Künstliche Intelligenz

Lokale Suchverfahren

Dr. Christian Meilicke
Research Group Data and Web Science
Universität Mannheim

ILIAS-Umfrage

- Online vs. Vor-Ort Vorlesung?
  - Bitte abstimmen
  - Kein Einfluss auf dieses Semester

Un/Informierte Suchverfahren

- Lösung des Problems = Weg von Startzustand zu Zielzustand

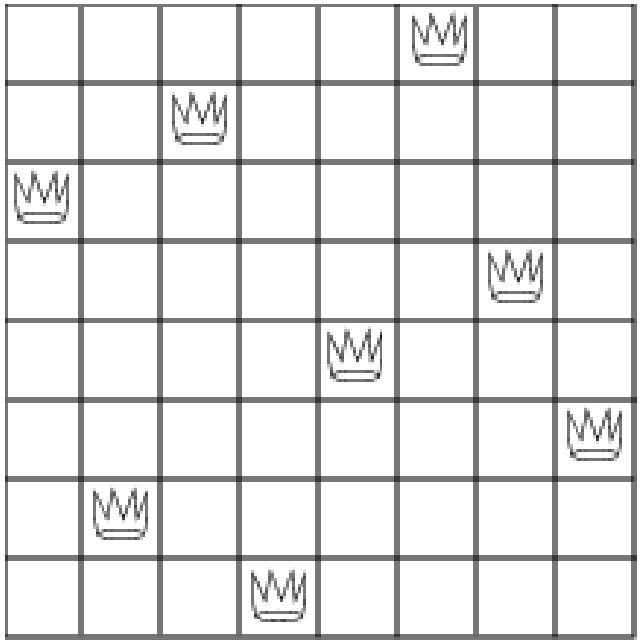
- Kosten der Lösung / Optimale Lösung
  - Uninformierte Suche
    - Implizit hat jeder Schritt konstante Kosten 1
    - Optimale Lösung = Lösung in geringster Tiefe
  - Informierte Suche
    - Explizit sind pro Schritt verschiedene Kosten
      Optimale Lösung = Lösung mit geringster Kostensumme

Was bisher geschah:

Breitensuche
Tiefensuche
Iterative Tiefensuche
Uniforme Kostensuche
Greedy Suche
A*Suche

Wen interessiert …

… in welcher Reihenfolge wir die Damen aufstellen?



(bitte nicht antworten, rhetorische Frage, die Antwort würde „Niemand“ lauten)

Der Weg ist das Ziel?

- Landkarten in 3 Farben färben
- 8 Damen Problem
- Modellkonstruktion für eine logische Formel
  - Im allgemeinen Constraint Satisfaction Probleme

- 8-Puzzle (15-Puzzle)
- Seemannssolitär
- Finde den Weg von Arad nach Bukarest
- Robinson Roulette

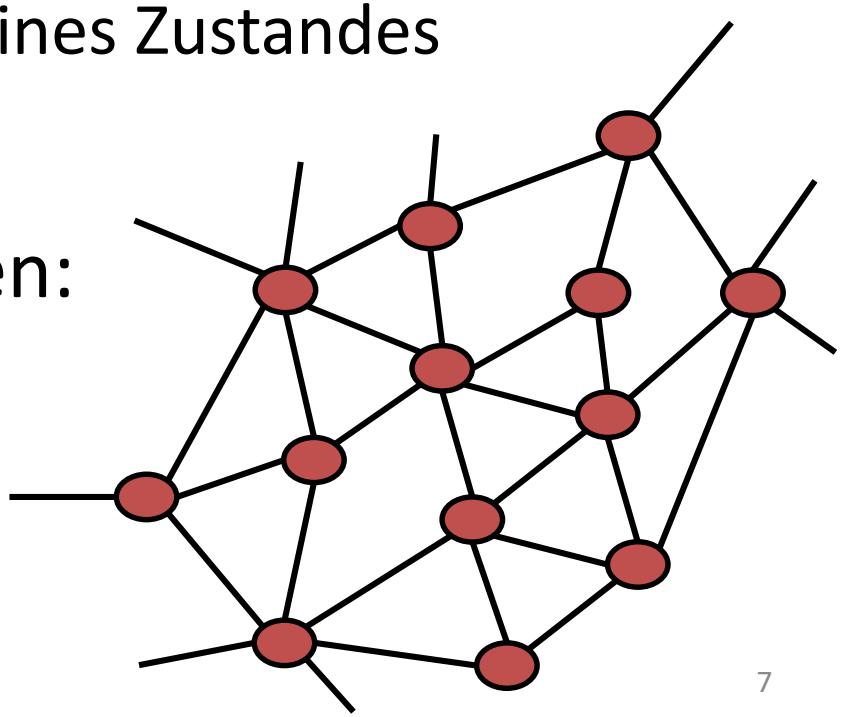
Lokale Suche

- Es wird kein Zustands-Pfad zum Ziel aufgebaut. Nur ein Zustand wird systematisch verändert in Richtung des Zielzustandes
- Wertigkeit eines Zustandes kann mit Hilfe einer Zielfunktion berechnet werden
- Optimierungsprobleme sind Beispiele für Probleme, die sich gut mit lokaler Suche lösen lassen
- Vorteile:
  - Geringer Speicherbedarf, in sehr großen Zustandsräumen anwendbar (wo optimale Verfahren nicht mehr anwendbar sind)
- Nachteil:
  - Optimalität ist nicht gewährleistet!

Nachbar eines Zustands

- Zentrale Begriffe bei den Verfahren, die einen Suchbaum systematisch aufbauen/durchlaufen:
  - Suchbaum (es gibt ein unten und oben)
  - Nachfolger (oder Kind) eines Zustandes

- Bei lokalen Suchverfahren:
  - Nachbar eines Zustandes



Der Weg ist das Ziel?

- Traveling Salesman Problem (TSP)
  - Besuche alle Städte $S_1, \ldots, S_n$ und lege dabei einen möglichst kurzen Weg zurück

- Kann auch mittels lokaler Suche gelöst werden, wie?

Nachbar beim TSP

- Traveling Salesman Problem (TSP)
  - Besuche alle Städte $S_1, \dots, S_n$ und lege dabei einen möglichst kurzen Weg zurück

- Ein Zustand ist eine Permutation der Sequenz $< S_1, \dots, S_n >$
- Ein Nachbar eines Zustand ist eine Sequenz bei der zwei Städte vertauscht sind
  - Jeder Zustand hat n * (n-1) / 2 Nachbarn

Lokale Suchverfahren

(I) Hill-Climbing

„Finde den Gipfel des Mt. Everest mit Gedächtnisverlust in dichtem Nebel“

(II) Simulated-Annealing (simuliertes Abkühlen)

„Finde die tiefste Senke indem du einen Tischtennisball schüttelst“

(III) Local Beam-Search (Lokaler-Strahl-Suche)

„Suchhundestaffel der Polizei im dichten Wald“

(IV) Genetische Algorithmen

„Sexuelle anstelle von asexueller Reproduktion.
Es lebe die Evolution!“

(I) Hill-Climbing Algorithmus

(einfachste Variante)

```
Hill-Climbing(Searchproblem problem)
   State current = createRandomState(problem)
   while (true):
      Set neighbors = getNeighbors(current)
      State bestNeighbor = getBestState(neighbors)
      if (valueOf(current) <= valueOf(bestNeighbor)):
         return current
      else:
         current = bestNeighbor
```

Hier geht es darum ein Minimierungsproblem zu lösen (analog für Maximierungsprobleme)

Beispiel: Acht-Damen Problem

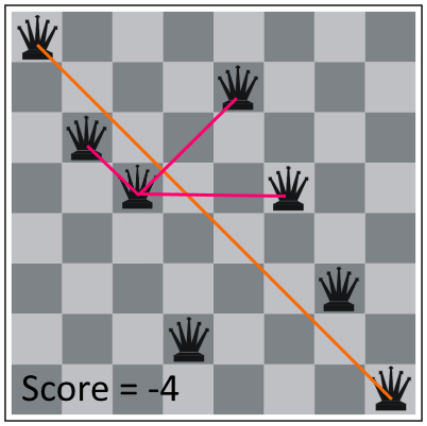
- Zunächst zufällige Positionierung der Damen
- Nachbarn eines Zustands, sind alle Zustände, bei denen eine der Damen in derselben Spalte, aber in einer anderen Reihe steht, sonst bleibt alles gleich.
- Wert eines Zustandes = Anzahl der Paare von Damen, die sich schlagen

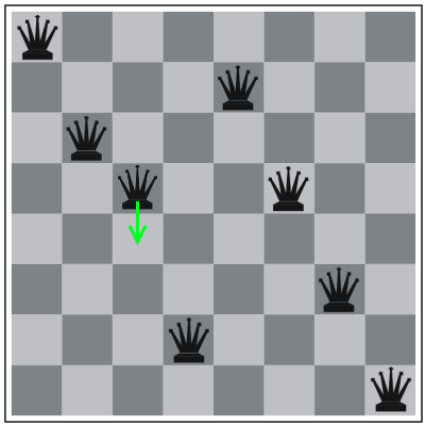
- Qualität der Lösungen:
  - Negativ: In 86% der Fälle wird keine optimale Lösung gefunden, nur für 14% der Startpositionen ist die Lösung optimal.
  - Positiv: Nur drei bis 4 Schritte sind notwendig bis keine Verbesserung mehr möglich ist!

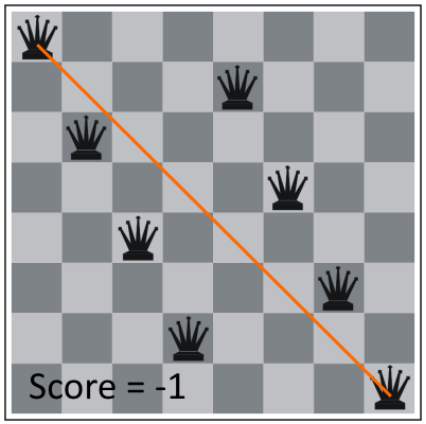
- Wie kommt es zu den negativen Ergebnissen?

Lokales Optimum

...

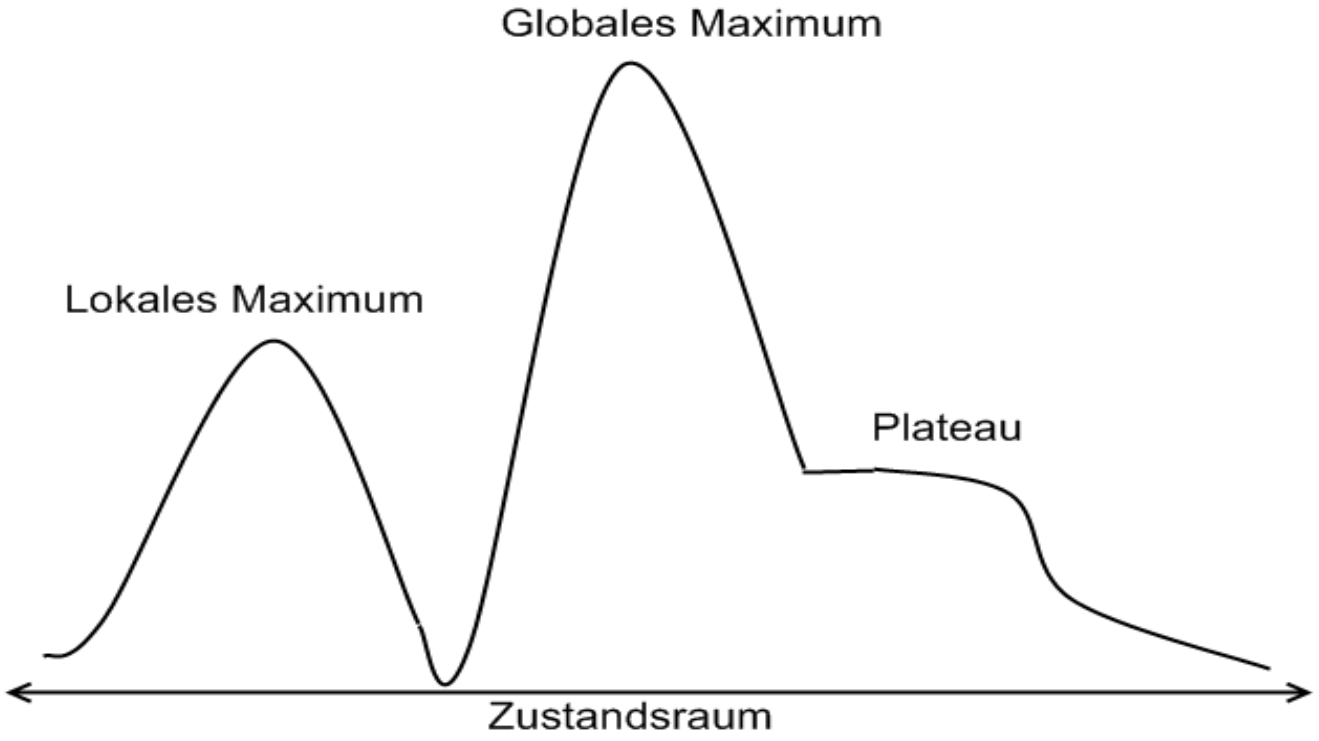






- Nach einigen Schritten erreichen wir einen Zustand, in dem sich vier Paare bedrohen
- Der „beste Nachbar“ ist ein Zustand in dem sich nur noch ein Damenpaar bedroht
- Problem: Von diesem Zustand aus läßt sich kein besser Zustand durch einen Zug erreichen!

Der Zustandsraum



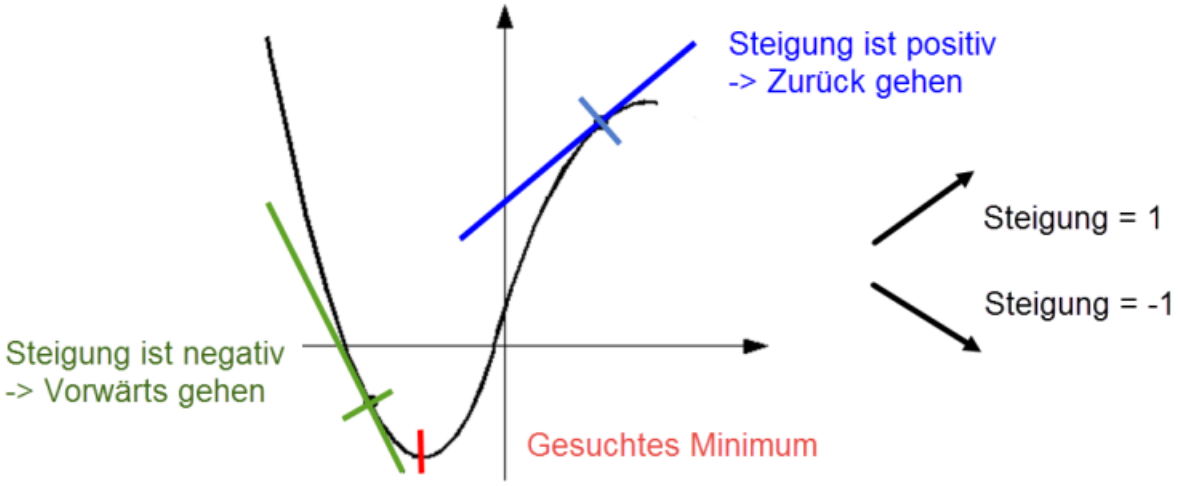
- Probleme offensichtlich, Anfangszustand ausschlaggebend, mögliche Lösungen? Vorschläge!

Varianten des Bergsteigens

- Erlaube „Seitenschritte“
  - Auch wenn sich der Wert des Zustands nicht verbessert (eine Weile!) weitergehen
  - Sehr effektiv beim 8-Damen-Problem: 100 Seitenschritte, steigert die Erfolgswahrscheinlichkeit von 14% auf 94%
- Stochastisches Hill-climbing
  - Wähle zufällig unter möglichen Aufwärtszügen aus
  - Konvergiert langsamer, aber bei bestimmten Zustandsräumen besser.
- Hill-climbing mit erster Wahl
  - Ziehe zufällige Nachfolger bis ein besserer als der aktuelle Zustand gefunden wurde (für Probleme mit sehr hoher Nachfolgerzahl)
- Hill-climbing mit Neustart
  - Starte erneut von anderem Zufallszustand, bis eine optimale Lösung gefunden wird (bzw. Abbruch nach n Versuchen).
- Gradientenabstieg (gradient descent)
  - Genauer siehe nächste Folie

Gradientenabstieg

- Problem: Gegeben $f : R^n \rightarrow R$, finde $x \in R$ unter dem $f(x)$ minimal (maximal) wird, wobei f eine differenzierbare Funktion ist
- Man wählt einen zufälligen Wert für x und geht in die Richtung (Schrittweite = alpha) , die durch den Gradienten vorgegeben wird



Gradientenabstieg

- Was ein Nachbar ist hängt davon ab, wie man α setzt
  - D.h. ein Zustand hat per se keine feste Anzahl an Nachbarn
    - Die reelen Zahlen sind kontinuierlich (und keine Felder eines Schachbretts)
  - α zu hoch => man springt um das Minimum herum
  - α zu klein => es dauert lange bis man beim Minimum ist
- Es wird immer der „Nachbar“ gewählt, der dem steilsten Abstieg entspricht
  - Genau wie beim Standard Hill Climbing
  - Analoge Probleme mit lokalen Maxima und Plateaus
- Wird oft im Kontext von Machine Learning verwendet
- **Nur anwendbar wenn Zustandraum über eine differenzierbare Funktion beschrieben werden kann**

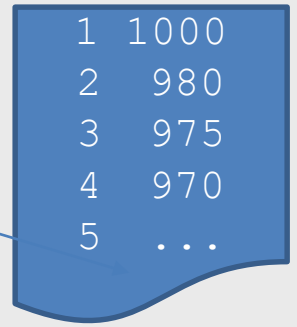
(II) Simulated Annealing

- Begriff „Annealing“ stammt aus der Metallverabeitung
  - Metalle werden gehärtet, indem man sie auf hohe Temperatur bringt und dann langsam abkühlt

- Idee (aus den 80ern): Kombiniere zufälliges Weitergehen mit Greedy-Entscheidungen.
  - Temperatur sinkt mit fortschreitender Zeit
  - Je höher die Temparatur, umso wahrscheinlicher ist es, dass ein Zufallsschritt (der die Lösung nicht verbessert) akzeptiert wird.
  - Je niedriger die Temparatur umso ähnlicher wird das Verfahren dem stochastischen Hill-climbing.
  - Sinkt die Temperatur langsam genug, so findet der Algorithmus eine optimale Lösung mit einer Wahrscheinlichkeit nahe 1.

Algorithmus (leicht vereinfacht, Maximierungsproblem)

```
Simulated-Annealing(Searchproblem problem, Schedule schedule)
   State current = createRandomState(problem)
   for round = 1 to ∞:
      Temperatur t = schedule[round]
      if t = 0:
         return current
      State next = getRandomNeighbor(current)
      ΔE = valueOf(next) – valueOf(current)
      if ΔE > 0:
         current = next
      else:
        p = getRandomValueInRange(0.0, 1.0)
        if p < some_function(ΔE,t): current = next
   return current
```



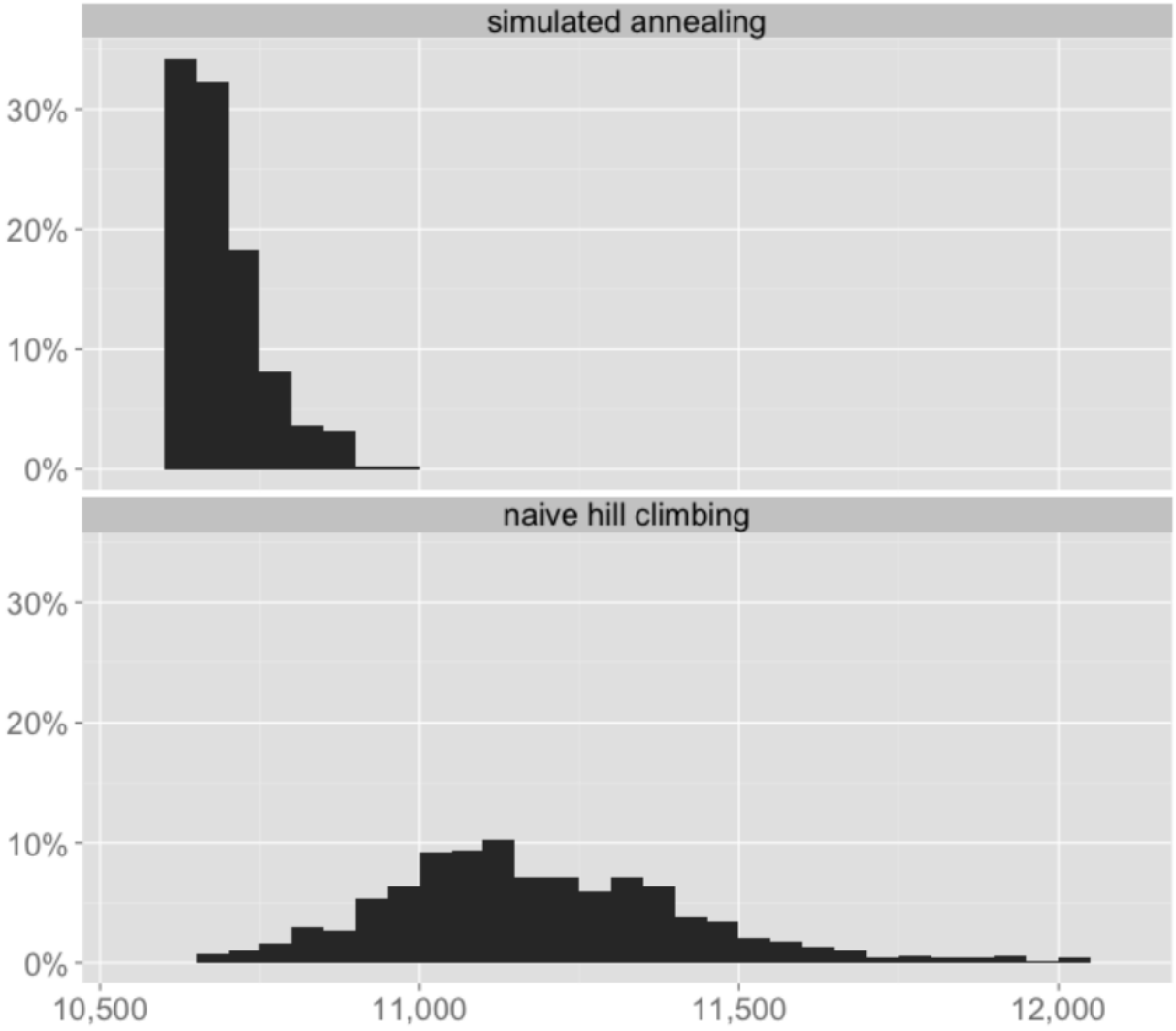
Mit sinkender Temperatur wird dies immer unwahrscheinlicher

Weiteres Beispiel

- Traveling Salesman Problem
- Gesucht wird im Raum der validen Lösungen, d.h., jeder Knoten hat genau 2 Kanten
- Online anschauen

•Screenshot taken from:
http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/

Vergleich

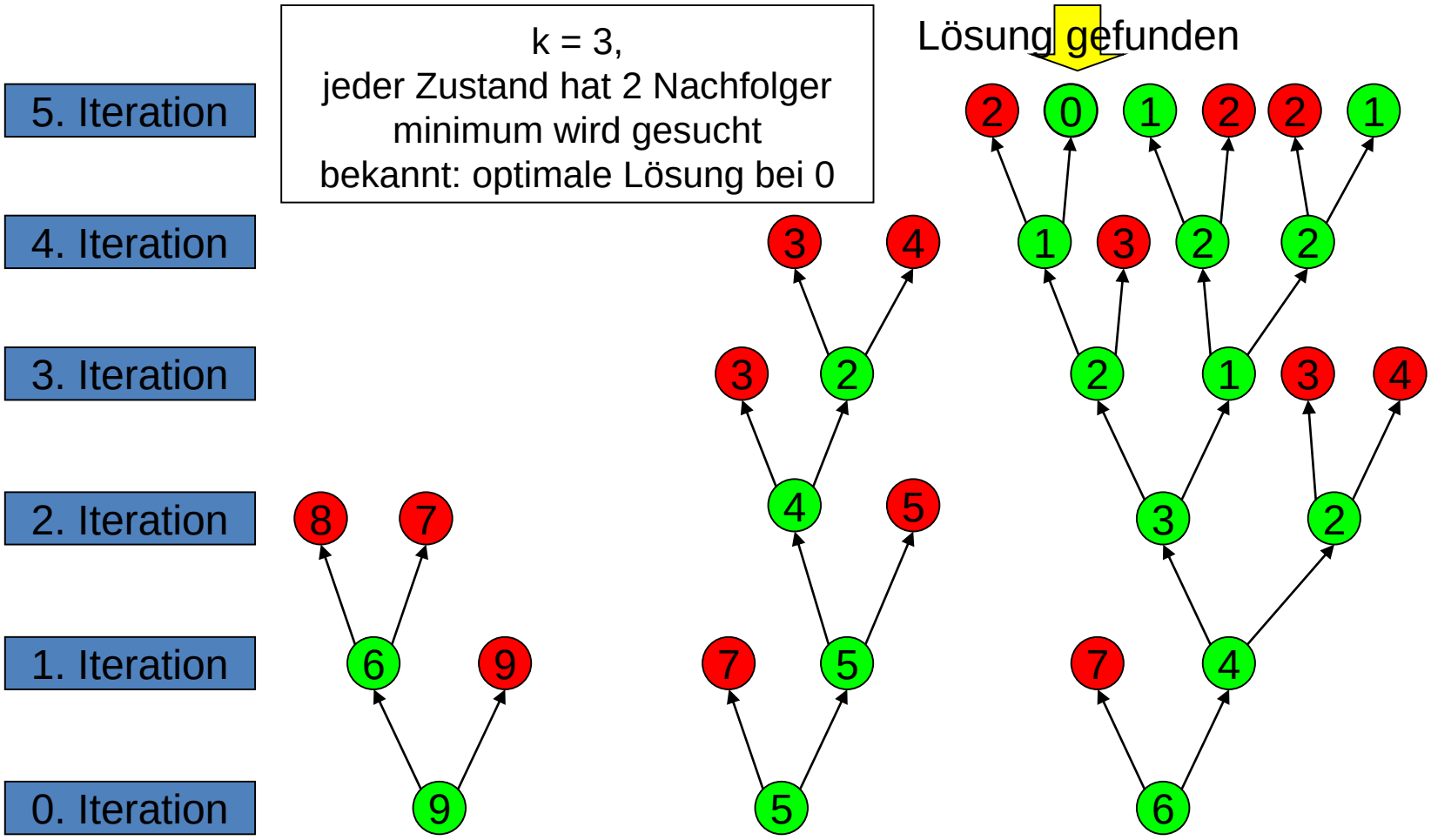

simulated annealing
30%
20%
10%
0%
naive hill climbing
30%
20%
10%
0%
10,500
11,000
11,500
12,000

(III) Local beam search

- Parallele Verwaltung von k Suchzuständen
- Zunächst k zufällige Startzustände erzeugen
- Bei jedem Schritt alle Nachfolger aller k Zustände erzeugen
- Beste k Nachfolger wählen und mit diesen weitermachen

=> Suche wird dort fortgesetzt, wo die Aussicht auf Erfolg am höchsten scheint

Graphische Darstellung



Anmerkungen

- Wieso ist dieses Verfahren verschieden von Hillclimbing mit k Neustarts?
- Problem: Zu schnelle Konzentration auf wenige Strahlen.
- Wie kann man sinnvoll eine stochastische Variante implementieren?

(IV) Genetische Algorithmen

- Variante der stochastischen Strahlsuche
- Neue Suchzustände zusätzlich (!) durch Kombination vorheriger Zustände (‚Kreuzung', ‚Kombination', ‚sexuelle Reproduktion')
- Idee basiert auf der Evolutionstheorie:

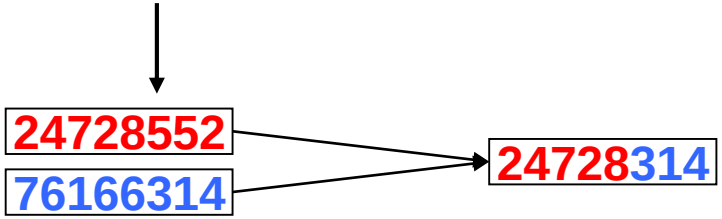
  - Bei der Reproduktion von Organismen (hier: Suchzustände) treten Variationen auf
  - Variationen durch
    - Mutation und/oder
    - Kombination
  - Durch Selektion überleben die Variationen, die am besten mit der Umwelt zurechtkommen, mit höherer Wahrscheinlichkeit
  - Diese wiederum pflanzen sich fort und geben ihr „genetisches Material" weiter.

Terminologie

- Population: Menge der Zustände einer Iteration (im Speicher)
- Fitness-Funktion: Entspricht der Kosten- oder Evaluationfunktion
- Individuum: Suchzustand
- X-te Generation: Die Population der x-ten Iteration des Suchverfahrens

Implementierung

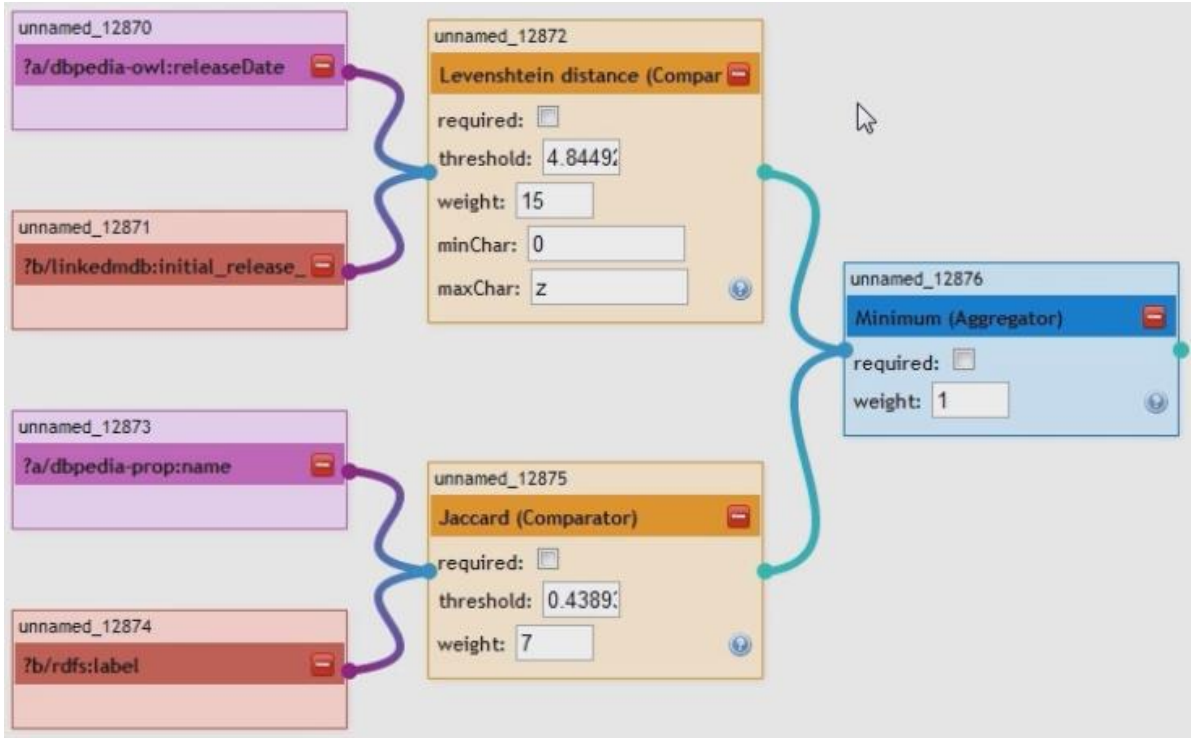
- Jedes Individuum wird durch eine Zeichenkette aus einem endlichen Alphabet dargestellt
  - Beispiel 8-Damen Problem: Eine Belegung des Bretts als 8 Ziffern von 1-8
  - 24748552 bedeutet z.B. in erster Spalte steht eine Dame auf Reihe 2, in der zweiten Spalte auf Reihe 4 usw.
- Kombination durch Auswahl eines Kreuzungspunkts
- Beispiel:



- Mutation leicht zu implementieren als zufällige Variation einer oder mehrerer Ziffern

Weiteres Beispiel

- Auffinden von Duplikaten in Datenbanken
- Gesucht ist Operatorbaum, der optimal Duplikate von zufälligen Paaren unterscheidet



Robert Isele. Learning Expressive Linkage Rules for Entity Matching using Genetic Programming. PhD Thesis Universität Mannheim

Noch ein Beispiel: Anwendung auf Spiele

- Darstellung der Heuristik als Zeichenkette um so Mutation und Kreuzung zu ermöglichen

- Fitnessfunktion wird bestimmt durch ein Turnier, bei dem die Individuuen zufällig gegeneinander antreten
  - Turnier findet nach jeder Iteration statt

- Anzahl der Siege entscheidet, ob Individuuem (Heuristik) im Genpool bliebt

Fazit zu genetischen Algorithmen

- Theoretisch sehr attraktiv und klingt cool, aber…
- Probleme:
  - Nur dann effektiv, wenn ein Zustand ‚unabhängige' Teillösungen (Blöcke von ‚Genstrings') enthält, deren Kombination evtl. Sinn macht
  - Art der Darstellung der Individuen hat enormen Einfluss auf das Verfahren
  - Identifizieren der Bedingungen unter denen genetische Algorithmen gut funktionieren ist schwer

Und nochmal …

- Hill-Climbing
  - „Finde den Gipfel des Mt. Everest mit Gedächtnisverlust in dichtem Nebel“
- Simulated-Annealing (simuliertes Abkühlen)
  - „Finde die tiefste Senke indem du einen Tischtennisball schüttelst“
- Local Beam-Search (Lokaler-Strahl-Suche)
  - „Suchhundestaffel der Polizei im dichten Wald“
- Genetische Algorithmen
  - „Sexuelle anstelle von asexueller Reproduktion, es lebe die Evolution!“

Zusammenfassung: Lokale Suchverfahren

- Vorteile: Sind auf umfangreiche Probleme anwendbar, in denen optimale Verfahren scheitern müssen

- Nachteile
  - Nicht optimal
  - Güte einer gefunden Lösung vom Zustandsraum abhängig
  - Güte einer gefunden Lösung schwer einschätzbar (wenn nicht bekannt wie die optimale Lösung aussieht)

- Grundlegender Ansatz kann optimiert werden durch …
  - Iterative Durchführung
  - Kontrolliertes randomisiertes Verhalten
  - Informationsaustausch
  - Kombinieren von ‚guten Zwischenergebnisen‘

So geht’s weiter

- Nächste Woche: Klassische Spielbaumverfahren
- Übernächste Woche: Monte Carlo Tree Search

- Und jetzt: Duplikateleminierung mit HashSets
  - Whiteboard und Eclipse