Karlsruher Institut für Technologie

Algorithmen für Routenplanung

15. Vorlesung, Sommersemester 2017

Tobias Zündorf | 5. Juli 2017

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

Organisatorisches

Ergebnisse der Vorlesungsevaluation

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

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 $(\tau_t, b_u, u, c_{(u,\ldots,v)})$ 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 $c_{(u,\ldots,v)}$ für den Pfad von u nach v (Initial $\perp$)

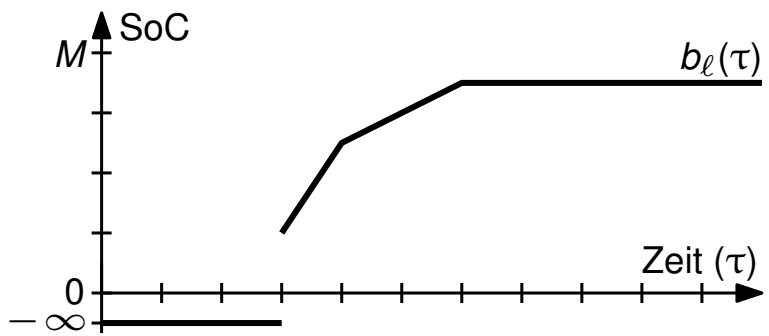
Charging Function Propagation (CFP)

Label: Ein Label ℓ am Knoten v ist ein Tupel $(\tau_t, b_u, u, c_{(u,\ldots,v)})$ 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 $c_{(u,\ldots,v)}$ 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 $b_\ell(\tau)$:
  $b_\ell(\tau) := b' - c_{(u,\ldots,v)}(b')$ mit
  $b' := \mathrm{cf}_u(b_u, \tau - \tau_t)$



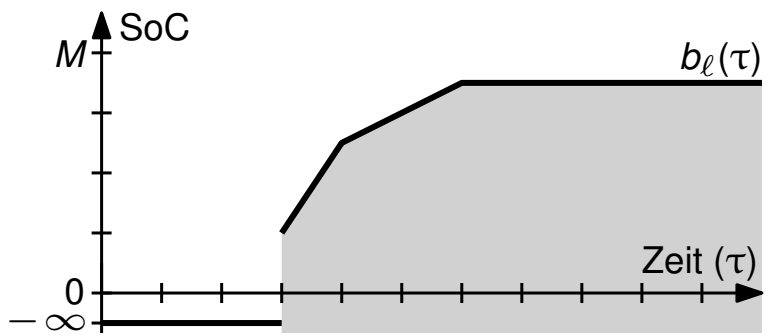
Charging Function Propagation (CFP)

Label: Ein Label ℓ am Knoten v ist ein Tupel $(\tau_t, b_u, u, c_{(u,\ldots,v)})$ 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 $c_{(u,\ldots,v)}$ für den Pfad von u nach v (Initial $\perp$)

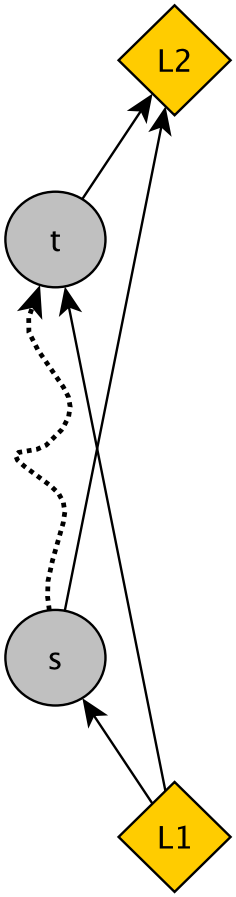
Interpretation:

- 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 $b_\ell(\tau)$:
  $b_\ell(\tau) := b' - c_{(u,\ldots,v)}(b')$ mit
  $b' := \mathrm{cf}_u(b_u, \tau - \tau_t)$

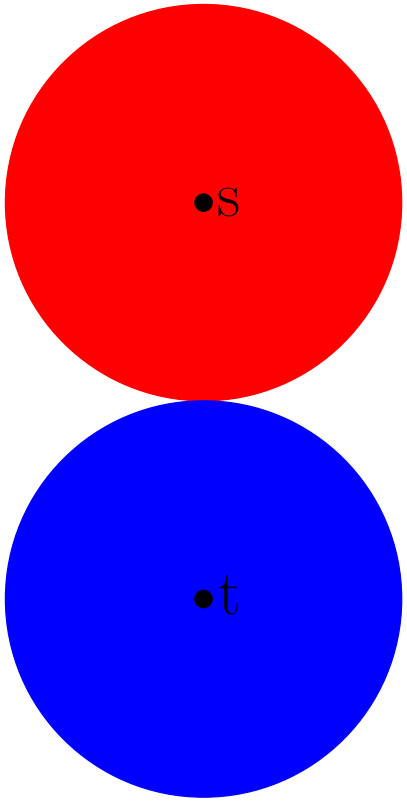


CFP – Beschleunigungstechniken

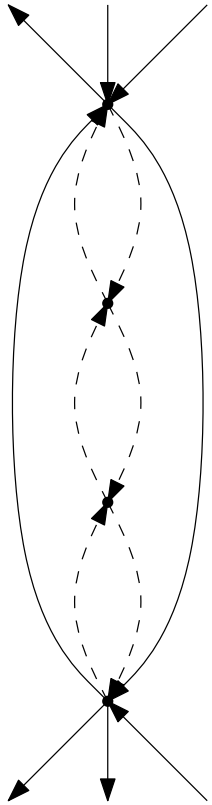
Landmarken



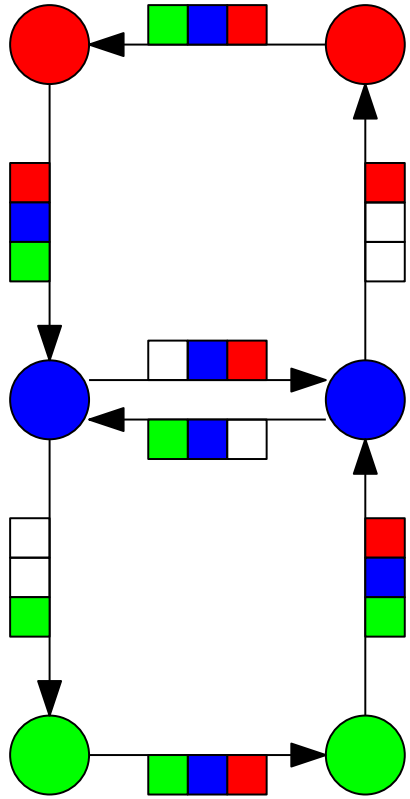
Bidirektionale Suche



Kontraktion



Arc-Flags



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

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%)

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

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

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

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 ω:

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

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

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 c: Berechnet min Energieverbrauch π_c
  - Auf Metrik ω: Berechnet min. Fahrzeit π_ω, falls $b_s = 0$

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 c: Berechnet min Energieverbrauch π_c
  - Auf Metrik ω: Berechnet min. Fahrzeit π_ω, falls $b_s = 0$
- Setze dann:

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

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

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

Straßen Netzwerk:

- Europa (Eur) & Deutschland (Ger)

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 $c < 0$	# 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

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

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

Einfaches Modell

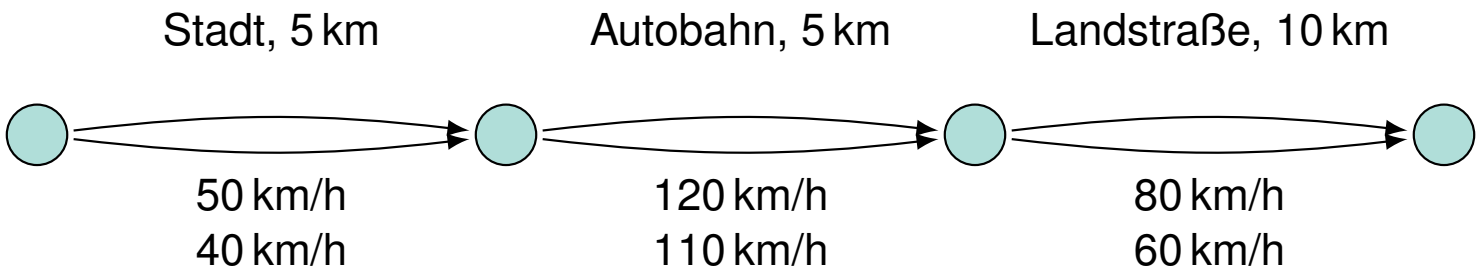
Ziel: Energie sparen durch Anpassung der Geschwindigkeit

- Pro Straßensegment: Interval mit min./max. Geschwindigkeit

Erster Ansatz:

- Diskretisiere mögliche Geschwindigkeiten pro Segment
- Eine Kante pro Geschwindigkeit

Routing auf Multigraph



Einfaches Modell

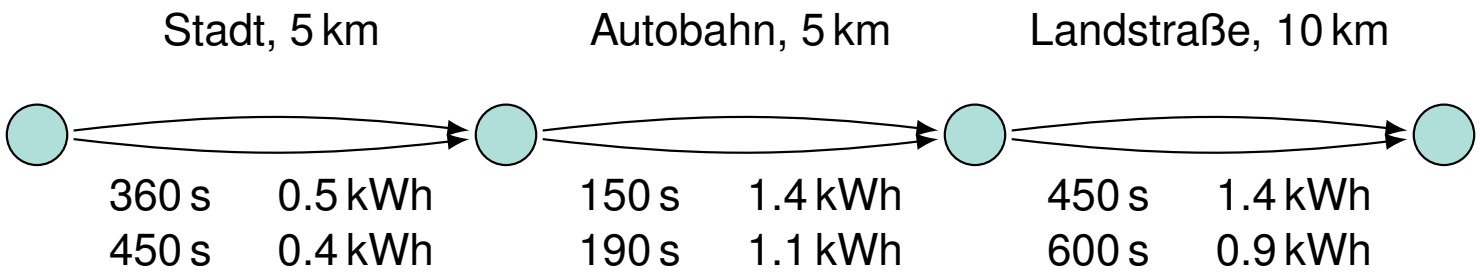
Ziel: Energie sparen durch Anpassung der Geschwindigkeit

- Pro Straßensegment: Interval mit min./max. Geschwindigkeit

Erster Ansatz:

- Diskretisiere mögliche Geschwindigkeiten pro Segment
- Eine Kante pro Geschwindigkeit

Routing auf Multigraph



Einfaches Modell

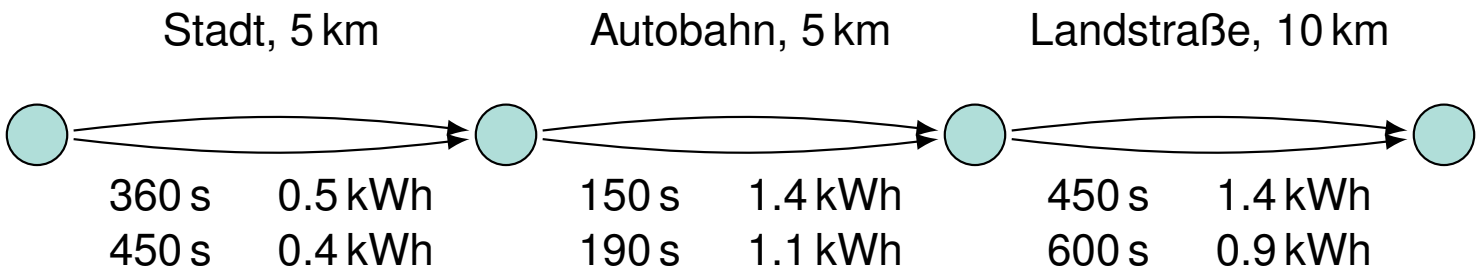
Ziel: Energie sparen durch Anpassung der Geschwindigkeit

- Pro Straßensegment: Interval mit min./max. Geschwindigkeit

Erster Ansatz:

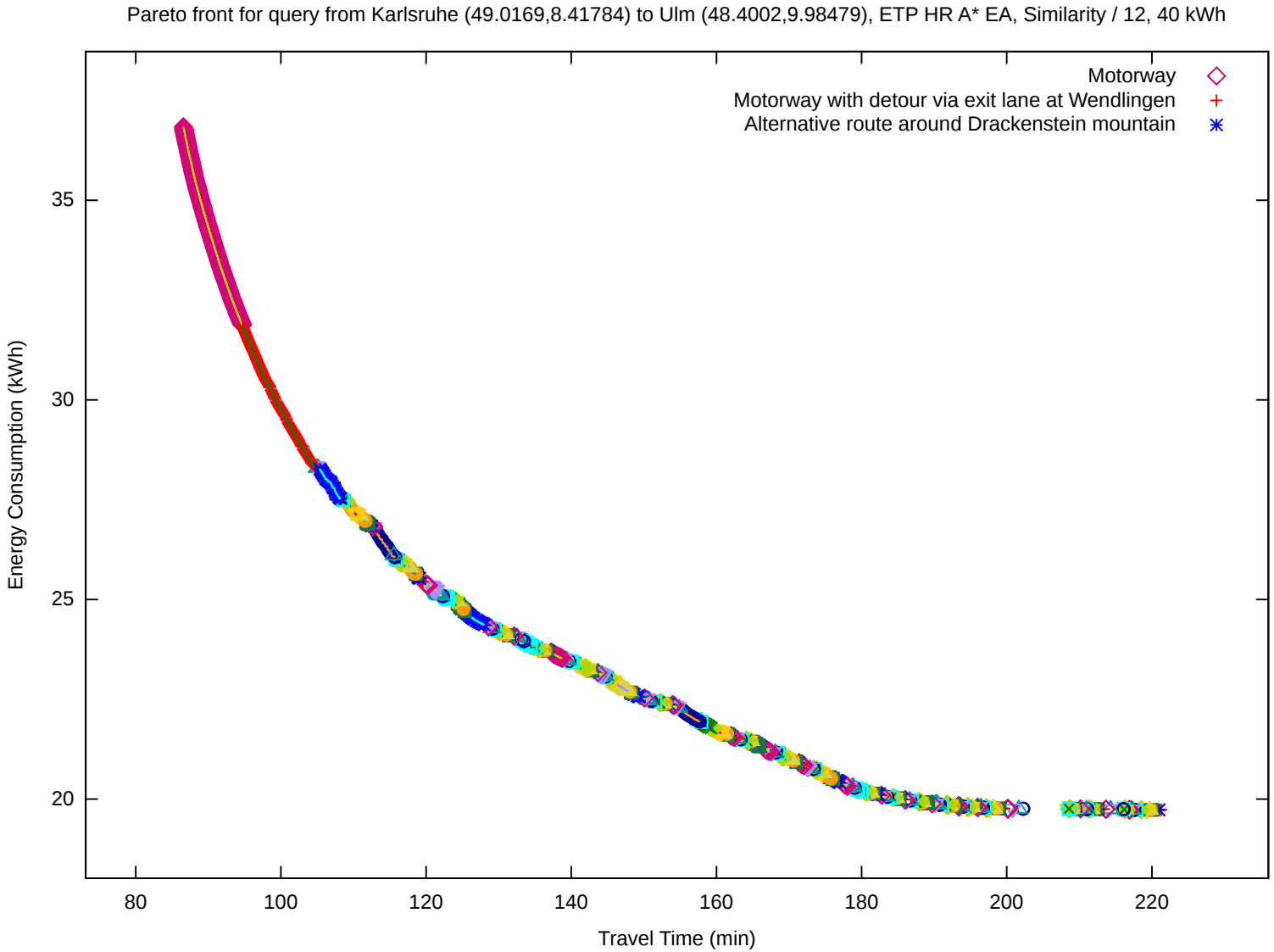
- Diskretisiere mögliche Geschwindigkeiten pro Segment
- Eine Kante pro Geschwindigkeit

Routing auf Multigraph



Einfache Umsetzung, aber hohe Lösungskomplexität

Einfaches Modell – Beispiel Pareto Menge



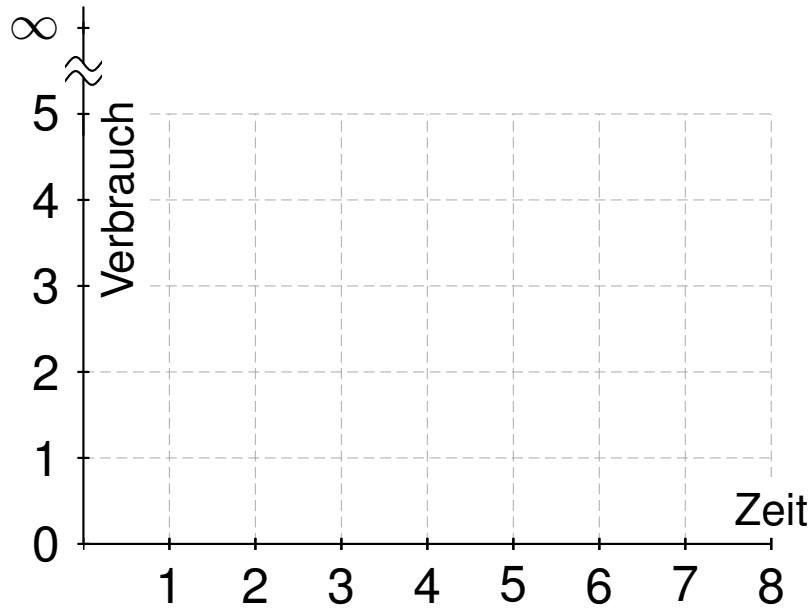
Realistisches Modell

Energieverbrauch auf einem Segment: $\lambda_1 v^2 + \lambda_2$.
v: Geschwindigkeit

- Tradeoff Funktion

$$c(x) = \frac{\alpha}{(x-\beta)^2} + \gamma$$

- Bildet Fahrzeit auf Verbrauch ab
- Minimale Fahrzeit $\underline{x}$
- Maximale Fahrzeit $\overline{x}$



Minimaler Verbrauch entlang eines Pfades?

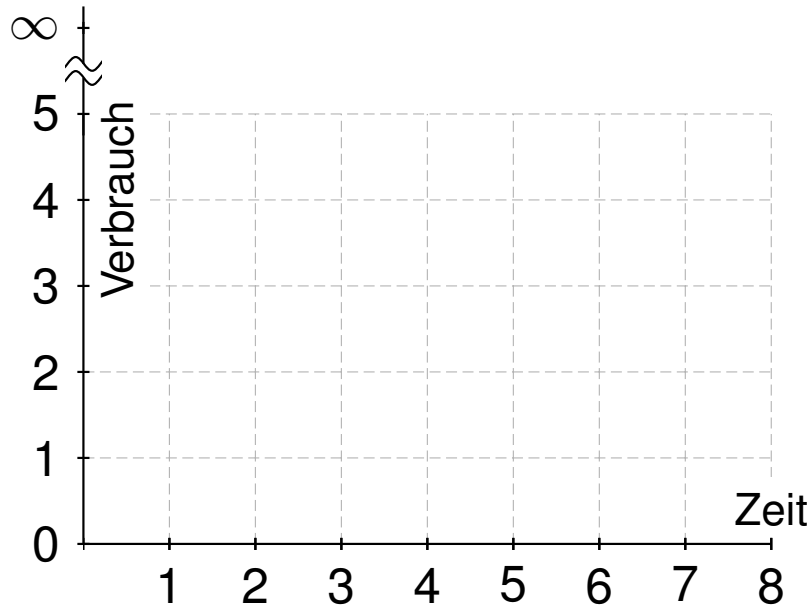
Realistisches Modell

Energieverbrauch auf einem Segment: $\lambda_1 v^2 + \lambda_2$.
v: Geschwindigkeit

- Tradeoff Funktion

$$c(x) = \frac{\alpha}{(x-\beta)^2} + \gamma$$

- Bildet Fahrzeit auf Verbrauch ab
- Minimale Fahrzeit $\underline{x}$
- Maximale Fahrzeit $\overline{x}$



Minimaler Verbrauch entlang eines Pfades?

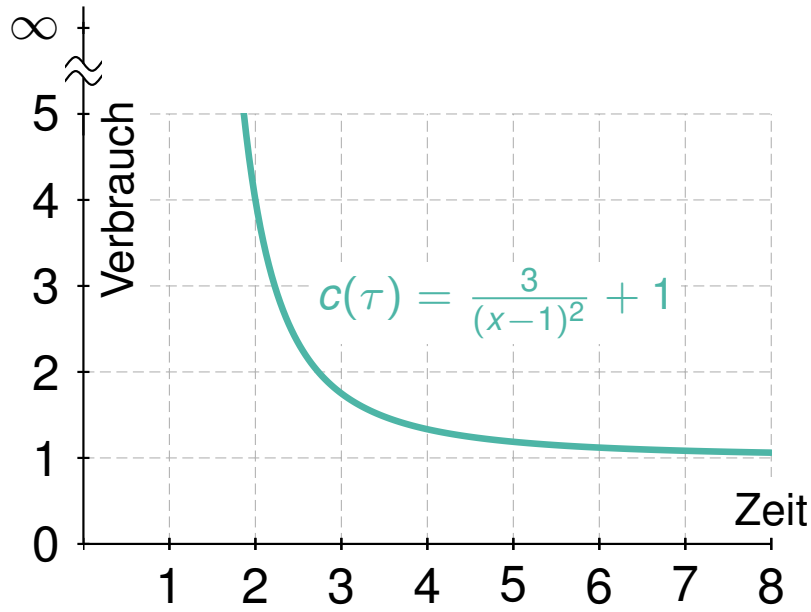
Realistisches Modell

Energieverbrauch auf einem Segment: $\lambda_1 v^2 + \lambda_2$.
v: Geschwindigkeit

- Tradeoff Funktion

$$c(x) = \frac{\alpha}{(x-\beta)^2} + \gamma$$

- Bildet Fahrzeit auf Verbrauch ab
- Minimale Fahrzeit $\underline{x}$
- Maximale Fahrzeit $\overline{x}$



Minimaler Verbrauch entlang eines Pfades?

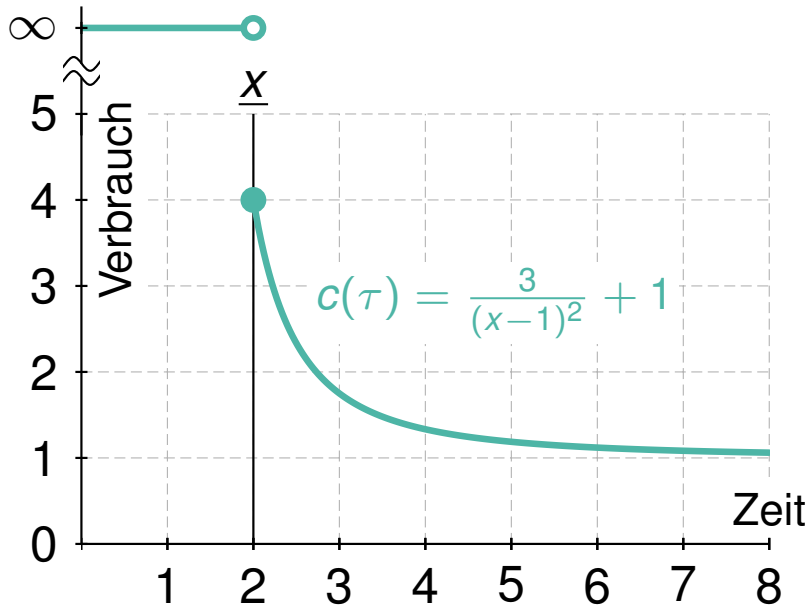
Realistisches Modell

Energieverbrauch auf einem Segment: $\lambda_1 v^2 + \lambda_2$.
v: Geschwindigkeit

- Tradeoff Funktion

$$c(x) = \frac{\alpha}{(x-\beta)^2} + \gamma$$

- Bildet Fahrzeit auf Verbrauch ab
- Minimale Fahrzeit $\underline{x}$
- Maximale Fahrzeit $\overline{x}$



Minimaler Verbrauch entlang eines Pfades?

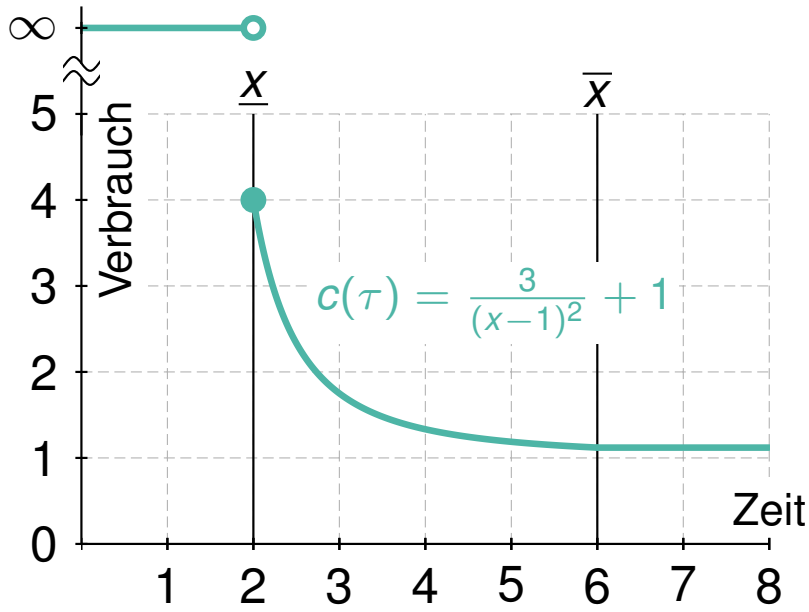
Realistisches Modell

Energieverbrauch auf einem Segment: $\lambda_1 v^2 + \lambda_2$.
v: Geschwindigkeit

- Tradeoff Funktion

$$c(x) = \frac{\alpha}{(x-\beta)^2} + \gamma$$

- Bildet Fahrzeit auf Verbrauch ab
- Minimale Fahrzeit $\underline{x}$
- Maximale Fahrzeit $\overline{x}$



Minimaler Verbrauch entlang eines Pfades?

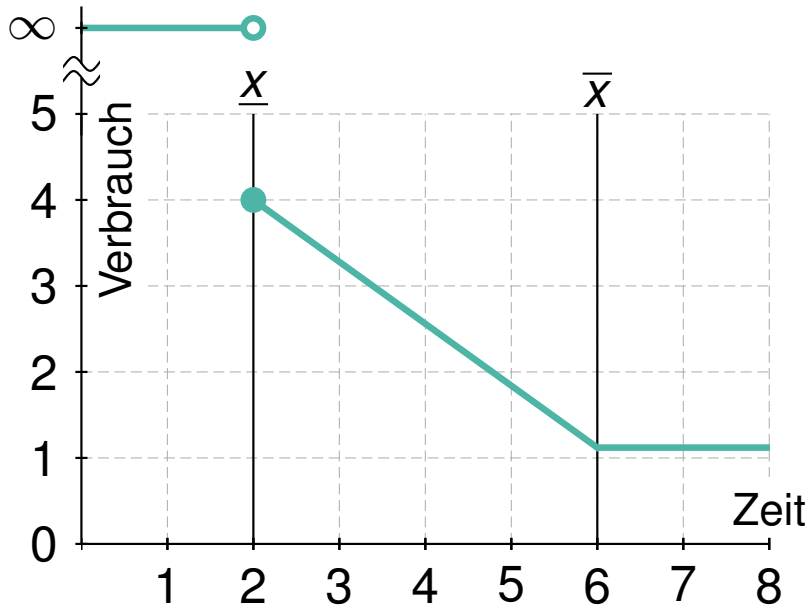
Realistisches Modell

Energieverbrauch auf einem Segment: $\lambda_1 v^2 + \lambda_2$.
v: Geschwindigkeit

- Tradeoff Funktion

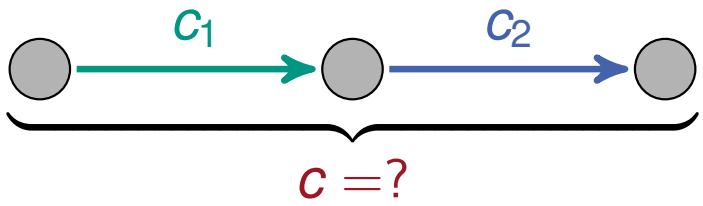
$$c(x) = \alpha x + \beta$$

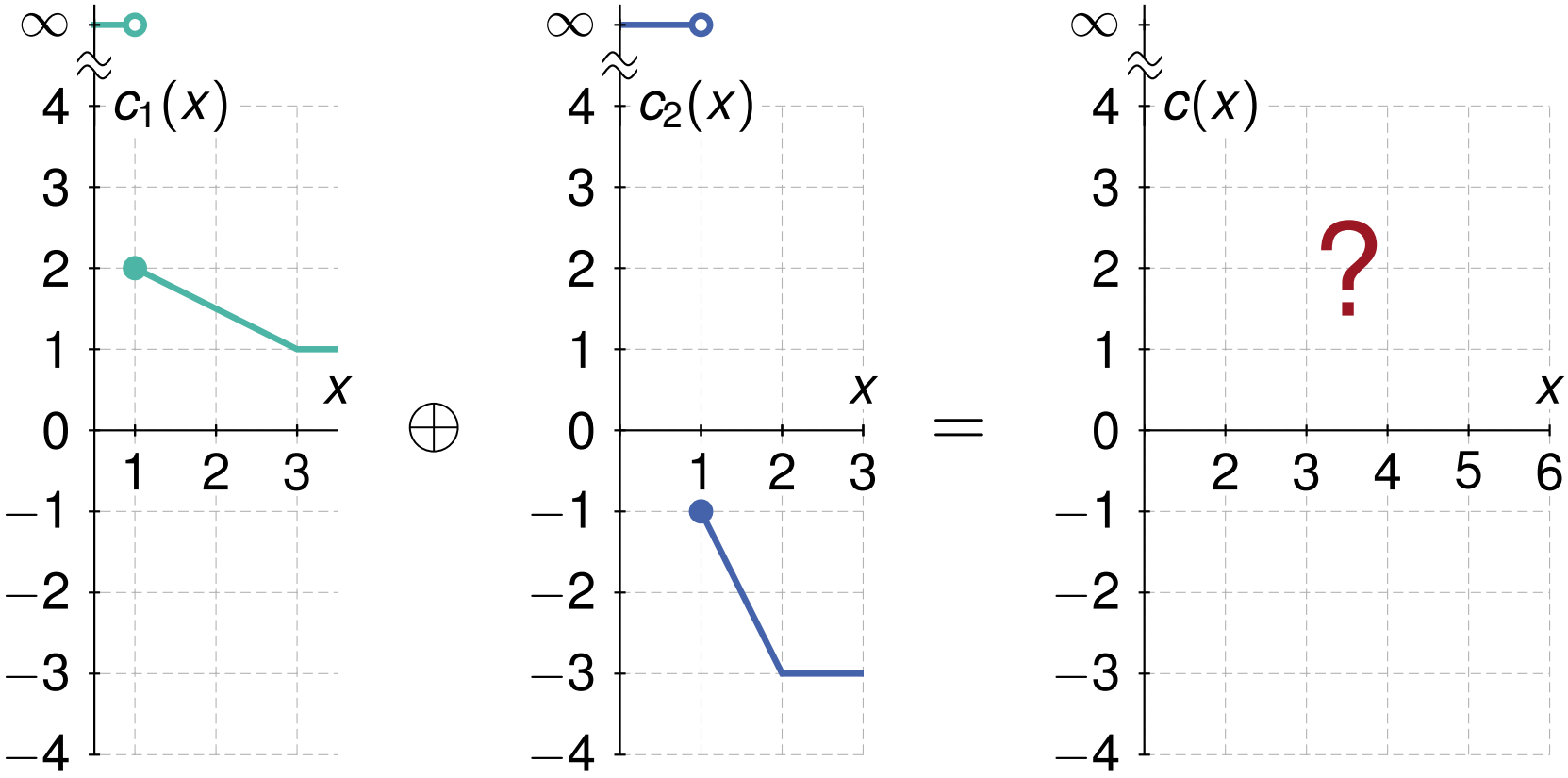
- Bildet Fahrzeit auf Verbrauch ab
- Minimale Fahrzeit $\underline{x}$
- Maximale Fahrzeit $\overline{x}$



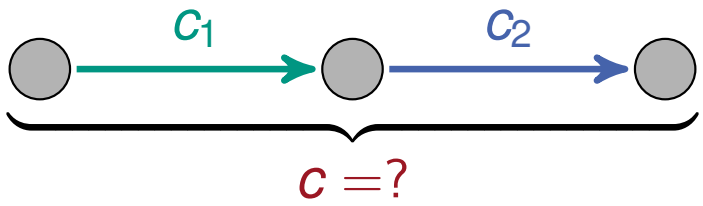
Minimaler Verbrauch entlang eines Pfades?

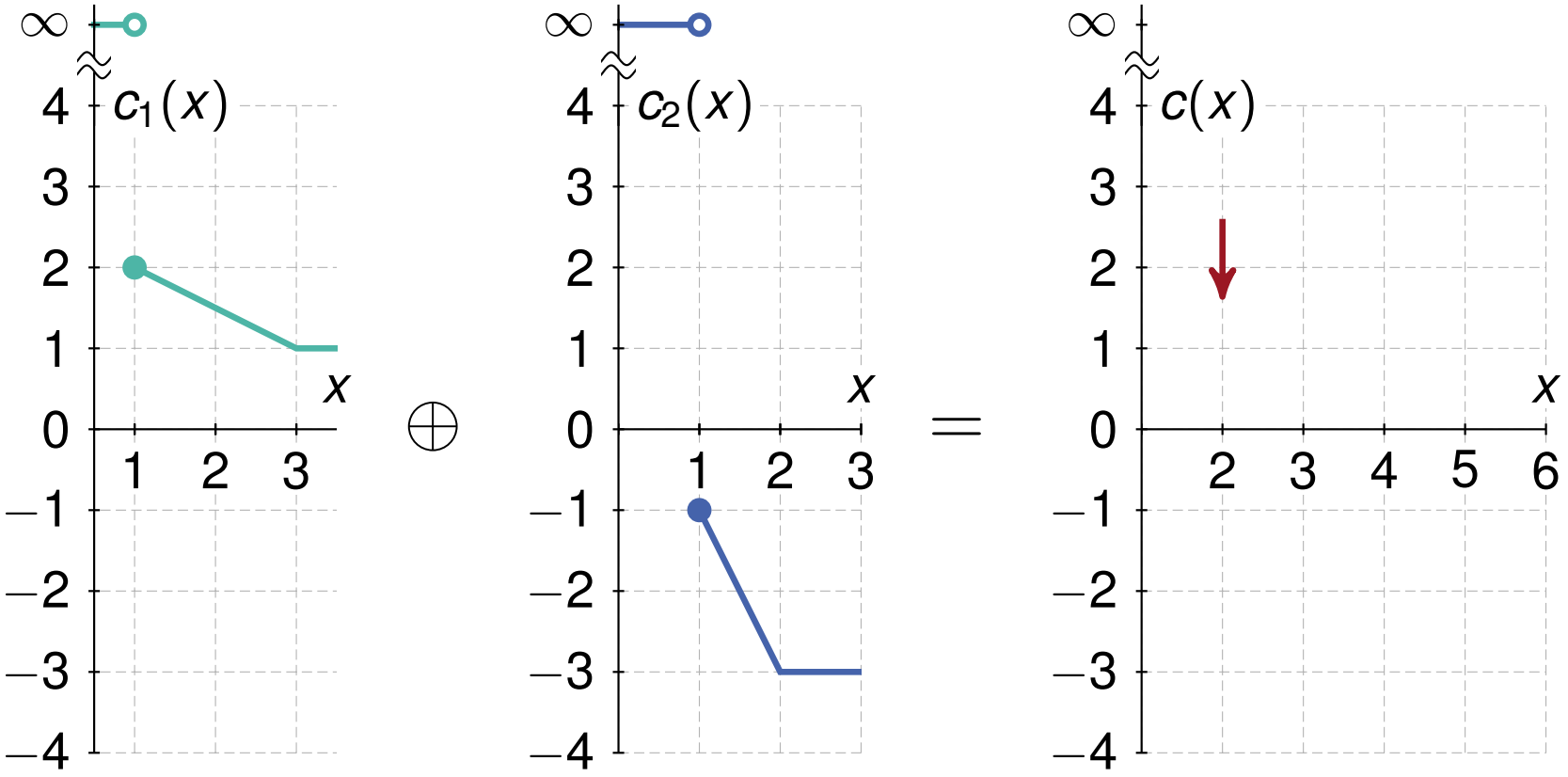
Tradeoff Funktionen für Pfade





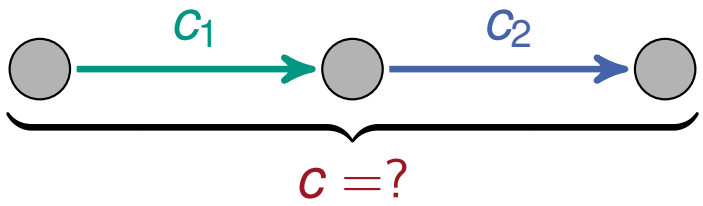
Tradeoff Funktionen für Pfade

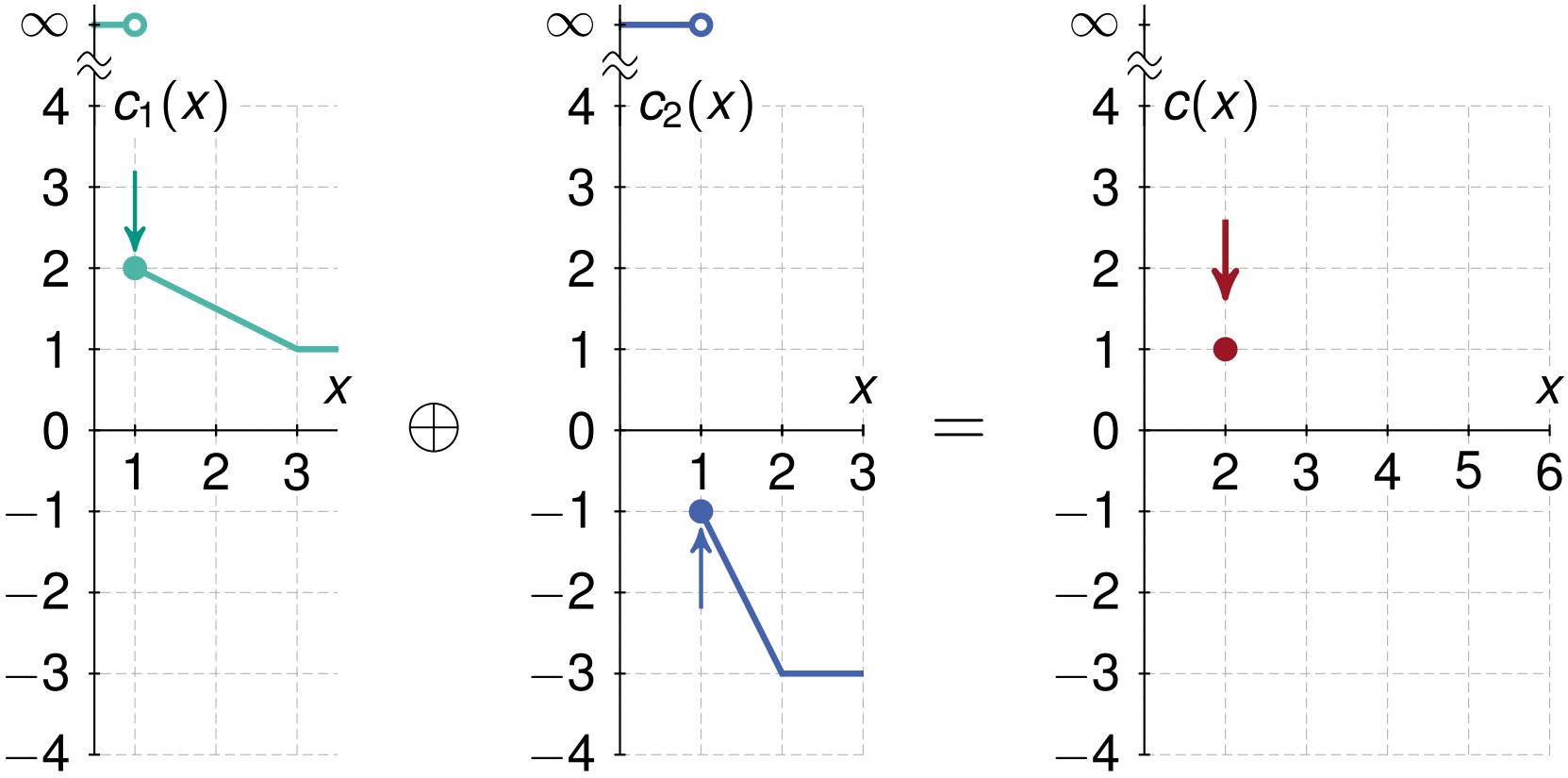




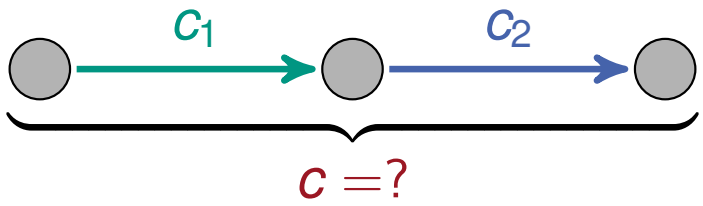
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

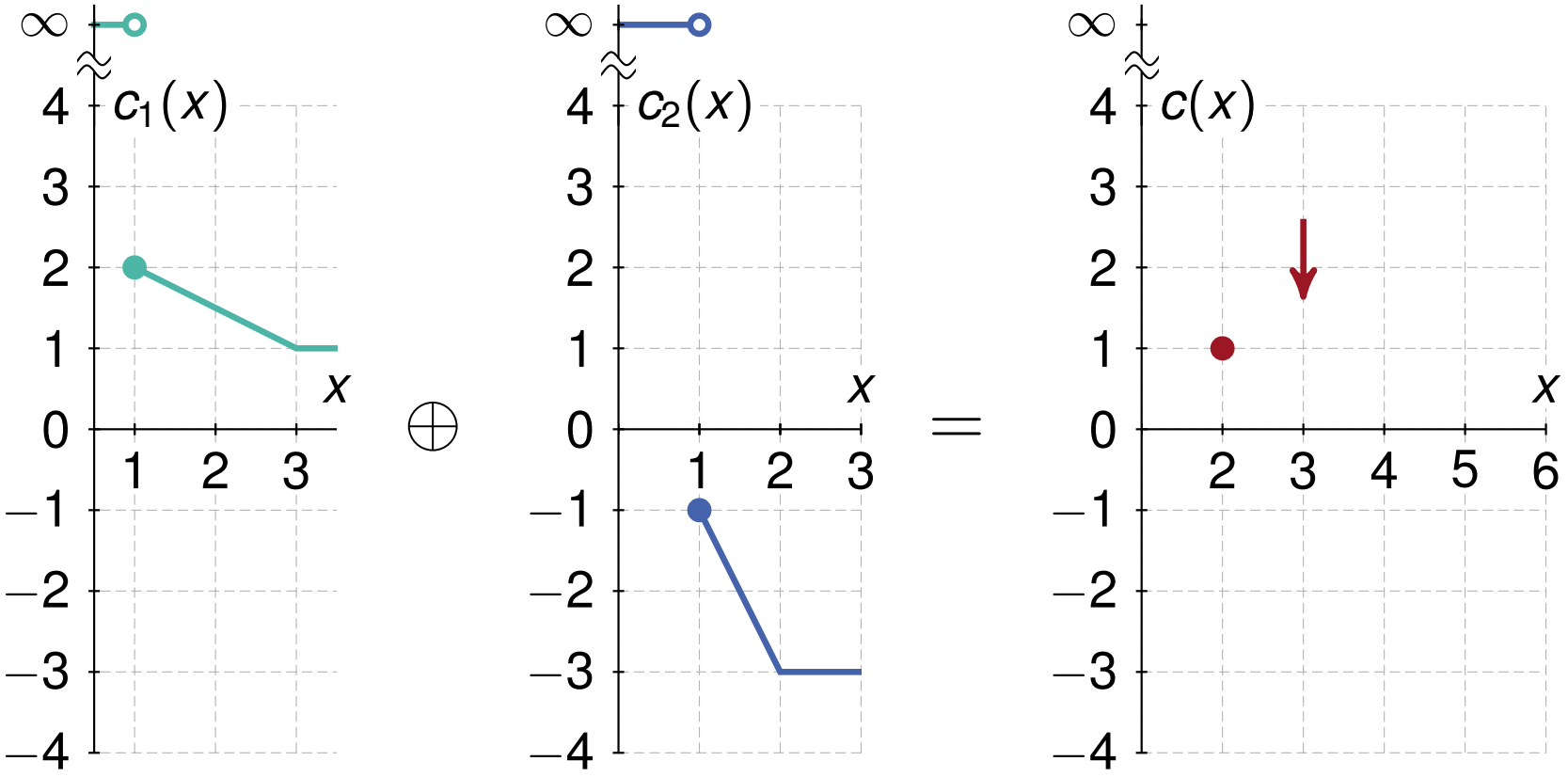
Tradeoff Funktionen für Pfade



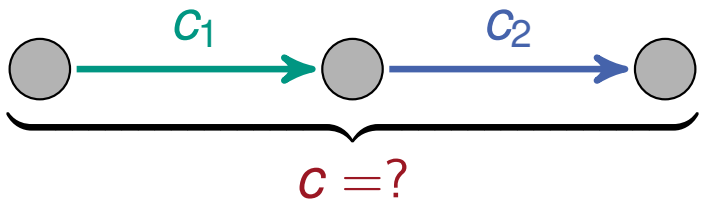


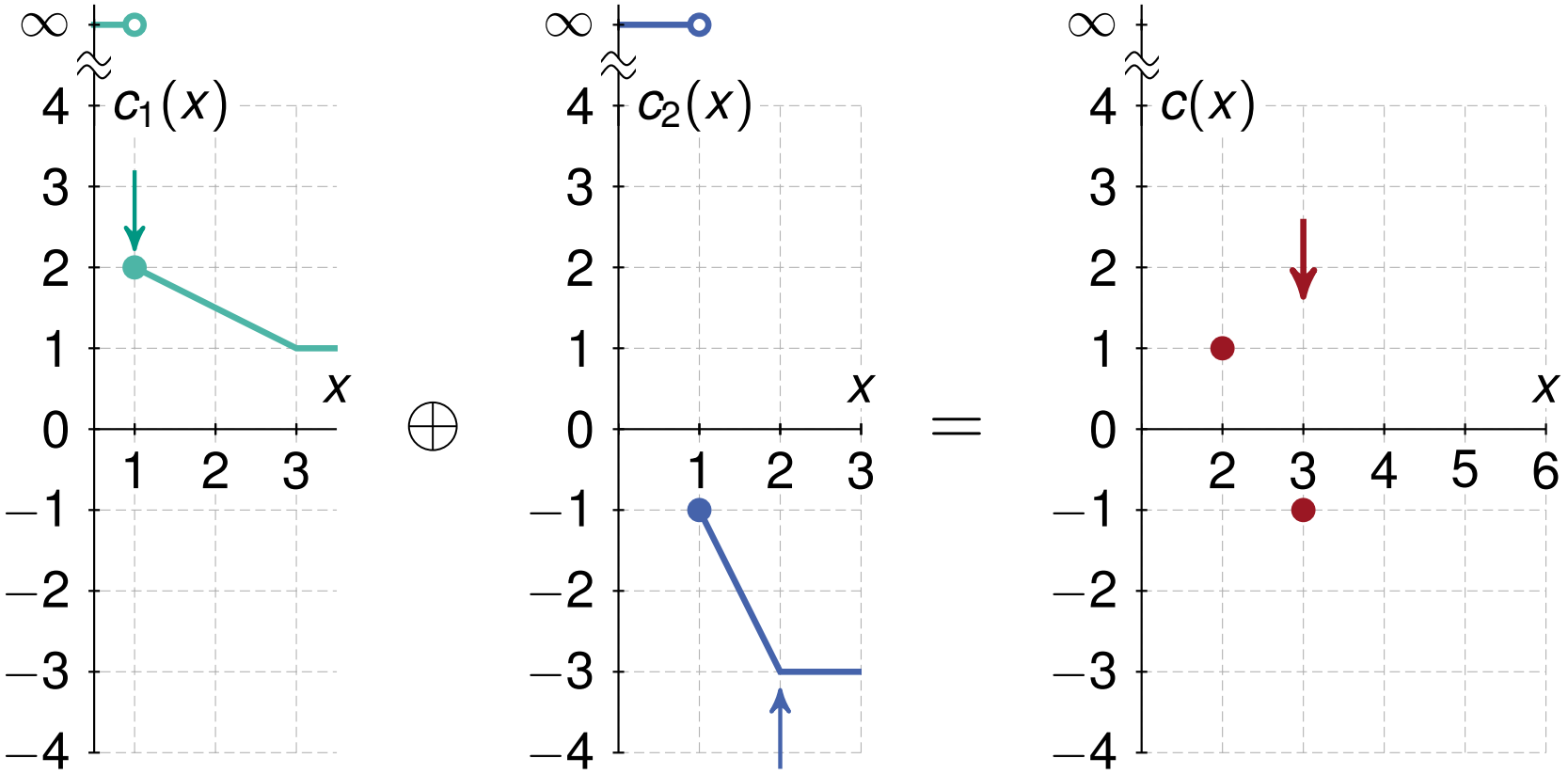
Tradeoff Funktionen für Pfade



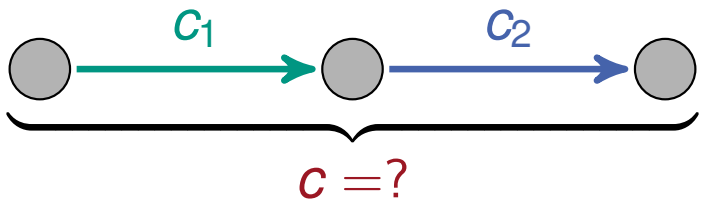


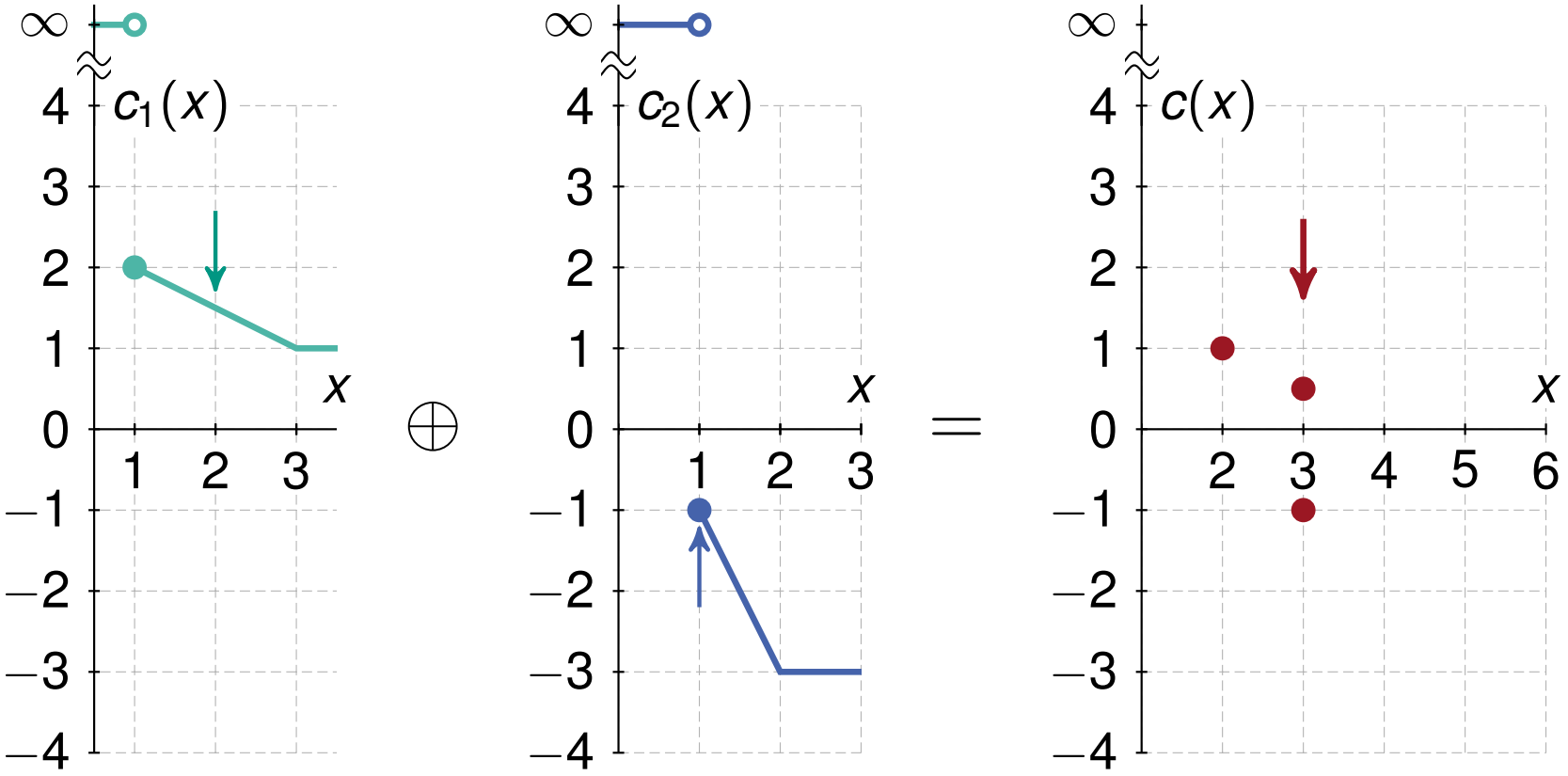
Tradeoff Funktionen für Pfade



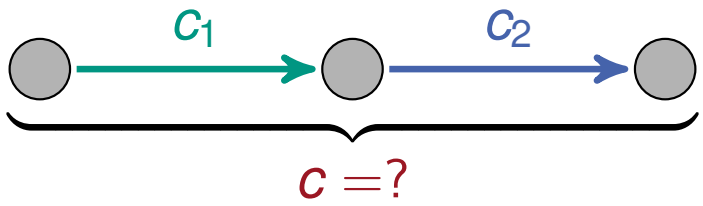


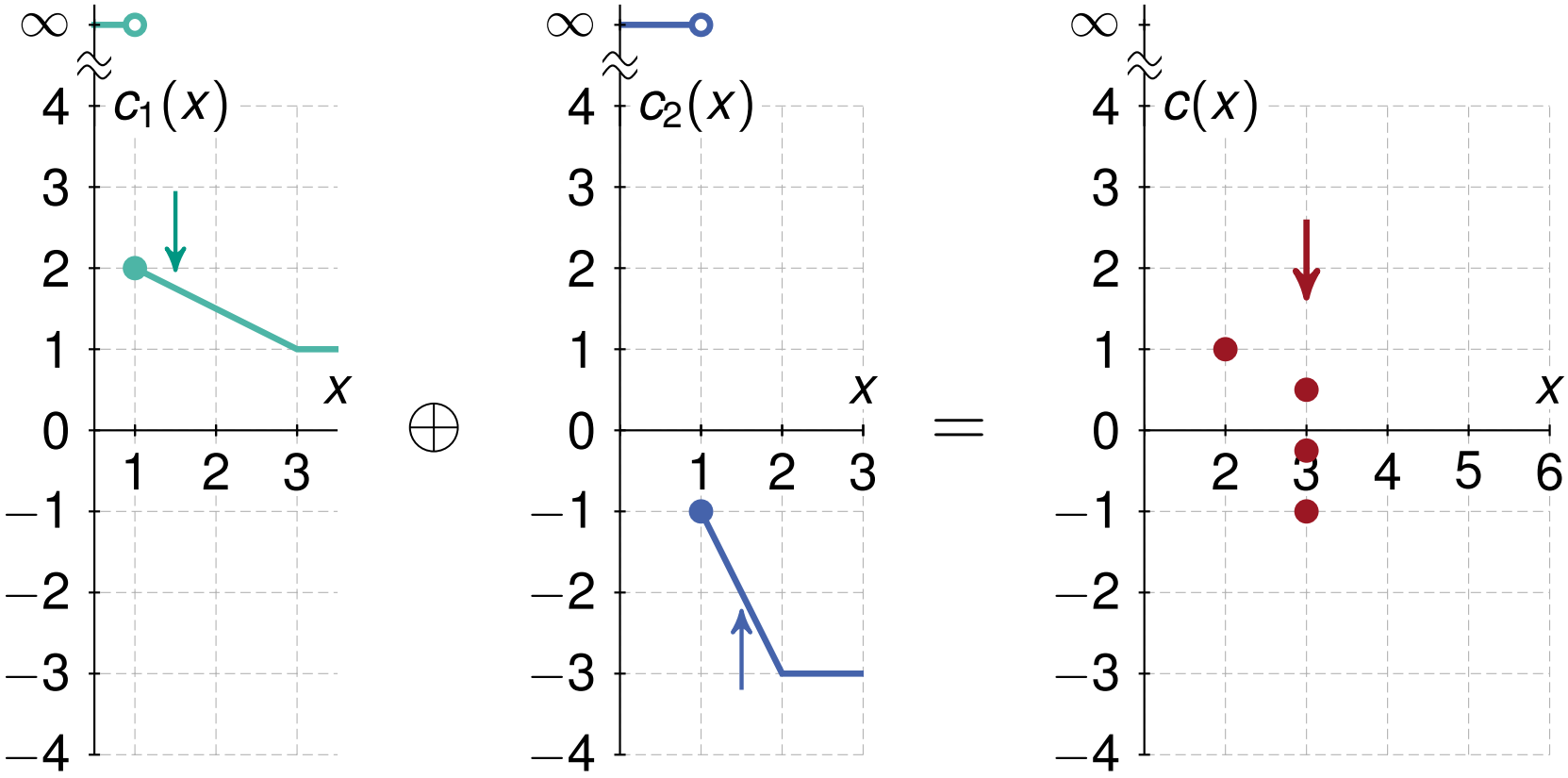
Tradeoff Funktionen für Pfade



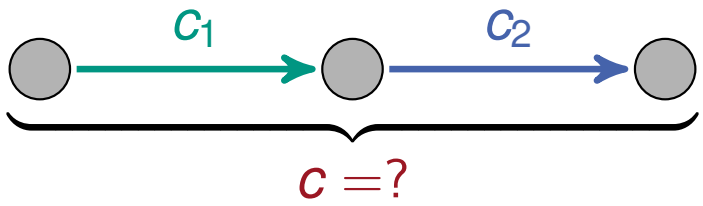


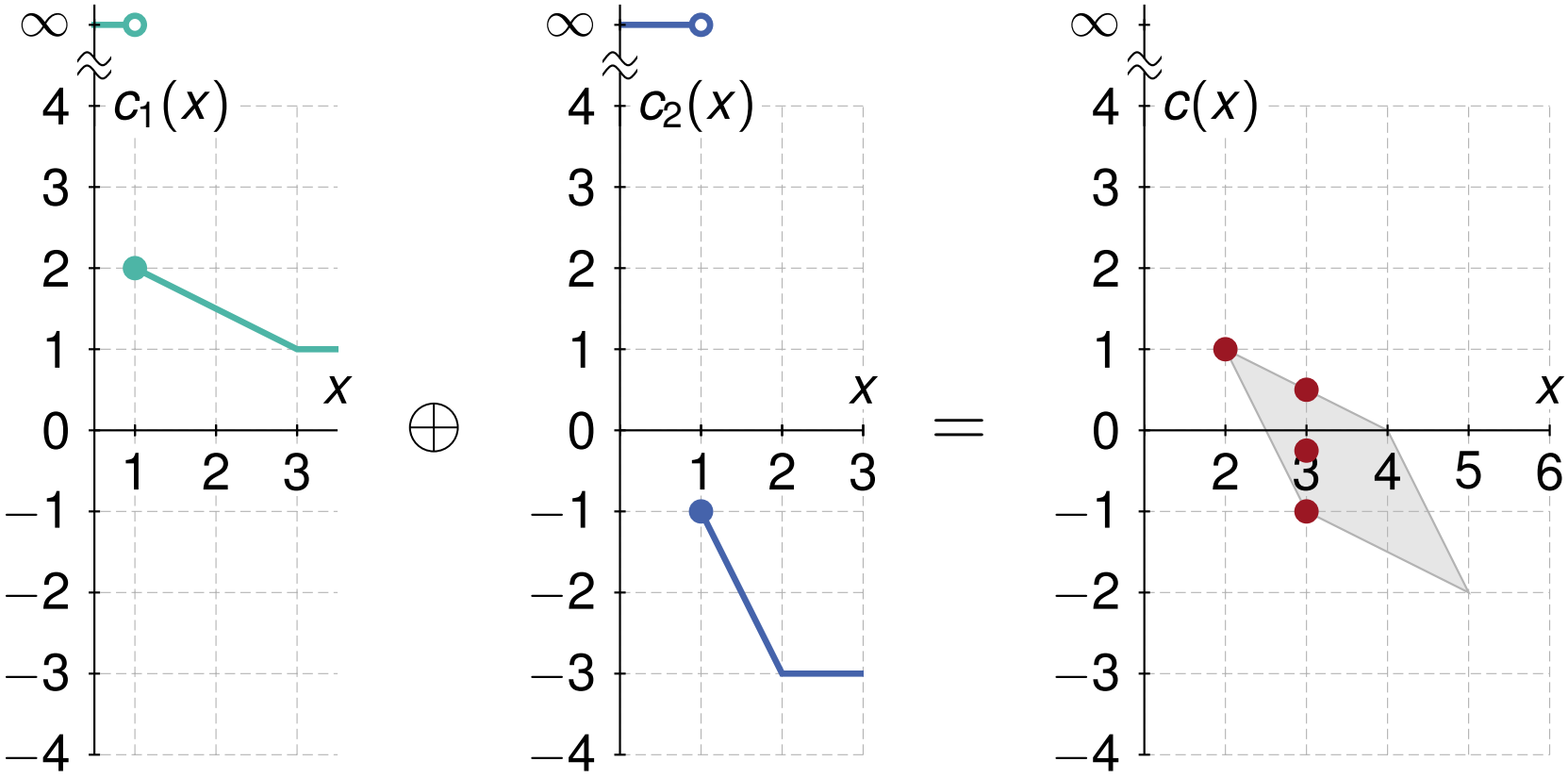
Tradeoff Funktionen für Pfade



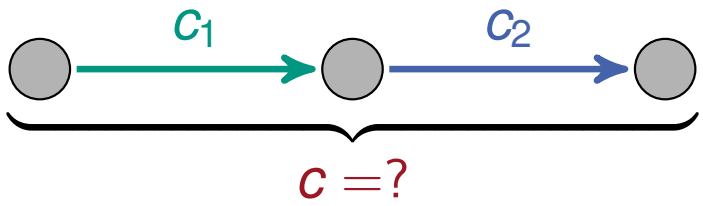


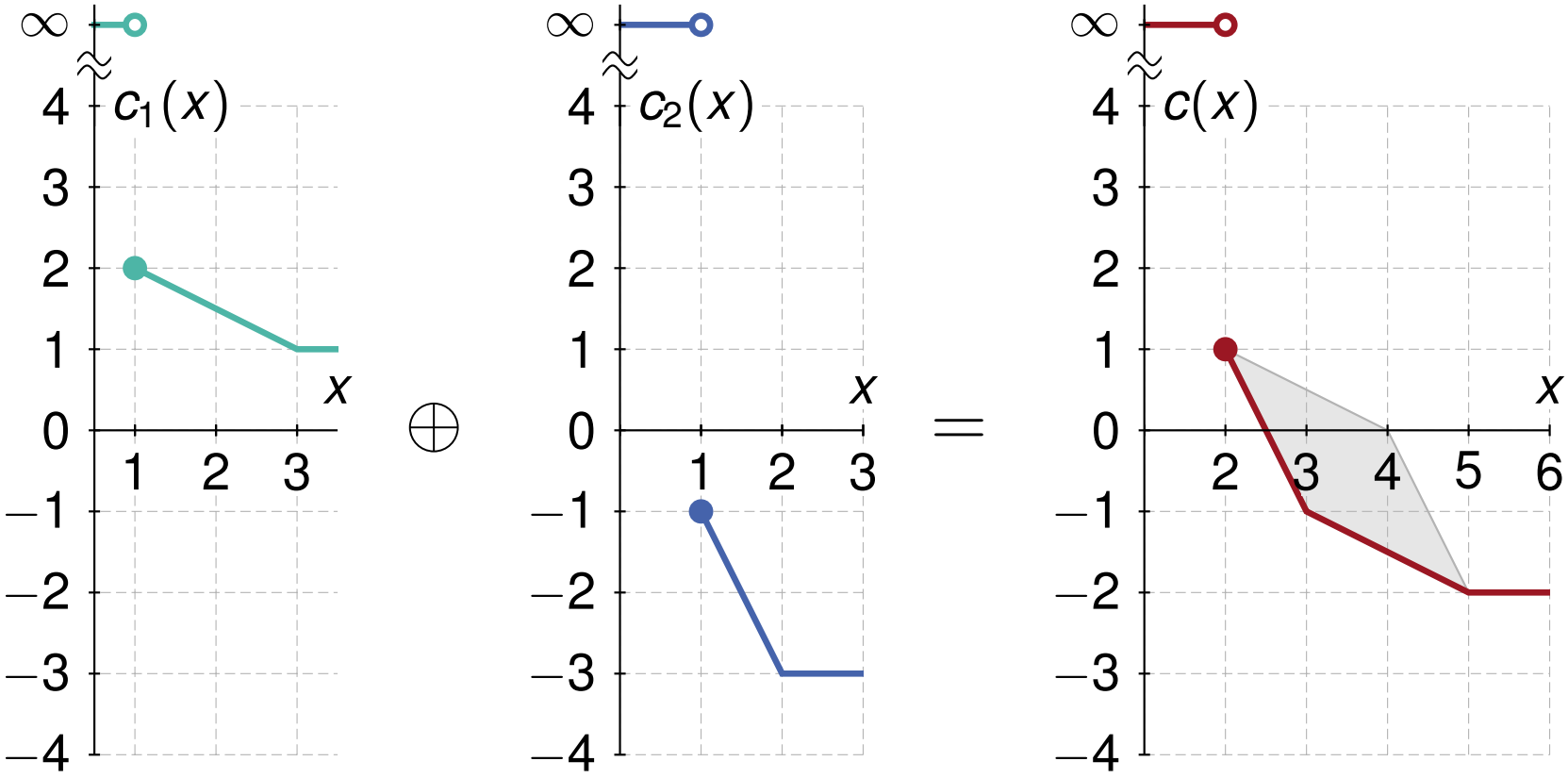
Tradeoff Funktionen für Pfade



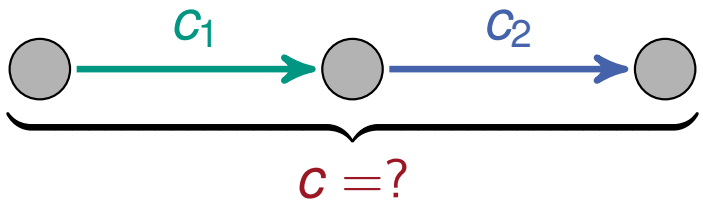


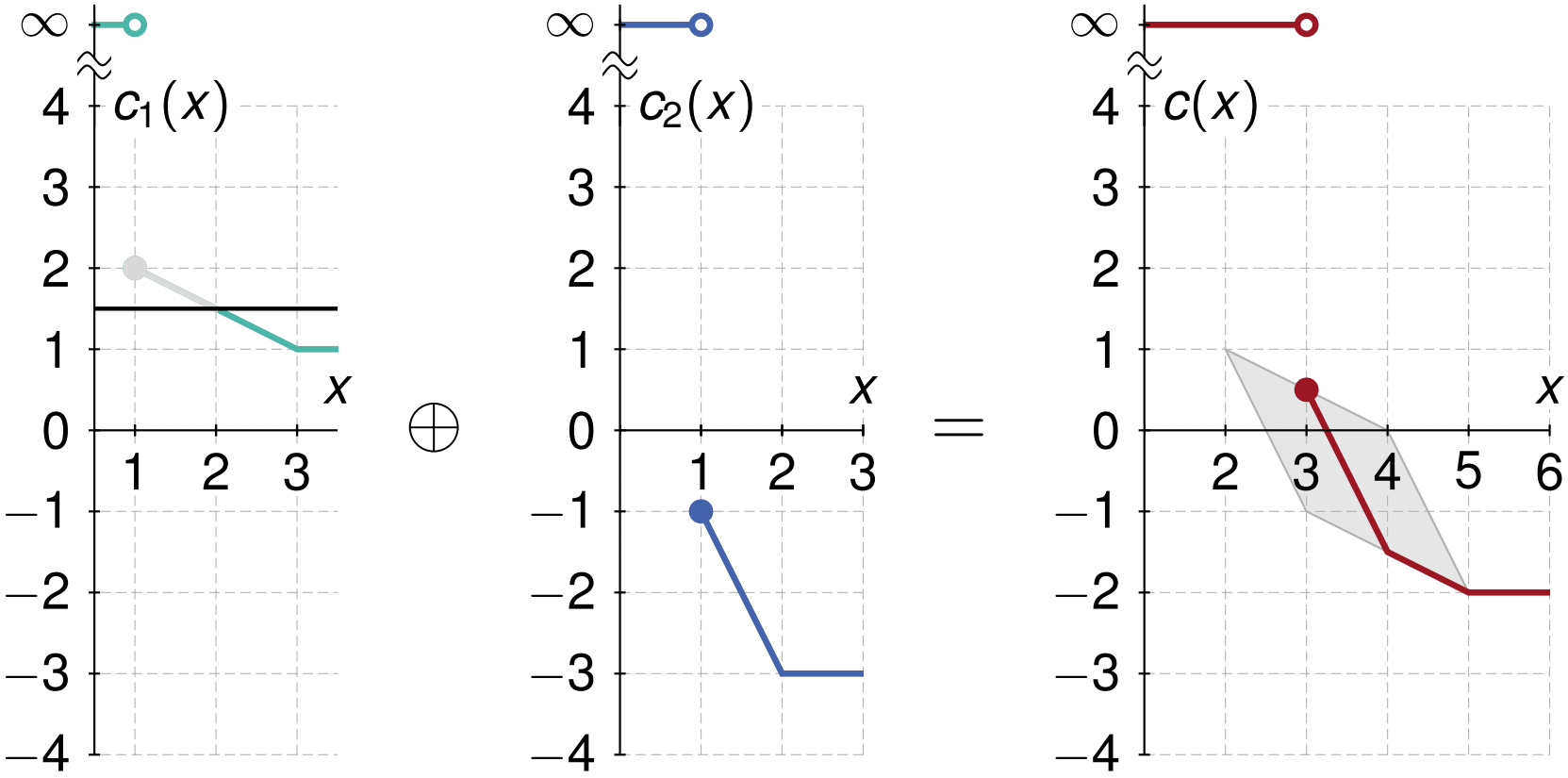
Tradeoff Funktionen für Pfade



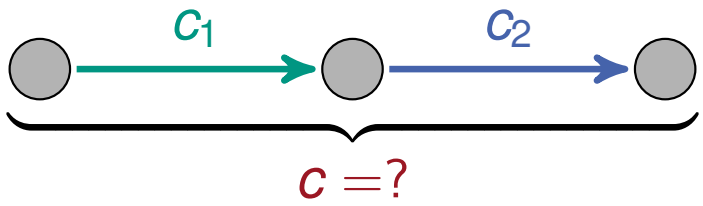


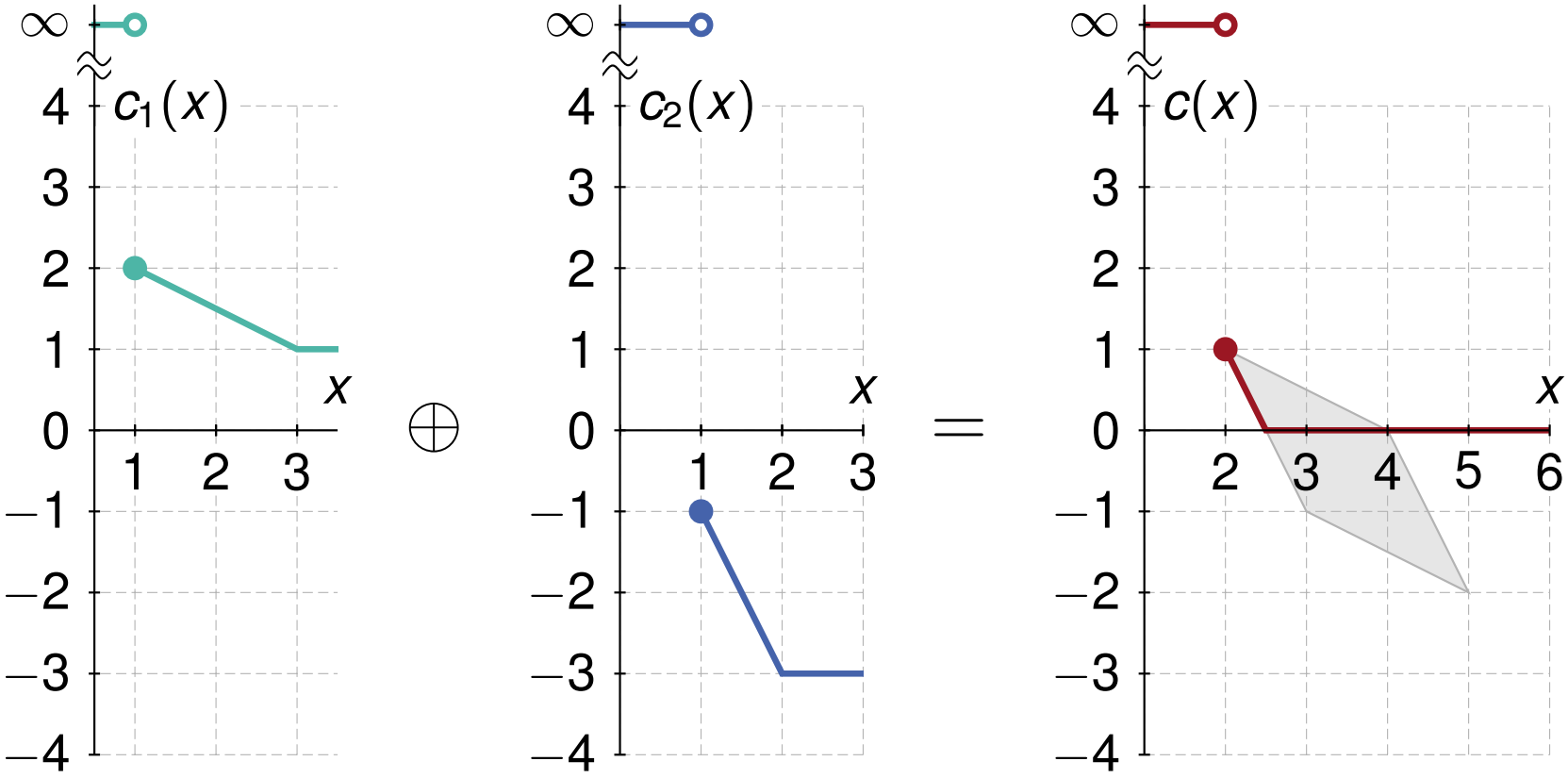
Tradeoff Funktionen für Pfade



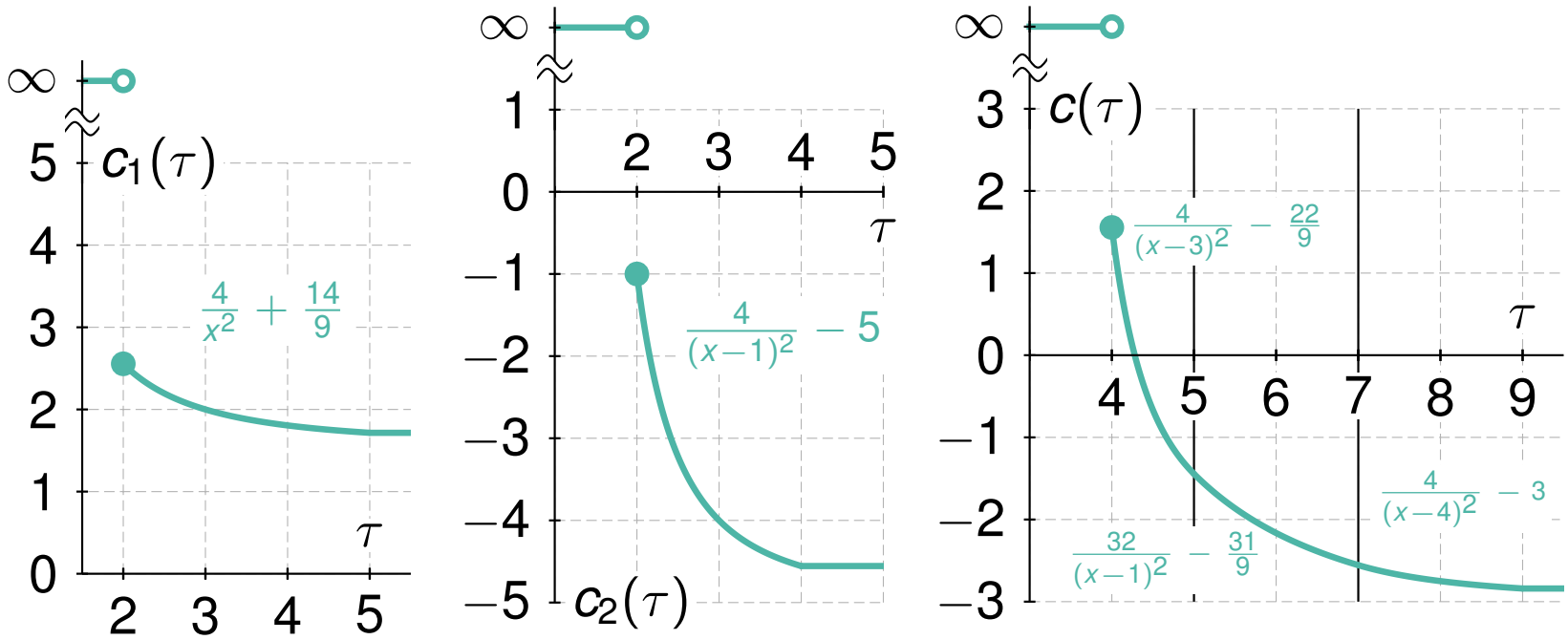


Tradeoff Funktionen für Pfade





Beispiel: Linking

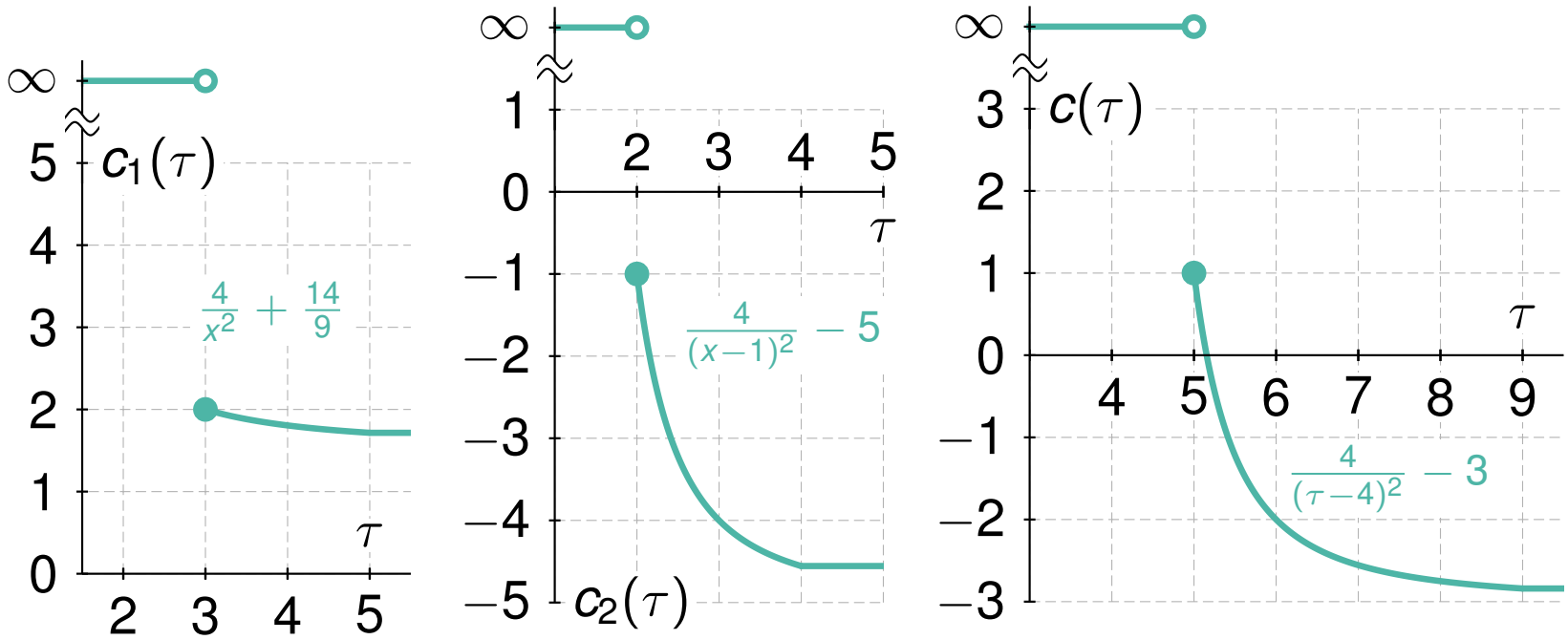


Initialer SoC (an c_1): 4, min. SoC: 0, max. SoC: 8

Battery Constraints beeinflussen Resultat zusätzlich:

- Akku fast leer: Geschwindigkeit muss auf c_1 reduziert werden
- Akku voll: Langsam fahren auf c_2 lohnt nicht

Beispiel: Linking

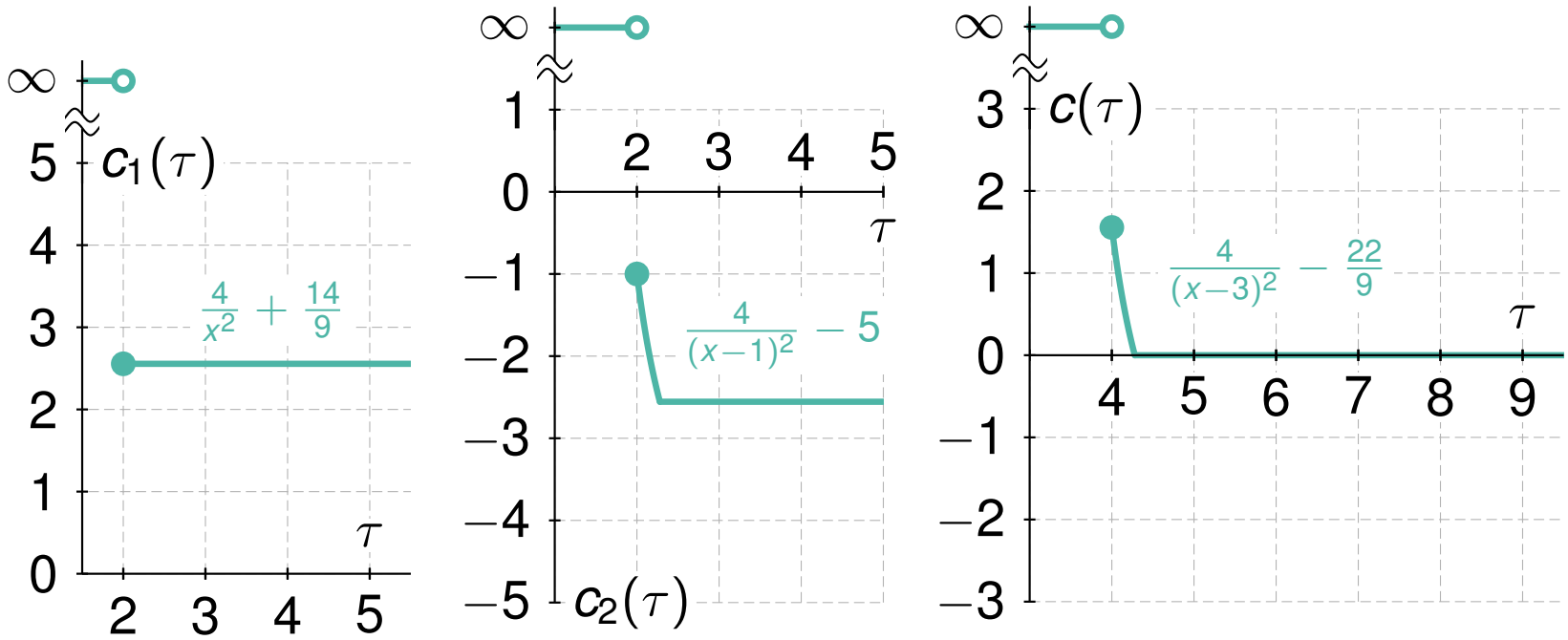


Initialer SoC (an c_1): 2, min. SoC: 0, max. SoC: 8

Battery Constraints beeinflussen Resultat zusätzlich:

- Akku fast leer: Geschwindigkeit muss auf c_1 reduziert werden
- Akku voll: Langsam fahren auf c_2 lohnt nicht

Beispiel: Linking



Initialer SoC (an c_1): 8, min. SoC: 0, max. SoC: 8

Battery Constraints beeinflussen Resultat zusätzlich:

- Akku fast leer: Geschwindigkeit muss auf c_1 reduziert werden
- Akku voll: Langsam fahren auf c_2 lohnt nicht

Basisalgorithmus

Herausforderungen:

- Integration von Funktionen in Suchalgorithmus
- Battery Constraints „on-the-fly“

Experimente:
Straßennetz Europa, EV mit 2 kWh Akku (ca. 10 km Reichweite)

Algorithmus	# Labels	# Vgl.	Zeit [ms]
Multigraph	30 990 k	21 301 M	47 755
Tradeoff Fkt.	103 k	4 M	444

(Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM)

- Pfade im Multigraph länger

Realistisches Modell zahlt sich aus

Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

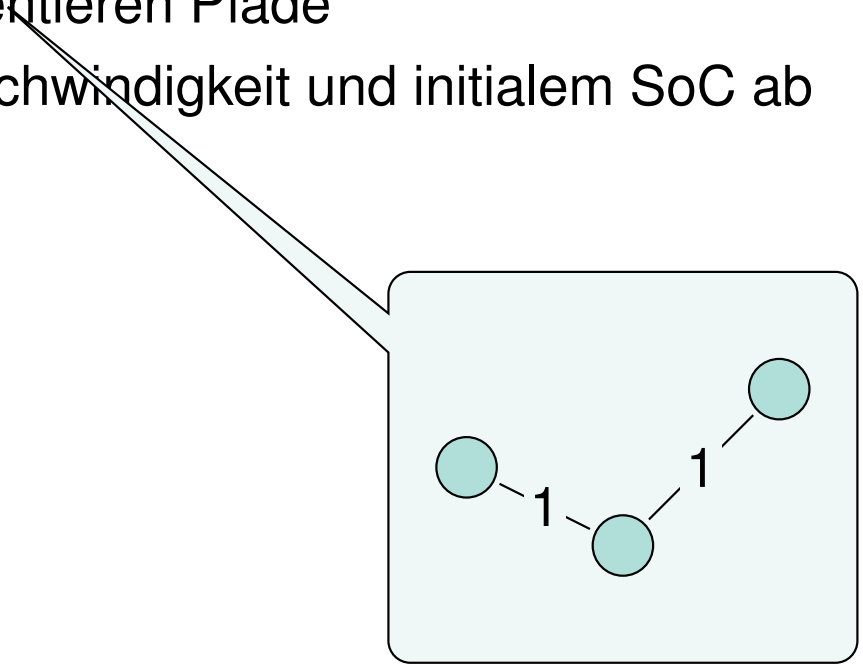
- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben

Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben

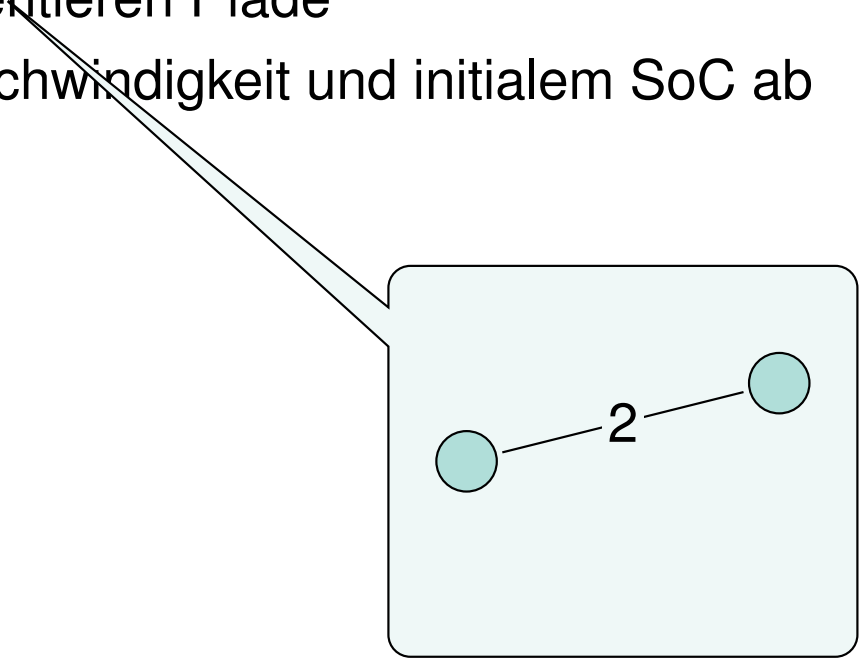


Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

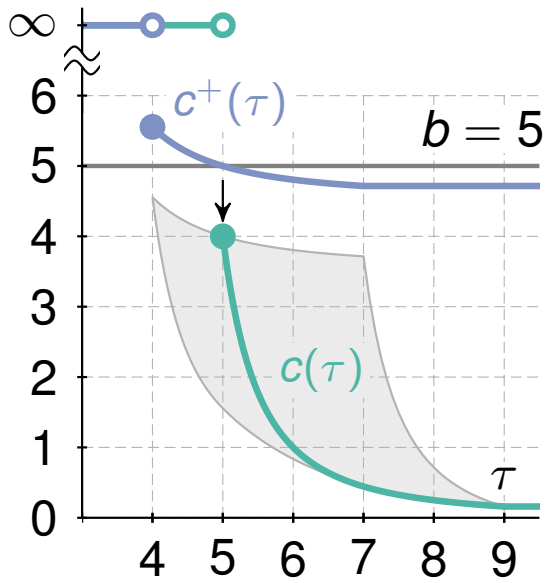
- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben

Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:
Verbrauch niemals negativ

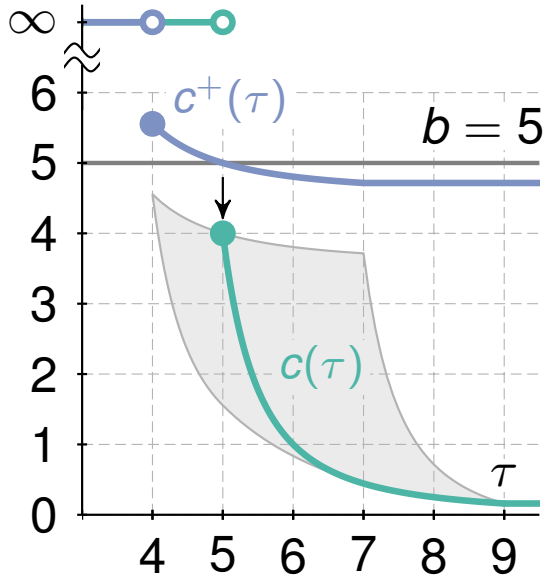
- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen

Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

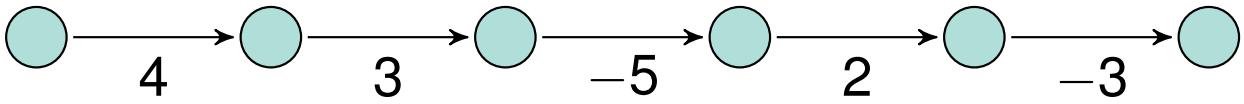
- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:

Verbrauch niemals negativ

- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen

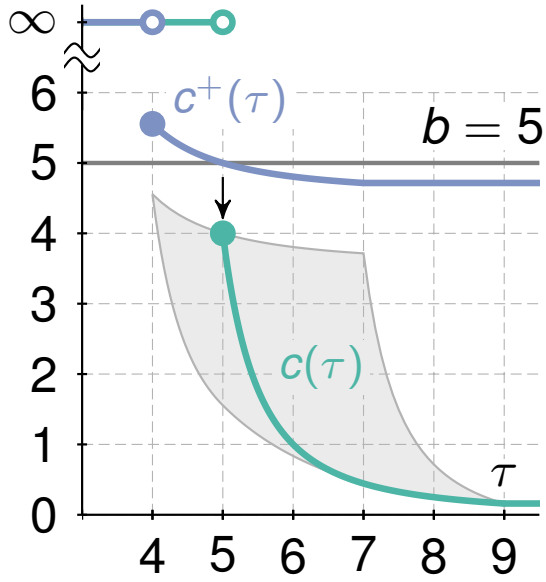


Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

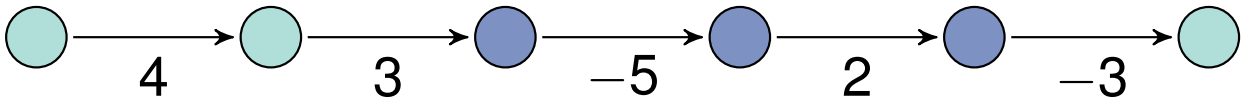
Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:
Verbrauch niemals negativ

- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen

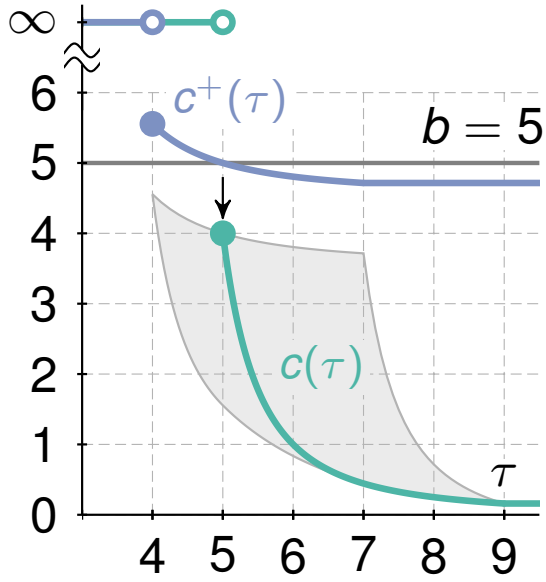


Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

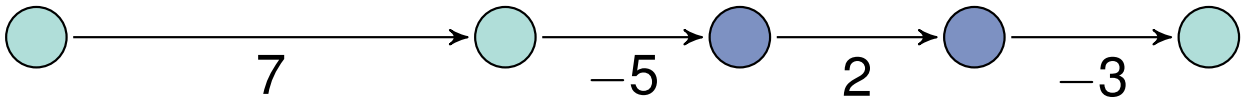
Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:
Verbrauch niemals negativ

- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen

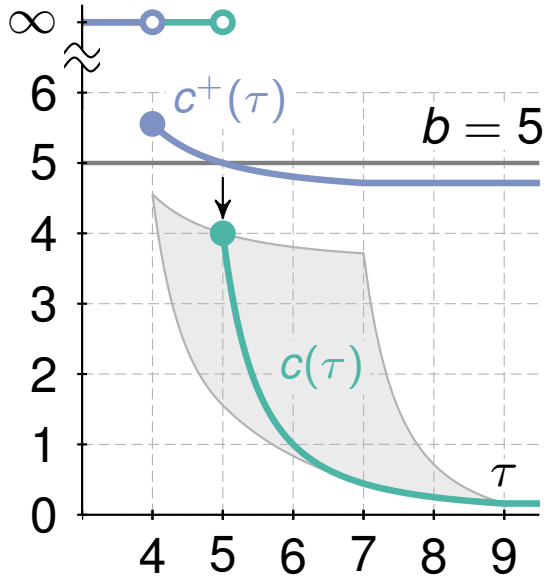


Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

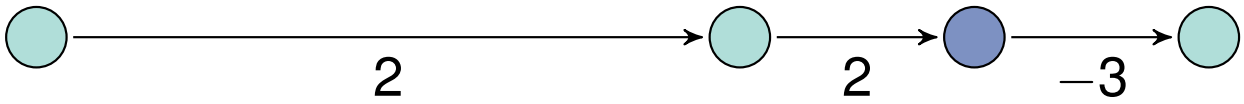
Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:
Verbrauch niemals negativ

- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen

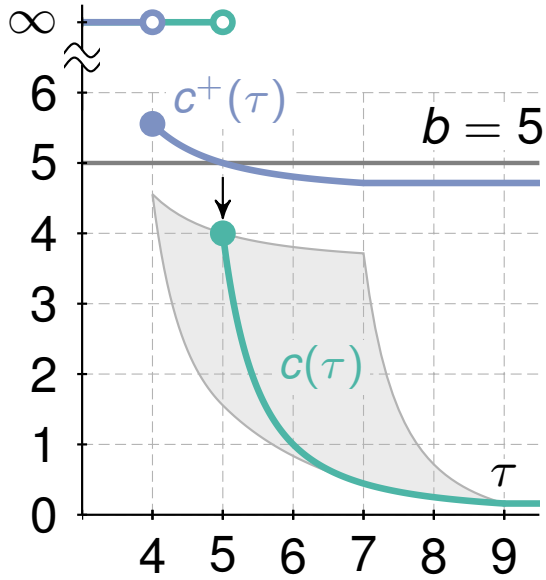


Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

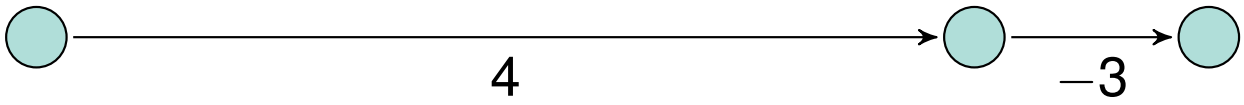
Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:
Verbrauch niemals negativ

- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen

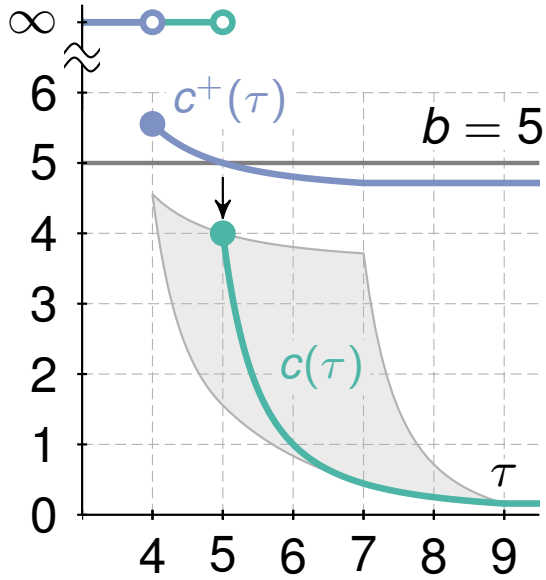


Beschleunigung

Idee: Kontraktion; Shortcuts repräsentieren Pfade

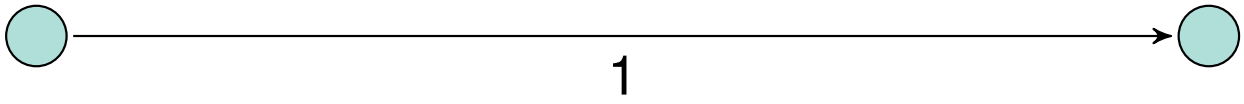
Problem: Verbrauch hängt von Geschwindigkeit und initialem SoC ab

- Bivariate Funktion $f(x, b)$
- Schwer zu handhaben



Betrachte „entladende Pfade“:
Verbrauch niemals negativ

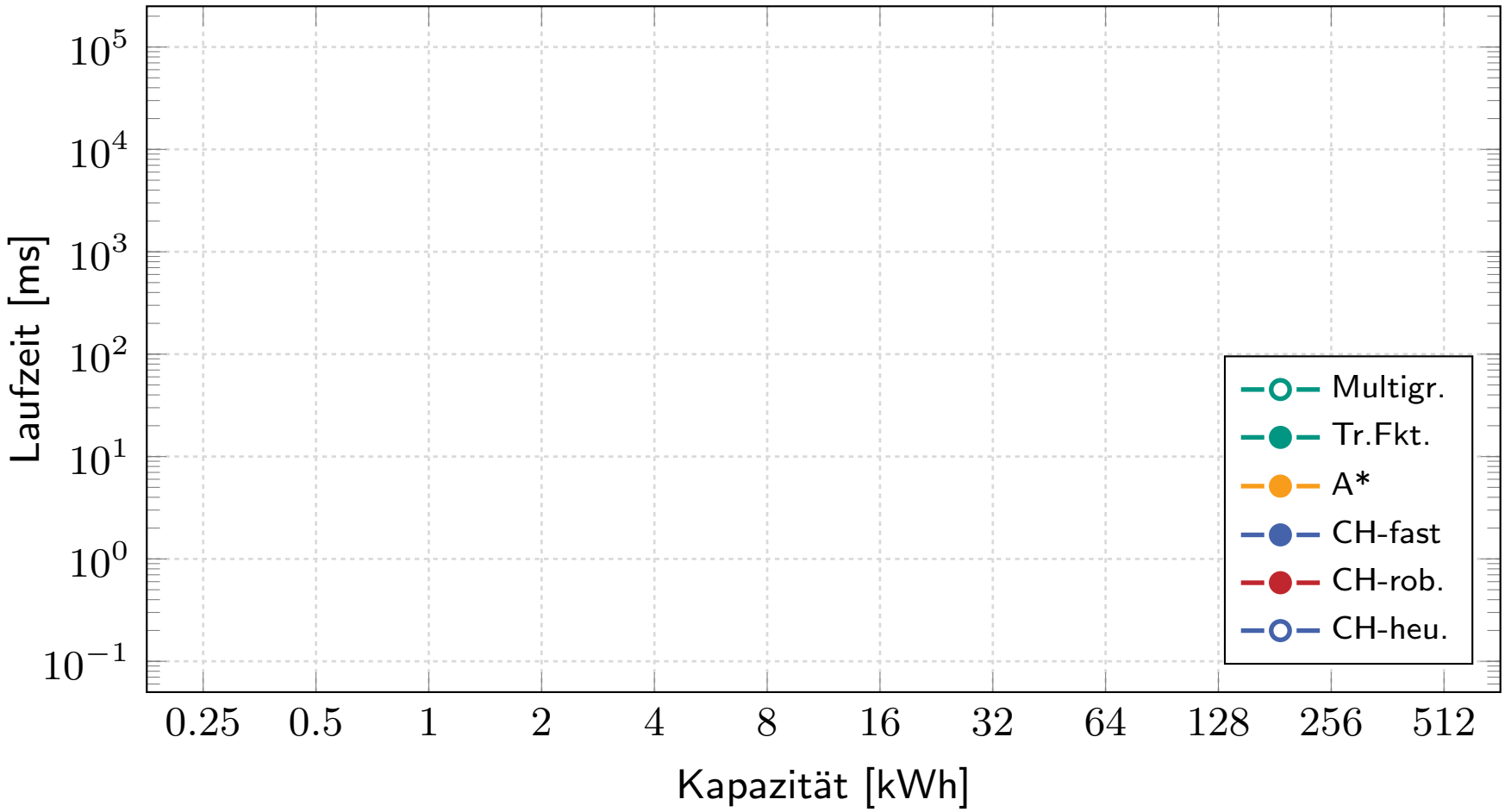
- Nur eine Bedingung muss geprüft werden
- Effizient darstellbar
- Effiziente Operationen



Experimente

Straßennetz Europa, max. Laufzeit: 60 min.

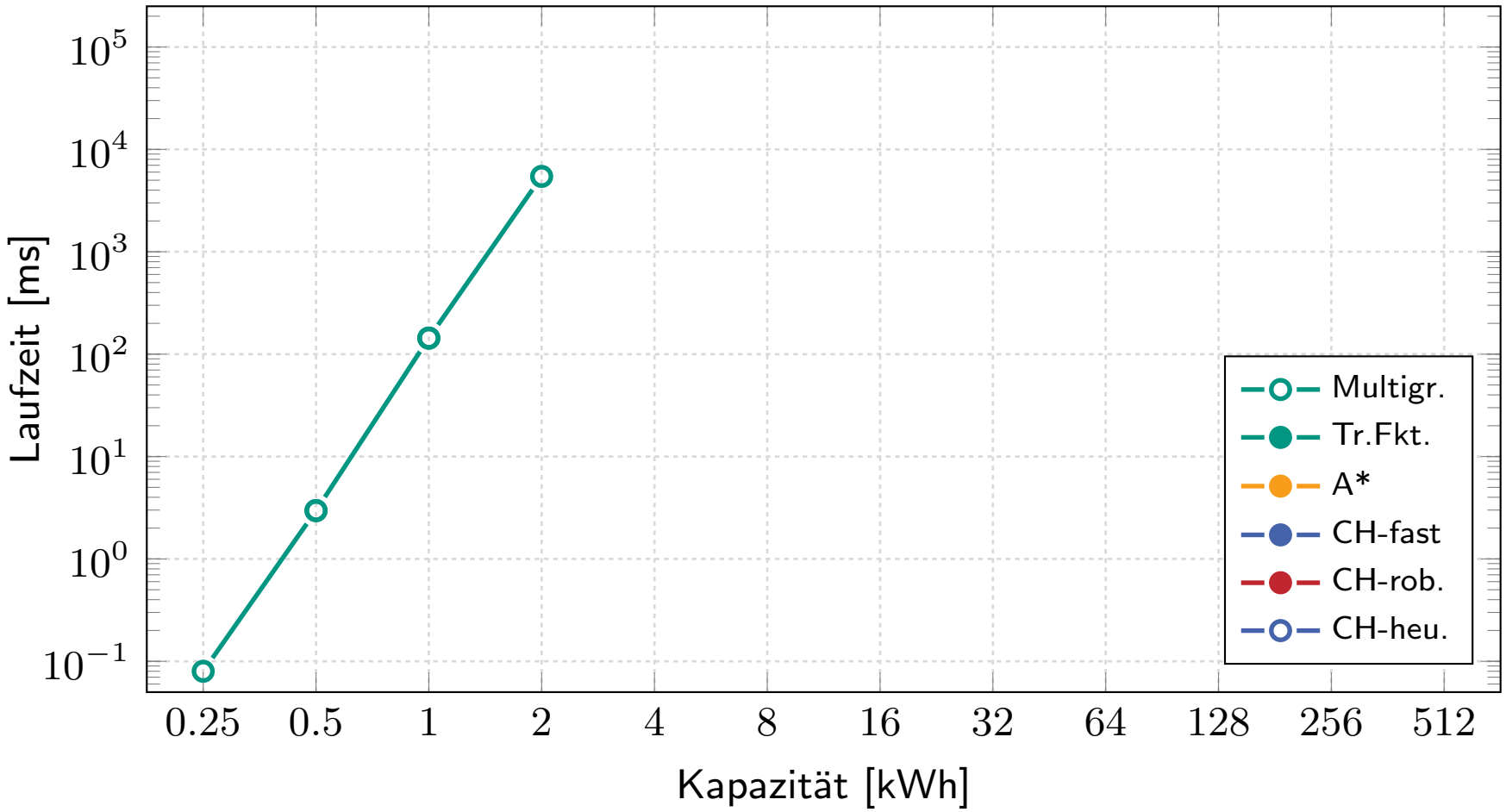
Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM



Experimente

Straßennetz Europa, max. Laufzeit: 60 min.

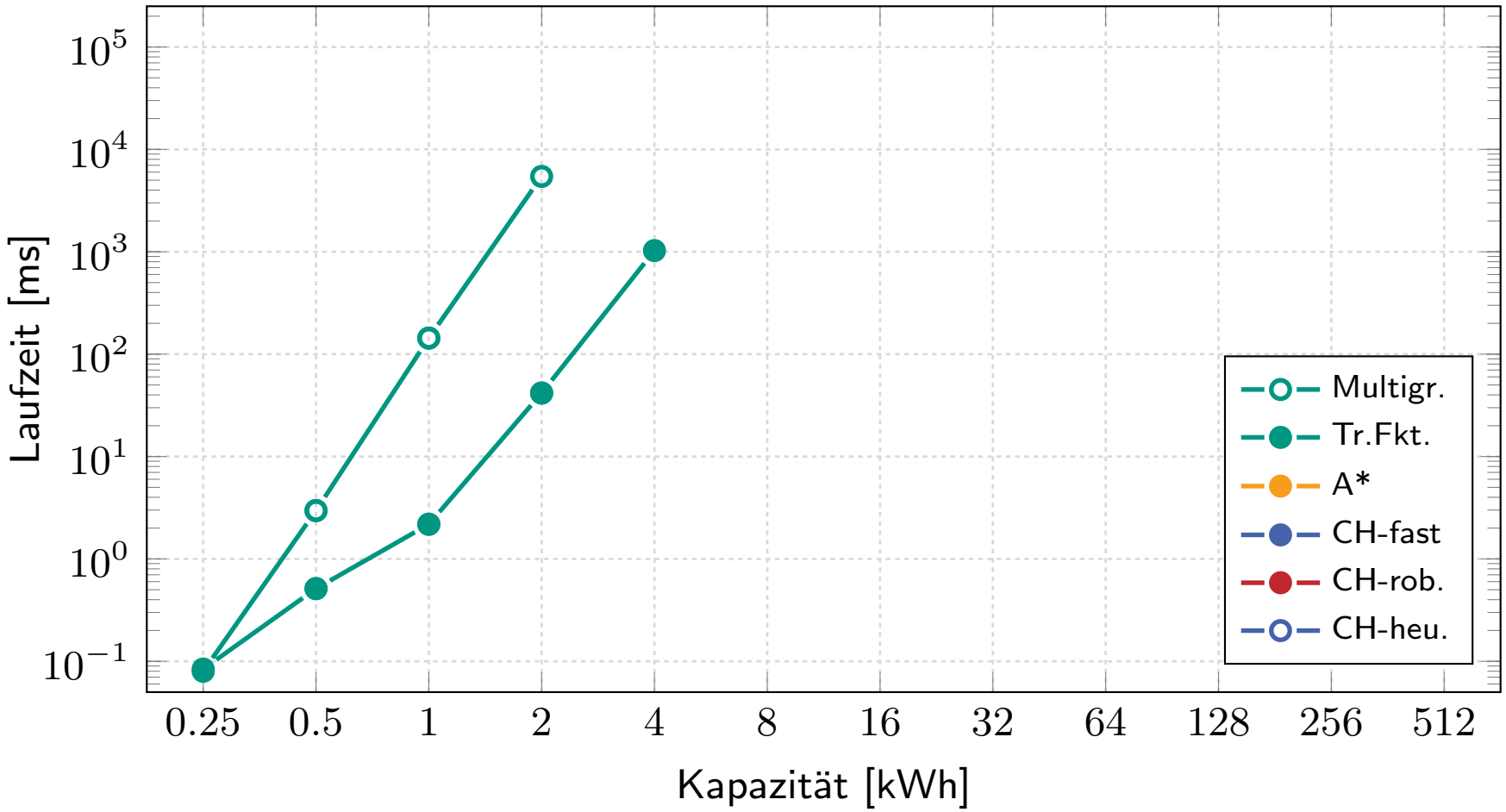
Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM



Experimente

Straßennetz Europa, max. Laufzeit: 60 min.

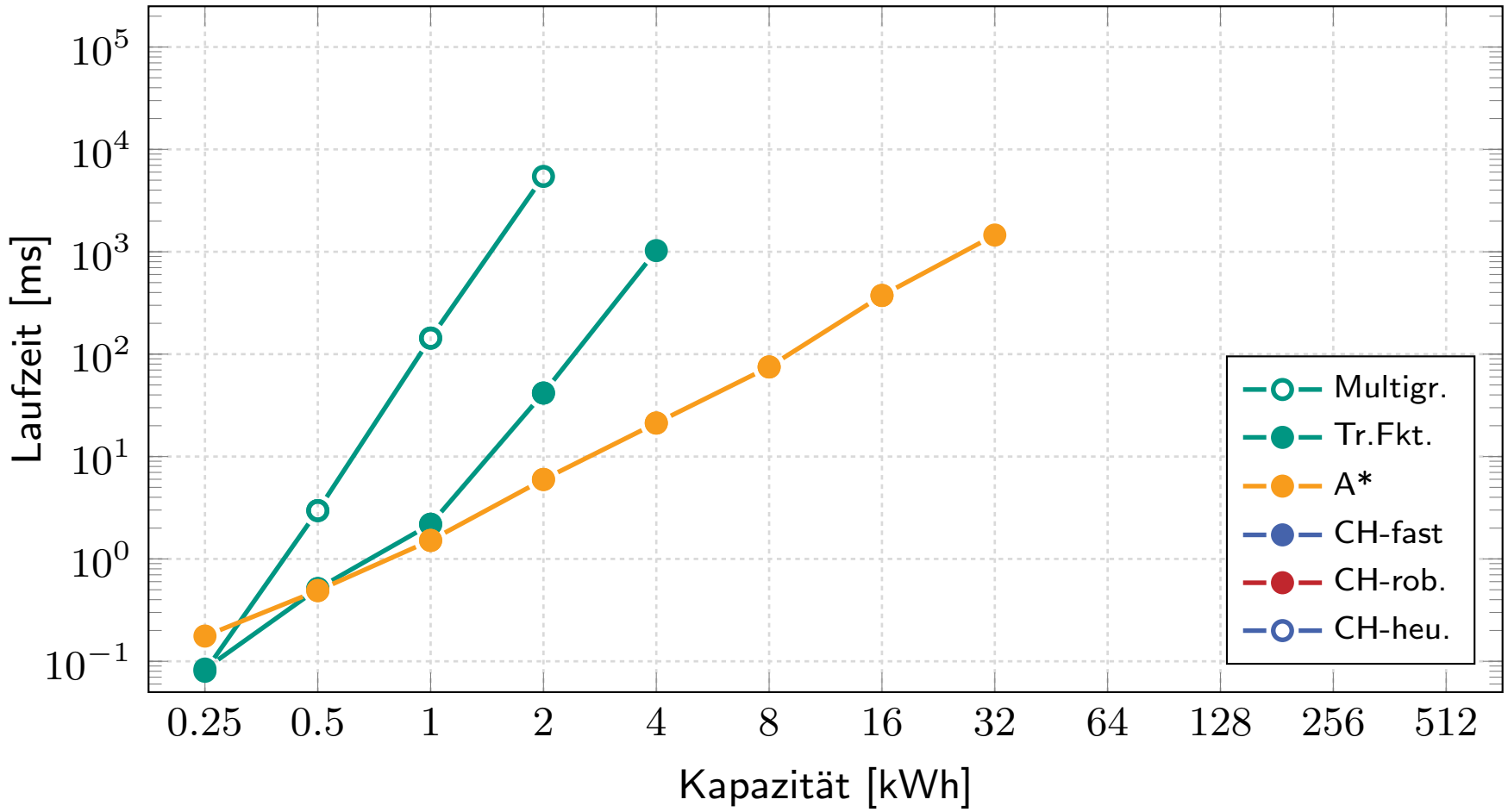
Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM



Experimente

Straßennetz Europa, max. Laufzeit: 60 min.

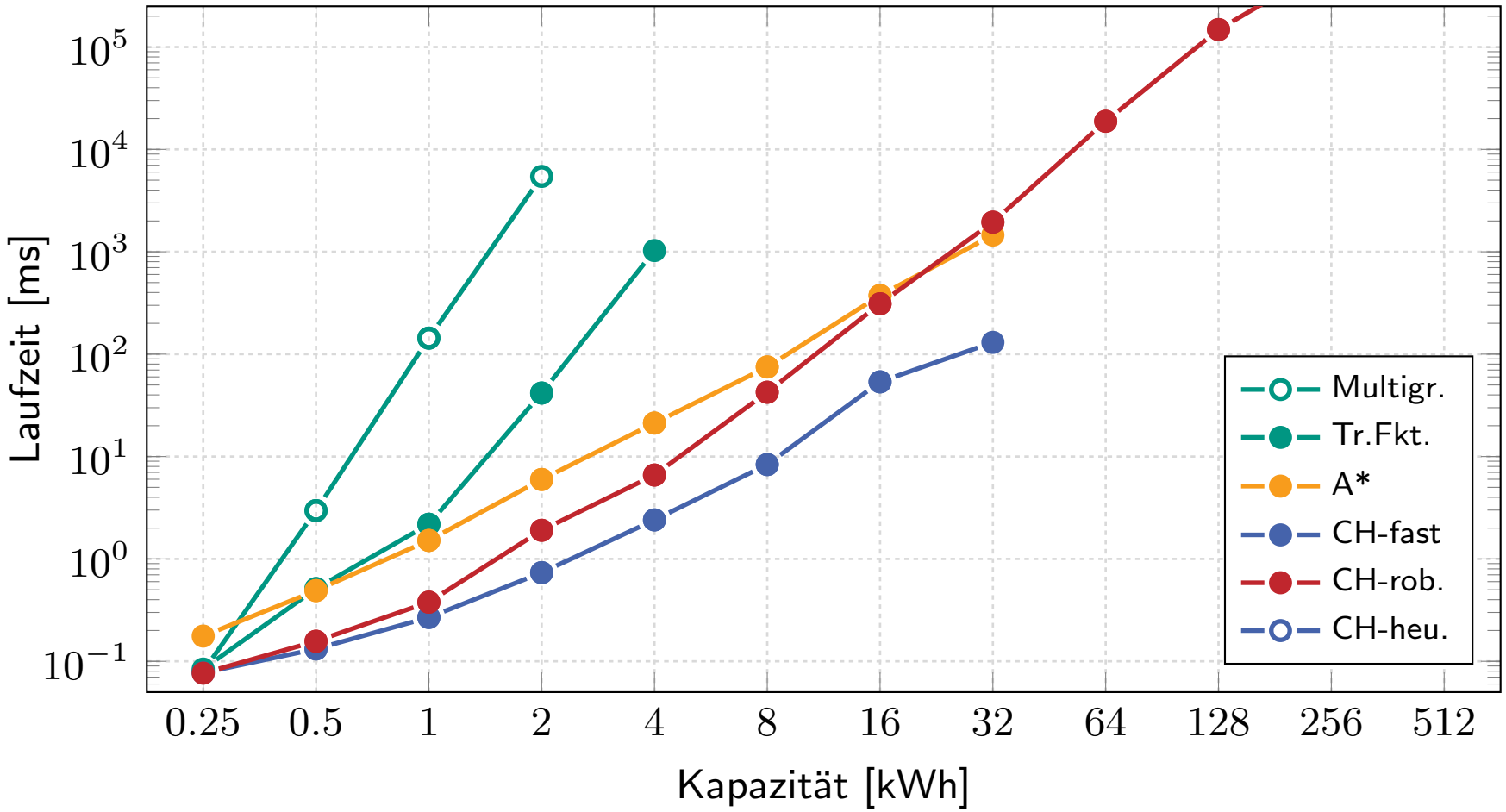
Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM



Experimente

Straßennetz Europa, max. Laufzeit: 60 min.

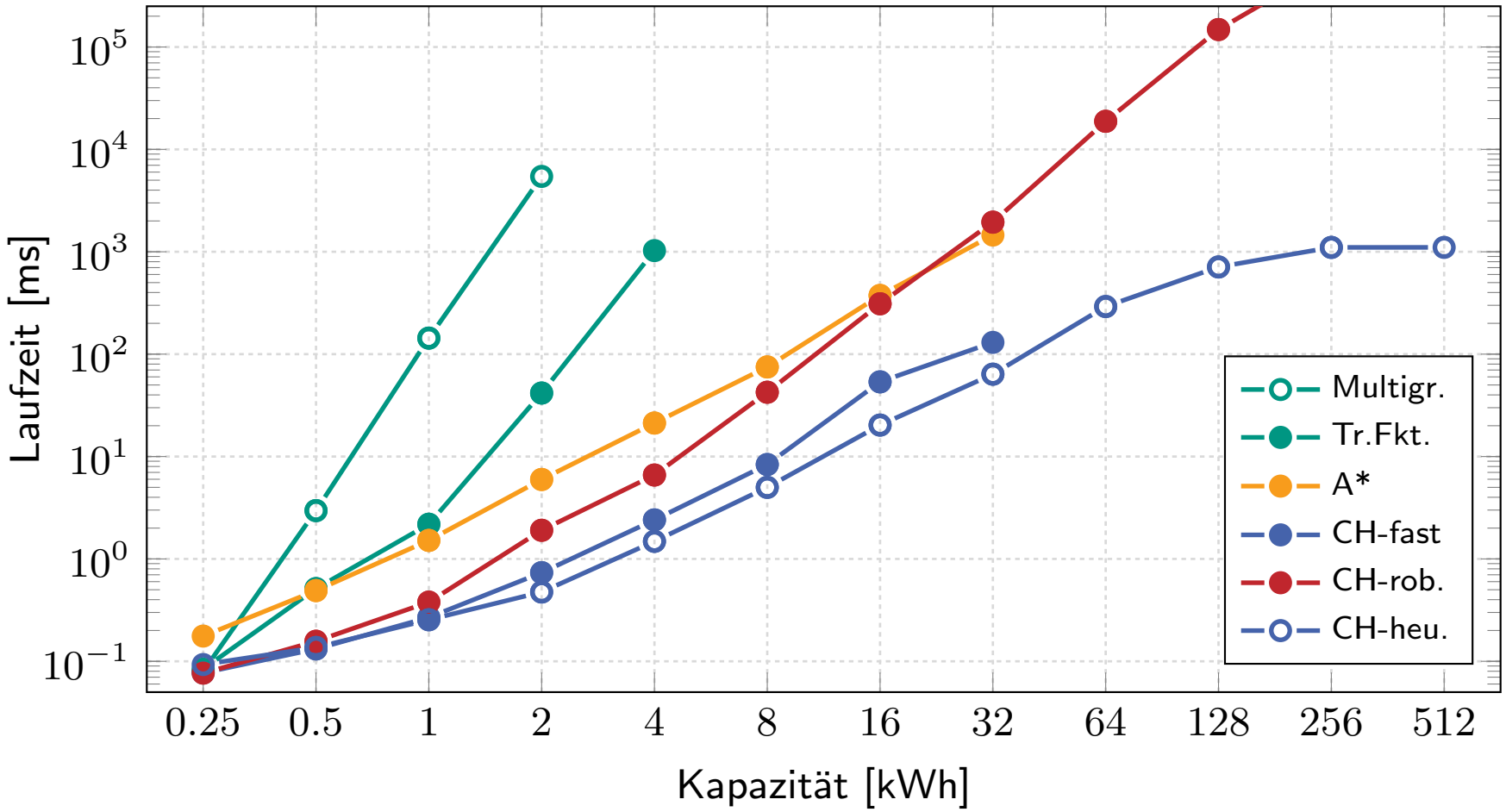
Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM



Experimente

Straßennetz Europa, max. Laufzeit: 60 min.

Hardware: Intel Xeon E5-1630v3, 3.7 GHz, 128 GiB RAM



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