Karlsruher Institut für Technologie

Algorithmen für Routenplanung

15. Vorlesung, Sommersemester 2016

Moritz Baum | 15. Juni 2016

INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK · PROF. DR. DOROTHEA WAGNER

Elektromobilität

Elektrofahrzeuge (EVs):

- Transportmittel der Zukunft
- Emissionsfreie Mobilität

Aber:

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

$\Rightarrow$ Berücksichtigung von Energieverbrauch bei der Routenplanung

Energieverbrauch als Metrik

Besonderheiten:

- Rekuperation
  - Rückgewinnung von Energie möglich (bergab fahren, bremsen)
  - Negative Kantengewichte möglich

  Aber: keine negativen Zyklen (physikalische Gesetze)

- Akkukapazität (Battery Constraints)
  - Akku darf nicht leer laufen
    (Andernfalls Kante nicht benutzbar)
  - Akku kann nicht überschritten werden
    (Kanten können genutzt werden, aber Energie verfällt ggf.)
  - Muss für jeden Knoten eines Pfades gelten

$\Rightarrow$ Ladestand (state of charge, SoC) während Query berücksichtigen

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

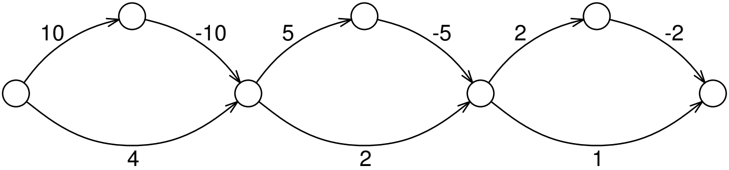
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

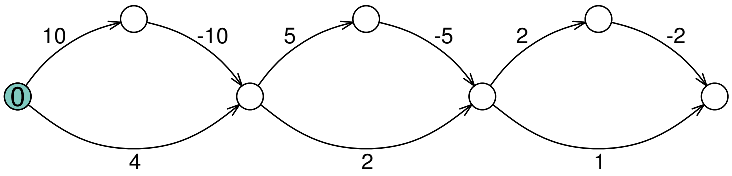


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



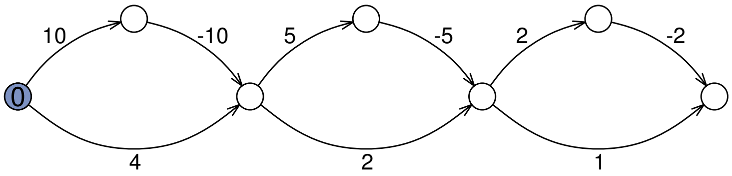
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



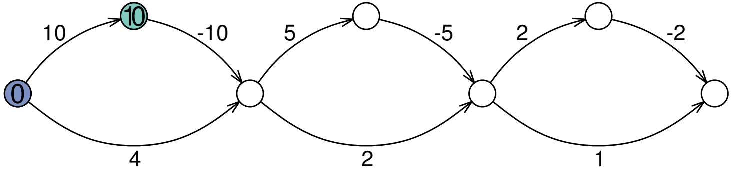
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



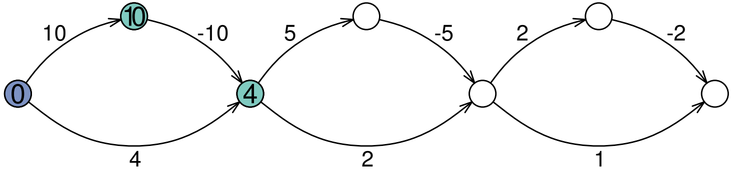
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



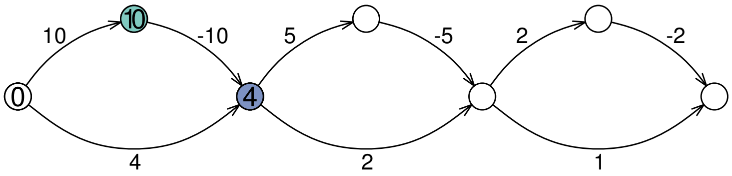
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



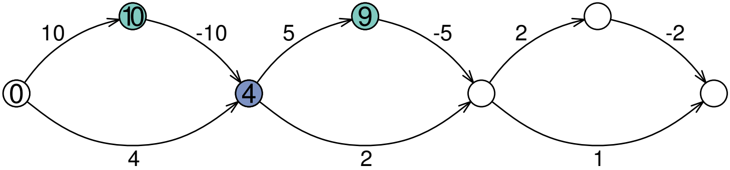
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

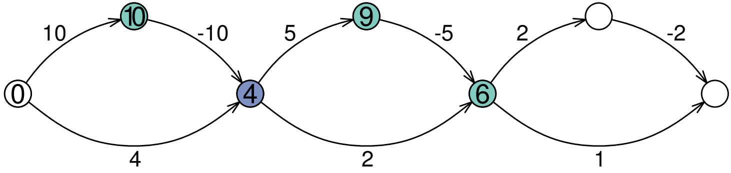
Energieoptimale Routen

Karlsruher Institut für Technologie

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

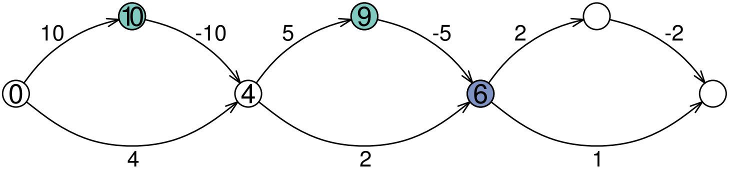


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



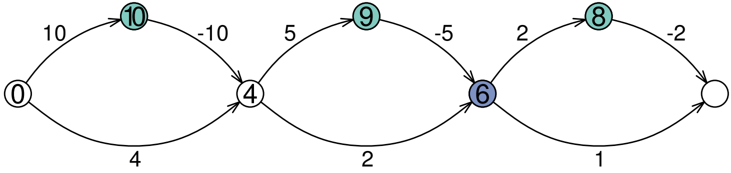
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



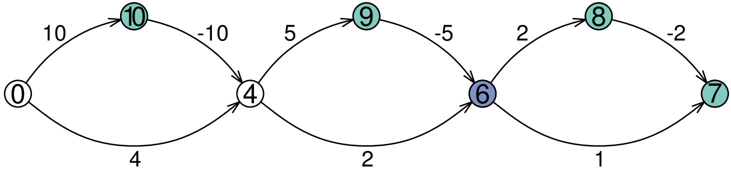
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

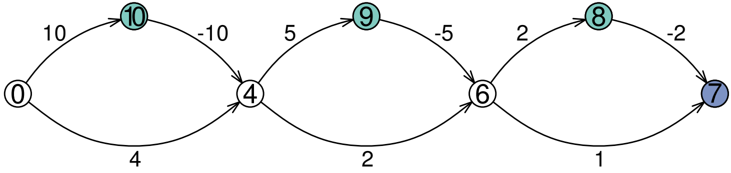


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

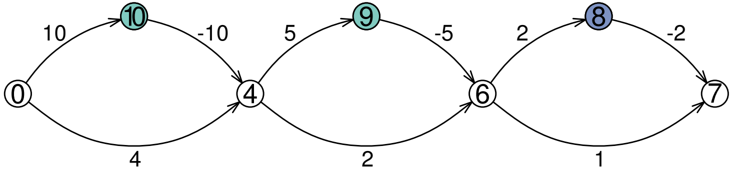


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



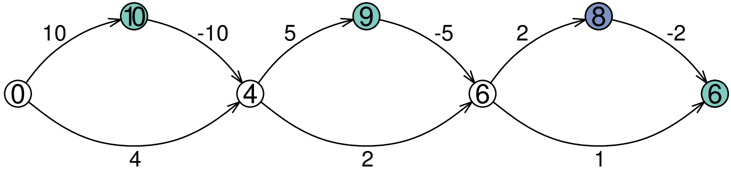
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

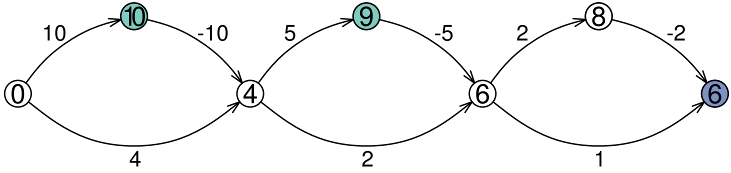


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

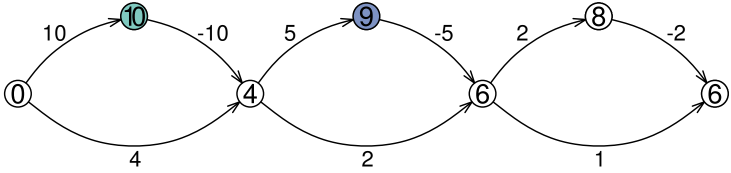


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

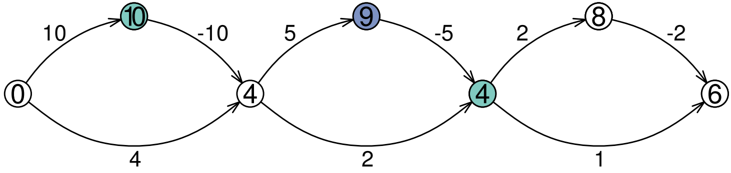


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



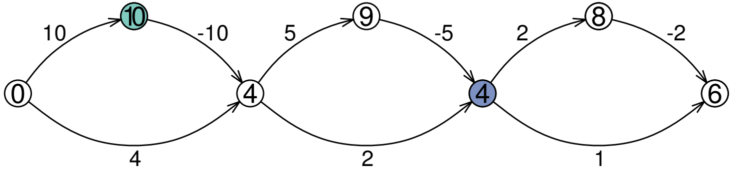
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

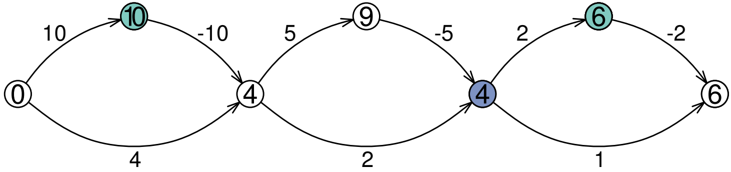


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

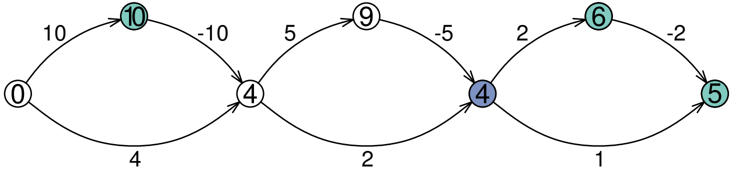


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

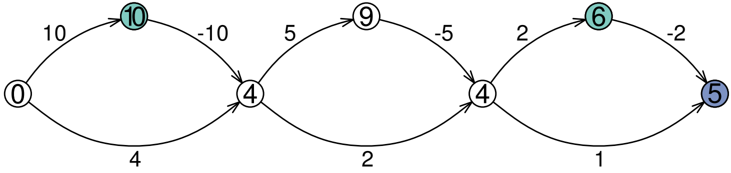


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

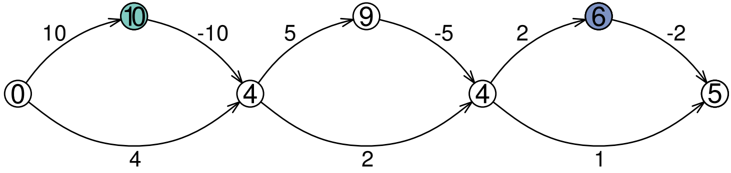


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

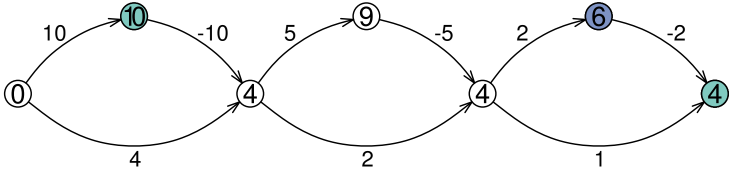


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



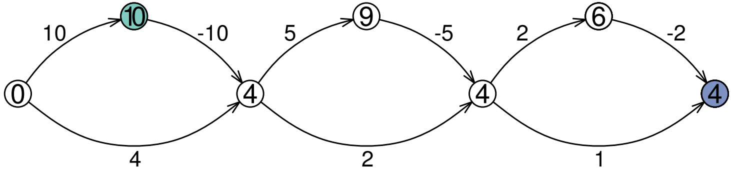
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



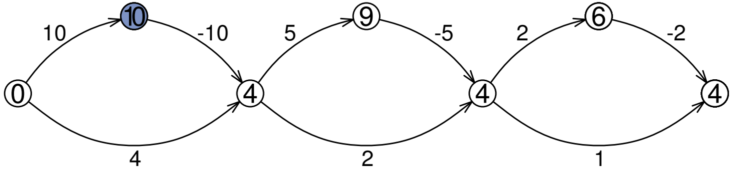
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

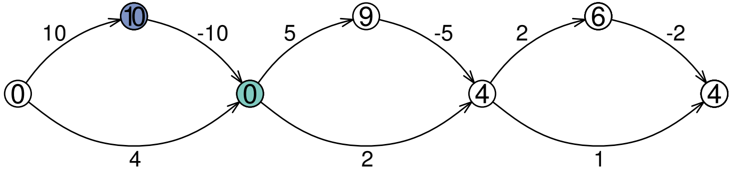


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



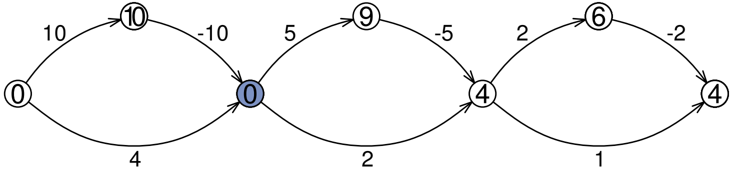
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



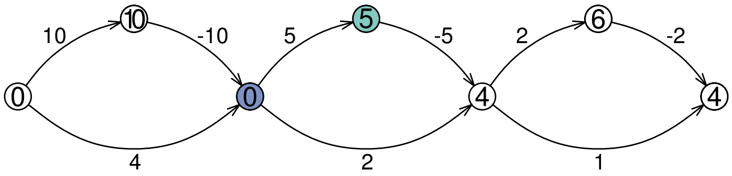
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

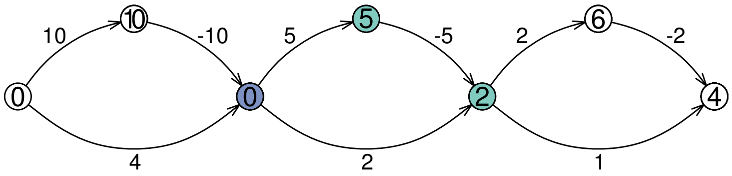


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



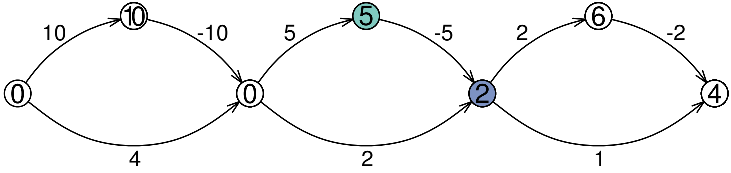
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

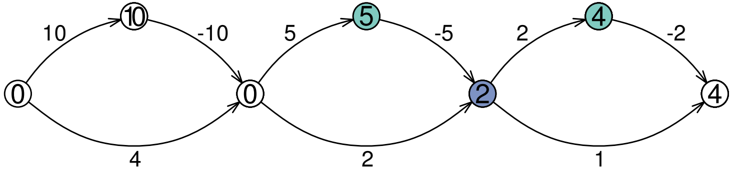


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

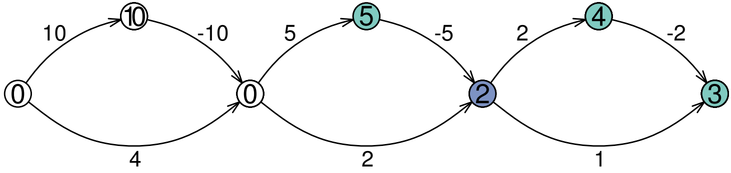


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

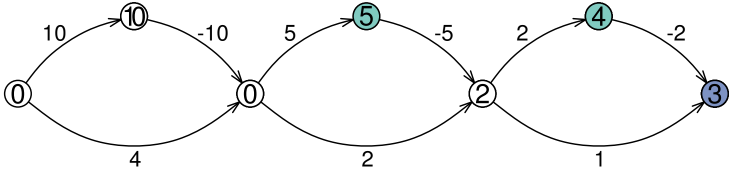


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

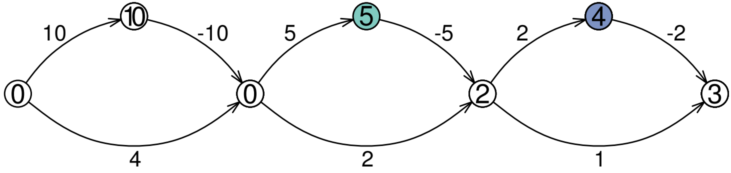


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra’s Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



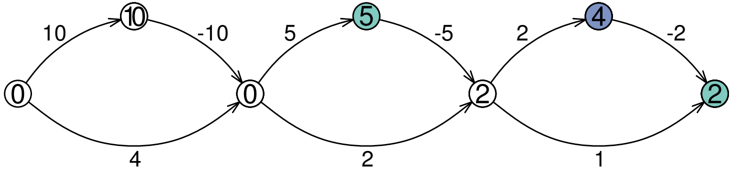
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

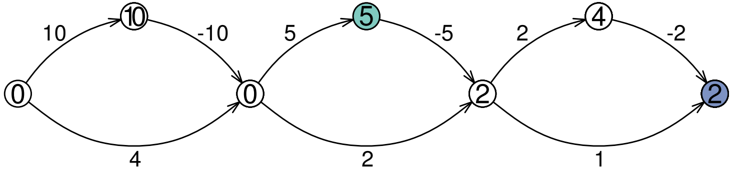


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



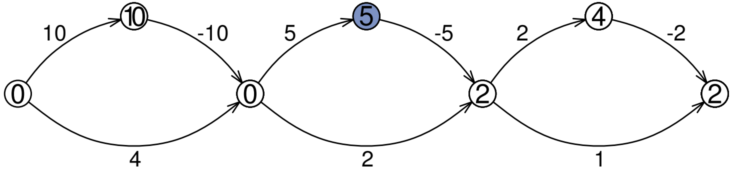
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



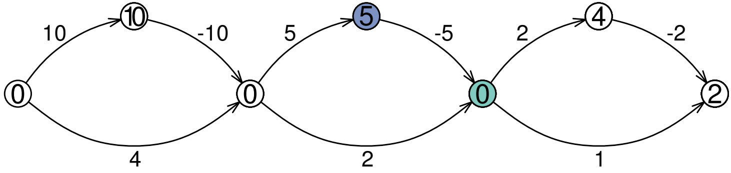
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



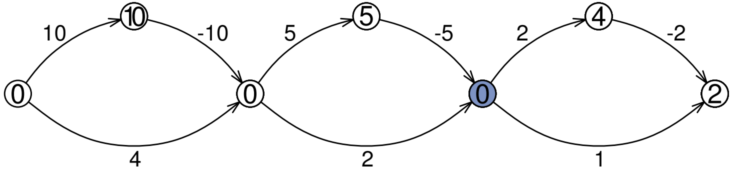
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

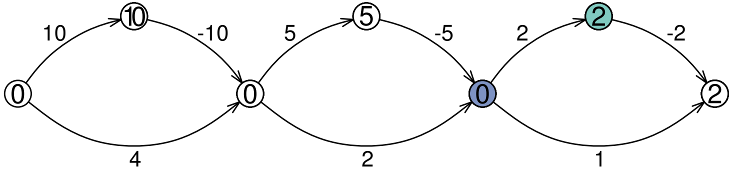


Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



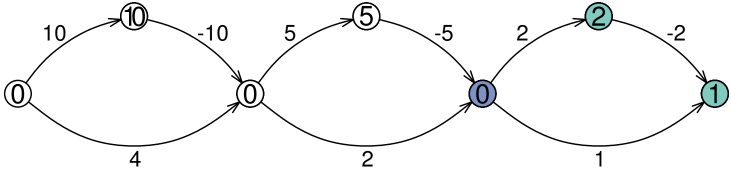
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



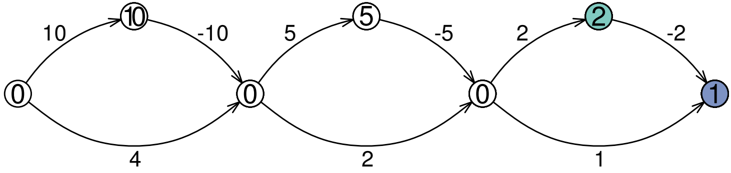
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



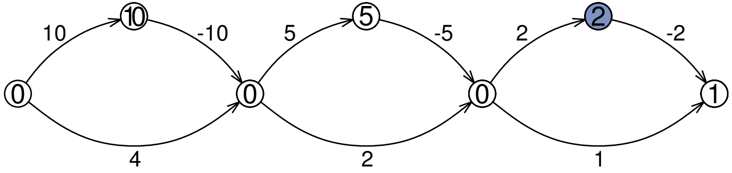
Energieoptimale Routen

Karlsruher Institut für Technologie

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



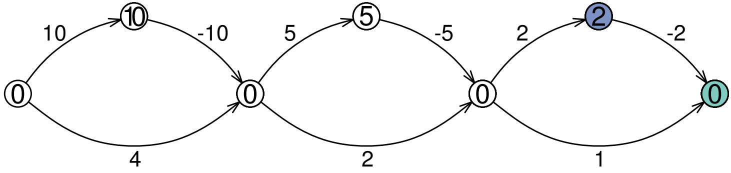
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten *s* und Zielknoten *t*,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



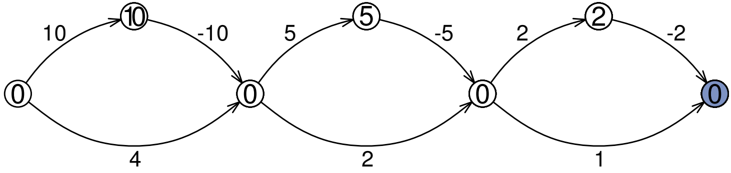
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

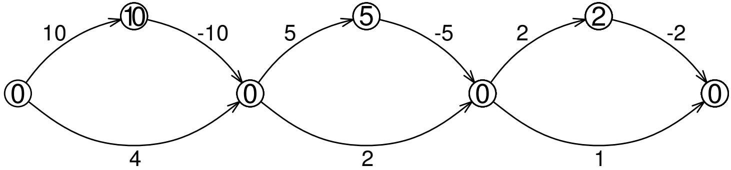


Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



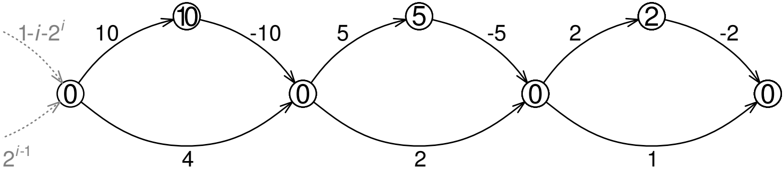
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t,
berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

Bellman Ford:

- Funktioniert auch mit negativen Kantengewichten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Ziel: Gegeben Startknoten s und Zielknoten t, berechne eine Route die den Energieverbrauch minimiert

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr

Bellman Ford:

- Funktioniert auch mit negativen Kantengewichten

Auf Realwelt-Instanzen ist Dijkstras Algorithmus schneller

Energieoptimale Routen

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr
- Nicht mehr Label-Setting

$\Rightarrow$ Stoppkriterium nicht mehr korrekt

Frage: Stoppkriterium wiederherstellbar?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Dijkstra's Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr
- Nicht mehr Label-Setting

$\Rightarrow$ Stoppkriterium nicht mehr korrekt

Frage: Stoppkriterium wiederherstellbar?

- Wenn t abgearbeitet wird, könnten sich noch Pfade in der Queue befinden, die Label $d[t]$ durch Anhängen eines negativen Subpfades verbessern
- Ziel: berechne Schranke $P_{\text{min}} \leq 0$ für Länge dieses Subpfades
- Dann Abbruch, sobald $\text{minKey}(Q) + P_{\text{min}} \geq d[t]$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Berechnung von P_{min}

Gegeben: Graph $G = (V, E)$ (ohne negative Zyklen)
Gesucht: (Global) kürzester Pfad in G

1. Ansatz:

- Füge (virtuelle) Knoten s', t' zu G hinzu
- Mit Kanten (s', u), (u, t') für alle $u \in V$ (mit Energieverbrauch 0)
- Berechne kürzesten s'-t'-Weg (mit Label-Correcting Dijkstra)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Berechnung von P_{min}

Gegeben: Graph $G = (V, E)$ (ohne negative Zyklen)
Gesucht: (Global) kürzester Pfad in G

2. Ansatz (simuliert 1. Ansatz, ist in der Praxis schneller):

- Initialisiere Distanzlabel $\underline{d}[v] = 0$ für alle $v \in V$
- Für jeden Knoten $v \in V$:
  - Starte (Label-Correcting) Suche von v
  - Distanzlabel $\underline{d}[\cdot]$ wird zwischen den Suchen *nicht* reinitialisiert
- Berechne währenddessen: $P_{min} := min_{v \in V} \underline{d}[v]$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Berechnung von P_{min}

Gegeben: Graph $G = (V, E)$ (ohne negative Zyklen)
Gesucht: (Global) kürzester Pfad in G

2. Ansatz (simuliert 1. Ansatz, ist in der Praxis schneller):

- Initialisiere Distanzlabel $\underline{d}[v] = 0$ für alle $v \in V$
- Für jeden Knoten $v \in V$:
  - Starte (Label-Correcting) Suche von v
  - Distanzlabel $\underline{d}[\cdot]$ wird zwischen den Suchen *nicht* reinitialisiert
- Berechne währenddessen: $P_{min} := min_{v \in V}\ \underline{d}[v]$

Danach Stoppkriterium wieder anwendbar

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Berechnung von P_{min}

Gegeben: Graph $G = (V, E)$ (ohne negative Zyklen)
Gesucht: (Global) kürzester Pfad in G

2. Ansatz (simuliert 1. Ansatz, ist in der Praxis schneller):

- Initialisiere Distanzlabel $\underline{d}[v] = 0$ für alle $v \in V$
- Für jeden Knoten $v \in V$:
  - Starte (Label-Correcting) Suche von v
  - Distanzlabel $\underline{d}[\cdot]$ wird zwischen den Suchen *nicht* reinitialisiert
- Berechne währenddessen: $P_{min} := min_{v \in V}\ \underline{d}[v]$

Danach Stoppkriterium wieder anwendbar

Beobachtung: Nach Berechnung von P_{min} gilt: $\underline{d}[t] \leq dist(v, t), \forall v \in V$

$\Rightarrow$ Stoppkriterium lässt sich verbessern zu: $\text{minKey}(Q) + \underline{d}[t] \geq d[t]$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Dijkstras Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Stoppkriterium lässt sich wiederherstellen
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr
- Nicht mehr Label-Setting

Frage: Label-Setting Eigenschaft wiederherstellbar?

Energieoptimale Routen

Dijkstras Algorithmus:

- Setzt nichtnegative Kantengewichte voraus
- Stoppkriterium lässt sich wiederherstellen
- Bleibt korrekt, aber keine polynomielle Laufzeitgarantie mehr
- Nicht mehr Label-Setting

Frage: Label-Setting Eigenschaft wiederherstellbar?

Idee: Finde zulässiges Potential $\pi\colon V \to \mathbb{R}$, sodass
$\mathrm{len}(u, v) - \pi(u) + \pi(v) \geq 0$ für jede Kante $(u, v) \in E$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Knotenpotentiale

Idee: Finde zulässiges Potential $\pi\colon V \to \mathbb{R}$, sodass
$\text{len}(u, v) - \pi(u) + \pi(v) \geq 0$ für jede Kante $(u, v) \in E$

Dann Dijkstras Algorihmus (mit Stoppkriterium) auf Graph mit reduzierten Gewichten anwendbar

1. Distanzbasiertes Potential:

- Setze $\pi(v) = d(v, v^*)$, für beliebigen Knoten v^*
- Berechnung mittels (Label-Correcting) Query von v^*
- Zulässigkeit folgt aus Dreiecksungleichung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Knotenpotentiale

Idee: Finde zulässiges Potential $\pi\colon V \to \mathbb{R}$, sodass
$\text{len}(u, v) - \pi(u) + \pi(v) \geq 0$ für jede Kante $(u, v) \in E$

Dann Dijkstras Algorihmus (mit Stoppkriterium) auf Graph mit reduzierten Gewichten anwendbar

2. Höheninduziertes Potential:

- Ann.: Höhenkoordinate $h(v)$ eines jeden Knoten gegeben
- $\pi(v) := \alpha \cdot h(v)$ für alle v, so dass $c(u, v) + \alpha(h(v) - h(u)) \geq 0$
- Kann mit Sweep über alle Kanten berechnet werden
- Keine Garantie für Zulässigkeit des Potentials
  (klappt für realistische Kantengewichte)

Energieoptimale Routen

Bisher:

- Route minimiert den absoluten Energieverbrauch
- Route ist für konkreten Ladezustand an s ggf. nicht realisierbar

Jetzt:

- Berechne Route abhängig vom initialen Ladezustand (SoC)
- Maximiere resultierenden SoC an t
- Betrachte nur Pfade die *zulässig* sind:
  - Energie ist ausreichend: SoC wird nie negativ
  - Akku wird nie überschritten: SoC $> M \;\rightarrow\;$ SoC $= M$
    ($M :=$ Akku-Kapazität)

Energieoptimale Routen

Anfragetypen (analog zu Zeitabhängigkeit):

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

Berechnung mit angepasstem Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Energieoptimale Routen

Karlsruher Institut für Technologie

Anfragetypen (analog zu Zeitabhängigkeit):

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

Berechnung mit angepasstem Dijkstra

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

Berechnung mit Hilfe von:

- Verbrauchsfunktionen
- Geeigneten Link/Merge-Operationen

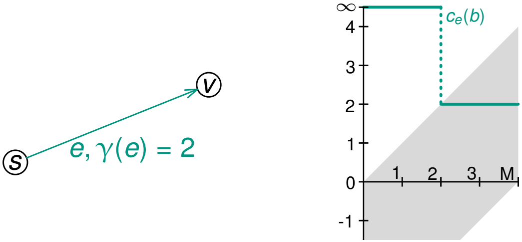
Verbrauchsfunktionen

Bisher:

- Konstanter Energieverbrauch pro Kante
- Explizites überprüfen der Battery Constraints im Algorithmus

Jetzt:

- Integriere Battery Constraints in Verbrauchsfunktionen
- Verbrauchsfunktion bildet SoC auf tatsächlichen Verbrauch ab



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

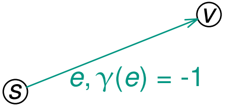
Verbrauchsfunktionen

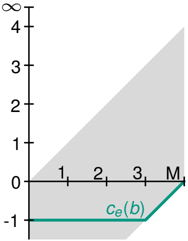
Bisher:

- Konstanter Energieverbrauch pro Kante
- Explizites überprüfen der Battery Constraints im Algorithmus

Jetzt:

- Integriere Battery Constraints in Verbrauchsfunktionen
- Verbrauchsfunktion bildet SoC auf tatsächlichen Verbrauch ab





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

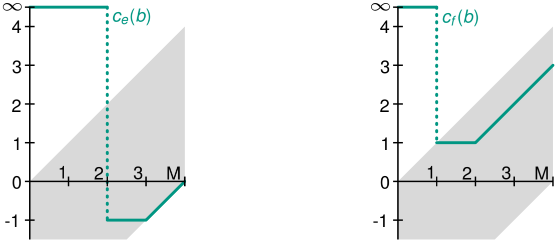
Verbrauchsfunktionen

Bisher:

- Konstanter Energieverbrauch pro Kante
- Explizites überprüfen der Battery Constraints im Algorithmus

Jetzt:

- Integriere Battery Constraints in Verbrauchsfunktionen
- Verbrauchsfunktion bildet SoC auf tatsächlichen Verbrauch ab



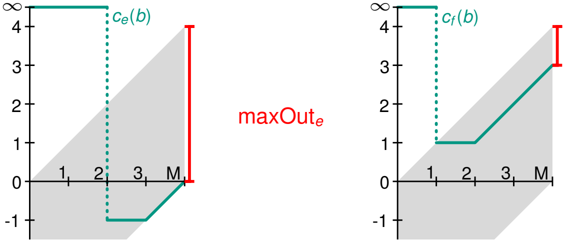
Verbrauchsfunktionen

Bisher:

- Konstanter Energieverbrauch pro Kante
- Explizites überprüfen der Battery Constraints im Algorithmus

Jetzt:

- Integriere Battery Constraints in Verbrauchsfunktionen
- Verbrauchsfunktion bildet SoC auf tatsächlichen Verbrauch ab



Anm: Potentiale weiterhin zulässig (cost_e ist untere Schranke auf Verbrauch)

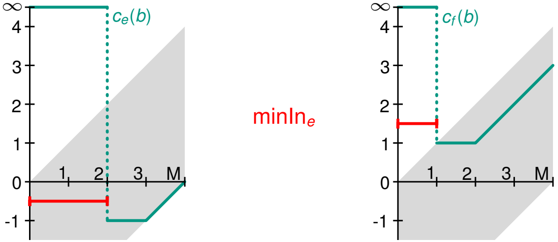
Verbrauchsfunktionen

Bisher:

- Konstanter Energieverbrauch pro Kante
- Explizites überprüfen der Battery Constraints im Algorithmus

Jetzt:

- Integriere Battery Constraints in Verbrauchsfunktionen
- Verbrauchsfunktion bildet SoC auf tatsächlichen Verbrauch ab



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

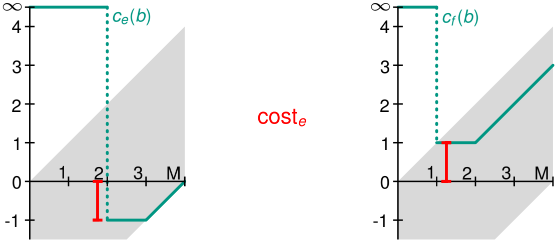
Verbrauchsfunktionen

Bisher:

- Konstanter Energieverbrauch pro Kante
- Explizites überprüfen der Battery Constraints im Algorithmus

Jetzt:

- Integriere Battery Constraints in Verbrauchsfunktionen
- Verbrauchsfunktion bildet SoC auf tatsächlichen Verbrauch ab



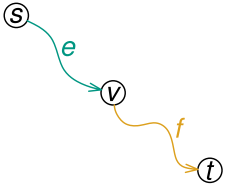
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

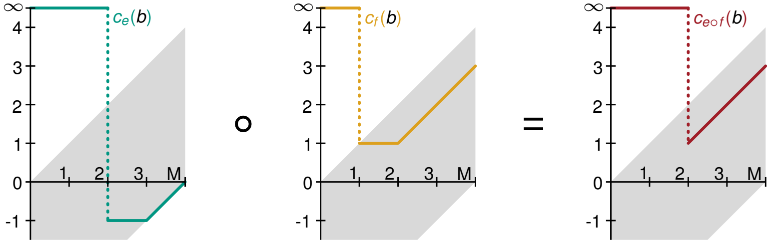
Verbrauchsfunktionen – Linking

Gesucht: Verbrauch beim traversieren von e *und* f

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

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





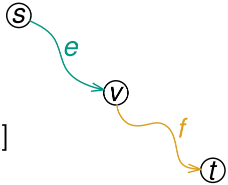
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

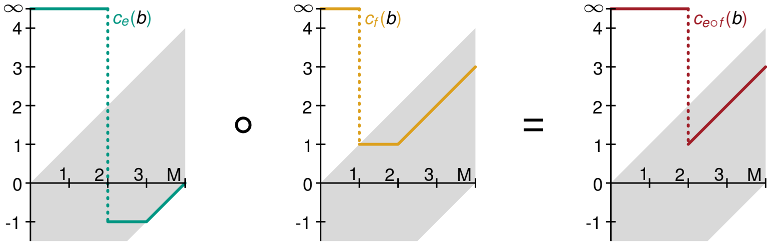
Verbrauchsfunktionen – Linking

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

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

(Bedingung: $\mathrm{maxOut}_e \geq \mathrm{minIn}_f$)





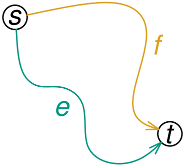
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

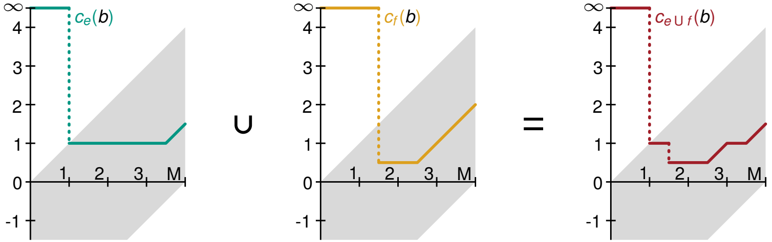
Verbrauchsfunktionen – Merging

Gesucht: Verbrauch beim traversieren von e *oder* f

- Benutze Pfad mit geringerem Verbrauch

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





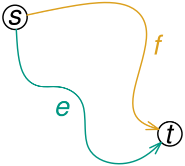
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Verbrauchsfunktionen – Merging

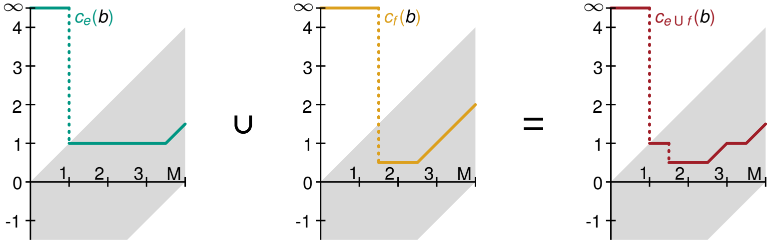
Gesucht: Verbrauch beim traversieren von e *oder* f

- Benutze Pfad mit geringerem Verbrauch

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



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



Energieoptimale Routen

Karlsruher Institut für Technologie

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

Besonderheiten: Energieverbrauch hängt von vielen Parametern ab

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

$\Rightarrow$ Schnelle Customization wichtig

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

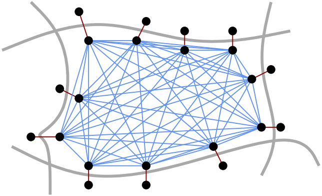
MLD für energieoptimale Routen

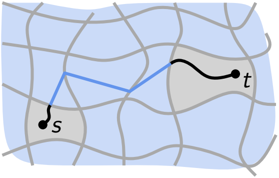
Idee MLD:

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

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

MLD für energieoptimale Routen

1: Metrik-unabhängige Vorberechnung

- Metrikunabhängig, keine Anpassung nötig

2: Metrik-abhängige Vorberechnung (Customization)

- Cliquenkanten sind Profile
- Lokale Profilsuchen
- Anpassung der Datenstrukturen

3: Queries

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente

Algorithm	CUSTOMIZING space [B/n]	CUSTOMIZING time [s]	QUERIES vertex scans	QUERIES time [ms]
LC	—	—	4 471 230	709.6
LC (stop. cr.)	0.0	5.20	3 001 014	486.6
Dijkstra π_{v^*}	4.0	3.91	2 361 997	288.0
Dijkstra π_h	4.0	0.69	2 359 140	380.6
Prof. π_h	4.0	0.70	2 904 764	741.2
MLD (LC)	10.5	4.42	3 676	2.7
MLD Uni π_h	14.5	5.12	2 410	1.9
MLD BPE π_h	14.5	5.12	2 266	1.4
MLD BDB π_h	14.5	5.12	2 917	1.1

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

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

Karlsruher Institut für Technologie

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?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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

Metrikvergleich

Metric	Unreach.	Extra Energy	Extra T/D
Energy vs. Travel Time	60 %	62 %	63 %
Energy vs. Distance	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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Metrikvergleich

Metric	Unreach.	Extra Energy	Extra T/D
Energy vs. Travel Time	60 %	62 %	63 %
Energy vs. Distance	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*

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wdh: Constrained Shortest Path

Definition: Constrained Shortest Path Problem

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

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

Anmerkung: Das entsprechende Optimierungsproblem lautet:

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

Theorem

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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 $> W$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

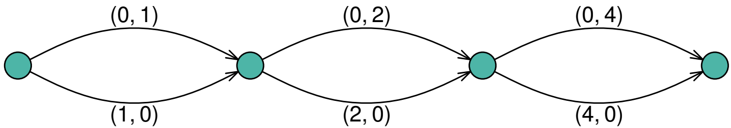
Wdh: Multi-Criteria Dijkstra für CSP

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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

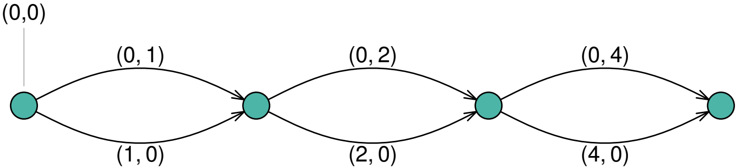
Wdh: Multi-Criteria Dijkstra für CSP

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



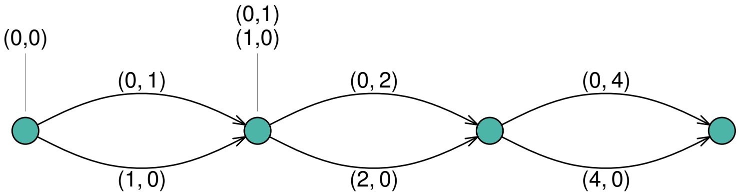
Wdh: Multi-Criteria Dijkstra für CSP

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



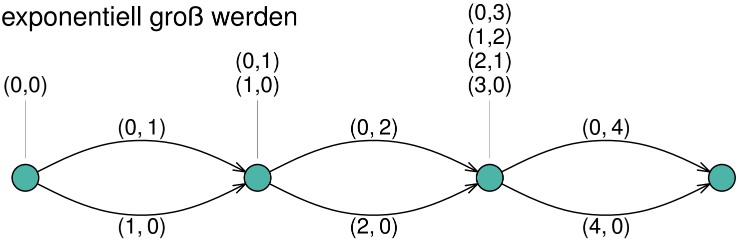
Wdh: Multi-Criteria Dijkstra für CSP

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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

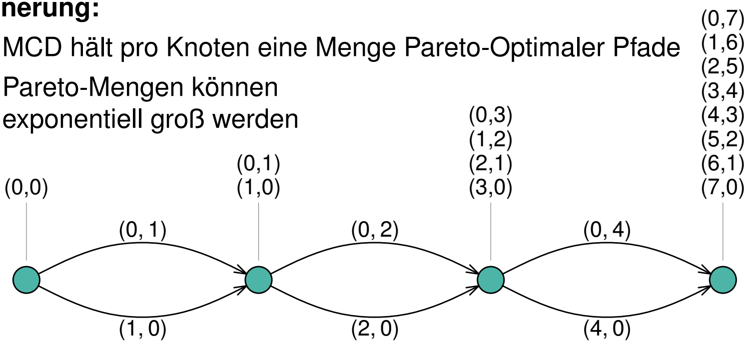
Wdh: Multi-Criteria Dijkstra für CSP

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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 $\underline{d}[t]$)
- Knoten Kontraktionen (Nutze Verbrauchsfunktionen)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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 $\underline{d}[t]$)
- Knoten Kontraktionen (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)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Erweiterte Szenarien

Variable Geschwindigkeit:

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

Ladestationen:

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

Ladestationen

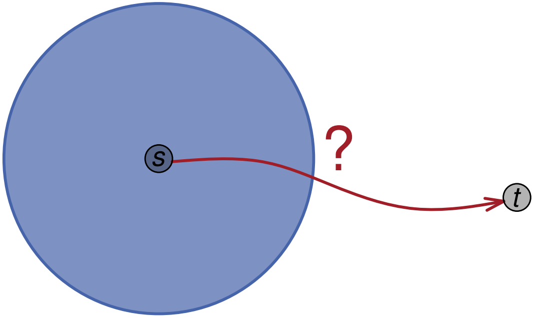
Finde schnellste Route von s nach t:

Erreichbares Gebiet

Ladestation

Ladestationen

Finde schnellste Route von s nach t:



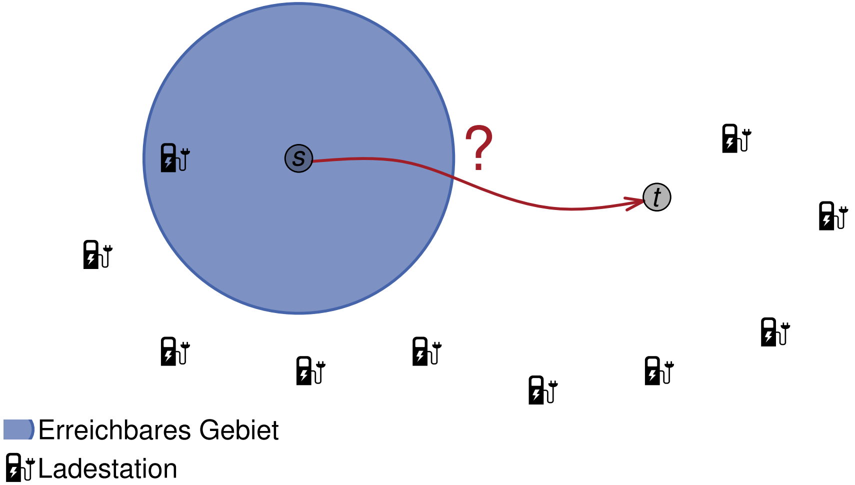
Erreichbares Gebiet

Ladestation

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

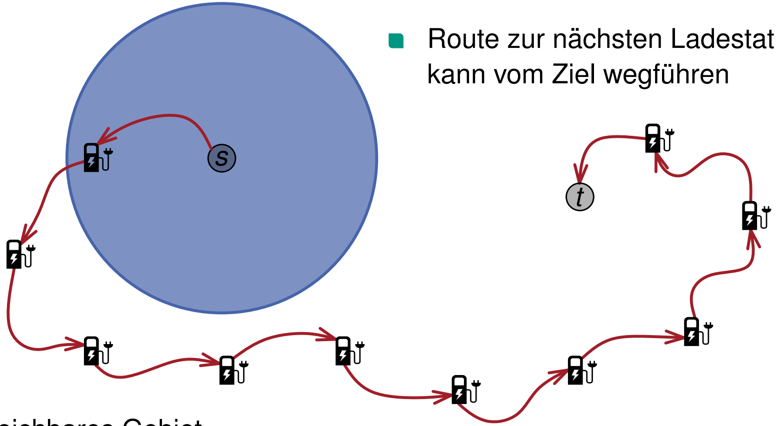
Finde schnellste Route von s nach t:



Ladestationen

Finde schnellste Route von s nach t:

- Route zur nächsten Ladestation kann vom Ziel wegführen

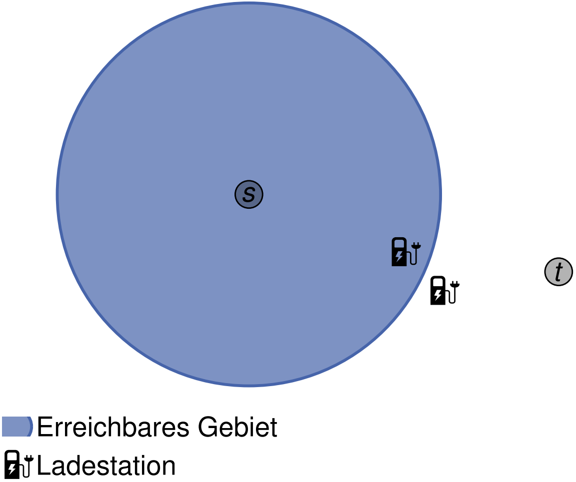


Erreichbares Gebiet

Ladestation

Ladestationen

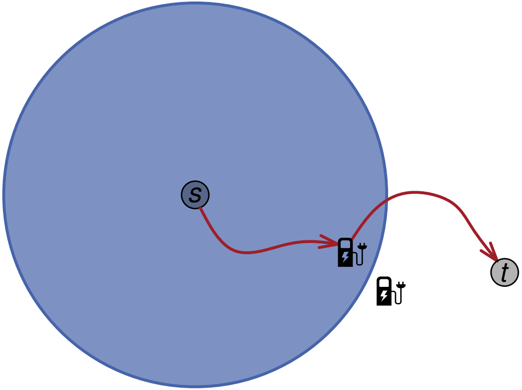
Finde schnellste Route von s nach t:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

Finde schnellste Route von s nach t:



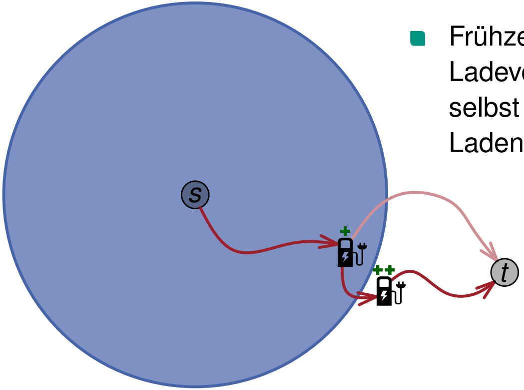
Erreichbares Gebiet

Ladestation

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

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



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

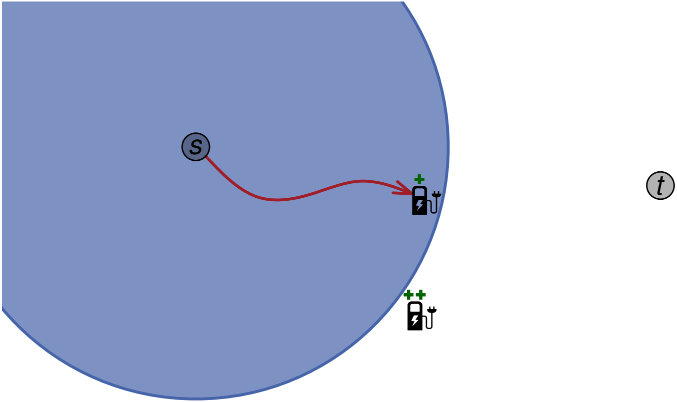
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

Finde schnellste Route von s nach t:



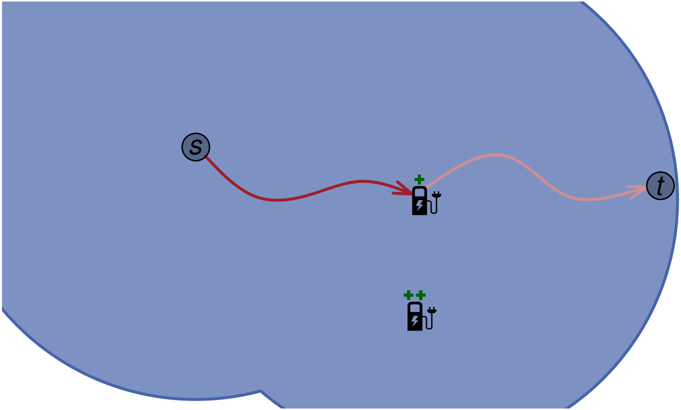
Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Ladestationen

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



Erreichbares Gebiet

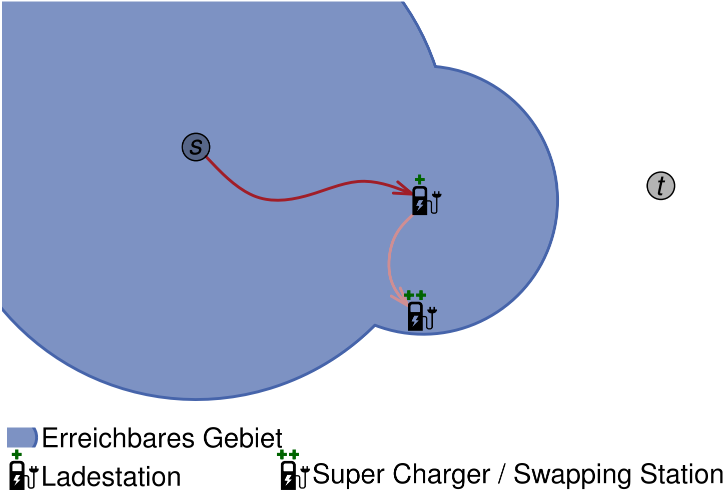
Ladestation

Super Charger / Swapping Station

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

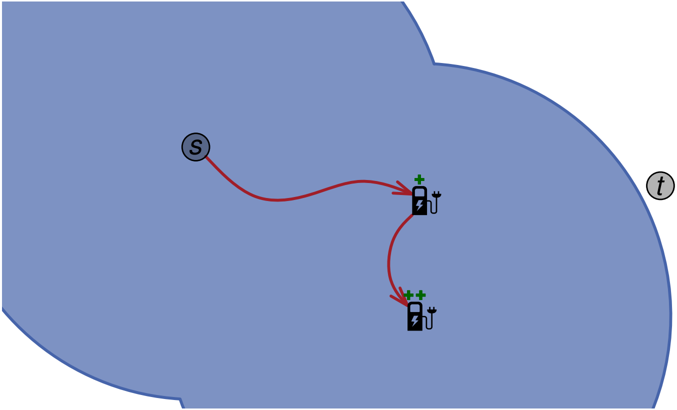
Finde schnellste Route von *s* nach *t*:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

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



Erreichbares Gebiet

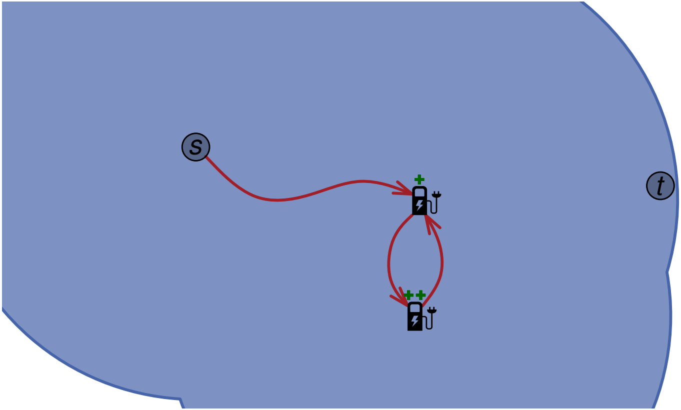
Ladestation

Super Charger / Swapping Station

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

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



Erreichbares Gebiet

Ladestation

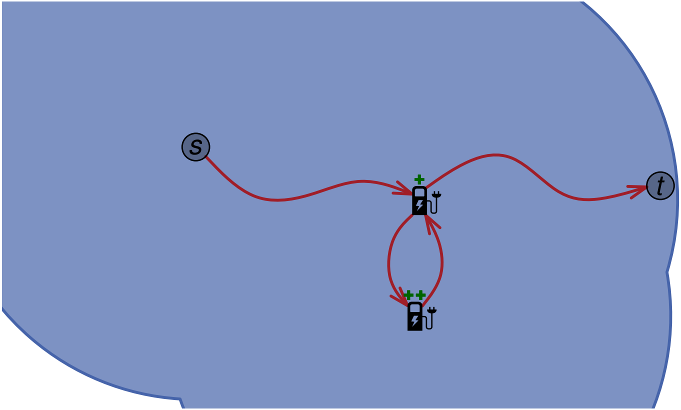
Super Charger / Swapping Station

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladestationen

Finde schnellste Route von s nach t:

- Zyklen möglich



Erreichbares Gebiet

Ladestation

Super Charger / Swapping Station

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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 τ_c auf
  - Durch laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

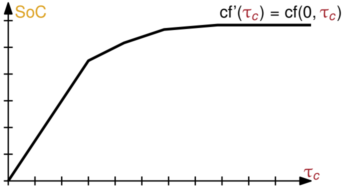
Ladefunktionen

Formal:

- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - Initialen SoC b_s und
  - Gewünschte Ladezeit τ_c auf
  - Durch laden erreichten SoC ab
- Monoton steigend (Länger laden ⇒ mehr Energie)
- Konkav (Akku voller ⇒ 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:

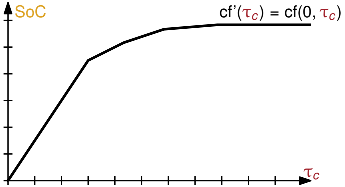
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - Initialen SoC b_s und
  - Gewünschte Ladezeit τ_c auf
  - Durch laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

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

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

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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladefunktionen

Formal:

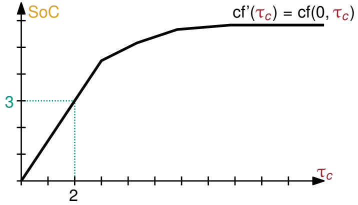
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - Initialen SoC b_s und
  - Gewünschte Ladezeit τ_c auf
  - Durch laden erreichten SoC ab
- Monoton steigend (Länger laden ⇒ mehr Energie)
- Konkav (Akku voller ⇒ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

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

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

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



Ladefunktionen

Formal:

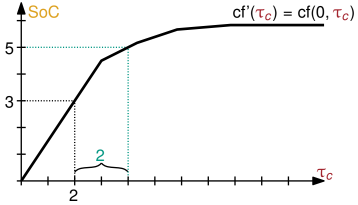
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - Initialen SoC b_s und
  - Gewünschte Ladezeit τ_c auf
  - Durch laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

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

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

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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ladefunktionen

Formal:

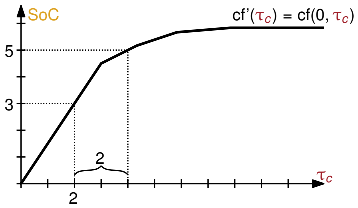
- Eine Funktion $\mathrm{cf}\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, bildet
  - Initialen SoC b_s und
  - Gewünschte Ladezeit τ_c auf
  - Durch laden erreichten SoC ab
- Monoton steigend (Länger laden $\Rightarrow$ mehr Energie)
- Konkav (Akku voller $\Rightarrow$ langsameres Laden)

Anmerkung: Realistische Ladefunktionen darstellbar durch:

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

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

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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

Algorithmus:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

Algorithmus:

- Basiert auf Dijkstra's 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 Dijkstra's 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 ℓ am Knoten v ist ein Tupel $(\tau_t, b_u, u, c_{(u,\dots,v)})$ mit:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

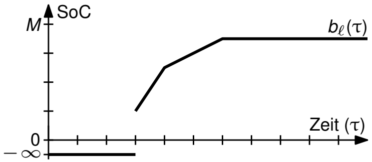
Charging Function Propagation (CFP)

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

- Reisezeit τ_t von s nach v (Inklusive Ladezeiten außer an u)
- SoC b_u mit dem die letzte Ladestation (u) erreicht wurde
- Zu letzt 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
  $b' := \mathrm{cf}_u(b_u, \tau - \tau_t)$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

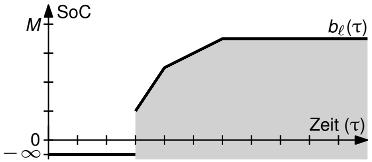
Charging Function Propagation (CFP)

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

- Reisezeit τ_t von s nach v (Inklusive Ladezeiten außer an u)
- SoC b_u mit dem die letzte Ladestation (u) erreicht wurde
- Zu letzt 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
  $b' := \mathrm{cf}_u(b_u, \tau - \tau_t)$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

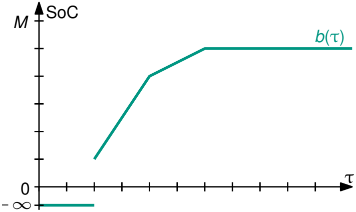
Charging Function Propagation (CFP)

Kanten Relaxierung: (Label $\ell = (\tau_t, b_u, u, c_{(u,\dots,v)})$)

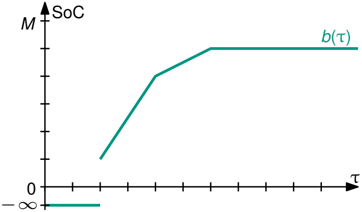
- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

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

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



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



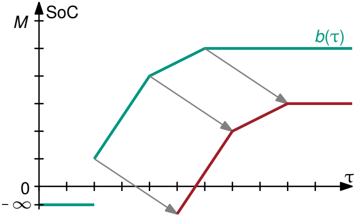
Charging Function Propagation (CFP)

Kanten Relaxierung: (Label $\ell = (\tau_t, b_u, u, c_{(u,\dots,v)})$)

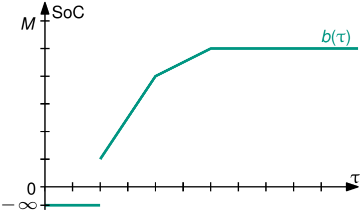
- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

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

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



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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

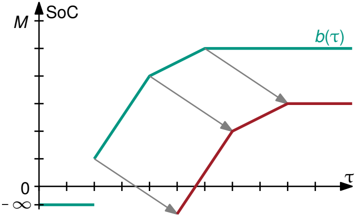
Charging Function Propagation (CFP)

Kanten Relaxierung: (Label $\ell = (\tau_t, b_u, u, c_{(u,\dots,v)})$)

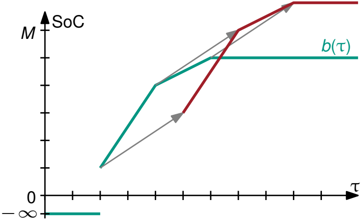
- Relaxieren der Kante $e = (v, w)$ verschiebt die SoC-Funktion $b_\ell(\tau)$
  - $b_\ell(\tau)$ wird um Fahrzeit $\tau_d(e)$ nach rechts verschoben
  - $b_\ell(\tau)$ wird um Verbrauch $\gamma(e)$ nach unten verschoben
- Anschließend werden Battery Constraints überprüft

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

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



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



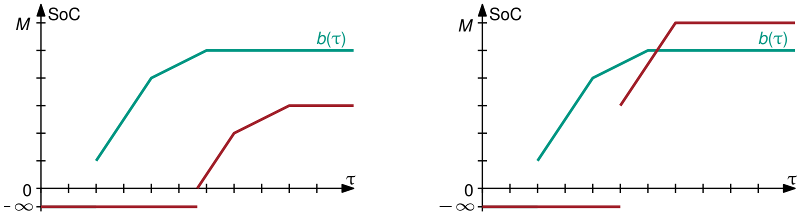
Charging Function Propagation (CFP)

Kanten Relaxierung: (Label $\ell = (\tau_t, b_u, u, c_{(u,\dots,v)})$)

- 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 $c_{(u,\dots,w)} \leftarrow c_{(u,\dots,v)} \circ c_e$

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



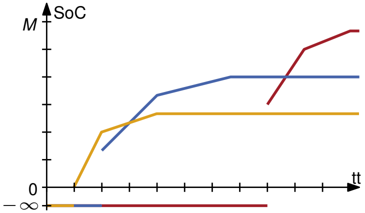
Charging Function Propagation (CFP)

Dominanz von SoC-Funktionen:

- Eine SoC-Funktion $b_\ell(\tau)$ dominert $b_\ell(\tau)$ ($b_\ell(\tau) \propto b_\ell(\tau)$) gdw.:

$$\forall \tau \geq 0\colon\ b_\ell(\tau) \geq b_\ell(\tau)$$

- Pro Knoten eine Pareto-Menge von SoC-Funktionen
- Ein neues Label wird erzeugt (Durch eine Kanten Relaxierung)
  $\Rightarrow$ Überprüfe Dominanz,
  Lösche dominierte Label



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

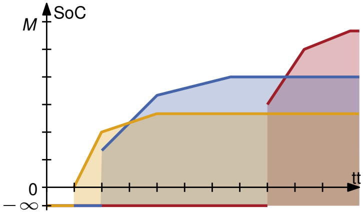
Charging Function Propagation (CFP)

Dominanz von SoC-Funktionen:

- Eine SoC-Funktion $b_\ell(\tau)$ dominert $b_\ell(\tau)$ ($b_\ell(\tau) \propto b_\ell(\tau)$) gdw.:

$$\forall \tau \geq 0\colon\ b_\ell(\tau) \geq b_\ell(\tau)$$

- Pro Knoten eine Pareto-Menge von SoC-Funktionen
- Ein neues Label wird erzeugt (Durch eine Kanten Relaxierung)
  $\Rightarrow$ Überprüfe Dominanz,
  Lösche dominierte Label



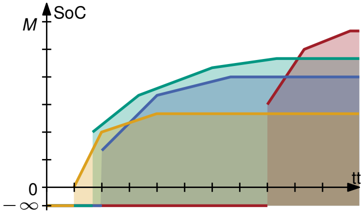
Charging Function Propagation (CFP)

Dominanz von SoC-Funktionen:

- Eine SoC-Funktion $b_\ell(\tau)$ dominert $b_\ell(\tau)$ ($b_\ell(\tau) \propto b_\ell(\tau)$) gdw.:

$$\forall \tau \geq 0 \colon\ b_\ell(\tau) \geq b_\ell(\tau)$$

- Pro Knoten eine Pareto-Menge von SoC-Funktionen
- Ein neues Label wird erzeugt (Durch eine Kanten Relaxierung)
  $\Rightarrow$ Überprüfe Dominanz,
  Lösche dominierte Label



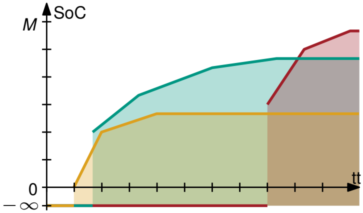
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Dominanz von SoC-Funktionen:

- Eine SoC-Funktion $b_\ell(\tau)$ dominert $b_\ell(\tau)$ ($b_\ell(\tau) \propto b_\ell(\tau)$) gdw.:

$$\forall \tau \geq 0 \colon b_\ell(\tau) \geq b_\ell(\tau)$$

- Pro Knoten eine Pareto-Menge von SoC-Funktionen
- Ein neues Label wird erzeugt (Durch eine Kanten Relaxierung)
  $\Rightarrow$ Überprüfe Dominanz,
  Lösche dominierte Label



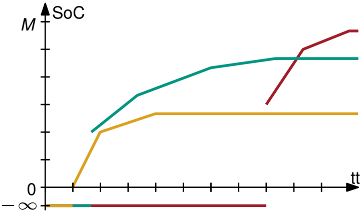
Charging Function Propagation (CFP)

Dominanz von SoC-Funktionen:

- Eine SoC-Funktion $b_\ell(\tau)$ dominert $b_\ell(\tau)$ ($b_\ell(\tau) \propto b_\ell(\tau)$) gdw.:

$$\forall \tau \geq 0\colon\ b_\ell(\tau) \geq b_\ell(\tau)$$

- Pro Knoten eine Pareto-Menge von SoC-Funktionen
- Ein neues Label wird erzeugt (Durch eine Kanten Relaxierung)
  $\Rightarrow$ Überprüfe Dominanz,
  Lösche dominierte Label



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

Settling von Ladestationen

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

Problem:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

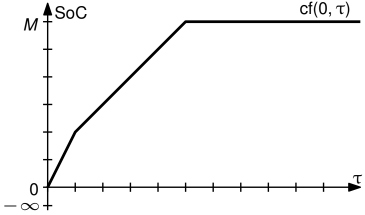
Settling von Ladestationen

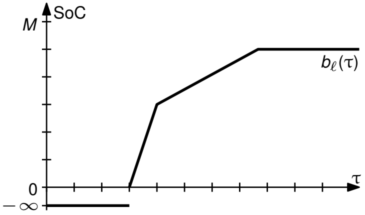
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

- Am ursprünglichem 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)

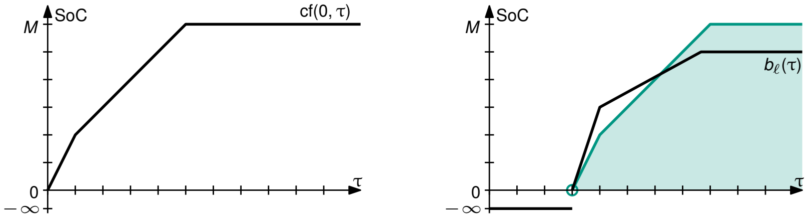
Settling von Ladestationen

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

Problem:

- Am ursprünglichem 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)

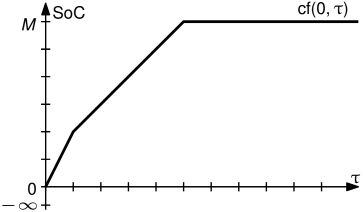
Settling von Ladestationen

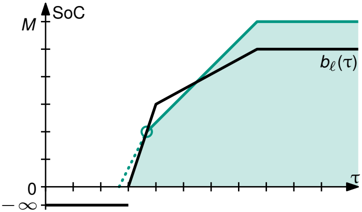
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

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

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





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

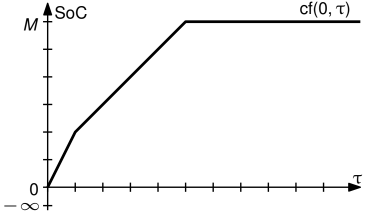
Settling von Ladestationen

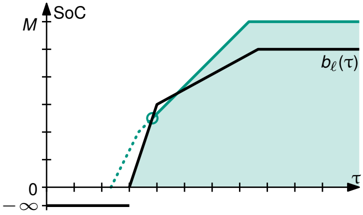
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

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

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





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

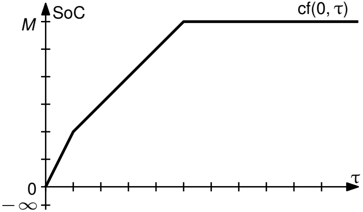
Settling von Ladestationen

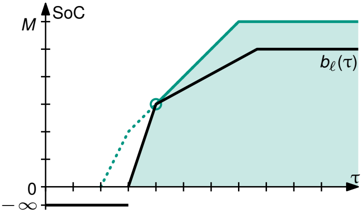
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

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

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





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

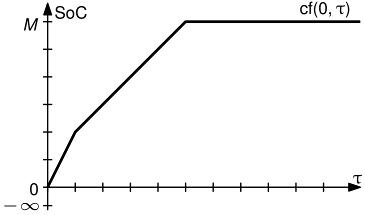
Settling von Ladestationen

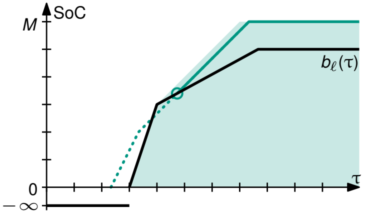
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

- Am ursprünglichem 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)

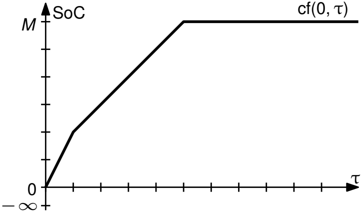
Settling von Ladestationen

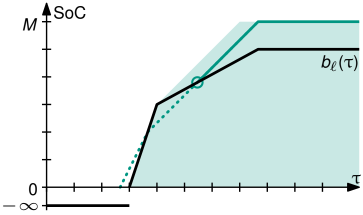
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

- Am ursprünglichem 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)

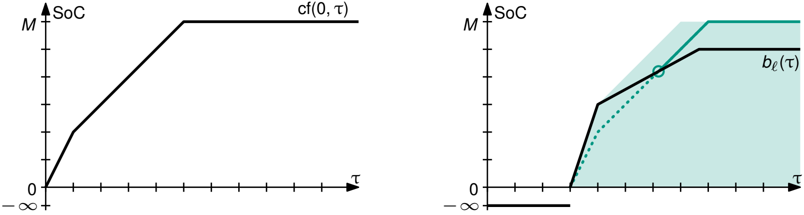
Settling von Ladestationen

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

Problem:

- Am ursprünglichem 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)

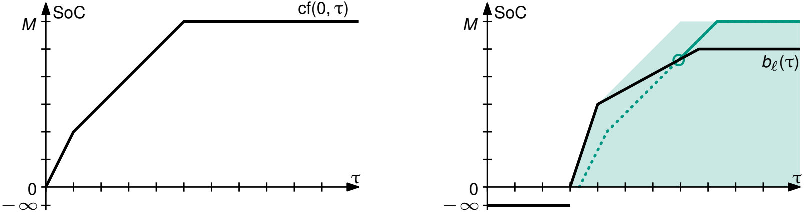
Settling von Ladestationen

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

Problem:

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

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



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

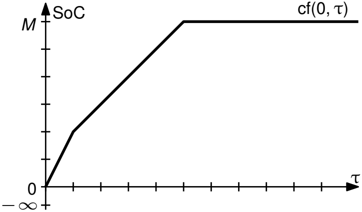
Settling von Ladestationen

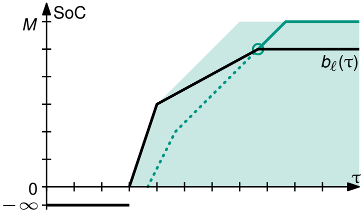
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

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

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





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

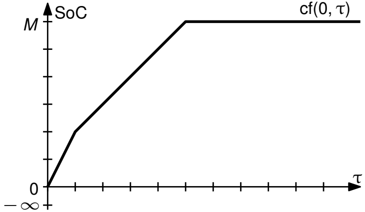
Settling von Ladestationen

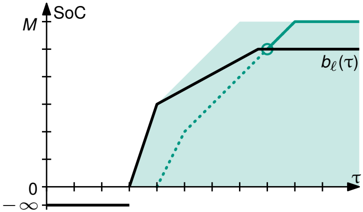
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

- Am ursprünglichem 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)

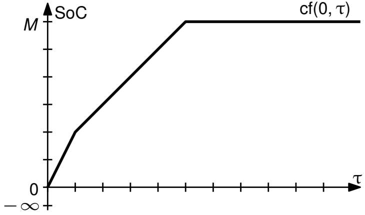
Settling von Ladestationen

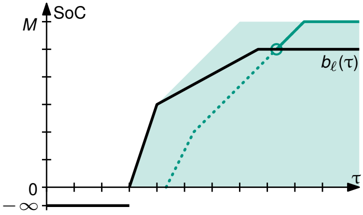
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

- Am ursprünglichem 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)

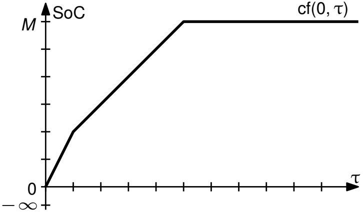
Settling von Ladestationen

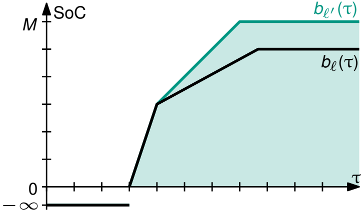
- Nur die letzte Ladestation wird im Label gespeichert
- Erreichen einer Ladestation $\Rightarrow$ Bestimme τ_c für die letzte Station

Problem:

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

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





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Charging Function Propagation (CFP)

Settling von Ladestationen

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

Problem:

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

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

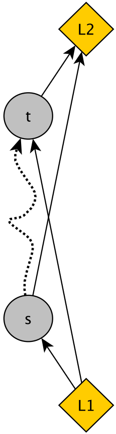
Gegeben:

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

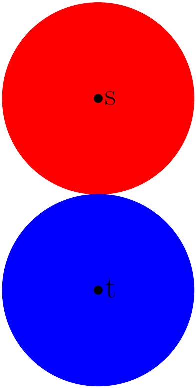
$\Rightarrow$ Erzeuge neues Label $\ell' = (\tau, b_\ell(\tau), v, 0)$

CFP – Beschleunigungstechniken

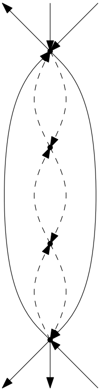
Landmarken



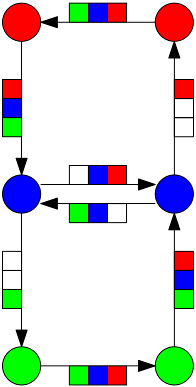
Bidirektionale Suche



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
  (Oben Halten, 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
  (Oben Halten, 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%)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CFP & A*

Erinnerung:

- A* benutzt Knotenpotential um Suche zum Ziel zu leiten
- Potential gibt untere Schranke für Fahrzeit zu t, pro Knoten
- Gute Technik für schwere/komplizierte Suchprobleme
- Klassicher Ansatz: Rückwärtssuche von t

Beobachtung:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CFP & A*

Karlsruher Institut für Technologie

Erinnerung:

- A* benutzt Knotenpotential um Suche zum Ziel zu leiten
- Potential gibt untere Schranke für Fahrzeit zu t, pro Knoten
- Gute Technik für schwere/komplizierte Suchprobleme
- Klassicher Ansatz: Rückwärtssuche von t

Beobachtung:

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

Idee:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CFP & A*

Karlsruher Institut für Technologie

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CFP & A*

Karlsruher Institut für Technologie

Gesucht:

- Potential $\pi : V \times [0, M] \to \mathbb{R}_{\geq 0}$, 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 ω:

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

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

CFP & A*

Karlsruher Institut für Technologie

Algorithmus: Läuft in 2 Phasen:

1: Rückwertssuche von *t* berechnet Potential π

2: Vorwärtssuche von *s* nach *t*, beschleunigt durch π

Potential Berechnung:

- Drei *unikriterielle* Dijkstra Suchen von *t* aus:
  - Auf Metrik τ_d: Berechnet min. Fahrzeit π_τ zu *t* (ohne Energieverbrauch)
  - Auf Metrik γ: Berechnet min Energieverbrauch π_γ
  - Auf Metrik ω: Berechnet min. Fahrzeit π_ω, falls $b_s = 0$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CFP & A*

Algorithmus: Läuft in 2 Phasen:

1: Rückwertssuche von t berechnet Potential π

2: Vorwärtssuche von s nach t, beschleunigt durch π

Potential Berechnung:

- Drei *unikriterielle* Dijkstra Suchen von t aus:
  - Auf Metrik τ_d: Berechnet min. Fahrzeit π_τ zu t (ohne Energieverbrauch)
  - Auf Metrik γ: Berechnet min Energieverbrauch π_γ
  - Auf Metrik ω: Berechnet min. Fahrzeit π_ω, falls $b_s = 0$
- Setze dann:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CHArge – CH & A* & CFP

Algorithmus:

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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

CHArge – CH & A* & CFP

Algorithmus:

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

Heuristiken:

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

Experimente

KIT Karlsruher Institut für Technologie

Straßen Netzwerk:

- Europa (Eur) & Deutschland (Ger) zur Verfügung gestellt von PTV AG

Energieverbrauch:

- PHEM – Entwickelt von der TU Graz [Hausberger et al. '09]
- SRTM Höhendaten (Shuttle Radar Topography Mission)
- Ladestations Positionen 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 [s]	
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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Experimente

	Instanz	M	Preproc.	Exact Query		Heuristic Query	
				Feas.	CHArge	H_ω	H_ω^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

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ende

Literatur:

- Moritz Baum, Julian Dibbelt, Thomas Pajor, Dorothea Wagner:
  Energy-Optimal Routes for Electric Vehicles
  In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013
- Jochen Eisner, Stefan Funke, Sabine Storandt:
  Optimal Route Planning for Electric Vehicles in Large Networks
  In: *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011
- Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, 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*, 2015