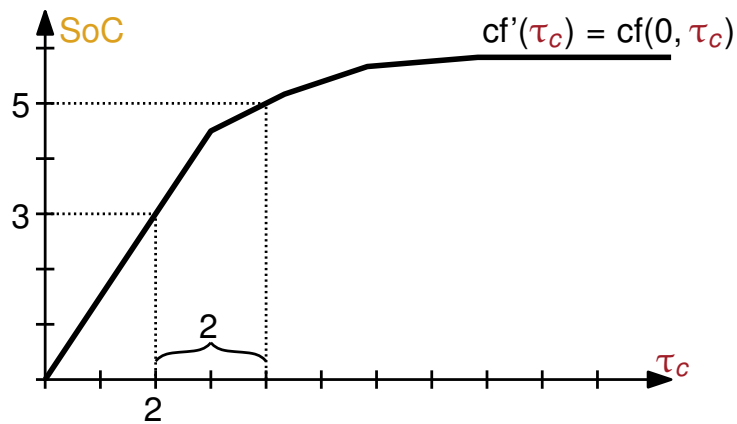
Formal:

- Eine Funktion $\mathrm{cf}: [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC b_s und
  - gewünschte Ladezeit τ_c auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf}': \mathbb{R}_{\geq 0} \to [0, M]$
  $\mathrm{cf}(b, \tau_c) := \mathrm{cf}'(\tau_c + \mathrm{cf}'^{-1}(b))$

$$\begin{aligned}\mathrm{cf}(3, 2) &= \mathrm{cf}'(2 + \mathrm{cf}'^{-1}(3)) \\ &= \mathrm{cf}'(2 + 2) \\ &= 5\end{aligned}$$



Literatur I

Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf.

Shortest feasible paths with charging stops for battery electric vehicles.

In *Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 44:1–44:10. ACM Press, 2015.

Moritz Baum, Julian Dibbelt, Lorenz Hübschle-Schneider, Thomas Pajor, and Dorothea Wagner.

Speed-consumption tradeoff for electric vehicle route planning.

In *Proceedings of the 14th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'14)*, volume 42 of *OpenAccess Series in Informatics (OASIcs)*, pages 138–151. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

Literatur II

Moritz Baum, Julian Dibbelt, Dorothea Wagner, and Tobias Zündorf.

Modeling and engineering constrained shortest path algorithms for battery electric vehicles.

In *Proceedings of the 25th Annual European Symposium on Algorithms (ESA'17)*, volume 87 of *Leibniz International Proceedings in Informatics*, pages 11:1–11:16, 2017.