Heuristische/Informierte Suche

- Ziel: Verringerung der Suchdauer
- Idee: Einbauen von zusätzlichem Wissen
- 2 Gruppen von Algorithmen bewerten
  - Lösungsschritte
  - Lösungen

Übersicht

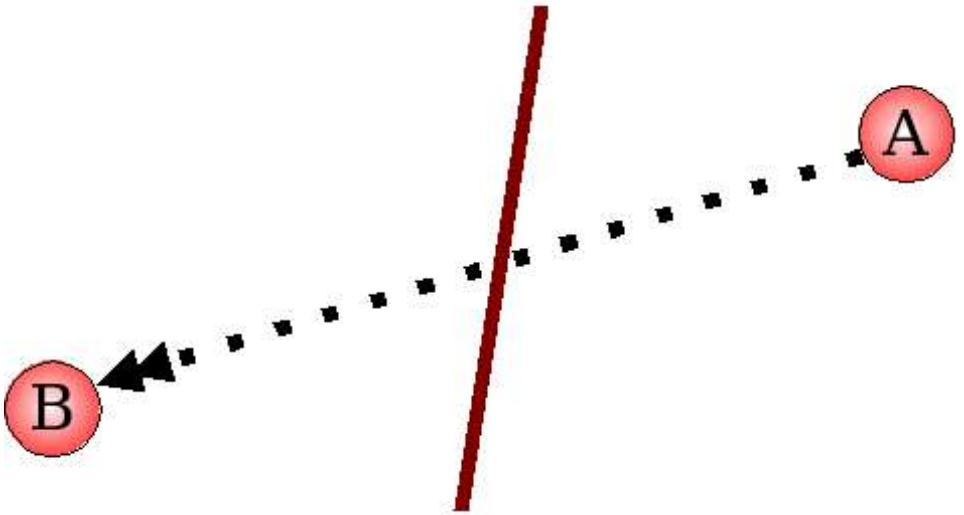
- Greedy-Search
  - (direktes Anlaufen der Lösung)
- Relaxed Problem
  - (Finden einer Heuristik)
- A* Search
  - (Greedy Search + uniform cost search)
- IDA* und SMA* Search
  - (A* mit geringem Speicherbedarf)

Übersicht

- Iterative Algorithmen
  - (Verbesserung von fertigen Lösungen)
- Hill-climbing
  - (bessere Lösungen verwenden, schlechtere verwerfen)
- Simulated Annealing
  - (u.U. Schlechtere verwenden/bessere verwerfen)
- Ausblick Genetic Algorithms
  - (Gedächtnis, Breitensuche, unabhängige Datendarstellung
    ⇒ Anwendung in sehr komplexen Problemen)

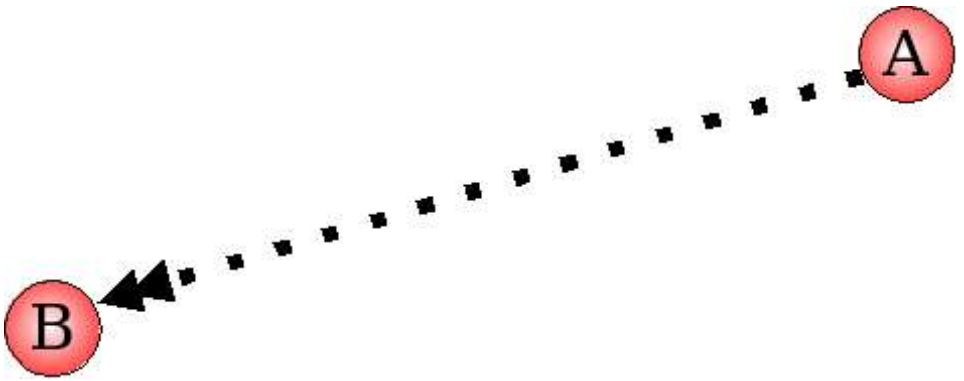
Einstieg (1)

“Schwieriges” Pfadsuche-Problem mit Wand



Einstieg (2)

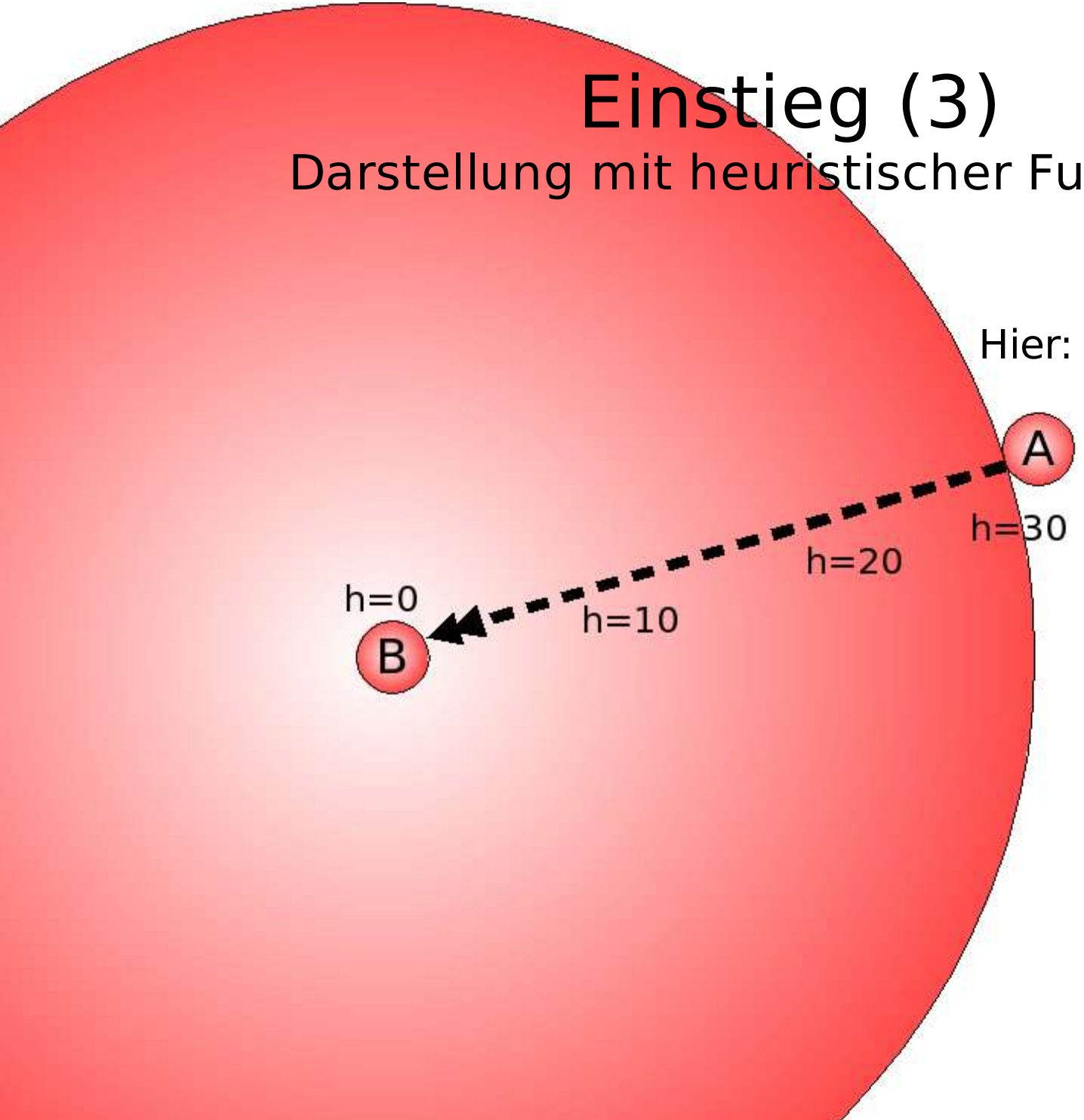
Vereinfachtes Pfadsuche-Problem ohne Wand



Einstieg (3)

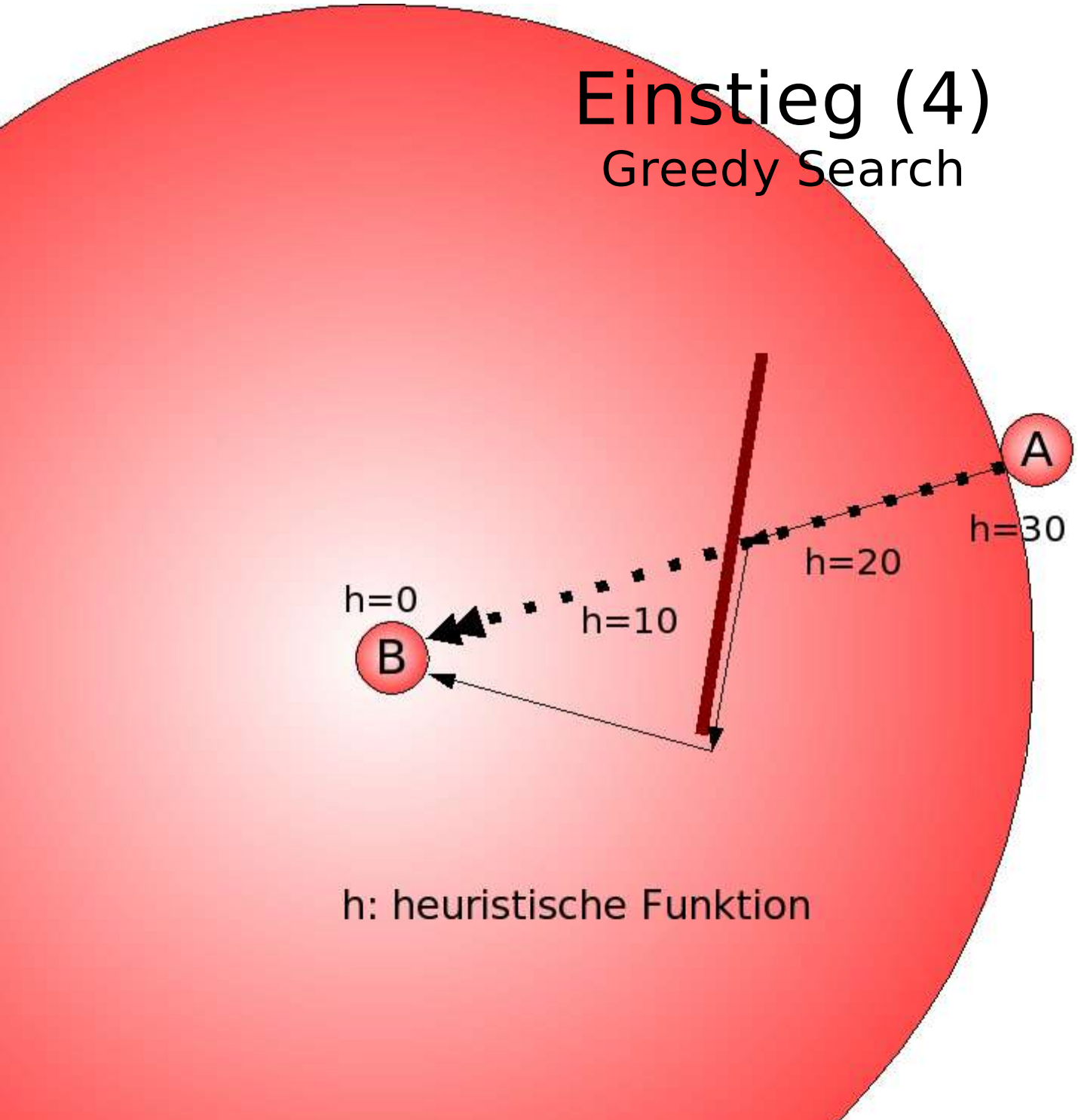
Darstellung mit heuristischer Funktion h

Hier: h ≙ Abstand zu B



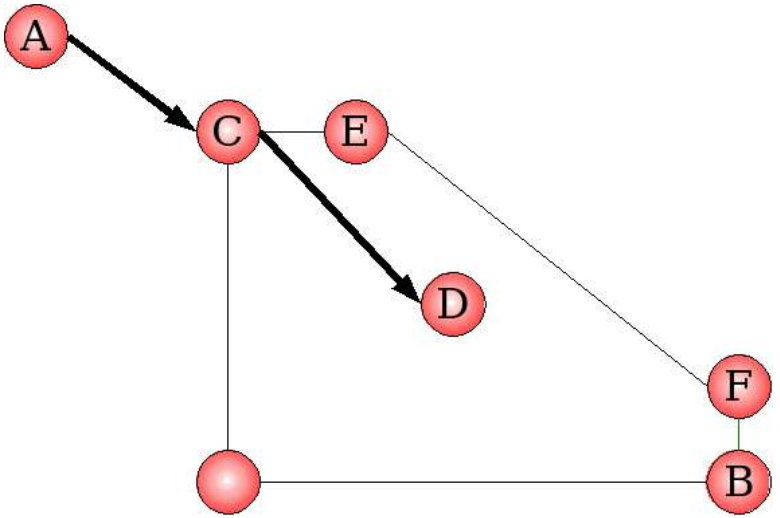
Einstieg (4)

Greedy Search

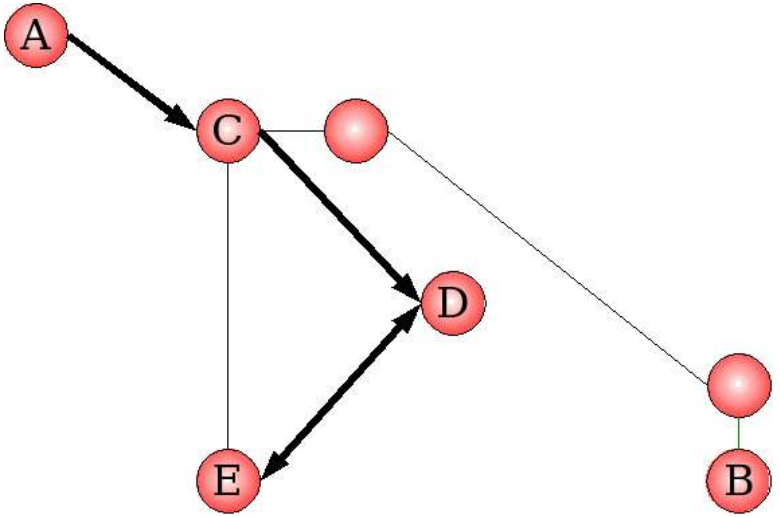


h: heuristische Funktion

Greedy search (1)
(Sackgasse und Schleife)



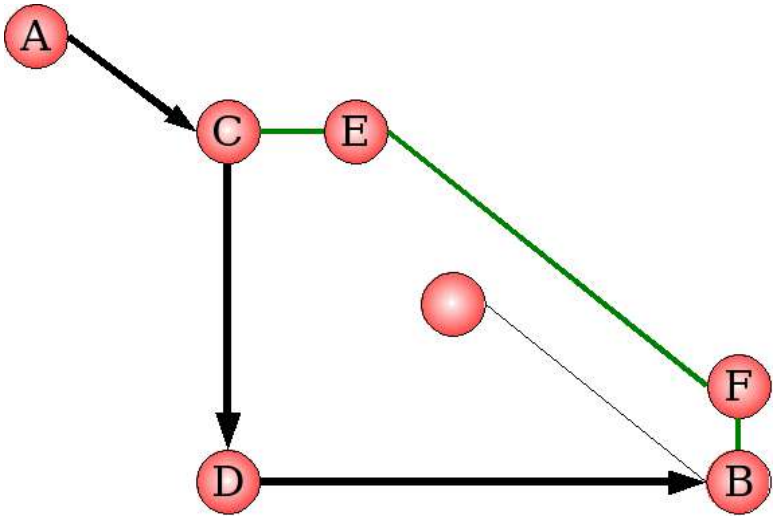
Sackgasse

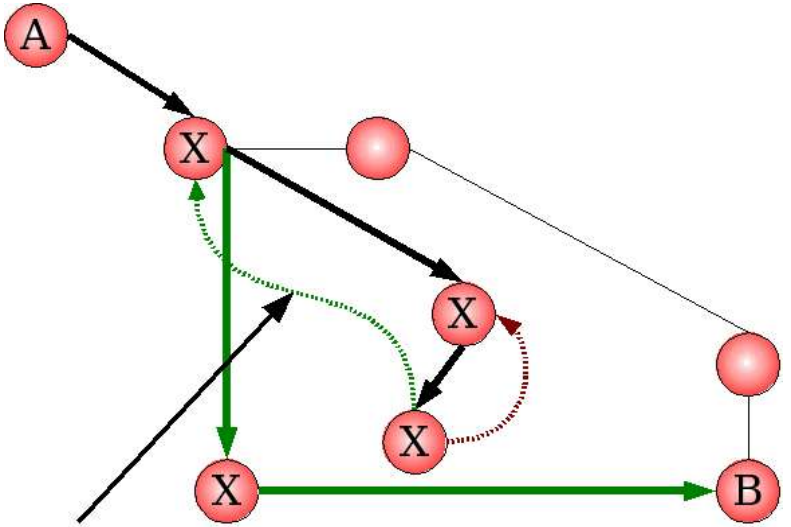


Schleife

⇒ **GREEDY SEARCH nicht vollständig**

Greedy search (2)
(ungünstiger Start und Gedächtnis)





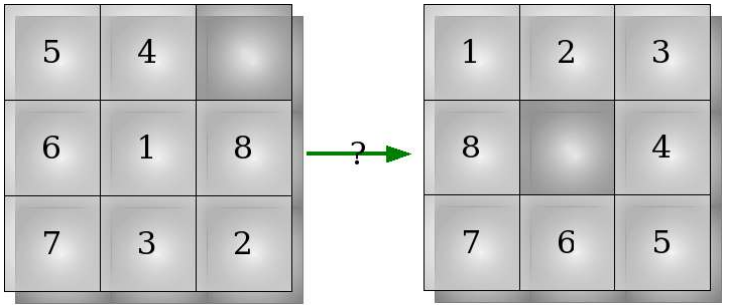
⇒ **GREEDY SEARCH nicht optimal**

⇒ **vollständig aber nicht optimal**

Heuristikenentwicklung

8 Puzzle Problem

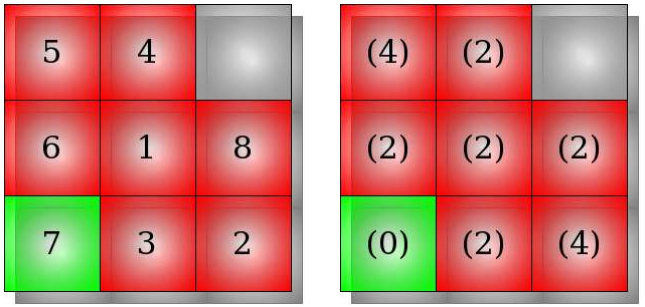
Startzustand



Zielzustand

Ein Teil darf
a) pro Schritt nur 1 Feld
b) nicht schräg
c) nur in ein freies Feld verschoben werden

Heuristik 1:
korrekte Positionen
Ignorierung der
Regeln a),b) und c)



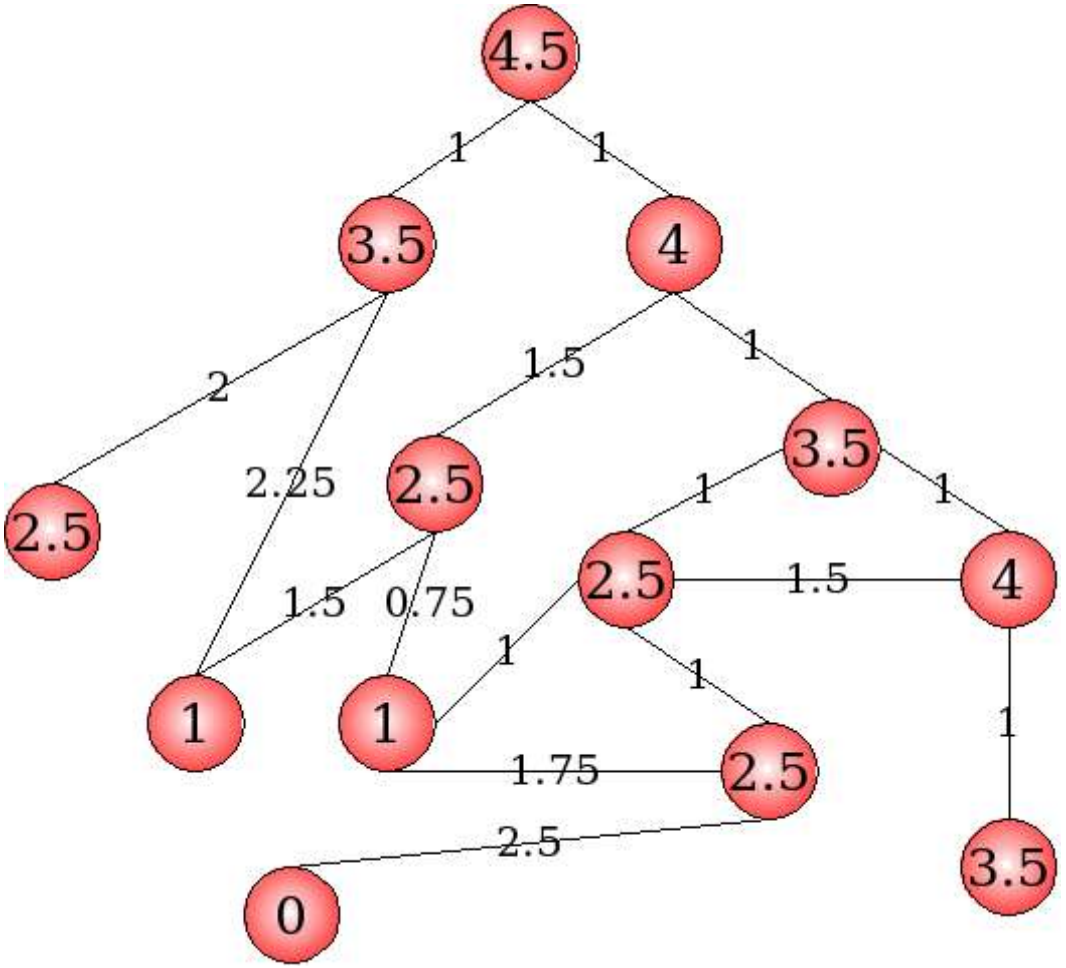
Heuristik 2:
Abstand zum Ziel
Ignorierung der
Regel c)

A* Search (1)
Übersicht

- Kombination aus Greedy und uniform-cost search
- Vollständig und optimal
- Meist Luftlinie (Greedy Search) + bisherige Kantenlänge (uniform cost search)
- Begrenzender Faktor ist Speicher
- Im schlechtesten Fall Speicher- und Suchzeit $O(b^m)$,
- Im Durchschnitt schneller als allgemeine Suche

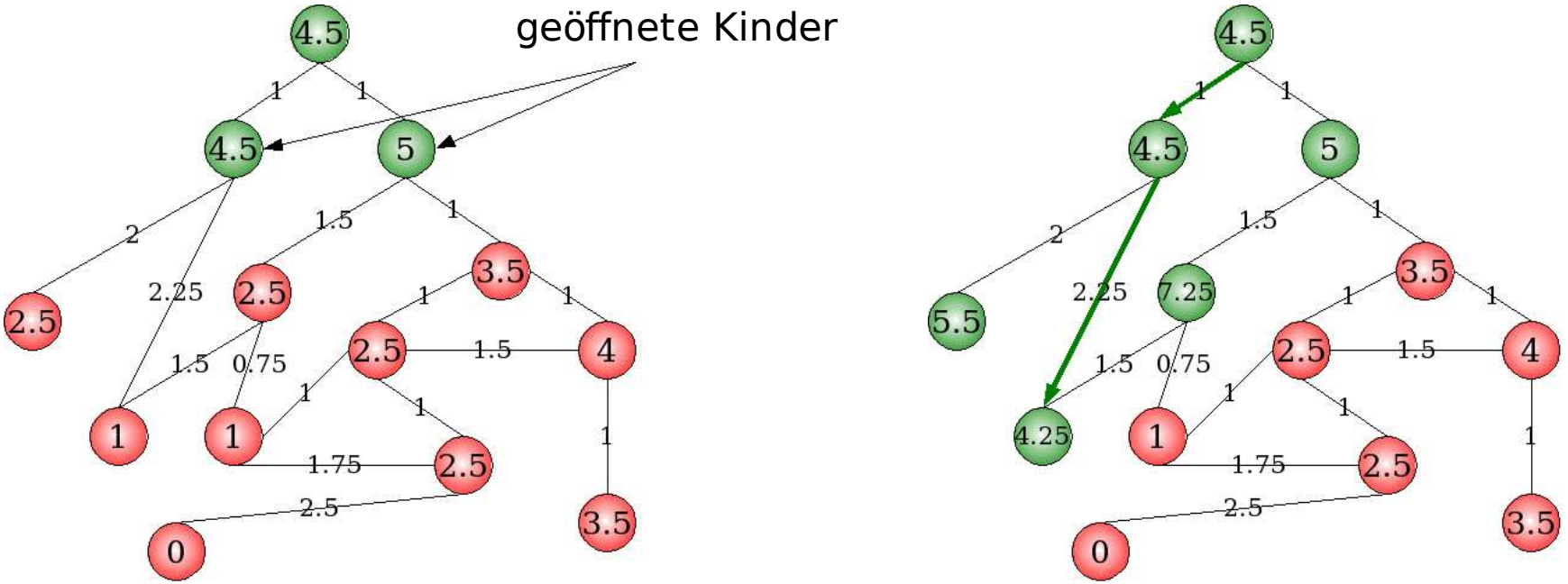
A* Search (2)

Beispiel Pfadsuche

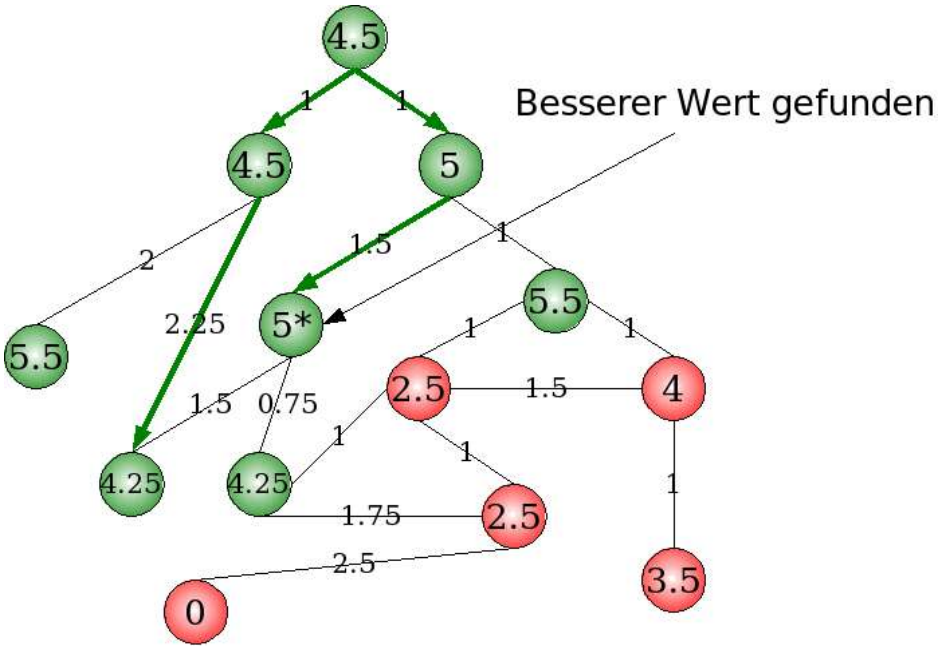


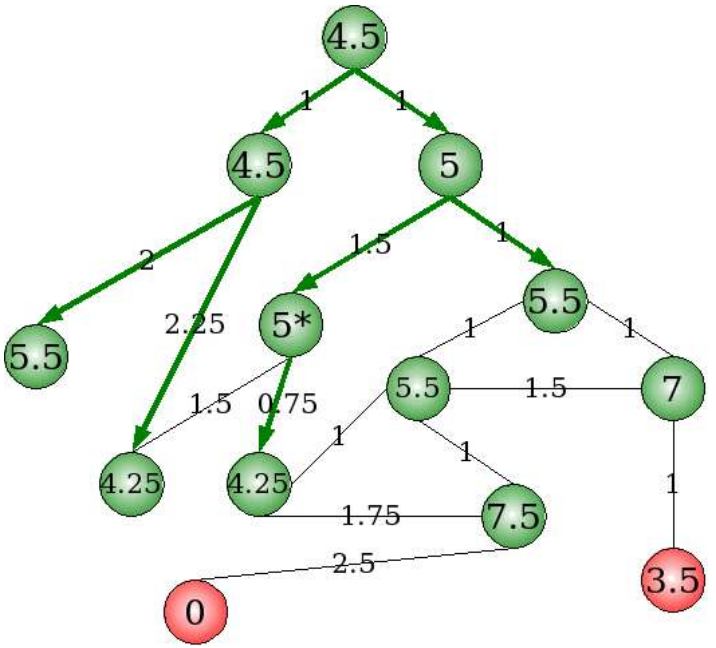
Zahlen an Kanten: Kantenlänge
Zahlen in Kreisen: Luftlinie zum Ziel

A* Search (3)

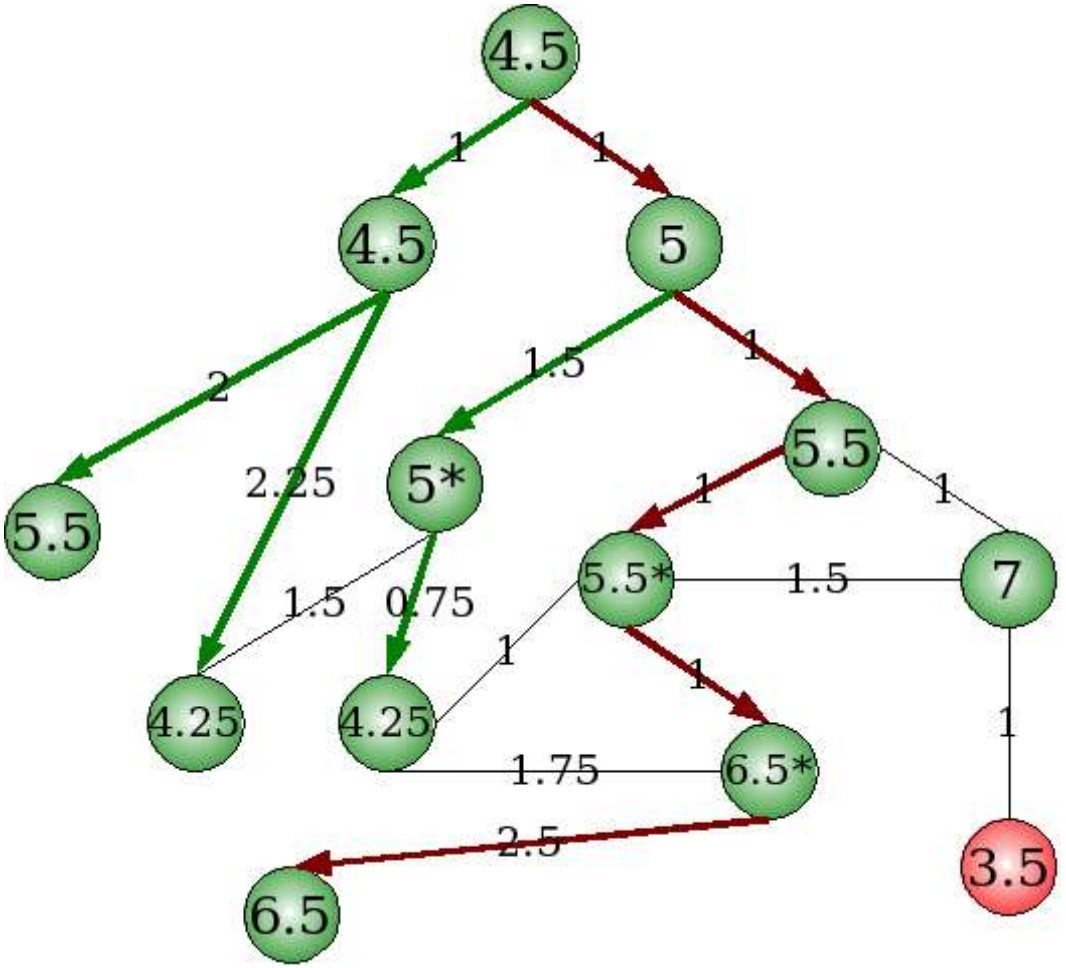

geöffnete Kinder
4.5
1
1
4.5
5
2
1.5
1
2.5
2.25
2.5
3.5
1
1
1.5
0.75
2.5
1.5
4
1
1
1
1
1.75
2.5
1
2.5
3.5
0
4.5
1
1
4.5
5
2
1.5
1
5.5
2.25
7.25
3.5
1
1
1.5
0.75
2.5
1.5
4
4.25
1
1
1
1.75
2.5
1
2.5
3.5
0

A* Search (4)


4.5
1
1
4.5
5
Besserer Wert gefunden
2
1.5
1
5.5
2.25
5*
1
1
5.5
1.5
0.75
2.5
1.5
4
1
1
4.25
4.25
1
1
1.75
2.5
2.5
3.5
0


4.5
1
1
4.5
5
2
1.5
1
5.5
2.25
5*
1
1
5.5
1.5
0.75
5.5
1.5
7
1
1
4.25
4.25
1
1
1.75
7.5
2.5
3.5
0

A* Search (5)



⇒ Optimaler Weg gefunden

Iterative Deepening A* Search (1)

- IDA* nutzt als Basis iterativ deepening search
- Knotentiefe wird durch Suchtiefe in heuristischer Funktion verändert
- Speicherverbrauch zwischen einem Schritt: Suchtiefe
- Neukalkulation aller bisherigen Knoten

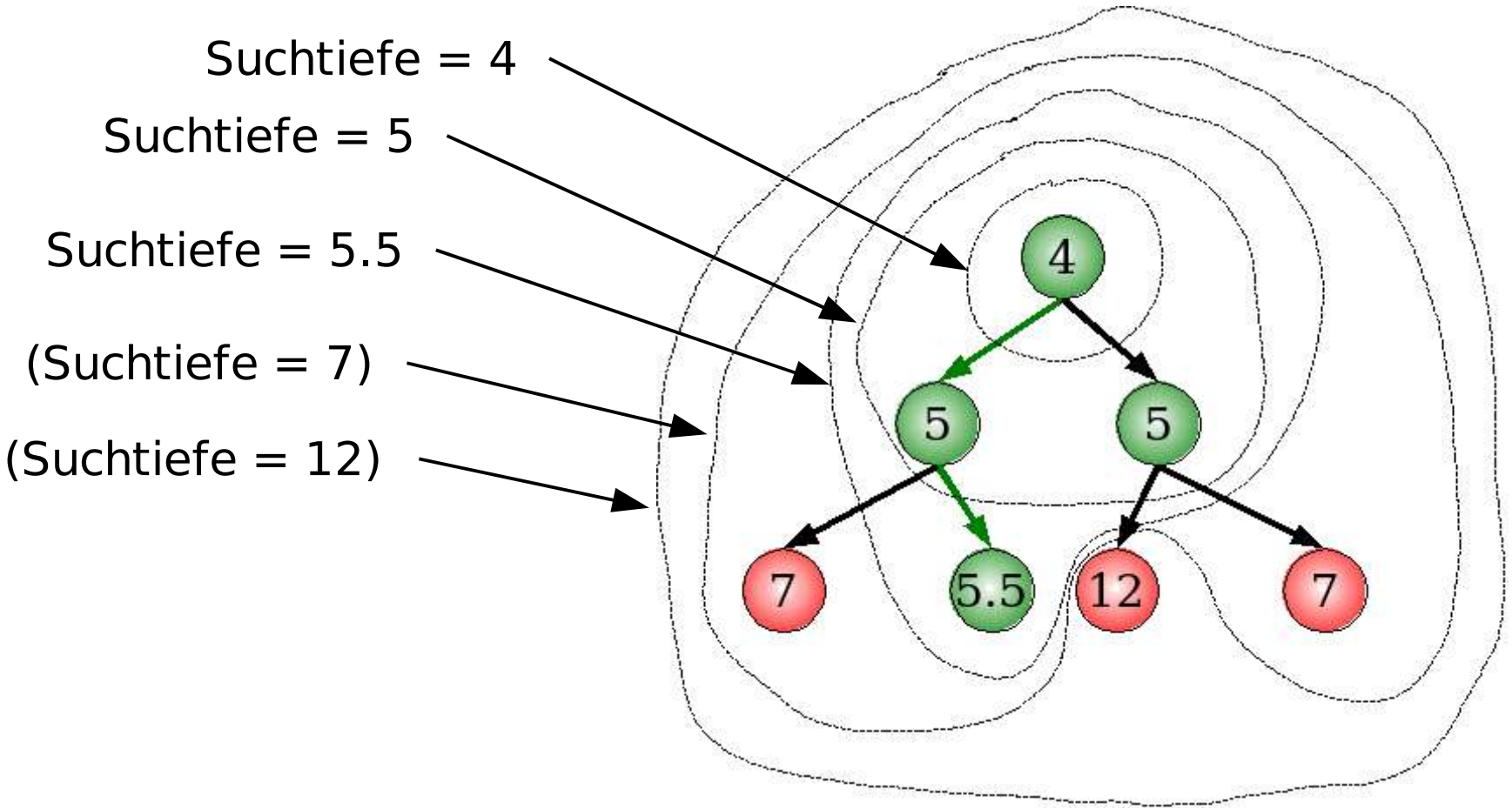
⇒ im schlimmsten Fall $O(b^m)$ Neuberechnungen

- Problem bei reellwertiger Heuristikfunktion

⇒ IDA* ist weder optimal noch vollständig

Iterative Deepening A* Search (2)

Beispiel



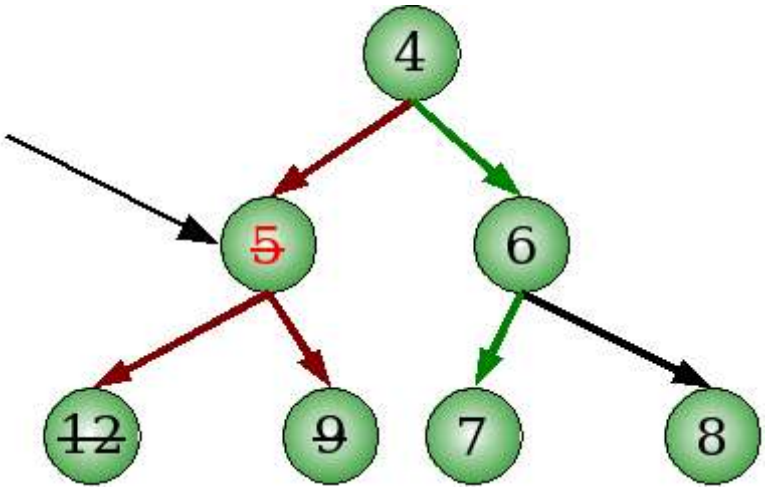
Simplified Memory-Bounded A* Search (1)

- Vorgang wie bei A*
- Speicherplatz knapp:
  - Teilgraph durch bestes Kind ersetzen
  - unter Umständen hoher Zeitaufwand
- Ausreichend viel Speicher: (Lösungspfad)
  - Optimal und vollständig
- Beliebig viel Speicher
  - Gleiches Verhalten wie A*
- Zeitaufwand $O(b^m)$

Simplified Memory-Bounded A* Search (1)

Beispiel mit Speicherplatz 4

Ersetze Teilgraphen
durch bestes Kind: 9



Iterative Algorithmen

- Basis: **zufällige Suche**
- komplette Lösung **schrittweise angepasst**
- **Grundsätzlicher Ablauf**:
- 1. Zufällige Lösung wird erstellt
- 2. Lösung wird zufällig verändert und mit der alten verglichen
- 3. besser ⇒ Neue Lösung fuer 2 verwenden
- 4. schlechter ⇒ alte Lösung fuer 2 verwenden
- Optimum oft unbekannt:

⇒ Suche endet sobald eine gültige Lösung “ausreichend” gut ist

Hillclimbing Algorithmus (1)

Fitness ≙ Bewertungsfunktion

Fitnesslandschaft ≙ Graph der Fitnessfunktion

Ziel: beste Fitness, z.B. Berggipfel A erreichen

Hier: Von E und F wird direkt auf Gipfel A gelaufen

Hillclimbing Algorithmus (2)

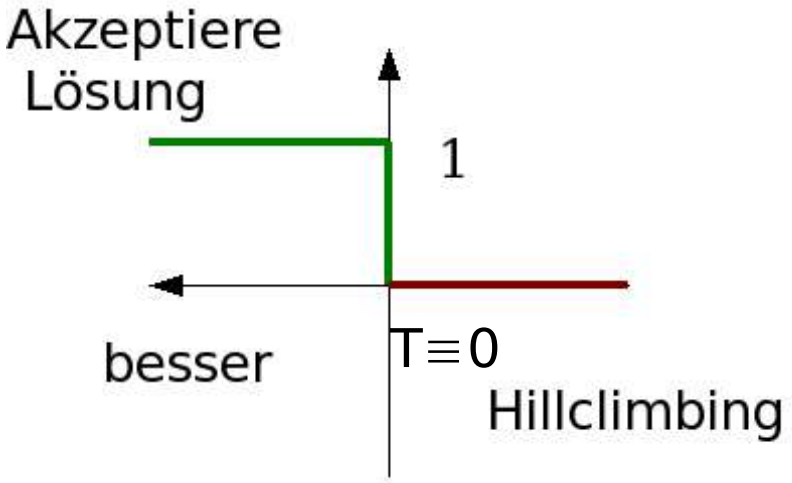
Beispiel

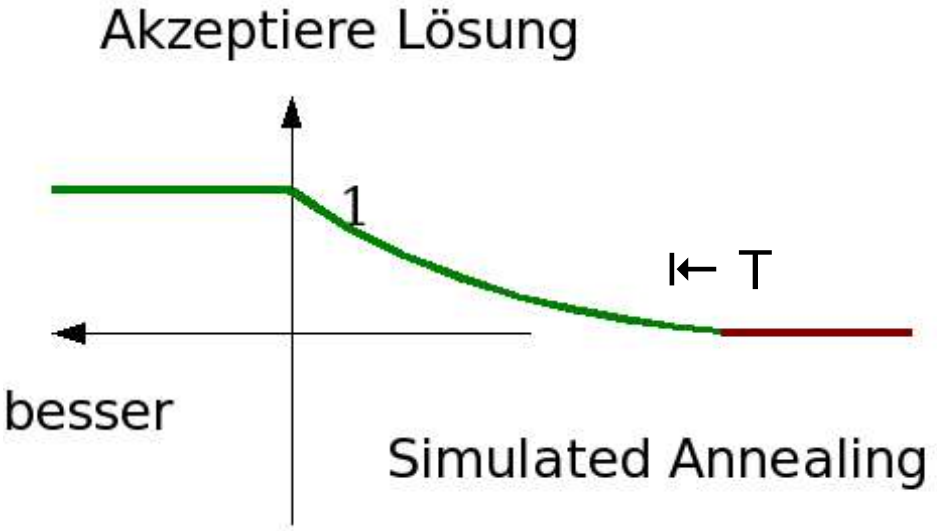
- B: ebene Fitnesslandschaft ⇒ zufällige Suche bis E gefunden
- C läuft nicht zu F sondern auf lokales Optimum nach rechts
- D läuft zwar Richtung A, bleibt jedoch ebenfalls hängen
- Ist A wirklich das Optimum? Andere Berggipfel rechts und links des Bilds?

Simulated Annealing Algorithmus (1)

Unterschiede zu Hillclimbing

- Akzeptiere mit Wahrscheinlichkeit T auch schlechtere Lösungen um lokale Optima zu überwinden
- Verringere die T stufenweise oder rellwertig langsam über die Zeit
- Option: Verwerfe u.U. Minimal bessere Lösungen





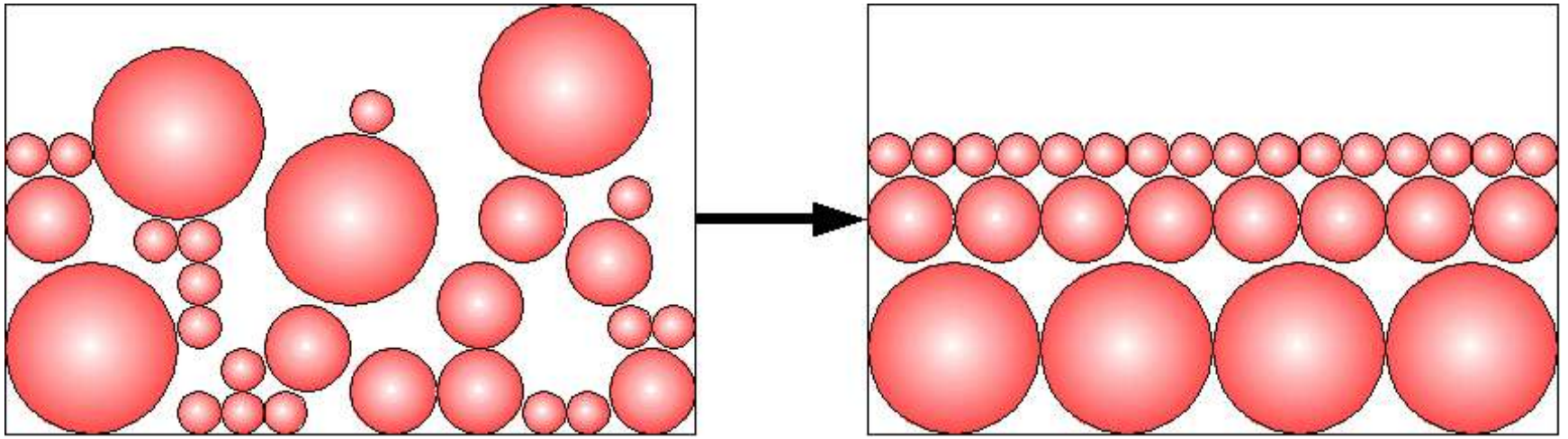
Simulated Annealing (2)

Beispiel

Ziel: Sand oben, Kies in der Mitte und Steine unten

Idee: Zustand zufällig verändern ⇒ Schütteln

⇒ Steine wandern nach unten und Sand nach oben



Simulated Annealing (3)

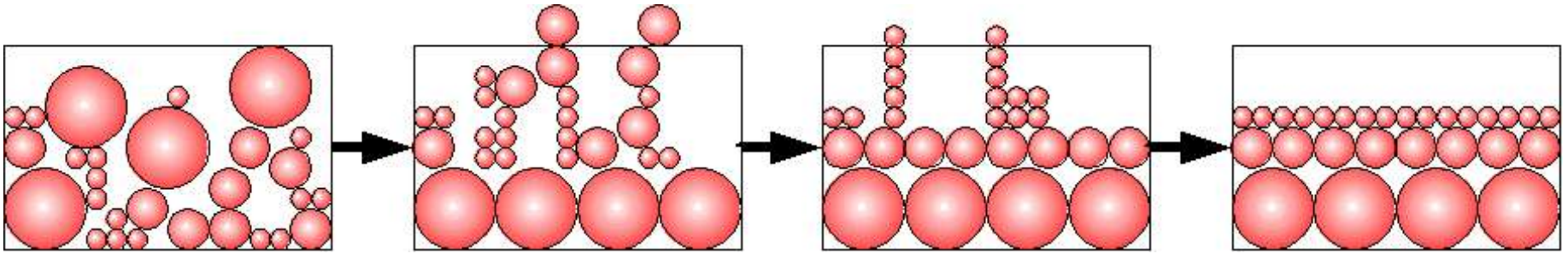
Beispiel

3 Temperaturstufen:

- T = 3: Steine bewegen sich

  Rest wird durchgeschüttelt, mitunter in schlechtere Positionen
- T = 2: Kies bewegt sich

  Sand wird durchgeschüttelt, mitunter in schlechtere Positionen
- T = 1: Sand bewegt sich bis optimaler Zustand bei
- T = 0 erreicht ist



Ausblick: Genetische Algorithmen

- Breitensuche mit mehreren “Agenten”
- Gedächtnis mit Hilfe von Allelen und Intronen
- Abkopplung von mutierendem Genmaterial und tatsächlicher Ausprägung
- Rekombination durch Genaustausch

⇒ Abflachung steiler Bereiche in der Fitnesslandschaft

- Aber kein “heiliger Gral”: Hauptarbeit liegt in Formulierung von Problem und Bewertungsfunktion

⇒ Abschätzungen, Kreativität, gute Kenntnisse in Abstraktion von Problemen und Grenzen der genetischen Algorithmen sind wichtig

Zusammenfassung (1)

- **Allgemeine Suchalgorithmen:**
  - blinde Suche
  - direkt fuer unterschiedliche Probleme verwendbar
- **Breitensuche**
- Gesamte Kantenlänge ⇒ **Uniform Cost search**
- Pfade bis Sackgasse verfolgen ⇒ (Begrenzte) **Tiefensuche**
- Schrittweise Erhöhung der Suchtiefe ⇒ **Iterative Tiefensuche**
- Paralelle Suche von Start und Ziel ⇒ **Bidirektionale Suche**
- CSP (**Backtracking**, **Forward checking**)

Zusammenfassung (2)

- **Heuristik**
  - Reduktion der Suchzeit durch zusätzliche Information
  - Finden über vereinfachtes Problem
- uniform cost search + greedy search ⇒ **A* Search** Algorithmus
- Optimieren fertiger Lösung ⇒ **Hillclimbing** Algorithmus
- Verfeinerung um lokalen Optima zu entkommen ⇒ **Simulated Annealing**
- Ausblick auf **genetische Algorithmen**