Karlsruher Institut für Technologie

# Algorithmen für Routenplanung

**12. Vorlesung, Sommersemester 2024**

Moritz Laupichler | 05. Juni 2024

# Elektromobilität

**Elektrofahrzeuge (EVs):**

- Transportmittel der Zukunft
- Emissionsfreie Mobilität

**Aber:**

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

⇒ Berücksichtigung von Energieverbrauch bei der Routenplanung

# Ladestopps

# Ladestationen

- Reichweite ist stark begrenzt
  - VW e-Up!: 260 km
  - Tesla Model S: 400–600 km
  - Je nach Witterung und Geschwindigkeit deutlich weniger!
- Lange Strecken unmöglich, selbst mit verbrauchsoptimalen Routen
- Nutzung von Ladestationen nicht zu vermeiden
- Problem: Ladestationen sind langsam und wenig verbreitet
  - Tesla Supercharger: ca. 40 min für Aufladen auf 80% Kapazität

# 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

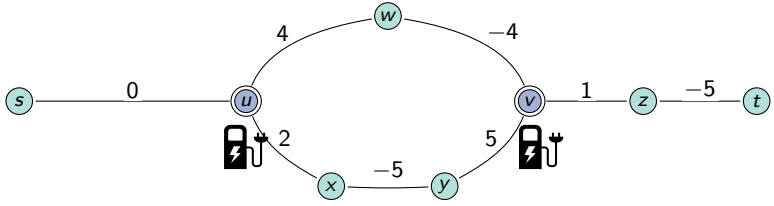
- $b_t$: SoC an $t$
- $r_t$: Geladene Energie, um $t$ zu erreichen

# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$



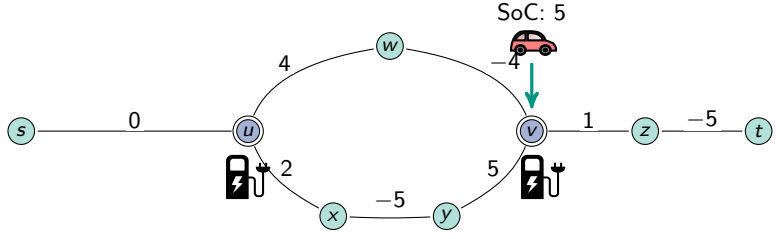
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_v - r_v = 5 - 0 = 5$



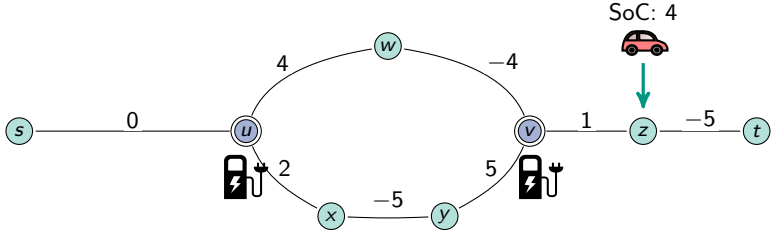
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_z - r_z = 4 - 0 = 4$



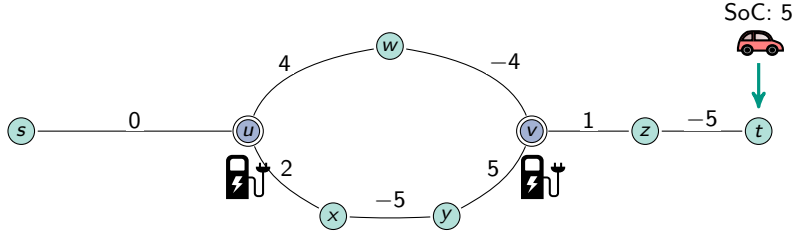
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_t - r_t = 5 - 0 = 5$



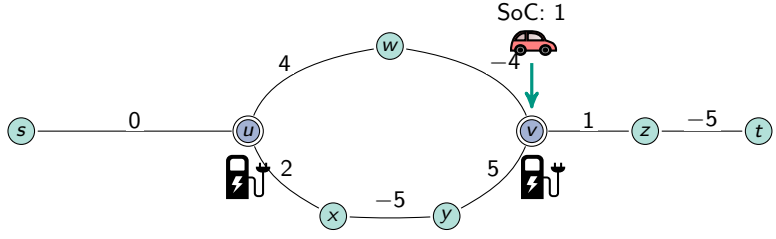
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_v - r_v = 1 - 0 = 1$



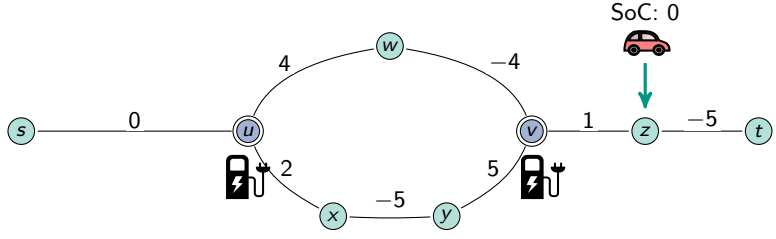
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_z - r_z = 0 - 0 = 0$



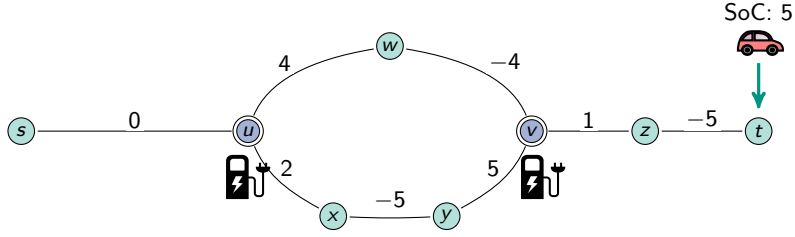
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_t - r_t = 5 - 0 = 5$



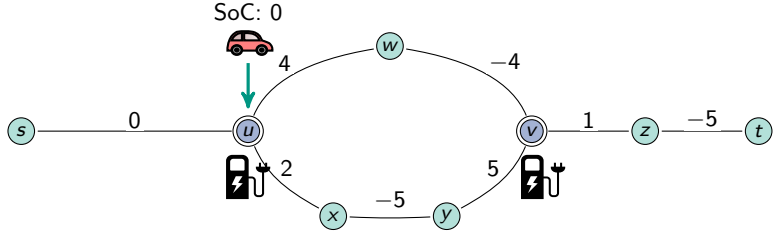
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_u - r_u = 0 - 0 = 0$



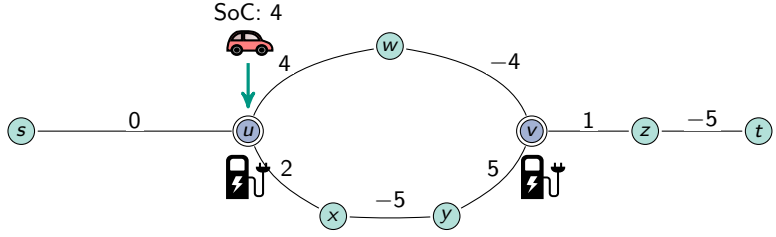
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_u - r_u = 4 - 4 = 0$

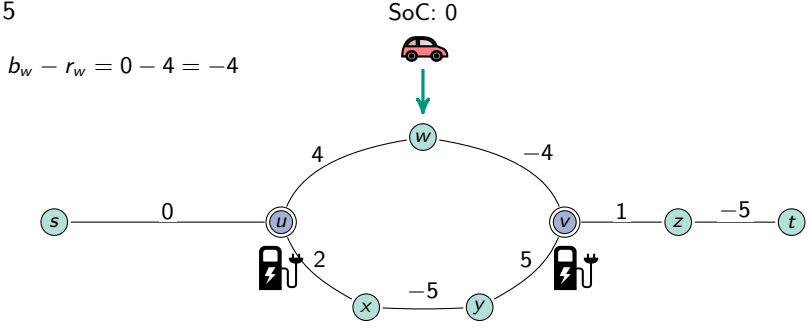


# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$



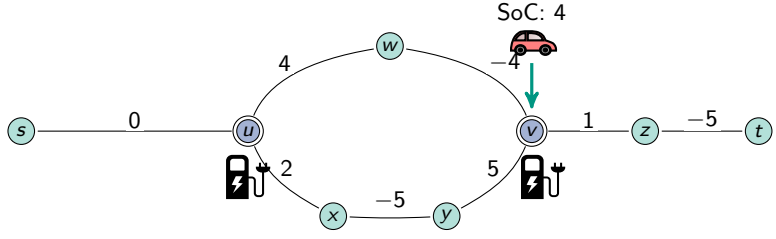
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_v - r_v = 4 - 4 = 0$



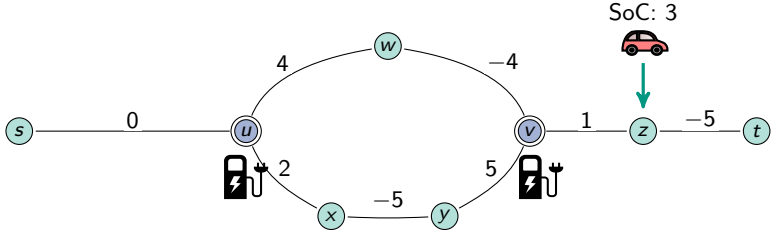
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

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



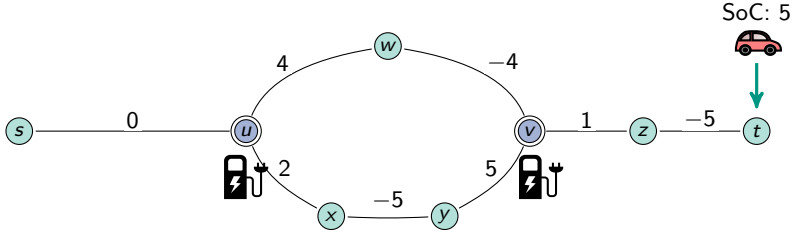
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_t - r_t = 5 - 4 = 1$



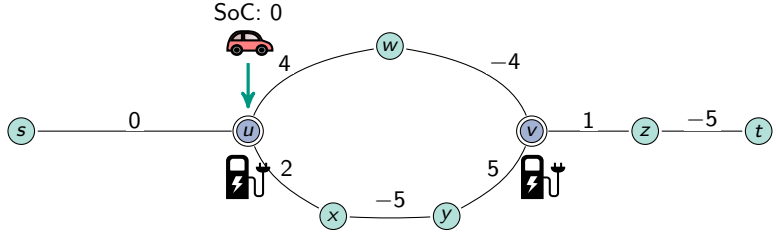
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_u - r_u = 0 - 0 = 0$



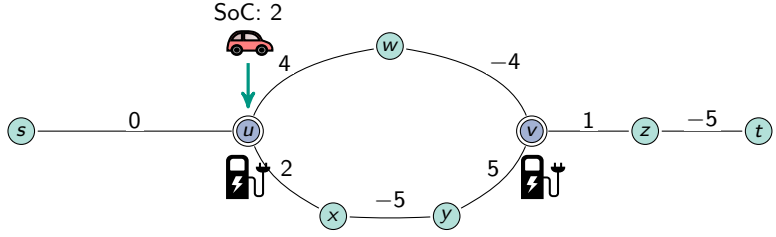
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_u - r_u = 2 - 2 = 0$



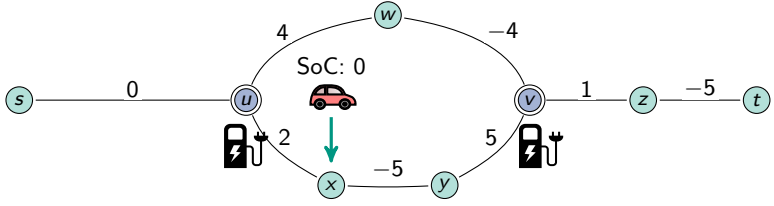
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

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



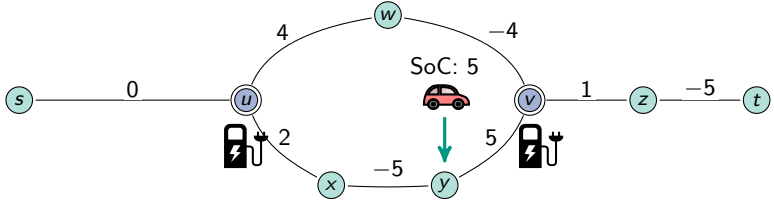
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$$b_y - r_y = 5 - 2 = 3$$



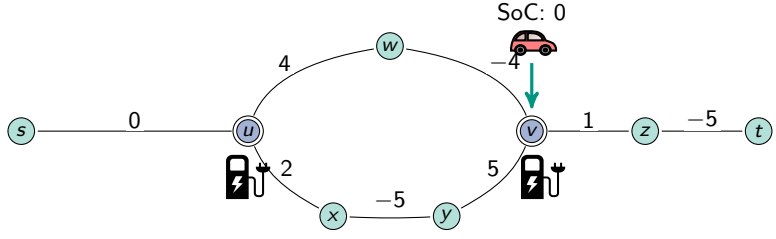
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

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



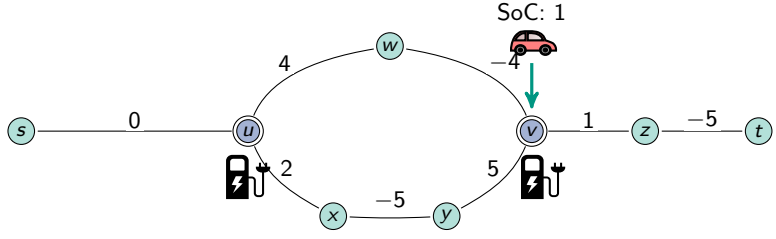
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

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



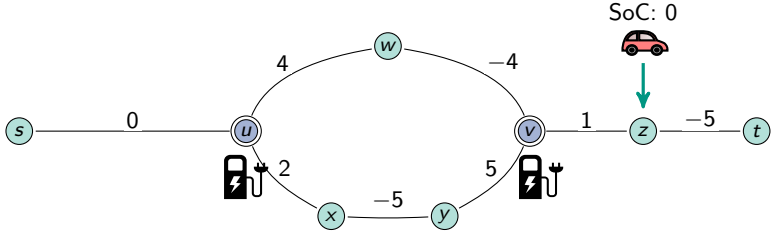
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

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



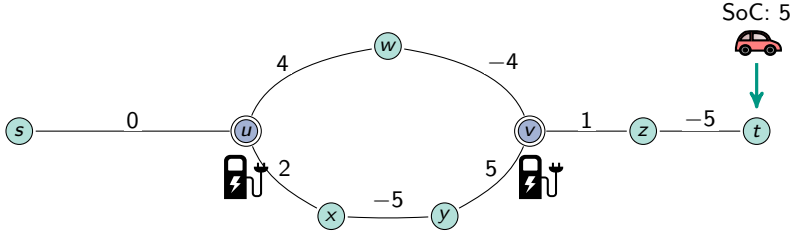
# Planung von Ladestopps

**Zwei Probleme:**

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

**Beispiel:** $M = 5$

$b_t - r_t = 5 - 3 = 2$

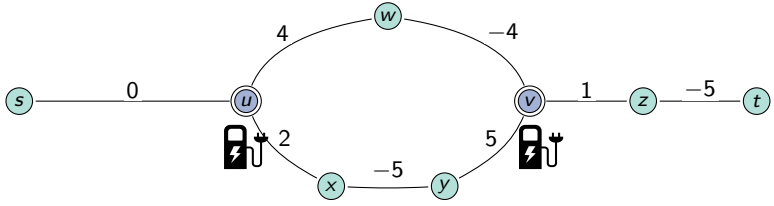


# Planung von Ladestopps

**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 zwischen jedem Paar von Ladestationen (+Ziel) ein Profil
  (Anzahl Pfade 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“

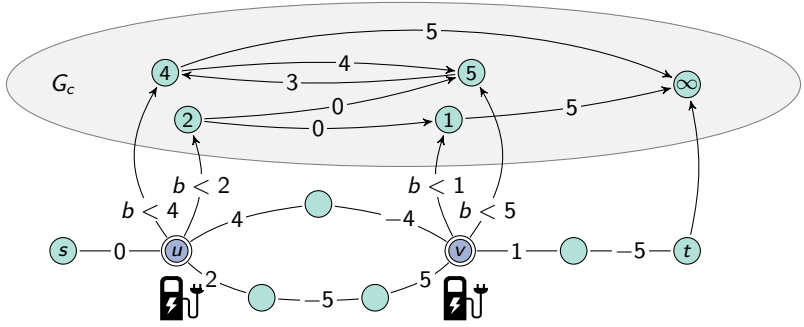
$\Rightarrow$ Konstruiere polynomiellen Suchgraph:

- 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)

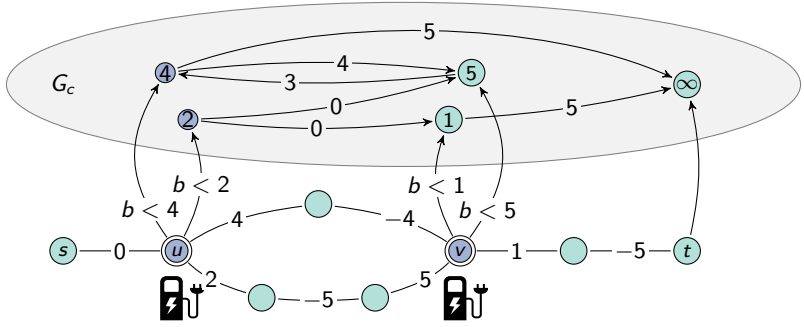



$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

# Suchgraph

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

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

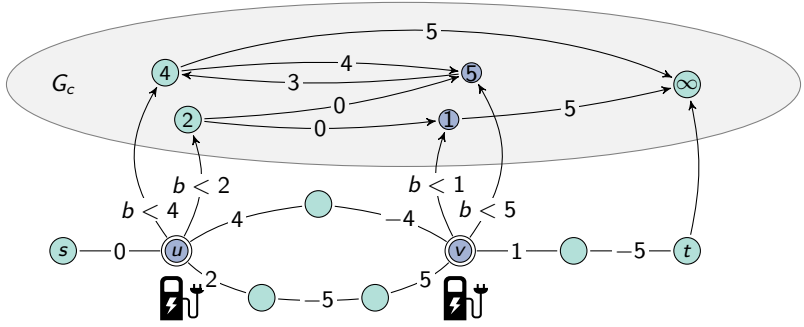



$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

# Suchgraph

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

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

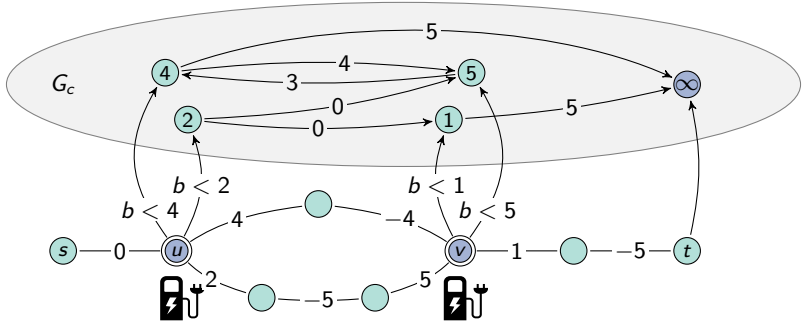



$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

# Suchgraph

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

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

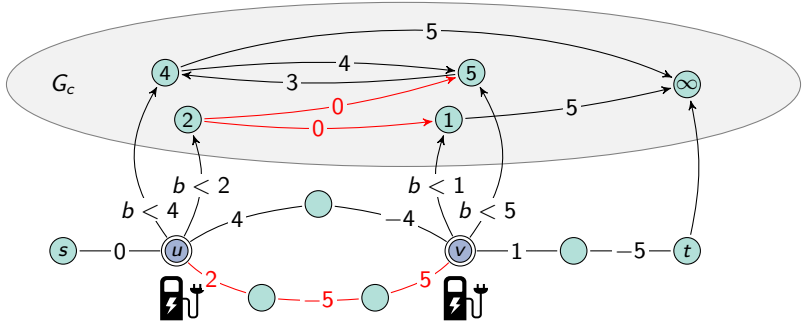



$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

# Suchgraph

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

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

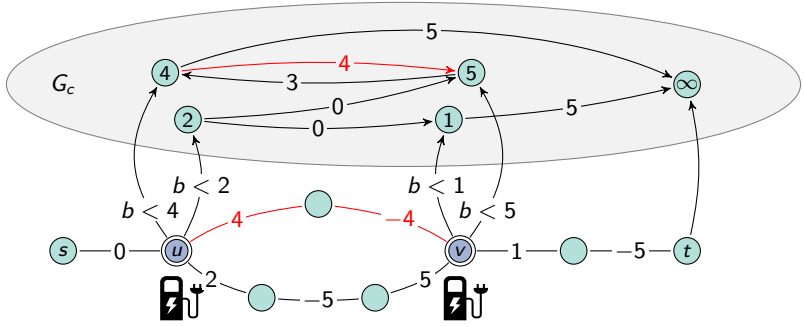



$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

# Suchgraph

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

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

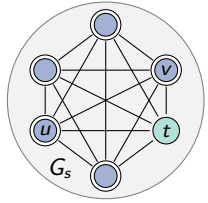



$\mathcal{O}(|V|)$ Pfade pro Ladestationspaar $\Rightarrow$ Suchgraph hat poly. Größe

# 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

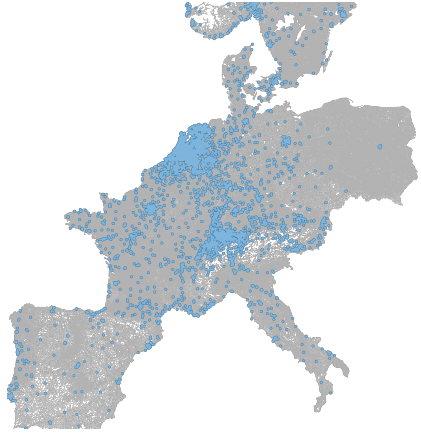
Speedup-Techniken:

- Contraction Hierarchies (CH): Partielle CH; kontrahiere keine Ladestationen
- A*-Suche: (ähnlich wie bei multikriterieller Suche)
  - Rückwärtssuche vom Ziel mit unendlich großer Batterie (ohne Ladestationen)
  - Liefert untere Schranke an Energieverbrauch zum Ziel $\rightarrow$ A*-Potential

# 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)

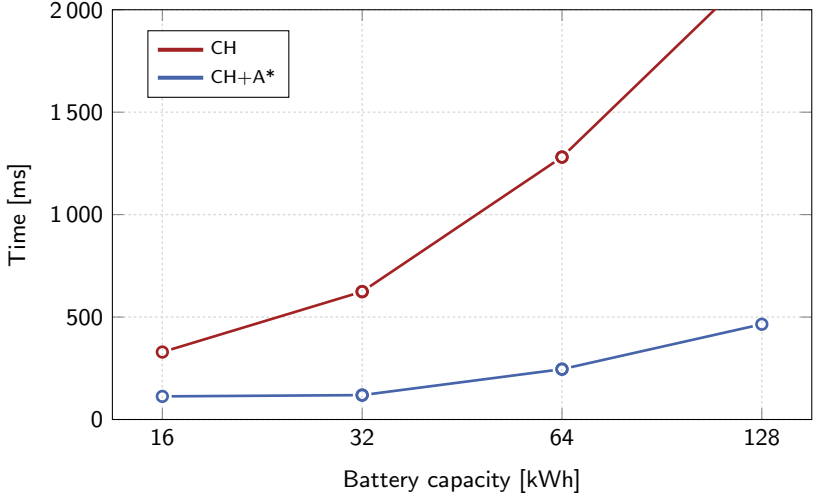
| Techniques | | | Peugeot iOn | | | Artificial | | |
|---|---|---|---|---|---|---|---|---|
| $G_S$ | CH | A* | Prepr. [s] | # V. Sc. | Q. [ms] | Prepr. [s] | # V. Sc. | Q. [ms] |
| ○ | ○ | ○ | — | 8 895k | 20 161 | — | 11 034k | 32 929 |
| ● | ○ | ○ | 1 487 | 760k | 710 | 15 062 | 7 754k | 6 286 |
| ● | ● | ○ | 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

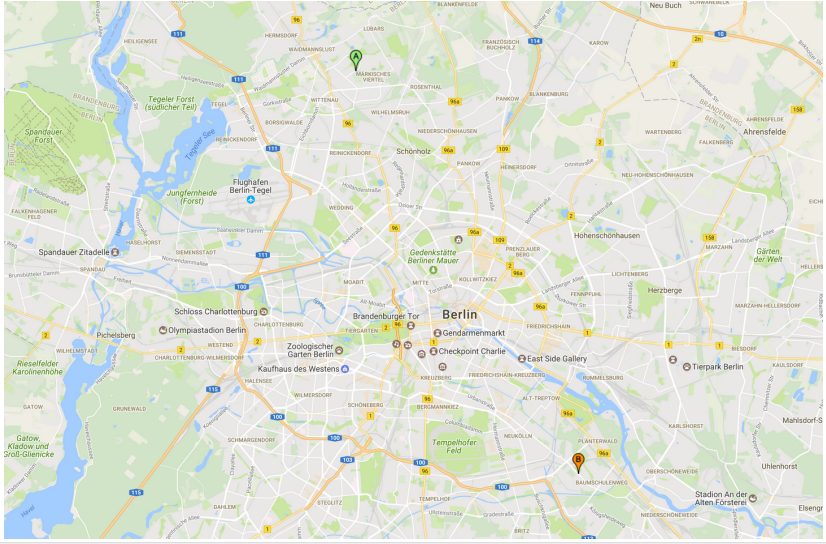
**Problem:** Wir versuchen Energie zu sparen, selbst wenn:

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

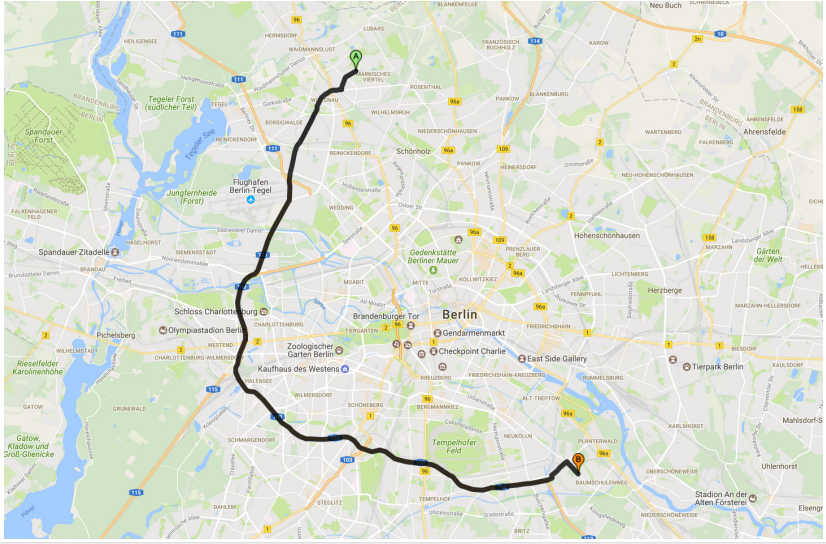
**Alternativen?**

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

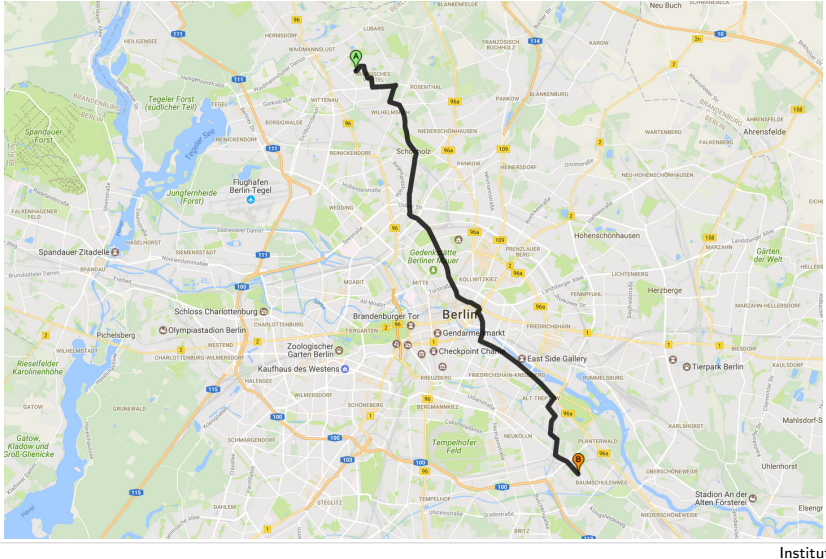
# Beispielrouten



# Beispielrouten



# Beispielroutenen



# 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?

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 $> W$

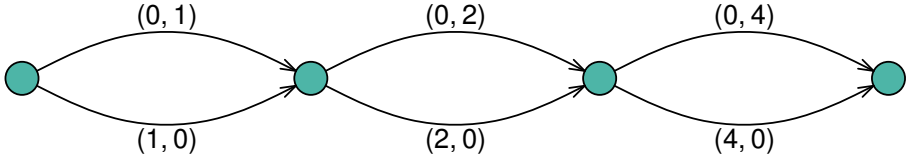
# Wdh: Multi-Criteria Dijkstra für CSP

**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



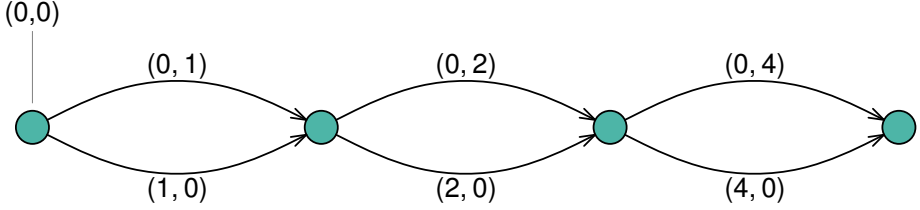
# Wdh: Multi-Criteria Dijkstra für CSP

**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



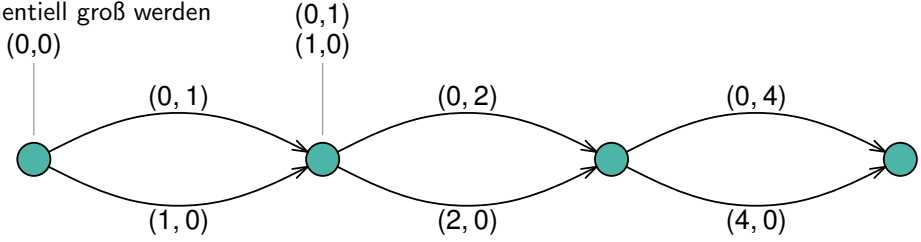
# Wdh: Multi-Criteria Dijkstra für CSP

**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



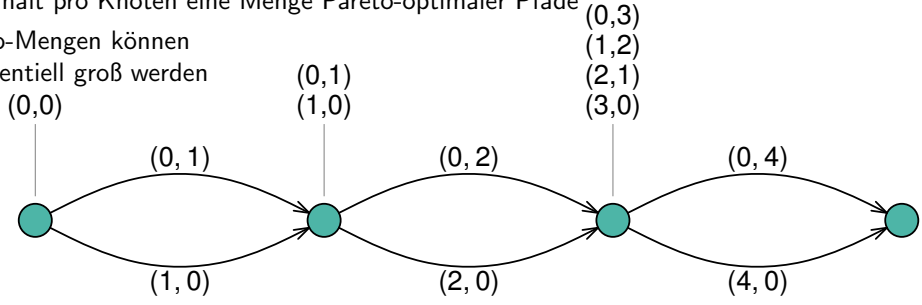
# Wdh: Multi-Criteria Dijkstra für CSP

**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



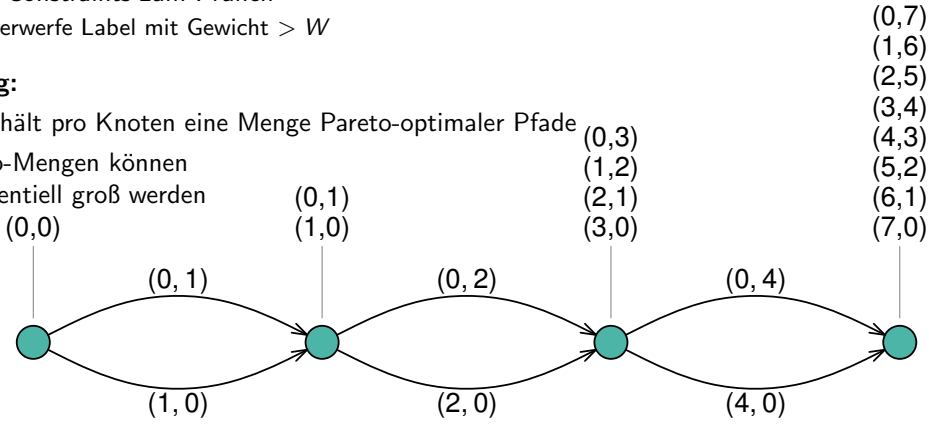
# Wdh: Multi-Criteria Dijkstra für CSP

**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)

# Ladestopps

# Ladestationen

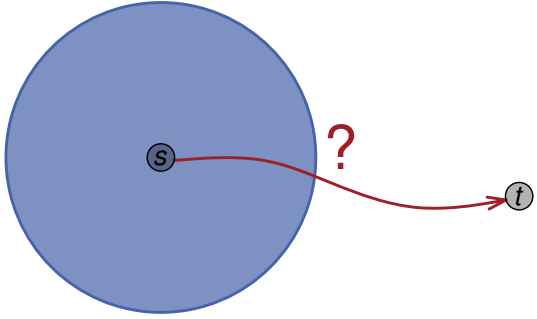
**Finde schnellste Route von $s$ nach $t$:**


Erreichbares Gebiet

Ladestation

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**

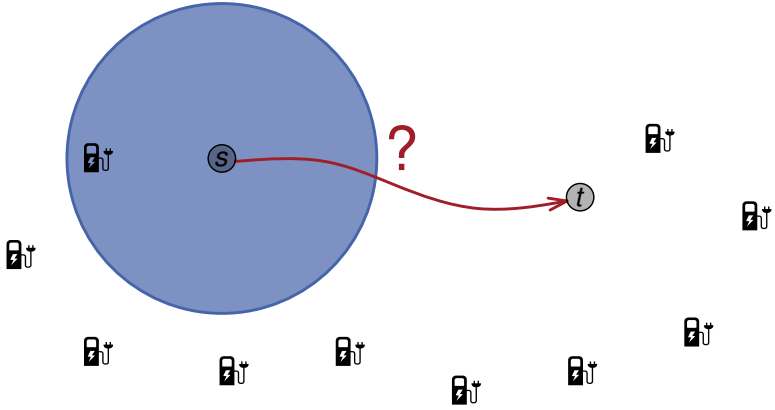



Erreichbares Gebiet

Ladestation

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**

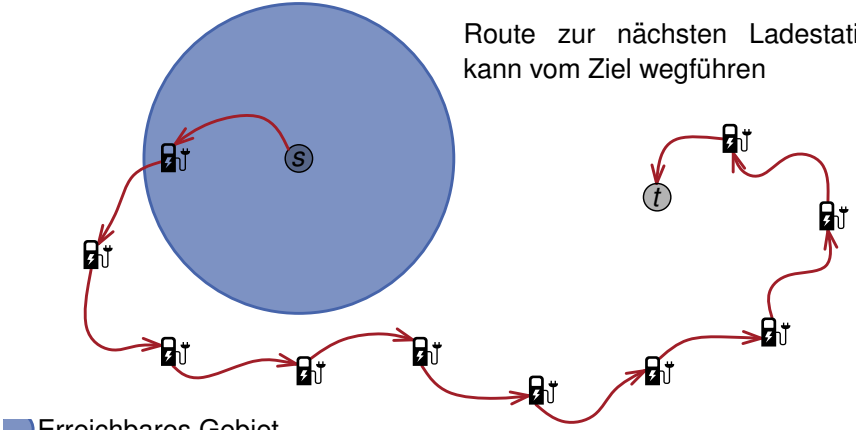



Erreichbares Gebiet

Ladestation

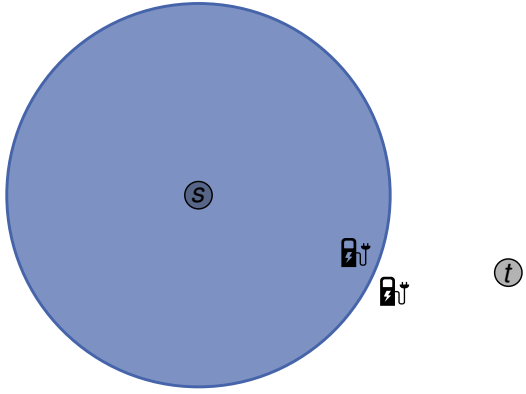
# Ladestationen

**Finde schnellste Route von *s* nach *t*:**



# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**

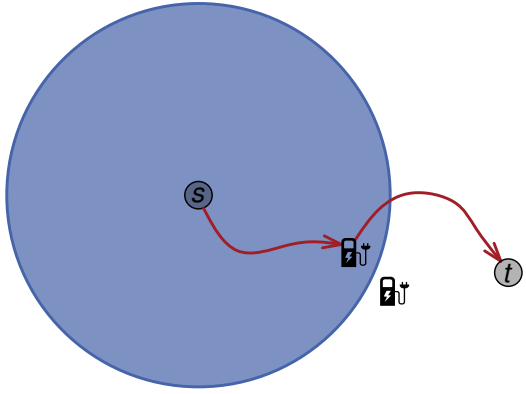



Erreichbares Gebiet

Ladestation

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**

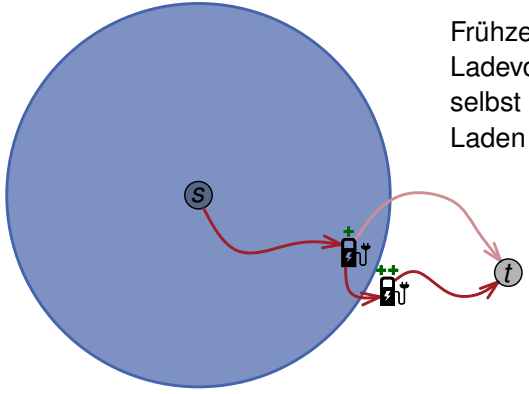



Erreichbares Gebiet

Ladestation

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**




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

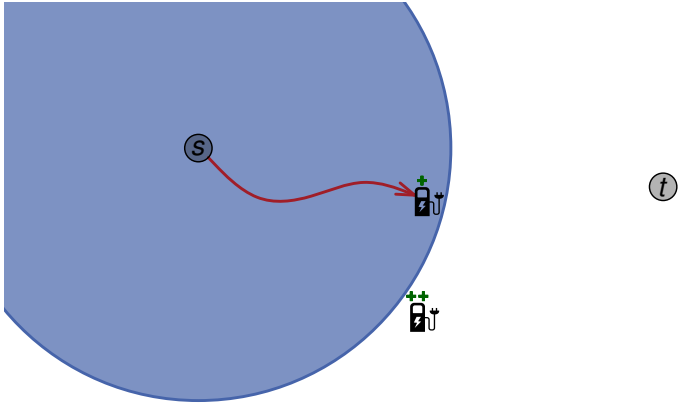
Erreichbares Gebiet

Ladestation

Supercharger/Swapping Station

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**



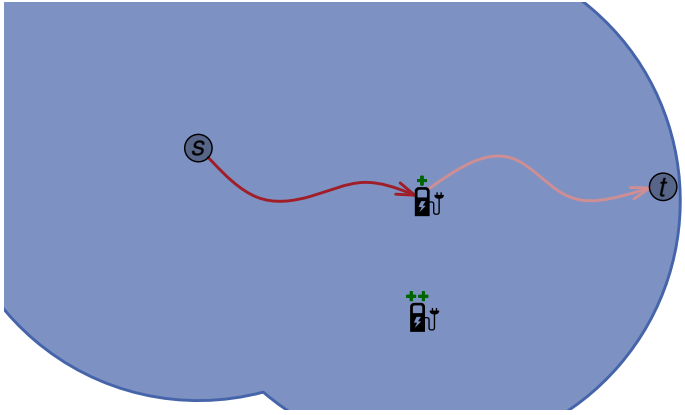

Erreichbares Gebiet

Ladestation

Supercharger/Swapping Station

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**



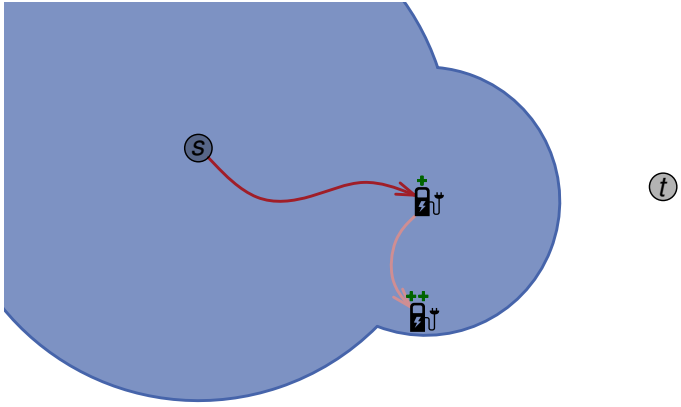

Erreichbares Gebiet

Ladestation

Supercharger/Swapping Station

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**



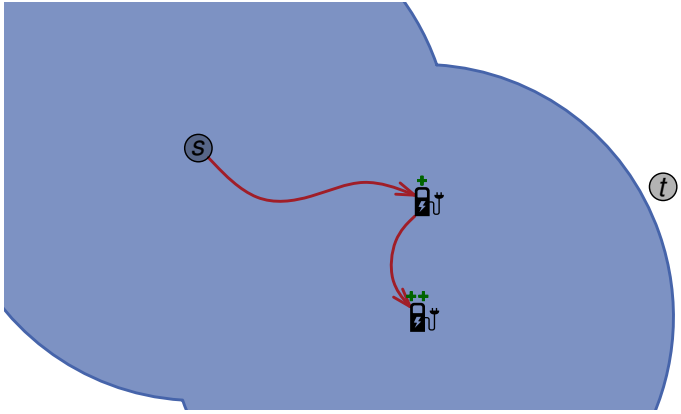

Erreichbares Gebiet

Ladestation

Supercharger/Swapping Station

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**



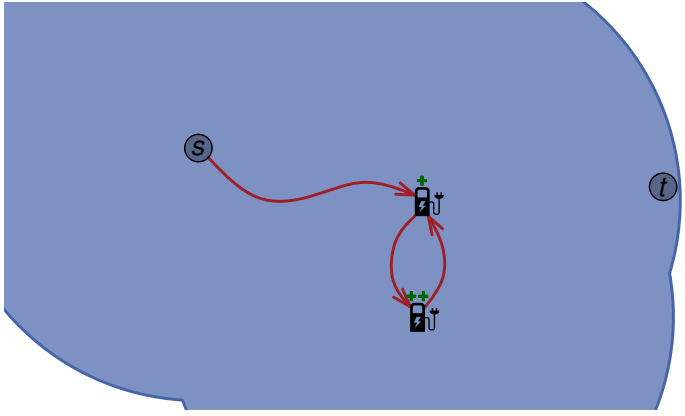

Erreichbares Gebiet

Ladestation

Supercharger/Swapping Station

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**



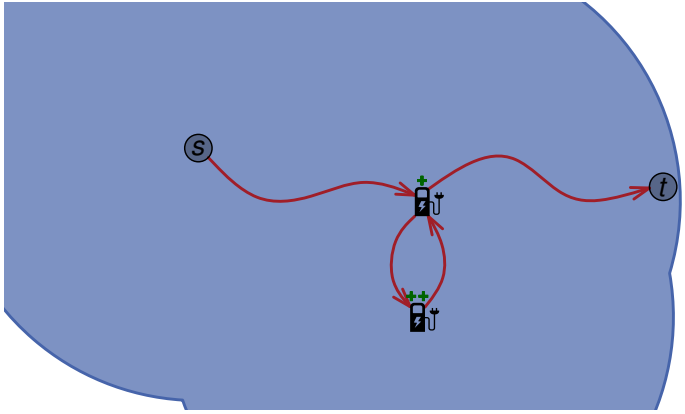

Erreichbares Gebiet

Ladestation

Supercharger/Swapping Station

# Ladestationen

**Finde schnellste Route von $s$ nach $t$:**




Zyklen möglich

Erreichbares Gebiet

Ladestation

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

# Ladefunktionen

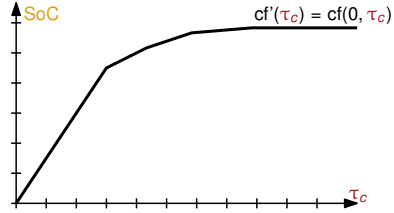
**Formal:**

- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC $b_s$ und
  - gewünschte Ladezeit $\tau_c$ 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 $\mathrm{cf'}\colon \mathbb{R}_{\geq 0} \to [0, M]$

$$\mathrm{cf}(b, \tau_c) := \mathrm{cf'}(\tau_c + \mathrm{cf'}^{-1}(b))$$



# Ladefunktionen

**Formal:**

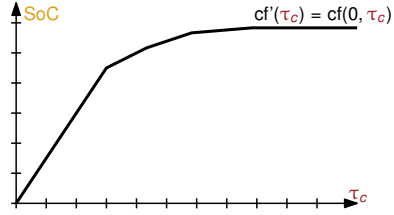
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC $b_s$ und
  - gewünschte Ladezeit $\tau_c$ auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

**Anmerkung:** Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf'}\colon \mathbb{R}_{\geq 0} \to [0, M]$

$$\mathrm{cf}(b, \tau_c) := \mathrm{cf'}(\tau_c + \mathrm{cf'}^{-1}(b))$$

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



# Ladefunktionen

**Formal:**

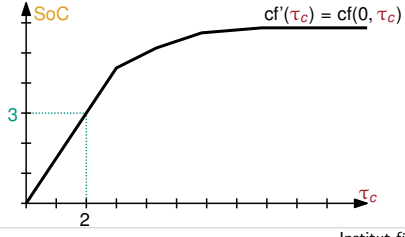
- Eine Funktion cf: $[0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC $b_s$ und
  - gewünschte Ladezeit $\tau_c$ auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

**Anmerkung:** Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion cf': $\mathbb{R}_{\geq 0} \to [0, M]$

$$\text{cf}(b, \tau_c) := \text{cf'}(\tau_c + \text{cf'}^{-1}(b))$$

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



# Ladefunktionen

**Formal:**

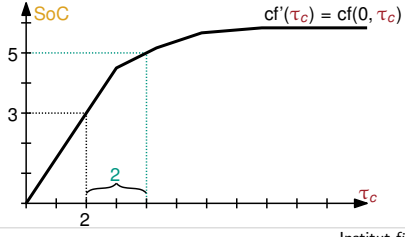
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC $b_s$ und
  - gewünschte Ladezeit $\tau_c$ auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

**Anmerkung:** Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf'}\colon \mathbb{R}_{\geq 0} \to [0, M]$

$$\mathrm{cf}(b, \tau_c) := \mathrm{cf'}(\tau_c + \mathrm{cf'}^{-1}(b))$$

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



# Ladefunktionen

**Formal:**

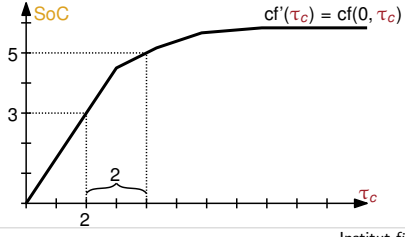
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - initialen SoC $b_s$ und
  - gewünschte Ladezeit $\tau_c$ auf
  - durch Laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

**Anmerkung:** Realistische Ladefunktionen darstellbar durch:

- Univariate Funktion $\mathrm{cf'}\colon \mathbb{R}_{\geq 0} \to [0, M]$

$$\mathrm{cf}(b, \tau_c) := \mathrm{cf'}(\tau_c + \mathrm{cf'}^{-1}(b))$$

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



# Charging Function Propagation (CFP)

**Algorithmus:**

- Basiert auf Dijkstras Algorithmus bzw. MCD
- Solange keine Ladestation besucht: Label = Tupel (Reisezeit, SoC)
- Battery Constraints, Pareto-Optimierung wie bisher

# Charging Function Propagation (CFP)

**Algorithmus:**

- Basiert auf Dijkstras Algorithmus bzw. MCD
- Solange keine Ladestation besucht: Label = Tupel (Reisezeit, SoC)
- Battery Constraints, Pareto-Optimierung wie bisher

**Problem:** Wenn Ladestation erreicht: Wie lange laden?

- Hängt vom gewähltem Pfad zu $t$ ab
- Optimaler SoC zum Weiterfahren unbekannt

# Charging Function Propagation (CFP)

**Algorithmus:**

- Basiert auf Dijkstras Algorithmus bzw. MCD
- Solange keine Ladestation besucht: Label = Tupel (Reisezeit, SoC)
- Battery Constraints, Pareto-Optimierung wie bisher

**Problem:** Wenn Ladestation erreicht: Wie lange laden?

- Hängt vom gewähltem Pfad zu $t$ ab
- Optimaler SoC zum Weiterfahren unbekannt

**Lösung:**

- Verschiebe die Entscheidung auf später!
- Merke zuletzt gesehene Ladestation

# Charging Function Propagation (CFP)

**Label:** Ein Label $\ell$ am Knoten $v$ ist ein Tupel $(\tau_t, b_u, u, c_{(u,\dots,v)})$ mit:

- Reisezeit $\tau_t$ von $s$ nach $v$ (Inklusive Ladezeiten außer an $u$)
- SoC $b_u$, mit dem die letzte Ladestation ($u$) erreicht wurde
- Zuletzt passierte Ladestation $u$ (Initial $\perp$)
- Verbrauchsfunktion $c_{(u,\dots,v)}$ für den Pfad von $u$ nach $v$ (Initial $\perp$)

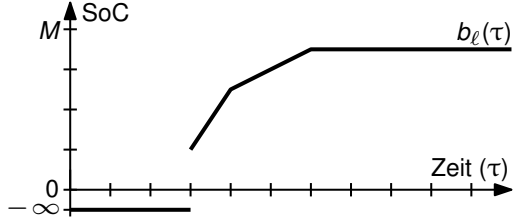
# Charging Function Propagation (CFP)

**Label:** Ein Label $\ell$ am Knoten $v$ ist ein Tupel $(\tau_t, b_u, u, c_{(u,\dots,v)})$ mit:

- Reisezeit $\tau_t$ von $s$ nach $v$ (Inklusive Ladezeiten außer an $u$)
- SoC $b_u$, mit dem die letzte Ladestation $(u)$ erreicht wurde
- Zuletzt passierte Ladestation $u$ (Initial $\perp$)
- Verbrauchsfunktion $c_{(u,\dots,v)}$ für den Pfad von $u$ nach $v$ (Initial $\perp$)

**Interpretation:**

- Label beschreibt eine *verschobene* Ladefunktion
- Bildet Reisezeit auf SoC ab (daher auch SoC-Funktion genannt)
- Funktion repräsentiert Menge von Pareto-optimalen Punkten
- Definition der SoC-Funktion $b_\ell(\tau)$:
  $b_\ell(\tau) := b' - c_{(u,\dots,v)}(b')$ mit
  $b' := \mathrm{cf}_u(b_u, \tau - \tau_t)$



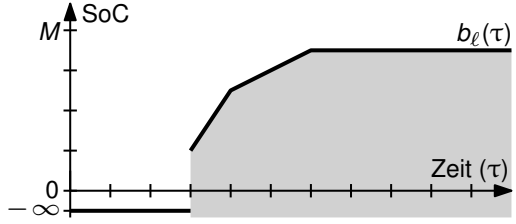
# Charging Function Propagation (CFP)

**Label:** Ein Label $\ell$ am Knoten $v$ ist ein Tupel $(\tau_t, b_u, u, c_{(u,\dots,v)})$ mit:

- Reisezeit $\tau_t$ von $s$ nach $v$ (Inklusive Ladezeiten außer an $u$)
- SoC $b_u$, mit dem die letzte Ladestation $(u)$ erreicht wurde
- Zuletzt passierte Ladestation $u$ (Initial $\perp$)
- Verbrauchsfunktion $c_{(u,\dots,v)}$ für den Pfad von $u$ nach $v$ (Initial $\perp$)

**Interpretation:**

- Label beschreibt eine *verschobene* Ladefunktion
- Bildet Reisezeit auf SoC ab (daher auch SoC-Funktion genannt)
- Funktion repräsentiert Menge von Pareto-optimalen Punkten
- Definition der SoC-Funktion $b_\ell(\tau)$:
  $b_\ell(\tau) := b' - c_{(u,\dots,v)}(b')$ mit
  $b' := \mathrm{cf}_u(b_u, \tau - \tau_t)$



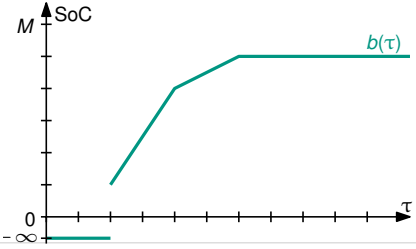
# Charging Function Propagation (CFP)

**Kantenrelaxierung:** (Label $\ell = (\tau_t, b_u, u, c_{(u,\ldots,v)})$)

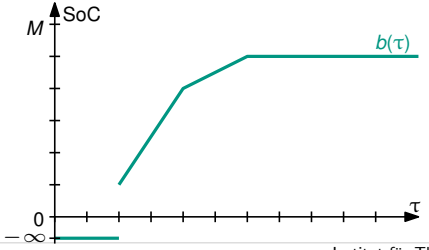
- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

**Formal:** $\tau_t \leftarrow \tau_t + \tau_d(e)$ und $c_{(u,\ldots,w)} \leftarrow c_{(u,\ldots,v)} \circ c_e$

$\tau_d(e) = 3, \gamma(e) = 2$




$\tau_d(e) = 3, \gamma(e) = -2$



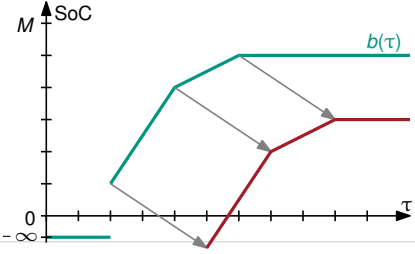
# Charging Function Propagation (CFP)

**Kantenrelaxierung:** (Label $\ell = (\tau_t, b_u, u, c_{(u,\ldots,v)})$)

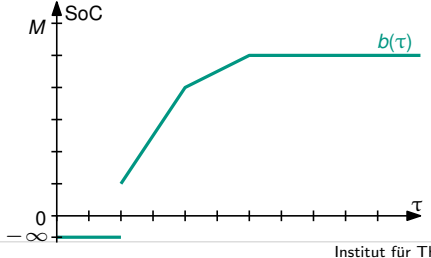
- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

**Formal:** $\tau_t \leftarrow \tau_t + \tau_d(e)$ und $c_{(u,\ldots,w)} \leftarrow c_{(u,\ldots,v)} \circ c_e$

$\tau_d(e) = 3, \gamma(e) = 2$




$\tau_d(e) = 3, \gamma(e) = -2$

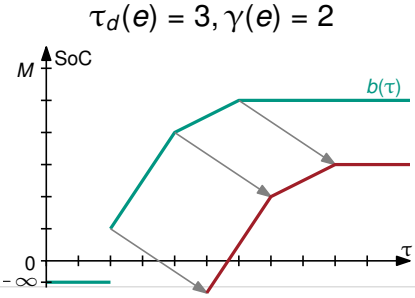


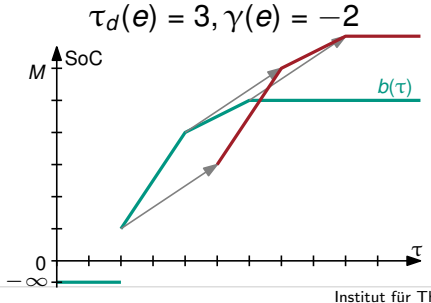
# Charging Function Propagation (CFP)

**Kantenrelaxierung:** (Label $\ell = (\tau_t, b_u, u, c_{(u,\ldots,v)})$)

- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

**Formal:** $\tau_t \leftarrow \tau_t + \tau_d(e)$ und $c_{(u,\ldots,w)} \leftarrow c_{(u,\ldots,v)} \circ c_e$



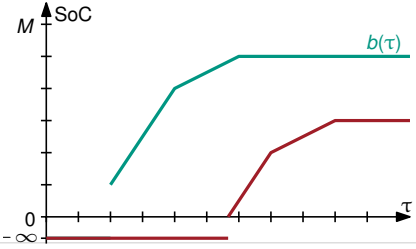
# Charging Function Propagation (CFP)

**Kantenrelaxierung:** (Label $\ell = (\tau_t, b_u, u, c_{(u,\ldots,v)})$)

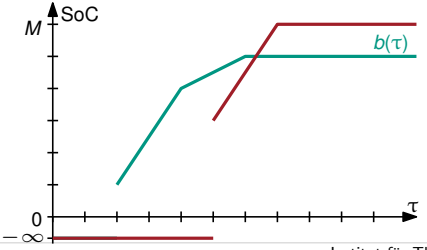
- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

**Formal:** $\tau_t \leftarrow \tau_t + \tau_d(e)$ und $c_{(u,\ldots,w)} \leftarrow c_{(u,\ldots,v)} \circ c_e$

$\tau_d(e) = 3, \gamma(e) = 2$




$\tau_d(e) = 3, \gamma(e) = -2$



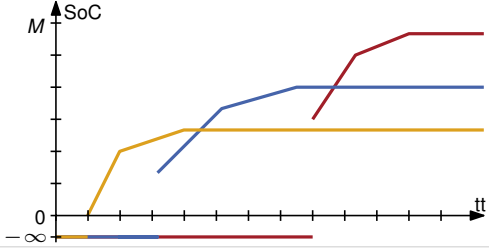
# Charging Function Propagation (CFP)

**Dominanz von SoC-Funktionen:**

- Für SoC-Funktionen $b_\ell(\tau)$ und $b_{\ell'}(\tau)$ definieren wir Dominanz ($\propto$) als:

$$b_\ell(\tau) \propto b_{\ell'}(\tau) \quad \Leftrightarrow \quad \forall \tau \geq 0 \colon b_\ell(\tau) \geq b_{\ell'}(\tau)$$

- Pro Knoten eine Menge von SoC-Funktionen
- Kantenrelaxierung erzeugt neues Label
  $\Rightarrow$ Überprüfe Dominanz (nur paarweise)
  Lösche dominierte Label



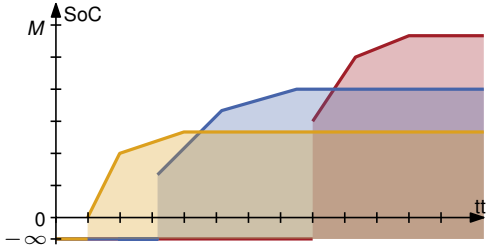
# Charging Function Propagation (CFP)

**Dominanz von SoC-Funktionen:**

- Für SoC-Funktionen $b_\ell(\tau)$ und $b_{\ell'}(\tau)$ definieren wir Dominanz ($\propto$) als:

$$b_\ell(\tau) \propto b_{\ell'}(\tau) \quad \Leftrightarrow \quad \forall \tau \geq 0 \colon b_\ell(\tau) \geq b_{\ell'}(\tau)$$

- Pro Knoten eine Menge von SoC-Funktionen
- Kantenrelaxierung erzeugt neues Label
  $\Rightarrow$ Überprüfe Dominanz (nur paarweise)
  Lösche dominierte Label



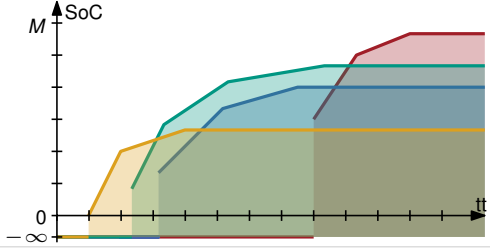
# Charging Function Propagation (CFP)

**Dominanz von SoC-Funktionen:**

- Für SoC-Funktionen $b_\ell(\tau)$ und $b_{\ell'}(\tau)$ definieren wir Dominanz ($\propto$) als:

$$b_\ell(\tau) \propto b_{\ell'}(\tau) \quad \Leftrightarrow \quad \forall \tau \geq 0 \colon b_\ell(\tau) \geq b_{\ell'}(\tau)$$

- Pro Knoten eine Menge von SoC-Funktionen
- Kantenrelaxierung erzeugt neues Label
  - $\Rightarrow$ Überprüfe Dominanz (nur paarweise)
  - Lösche dominierte Label



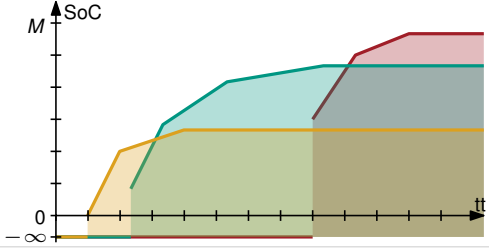
# Charging Function Propagation (CFP)

**Dominanz von SoC-Funktionen:**

- Für SoC-Funktionen $b_\ell(\tau)$ und $b_{\ell'}(\tau)$ definieren wir Dominanz ($\propto$) als:

$$b_\ell(\tau) \propto b_{\ell'}(\tau) \quad \Leftrightarrow \quad \forall \tau \geq 0 \colon b_\ell(\tau) \geq b_{\ell'}(\tau)$$

- Pro Knoten eine Menge von SoC-Funktionen
- Kantenrelaxierung erzeugt neues Label
  $\Rightarrow$ Überprüfe Dominanz (nur paarweise)
  Lösche dominierte Label



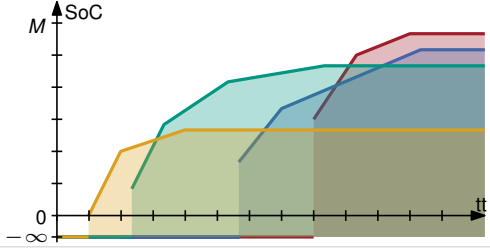
# Charging Function Propagation (CFP)

**Dominanz von SoC-Funktionen:**

- Für SoC-Funktionen $b_\ell(\tau)$ und $b_{\ell'}(\tau)$ definieren wir Dominanz ($\propto$) als:

$$b_\ell(\tau) \propto b_{\ell'}(\tau) \quad \Leftrightarrow \quad \forall \tau \geq 0 \colon b_\ell(\tau) \geq b_{\ell'}(\tau)$$

- Pro Knoten eine Menge von SoC-Funktionen
- Kantenrelaxierung erzeugt neues Label
  - $\Rightarrow$ Überprüfe Dominanz (nur paarweise)
  - Lösche dominierte Label



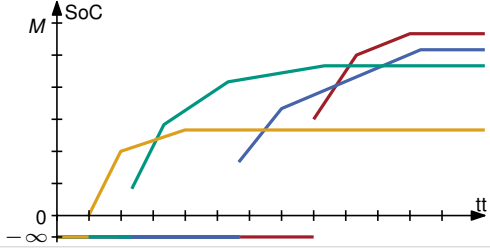
# Charging Function Propagation (CFP)

**Dominanz von SoC-Funktionen:**

- Für SoC-Funktionen $b_\ell(\tau)$ und $b_{\ell'}(\tau)$ definieren wir Dominanz ($\propto$) als:

$$b_\ell(\tau) \propto b_{\ell'}(\tau) \quad \Leftrightarrow \quad \forall \tau \geq 0 \colon b_\ell(\tau) \geq b_{\ell'}(\tau)$$

- Pro Knoten eine Menge von SoC-Funktionen
- Kantenrelaxierung erzeugt neues Label
  $\Rightarrow$ Überprüfe Dominanz (nur paarweise)
  Lösche dominierte Label



# Charging Function Propagation (CFP)

**Settling von Ladestationen:**

- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

# Charging Function Propagation (CFP)

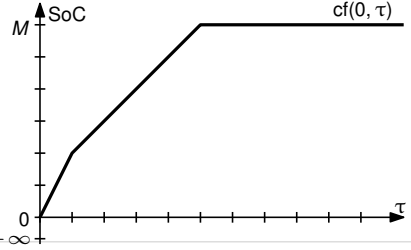
**Settling von Ladestationen:**

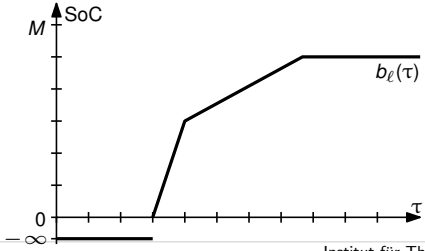
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

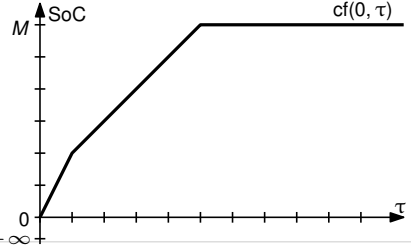
**Settling von Ladestationen:**

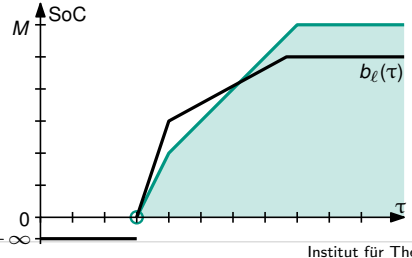
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

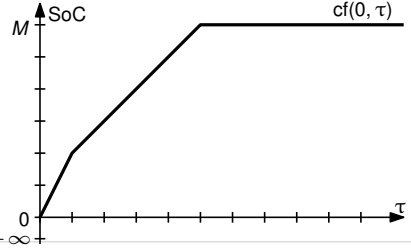
**Settling von Ladestationen:**

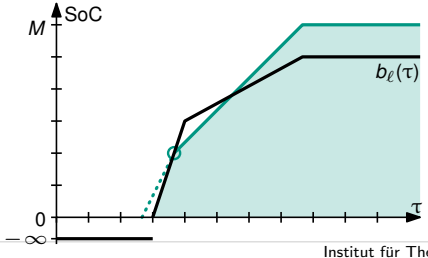
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

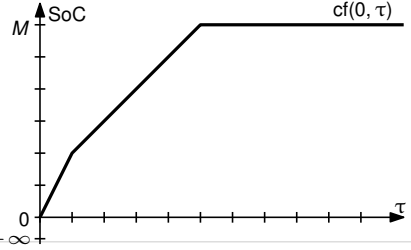
**Settling von Ladestationen:**

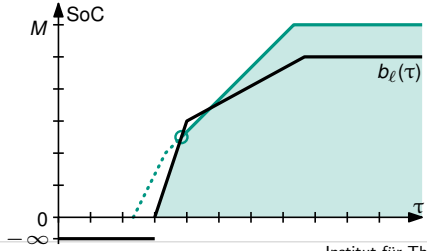
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

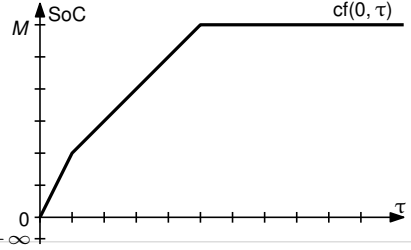
**Settling von Ladestationen:**

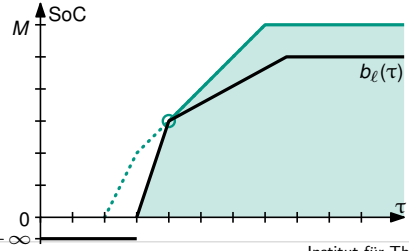
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

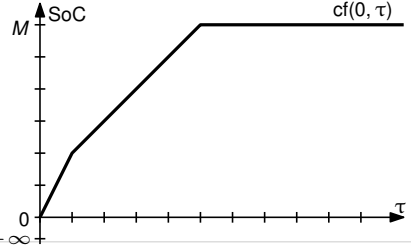
**Settling von Ladestationen:**

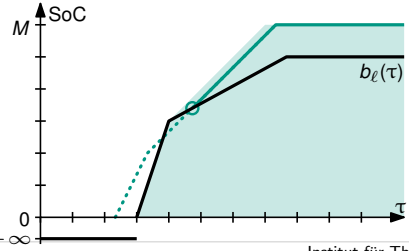
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

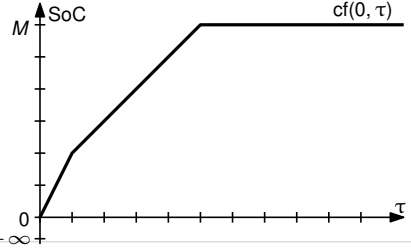
**Settling von Ladestationen:**

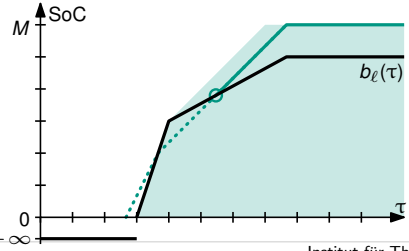
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

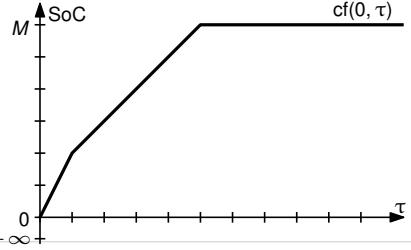
**Settling von Ladestationen:**

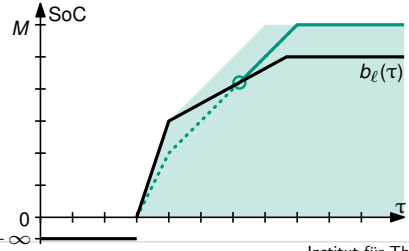
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

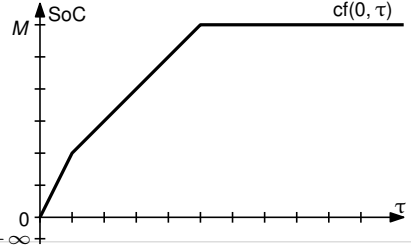
**Settling von Ladestationen:**

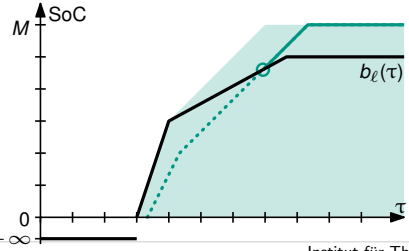
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

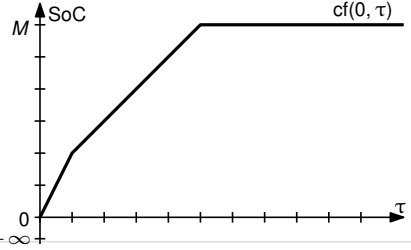
**Settling von Ladestationen:**

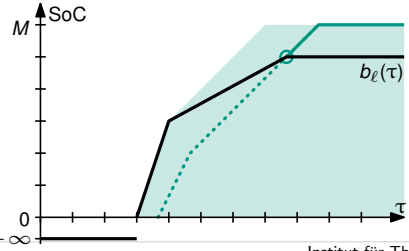
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

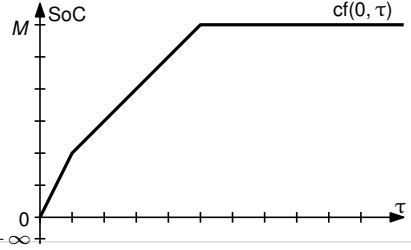
**Settling von Ladestationen:**

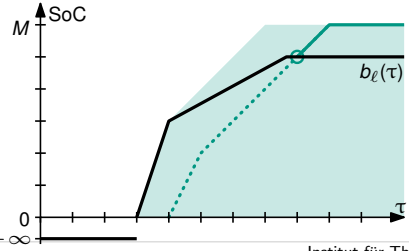
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

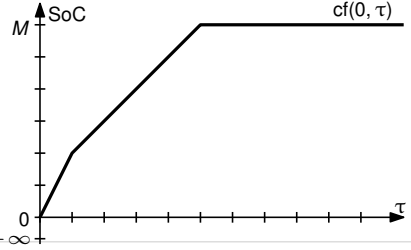
**Settling von Ladestationen:**

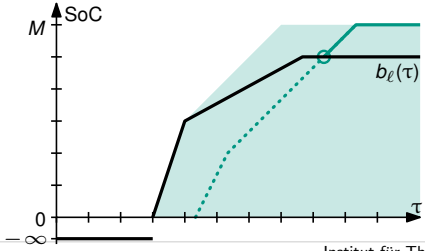
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

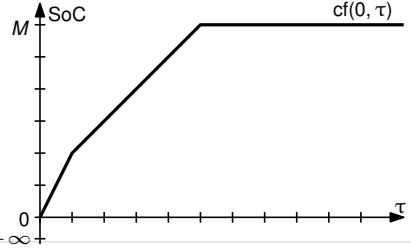
**Settling von Ladestationen:**

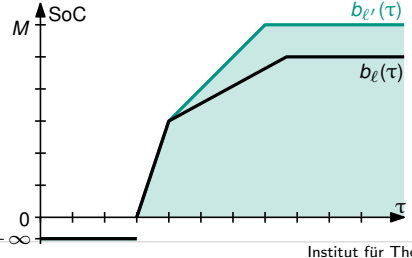
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

**Aber:** Wechsel der Station nur sinnvoll, wenn neue Station besser



# Charging Function Propagation (CFP)

**Settling von Ladestationen:**

- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme $\tau_c$ für die letzte Station

**Problem:**

- Am ursprünglichen Problem hat sich nichts geändert
- Auch für vorletzte Ladestation ist die Ladezeit unklar

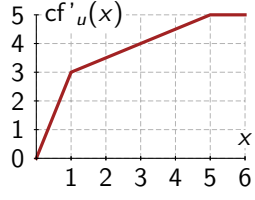
Wechsel der Ladestation lohnt sich nur an Stützpunkten von $b_\ell(\tau)$

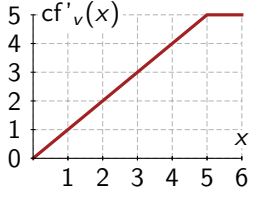
**Gegeben:**

- Label $\ell = (\tau_t, b_u, u, c_{(u,\dots,v)})$ an Knoten $v$
- $v$ ist Ladestation
- $\tau$ ist Stützstelle von $b_\ell$

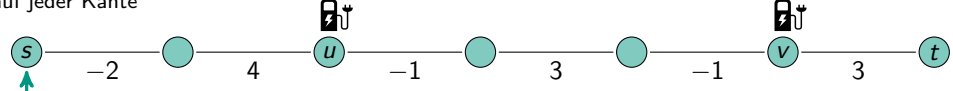
$\Rightarrow$ Erzeuge neues Label $\ell' = (\tau, b_\ell(\tau), v, 0)$

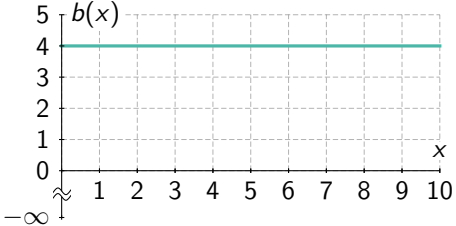
# Beispiel



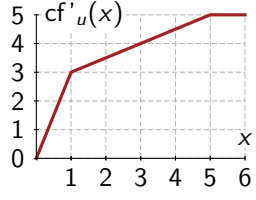



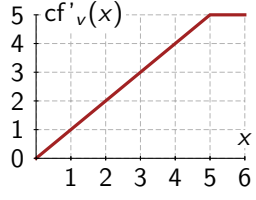

Annahme:
Fahrzeit 1
auf jeder Kante



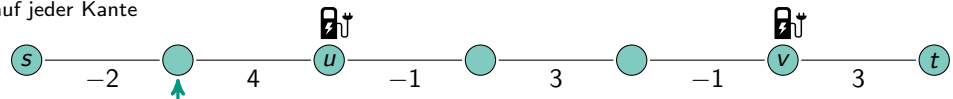



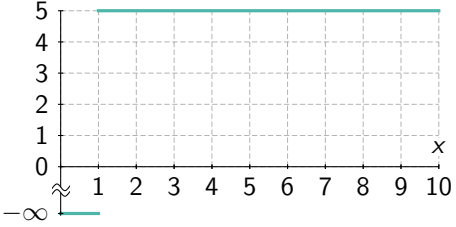
# Beispiel



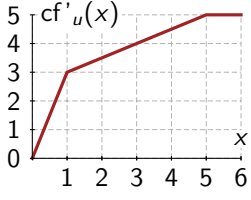

Annahme:
Fahrzeit 1
auf jeder Kante

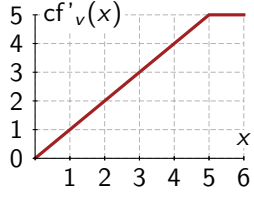


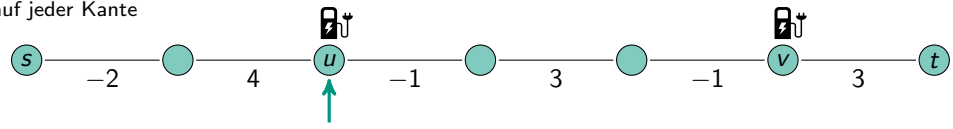



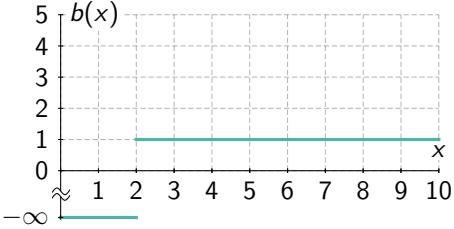
# Beispiel

Annahme:
Fahrzeit 1
auf jeder Kante

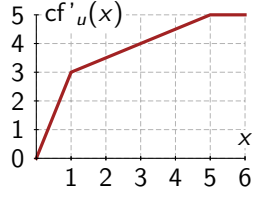


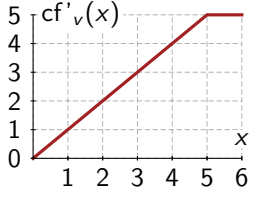



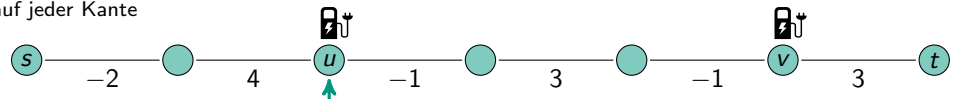



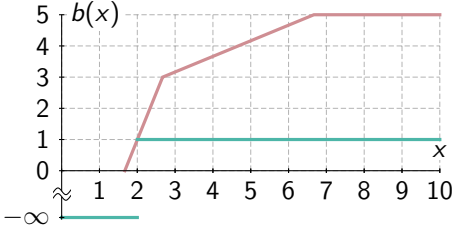
# Beispiel



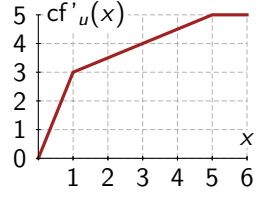



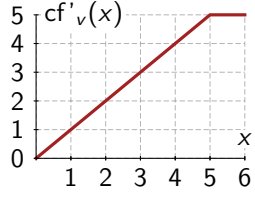

Annahme:
Fahrzeit 1
auf jeder Kante



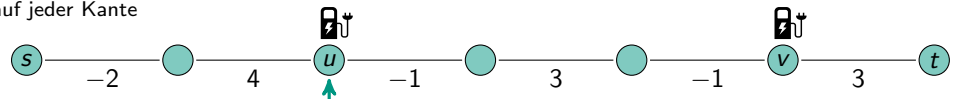



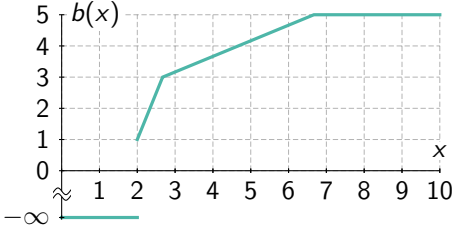
# Beispiel



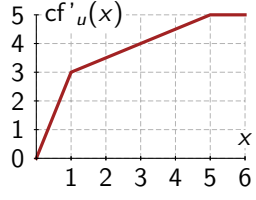

Annahme:
Fahrzeit 1
auf jeder Kante

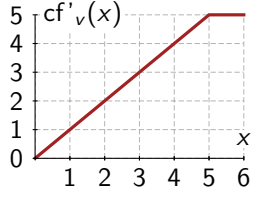


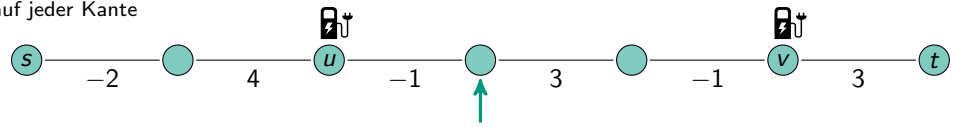



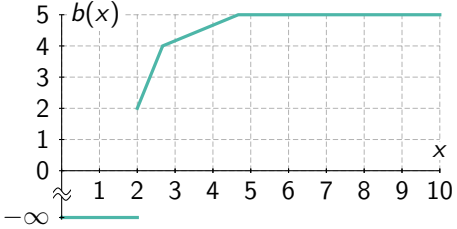
# Beispiel

Annahme:
Fahrzeit 1
auf jeder Kante



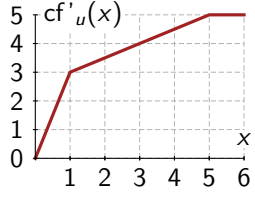



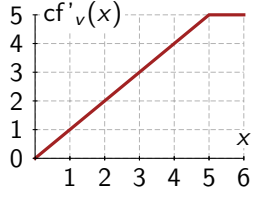



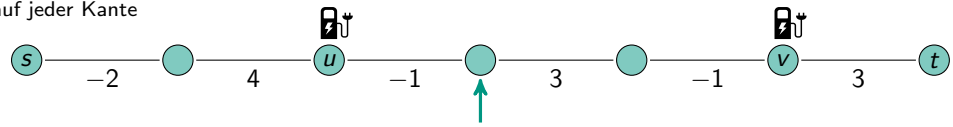



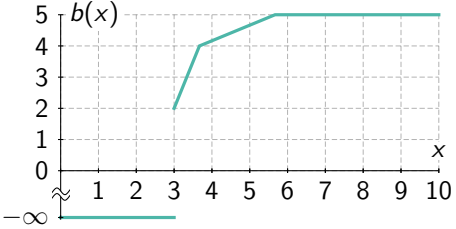
# Beispiel

Annahme:
Fahrzeit 1
auf jeder Kante

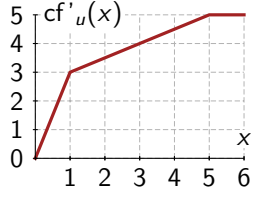


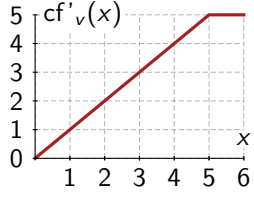



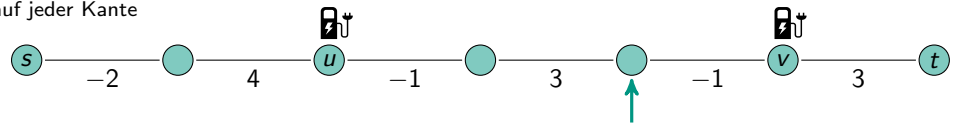



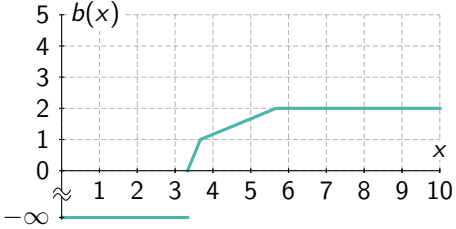
# Beispiel



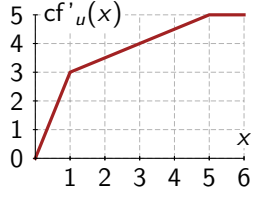

Annahme:
Fahrzeit 1
auf jeder Kante

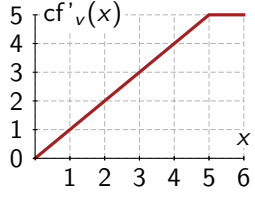


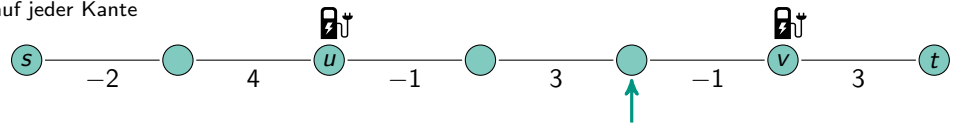



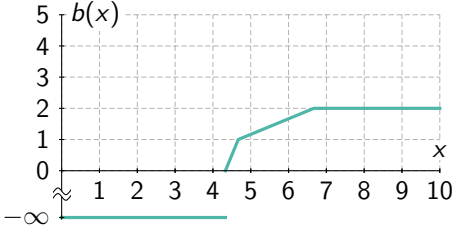
# Beispiel

Annahme:
Fahrzeit 1
auf jeder Kante



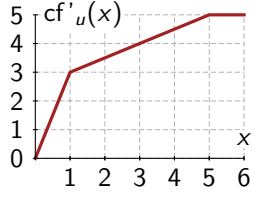



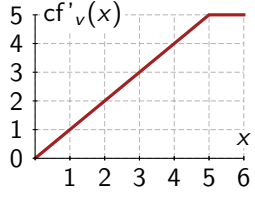



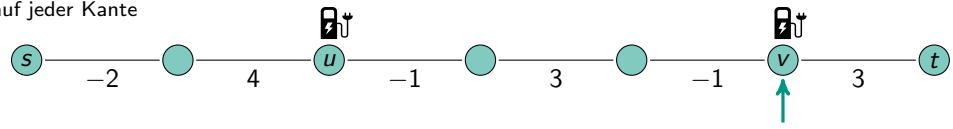



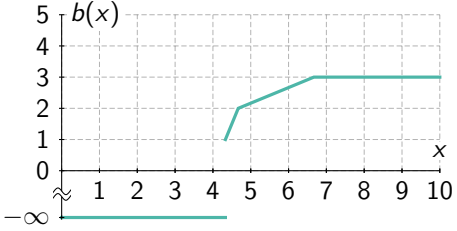
# Beispiel

Annahme:
Fahrzeit 1
auf jeder Kante



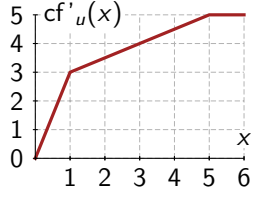



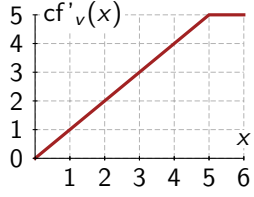



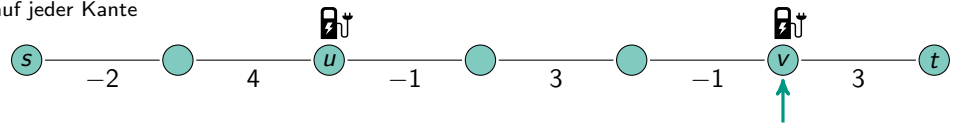



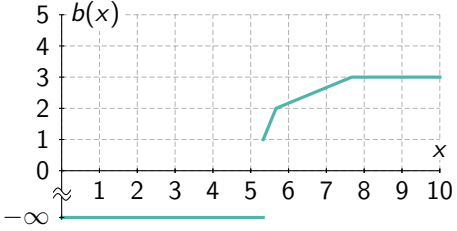
# Beispiel

Annahme:
Fahrzeit 1
auf jeder Kante

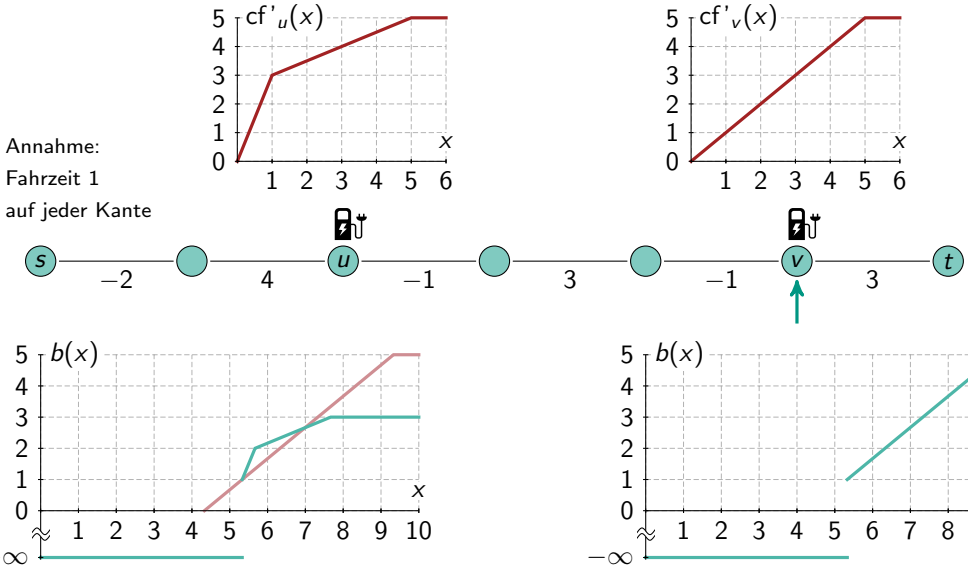


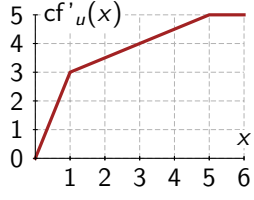



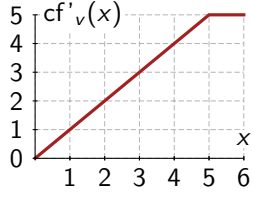



# Beispiel

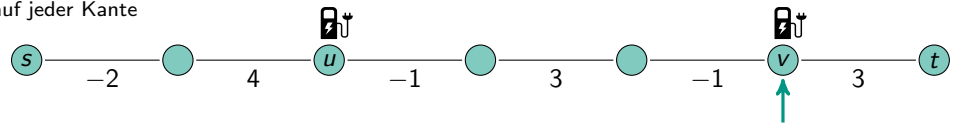

cf'u(x)
cf'v(x)
Annahme:
Fahrzeit 1
auf jeder Kante
s
u
v
t
−2
4
−1
3
−1
3
b(x)
b(x)
x
−∞

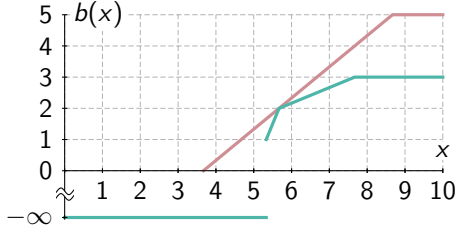
# Beispiel

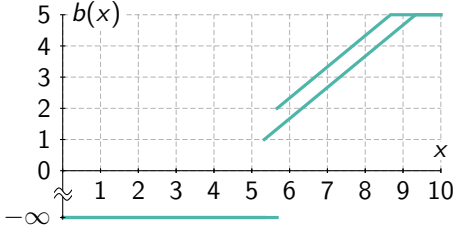


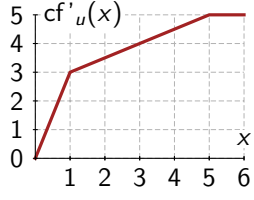

Annahme:
Fahrzeit 1
auf jeder Kante

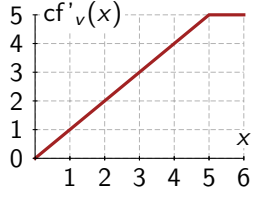


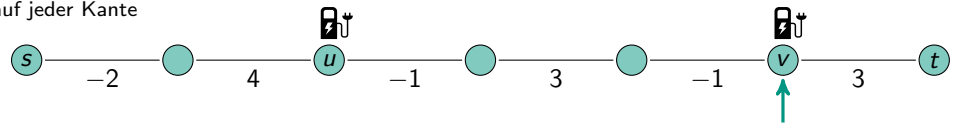



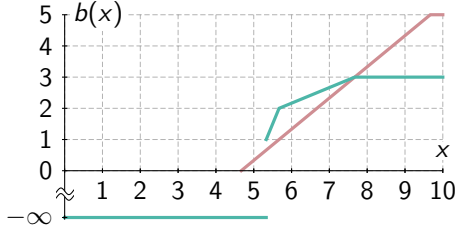
# Beispiel

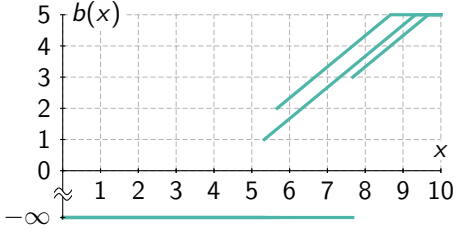


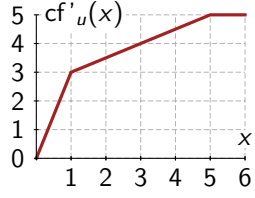

Annahme:
Fahrzeit 1
auf jeder Kante

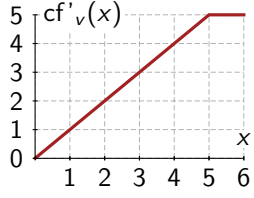


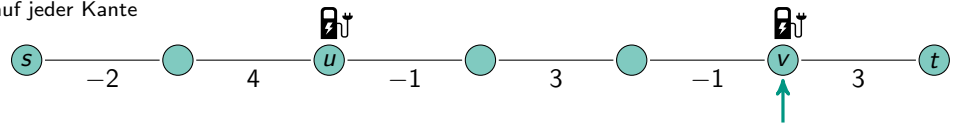



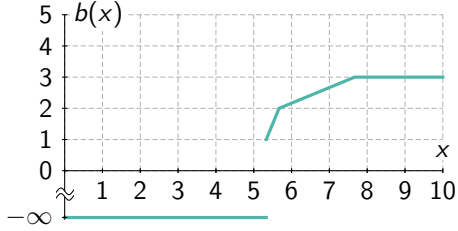
# Beispiel

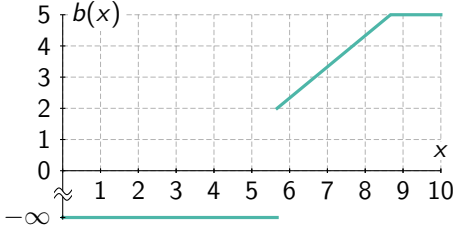


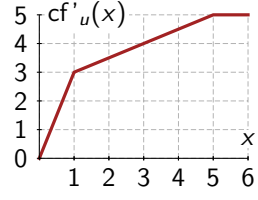

Annahme:
Fahrzeit 1
auf jeder Kante

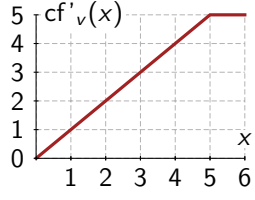


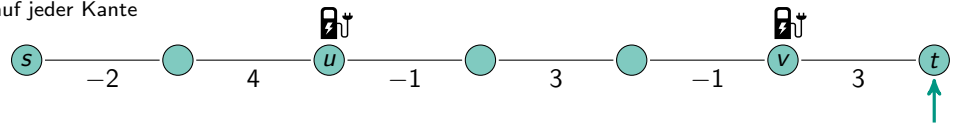



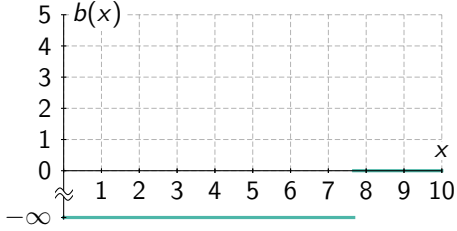
# Beispiel

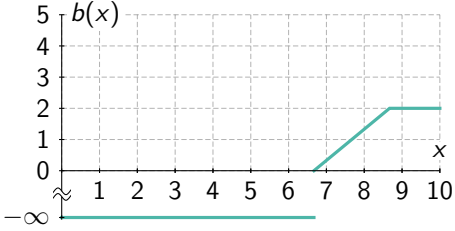


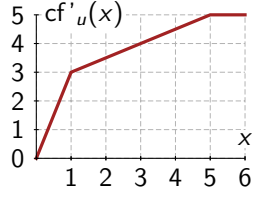

Annahme:
Fahrzeit 1
auf jeder Kante

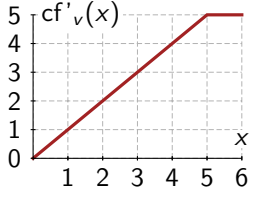


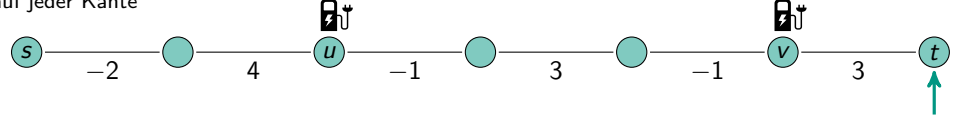



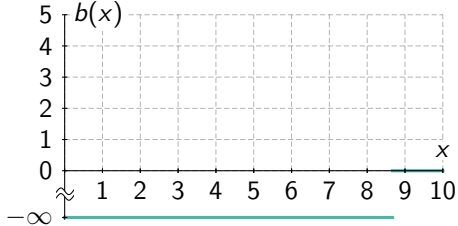
# Beispiel

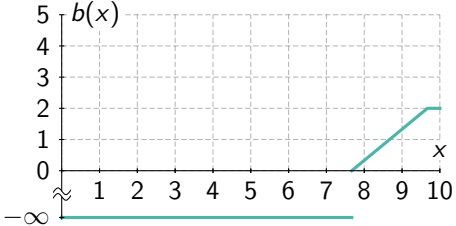



Annahme:
Fahrzeit 1
auf jeder Kante

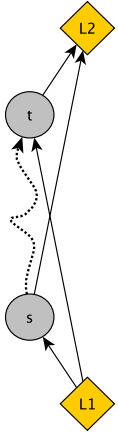


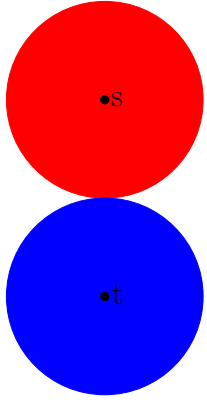



# CFP – Beschleunigungstechniken

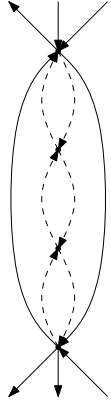
## Landmarken



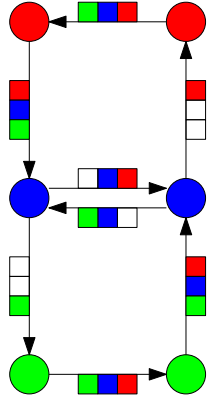

## Bidirektionale Suche




## Kontraktion



## Arc-Flags



# CFP & CH

**Probleme:**

- Kontraktion muss Kürzeste-Wege-Distanzen erhalten
  - SoC während Vorberechnung unbekannt
  - Battery Constraints müssen beachtet werden
  - Ladefunktionen müssen berücksichtigt werden

# CFP & CH

**Probleme:**

- Kontraktion muss Kürzeste-Wege-Distanzen erhalten
  - SoC während Vorberechnung unbekannt
  - Battery Constraints müssen beachtet werden
  - Ladefunktionen müssen berücksichtigt werden

**Lösung:**

- Benutze Verbrauchsfunktionen
  (Battery Constraints in Kantengewichten enthalten)
- Ladestation per Definition wichtig
  (Nicht kontrahieren $\Rightarrow$ Core-Graph)

# CFP & CH

**Probleme:**

- Kontraktion muss Kürzeste-Wege-Distanzen erhalten
  - SoC während Vorberechnung unbekannt
  - Battery Constraints müssen beachtet werden
  - Ladefunktionen müssen berücksichtigt werden

**Lösung:**

- Benutze Verbrauchsfunktionen
  (Battery Constraints in Kantengewichten enthalten)
- Ladestation per Definition wichtig
  (Nicht kontrahieren $\Rightarrow$ Core-Graph)

**Aber:**

- Shortcuts repräsentieren jeweils Pareto-Mengen (Fahrzeit, Verbrauch)
- Pareto-Mengen werden exponentiell groß
- Breche Vorberechnung ab $\Rightarrow$ unkontrahierter Core-Graph (~0.5%)

# CFP & A*

**Erinnerung:**

- A* benutzt Knotenpotential, um Suche zum Ziel zu leiten
- Potential eines Knotens ist untere Schranke für Reisezeit zu $t$
- Gute Technik für schwere/komplizierte Suchprobleme
- Klassischer Ansatz: Bestimme Potential durch Rückwärtssuche von $t$

**Beobachtung:**

- Fahrzeit zu $t$ hängt auch vom SoC ab
- Metriken beeinflussen sich gegenseitig

# CFP & A*

**Erinnerung:**

- A* benutzt Knotenpotential, um Suche zum Ziel zu leiten
- Potential eines Knotens ist untere Schranke für Reisezeit zu $t$
- Gute Technik für schwere/komplizierte Suchprobleme
- Klassischer Ansatz: Bestimme Potential durch Rückwärtssuche von $t$

**Beobachtung:**

- Fahrzeit zu $t$ hängt auch vom SoC ab
- Metriken beeinflussen sich gegenseitig

**Idee:**

- Reisezeit zu $t$ abhängig von aktueller Position und SoC
- Nutze Potential $\pi\colon V \times [0, M] \to \mathbb{R}_{\geq 0}$, welches beides berücksichtigt

# CFP & A*

**Gesucht:**

- Potential $\pi\colon V \times [0, M] \to \mathbb{R}_{\geq 0}$, bildet (Knoten, SoC) auf Zeit zu $t$ ab

**Beobachtung:**

- Ladestationen erlauben "Umwandlung" von Zeit in SoC

# CFP & A*

**Gesucht:**

- Potential $\pi\colon V \times [0, M] \to \mathbb{R}_{\geq 0}$, bildet (Knoten, SoC) auf Zeit zu $t$ ab

**Beobachtung:**

- Ladestationen erlauben "Umwandlung" von Zeit in SoC
- Benutze dafür neue Metrik
- Sei dazu $c_{\max}$ die maximale Ladegeschwindigkeit
  (Maximum über die Steigung aller Ladefunktionen)
- Neue Metrik $\omega$:

$$\omega(e) := \tau_d(e) + \frac{\gamma(e)}{c_{\max}}$$

- Beschreibt min. Reisezeit, falls alle Energie geladen werden muss

# CFP & A*

**Algorithmus:** Läuft in 2 Phasen:

**1:** Rückwärtssuche von $t$ berechnet Potential $\pi$

**2:** Vorwärtssuche von $s$ nach $t$, beschleunigt durch $\pi$

**Potential-Berechnung:**

- Drei *unikriterielle* Dijkstra-Suchen von $t$ aus:
  - Auf Metrik $\tau_d$: Berechnet min. Fahrzeit $\pi_\tau$ zu $t$ (ohne Energieverbrauch)
  - Auf Metrik $\gamma$: Berechnet min Energieverbrauch $\pi_\gamma$
  - Auf Metrik $\omega$: Berechnet min. Reisezeit $\pi_\omega$, falls $b_s = 0$

# CFP & A*

**Algorithmus:** Läuft in 2 Phasen:

**1:** Rückwärtssuche von $t$ berechnet Potential $\pi$

**2:** Vorwärtssuche von $s$ nach $t$, beschleunigt durch $\pi$

**Potential-Berechnung:**

- Drei *unikriterielle* Dijkstra-Suchen von $t$ aus:
  - Auf Metrik $\tau_d$: Berechnet min. Fahrzeit $\pi_\tau$ zu $t$ (ohne Energieverbrauch)
  - Auf Metrik $\gamma$: Berechnet min Energieverbrauch $\pi_\gamma$
  - Auf Metrik $\omega$: Berechnet min. Reisezeit $\pi_\omega$, falls $b_s = 0$
- Setze dann:

$$\pi(v, b) := \begin{cases} \pi_\tau(v) & \text{, falls } b \geq \pi_\gamma(v) \\ \pi_\omega(v) - \frac{b}{c_{\max}} & \text{, sonst} \end{cases}$$

# CHArge – CH & A* & CFP

**Algorithmus:**

- Vorberechnung CH:
  - Hält Ladestationen in der Ordnung oben
  - Lässt kleinen Core unkontrahiert
- Query:
  - CH-Aufwärtssuchen von $s$ und $t$, bis Core erreicht
    (Normaler Energie-CSP-Algorithmus, da keine Ladestationen)
  - Anschließend A* eingeschränkt auf den Core-Graphen

# CHArge – CH & A* & CFP

**Algorithmus:**

- Vorberechnung CH:
  - Hält Ladestationen in der Ordnung oben
  - Lässt kleinen Core unkontrahiert
- Query:
  - CH-Aufwärtssuchen von $s$ und $t$, bis Core erreicht
    (Normaler Energie-CSP-Algorithmus, da keine Ladestationen)
  - Anschließend A* eingeschränkt auf den Core-Graphen

**Heuristiken:**

- Weitere Beschleunigung durch Heuristiken möglich
- Pfade, die bezüglich $\omega$-Metrik minimal sind, sind oft optimal
- Relaxiere pro Shortcut nur $\omega$-minimale Pareto-Punkte

# Experimente

**Straßennetzwerk:**

- Europa (Eur) & Deutschland (Ger)

**Energieverbrauch:**

- Emissionsmodel: PHEM (TU Graz) [Hausberger et al. '09]
- SRTM-Höhendaten (Shuttle Radar Topography Mission)
- Ladestationspositionen von ChargeMap

| Instanzen | # Knoten | # Kanten | # Kanten mit $\gamma < 0$ | | # S |
|---|---|---|---|---|---|
| Ger | 4 692 091 | 10 805 429 | 1 119 710 | (10.36%) | 1 966 |
| Eur | 22 198 628 | 51 088 095 | 6 060 648 | (11.86%) | 13 810 |
| Osg | 5 588 146 | 11 711 088 | 1 142 391 | (9.75%) | 643 |

# Experimente

**CH-Vorberechnung:**

- Auswirkung der Core-Größe auf die Vorberechnung

| Core-Größe | | Vorberechnung | Query [ms] | |
|---|---|---|---|---|
| Avg. deg. | # Knoten | [h:m:s] | CS: only BSS | CS: realistic |
| 8 | 344 066 (7.33%) | 2:58 | 1 474.1 | 47 979.9 |
| 16 | 116 917 (2.49%) | 4:01 | 536.5 | 1 669.0 |
| 32 | 65 375 (1.39%) | 5:03 | 436.1 | 1 356.8 |
| 64 | 43 036 (0.91%) | 7:07 | 449.8 | 1 408.8 |
| 128 | 30 526 (0.65%) | 11:16 | 509.6 | 1 585.4 |
| 256 | 22 592 (0.48%) | 20:22 | 647.5 | 2 098.5 |
| 512 | 17 431 (0.37%) | 37:11 | 880.7 | 2 739.9 |
| 1024 | 13 942 (0.29%) | 1:05:51 | 1 264.6 | 3 934.2 |
| 2048 | 11 542 (0.24%) | 2:00:27 | 1 822.6 | 5 670.1 |
| 4096 | 9 842 (0.20%) | 4:17:36 | 2 706.6 | 8 420.1 |

# Experimente

| | Instanz | $M$ | Preproc. | Exact Query | | Heuristic Query | |
|---|---|---|---|---|---|---|---|
| | | | | Feas. | CHArge | $H_\omega$ | $H_\omega^A$ |
| Only BSS | Ger-c1966 | 16 kWh | 5:03 | 100 | 1 398 | 436 | 21 |
| | Ger-c1966 | 85 kWh | 4:59 | 100 | 1 013 | 48 | 28 |
| | Eur-c13810 | 16 kWh | 30:32 | 63 | 10 786 | 9943 | 207 |
| | Eur-c13810 | 85 kWh | 30:16 | 100 | 47 921 | 1022 | 41 |
| Mixed CS | Ger-c1966 | 16 kWh | 5:03 | 100 | 8 629 | 1 357 | 155 |
| | Ger-c1966 | 85 kWh | 4:59 | 100 | 2 614 | 342 | 34 |
| | Eur-c13810 | 16 kWh | 30:32 | 63 | 24 148 | 17 630 | 2 694 |
| | Eur-c13810 | 85 kWh | 30:16 | 100 | 86 193 | 26 867 | 600 |

Vorberechnungszeiten in Minuten:Sekunden, Query-Zeiten in Millisekunden

# 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.

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.