

Nicht vergessen

# Künstliche Intelligenz

## Lokale Suchverfahren

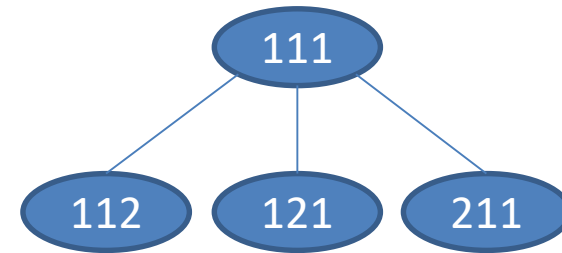
Dr. Christian Meilicke  
Research Group Data and Web Science  
Universität Mannheim

# Allgemeine Anmerkungen

- Wenn Kritik im Tutorium (aus eurer Sicht) nicht angemessen berücksichtigt wird, ist wiederholtes Nachfragen egoistisch
  - Kostet Zeit und Nerven
- Fast alle Aufgaben sind von mir gestellt
  - Wenn es Kritik gibt, bitte direkt an mich im Forum
  - Läßt sich nicht klären => Zoom Sprechstunde
  - Längere Aufgabenstellungen sind vielleicht genauer, aber darüber gibt's in der Regel auch Beschwerden => „zu lang“
- Nicht jedes mögliche Fehlverständnis muss direkt dazu führen dass Punkte gegeben werden oder Bewertung geändert wird
- **Leute, macht euch mal locker!**

# Tutorium - Rückfragen

- Aufgabe 1:
  - Breitensuche (und Tiefensuche) führt zu unatürlichem Verhalten des Fahrraddiebs
- Aufgabe 2:
  - Wolf frißt schneller als das Schaf
    - Blumenkohl, Wolf Schaf => Wolf, Blumenkohl
- Aufgabe 3:
  - „Frühere Ebene“ zweideutig bzw. in dem Fall tatsächlich andere Bedeutung als gemeint
    - Korrektur der Punktevergabe



# Übungsblätter vs. Klausur

- Die Idee der Übungsblätter: Vorbereitung auf die Klausur
- Jedes Jahr sind viele der Teilnehmer erschrocken wie schwer die Klausur ist
  - Kein Witz! 80% der Teilnehmer sagen nach der Klausur, dass die Klausur unerwartet schwer war
- Die Übungsblätter sind stellenweise ähnlich schwer die Klausuraufgaben
  - Probleme mit dem Verständnis der Übungsaufgaben => Probleme in der Klausur

# Kleine Änderung Bonusaufgaben

- Mit 0 Punkten bewertete Bonus-Aufgaben, die eventuell „Substanz“ haben, werden bei Leuten die am Ende auf der Kippe stehen von mir angeschaut und nachbewertet
- Isabel markiert diese, bei Bedarf gehen wir diese Ende des Semesters durch

# 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

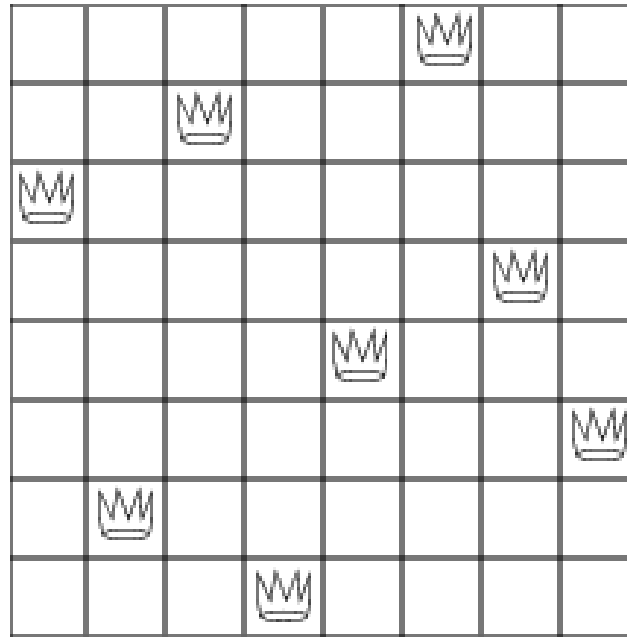
Uniforme Kostensuche

Greedy Suche

A\*Suche

# Wen interessiert ...

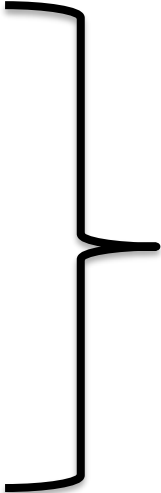
... in welcher Reihenfolge wir die Damen aufstellen?



*(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



Abstrakte  
Beschreibung  
der  
Eigenschaften  
eines  
unbekannten  
Zielzustands

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



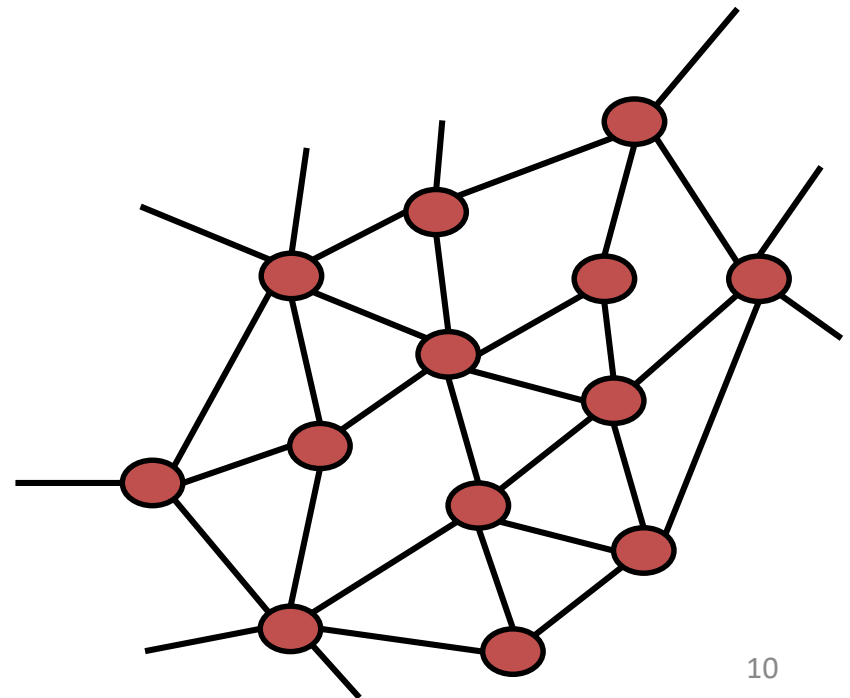
# Lokale Suche

- 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

Die Kunst bei lokalen Suchverfahren:  
Zustands-Raum sinnvoll definieren



# Der Weg ist das Ziel?

- Traveling Salesman Problem (TSP)
  - Besuche alle Städte  $S_1, \dots, S_n$  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  $\langle S_1, \dots, S_n \rangle$
- 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)

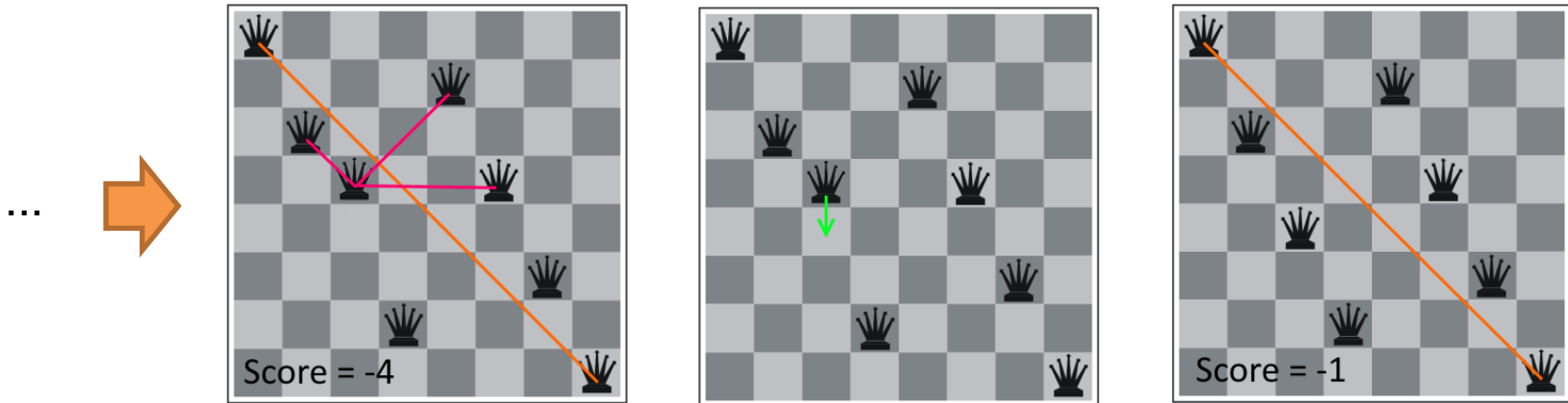
```
Hill-Climbing(Searchproblem problem)
    State current = createRandomState(problem)
    while (true):
        Set neighbors = getNeighbors(current)
        State bestNeighbor = getBestState(neighbors)
        if (valueOf(current) <= valueOf(bestNeighbor)):
            return current
        else: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

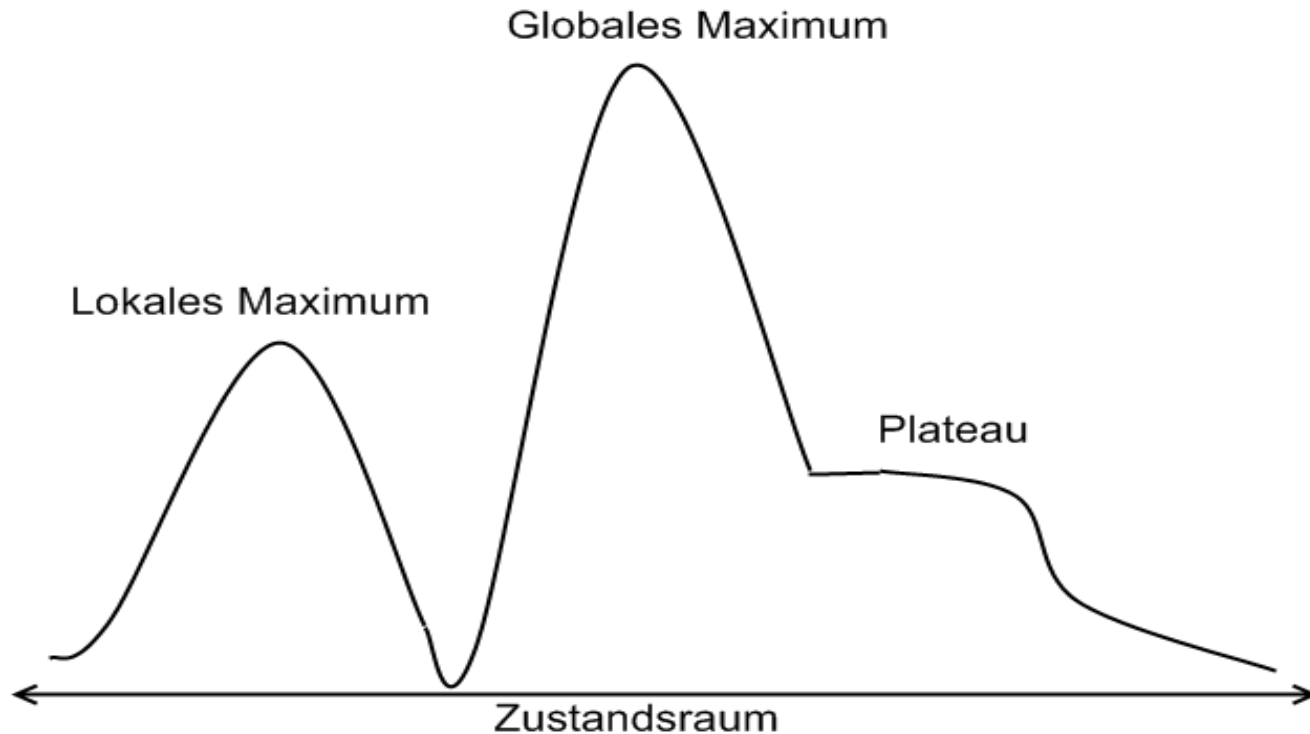
# Lokales Optimum



- 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 besserer Zustand durch einen Zug erreichen!



# Der Zustandsraum



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

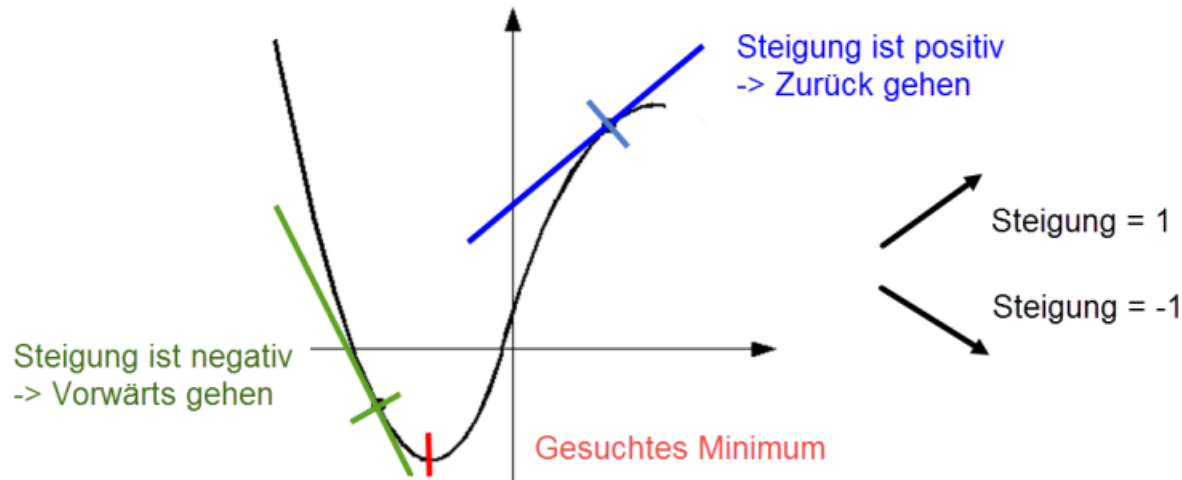
# Varianten des Bergsteigens

- 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 Aufwärtzügen 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 einem\* Zufallszustand, bis eine optimale Lösung gefunden wird (bzw. Abbruch nach n Versuchen).
- Gradientenabstieg (gradient descent)
  - Genauer siehe nächste Folie

\* kann auch derselbe sein,  
ist aber extrem unwahrscheinlich

# Gradientenabstieg

- Problem: Gegeben  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , finde  $x \in \mathbb{R}$  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



Motivation: Man weiß direkt, welches der beste Nachbar ist

# Gradientenabstieg

- Was ein Nachbar ist hängt davon ab, wie man  $\alpha$  setzt
  - D.h. ein Zustand hat per se keine feste Anzahl an Nachbarn
    - Die reellen Zahlen sind kontinuierlich (und keine Felder eines Schachbretts)
  - $\alpha$  zu hoch => man springt um das Minimum herum
  - $\alpha$  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 Zustandsraum über eine differenzierbare Funktion beschrieben werden kann**

# (II) Simulated Annealing

- Begriff „Annealing“ stammt aus der Metallverarbeitung
  - 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 Temperatur, umso wahrscheinlicher ist es, dass ein Zufallsschritt (der die Lösung nicht verbessert) akzeptiert wird.
  - Je niedriger die Temperatur 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  $\infty$ :
```

```
    Temperatur t = schedule[round]
```

```
    if t = 0:
```

```
        return current
```

```
    State next = getRandomNeighbor(current)
```

```
     $\Delta E$  = valueOf(next) - valueOf(current)
```

```
    if  $\Delta E > 0$ :
```

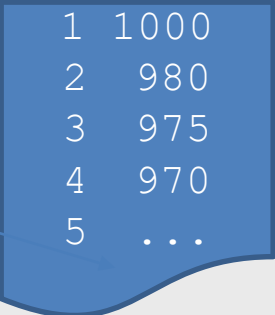
```
        current = next
```

```
    else:
```

```
        p = getRandomValueInRange(0.0, 1.0)
```

```
        if  $p < \text{some\_function}(\Delta E, t)$ : current = next
```

```
return current
```



1	1000
2	980
3	975
4	970
5	...

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

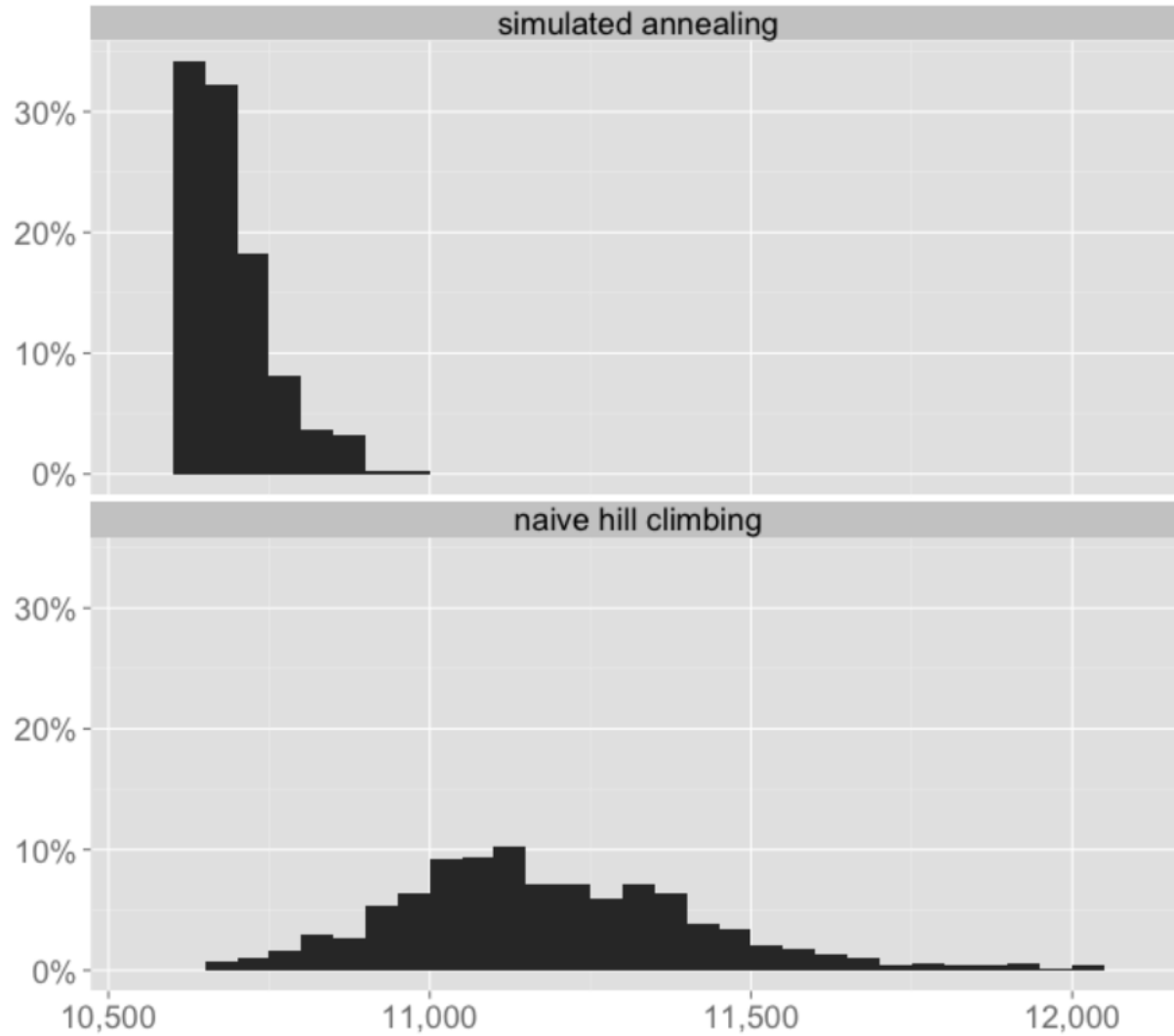
Distance: 12,135 miles  
Temperature: 58  
Iterations: 220,000



•Screenshot taken from:

<http://toddwshneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/>

# Vergleich





# (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

$k = 3$ ,  
 jeder Zustand hat 2 Nachfolger  
 minimum wird gesucht  
 bekannt: optimale Lösung bei 0

Lösung gefunden

5. Iteration

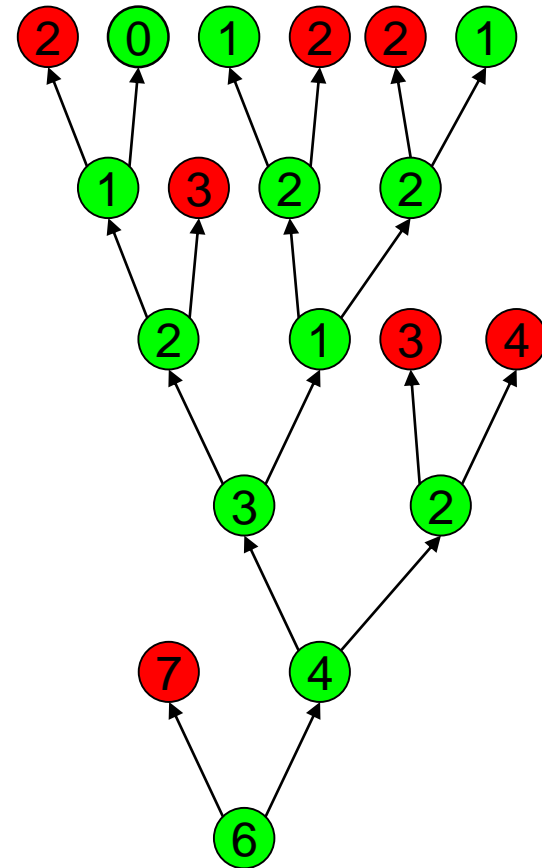
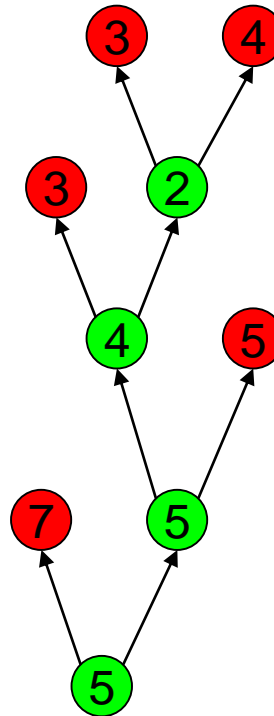
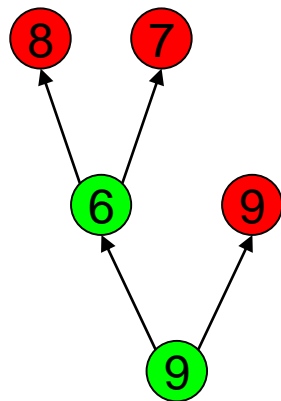
4. Iteration

3. Iteration

2. Iteration

1. Iteration

0. Iteration



# 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

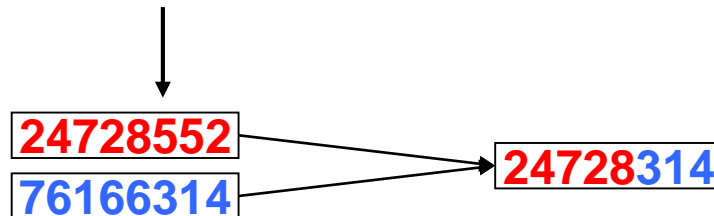
- Variante der stochastischen Strahlsuche
- Neue Suchzustände zusätzlich (!) durch Kombination vorheriger Zustände (,Kreuzung‘, ,Kombination‘, ,sexuelle Reproduktion‘)
- Idee basiert auf der Evolutionstheorie:
  - Bei der Reproduktion von Organismen (hier: Suchzustände) treten Variationen auf
  - Variationen 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

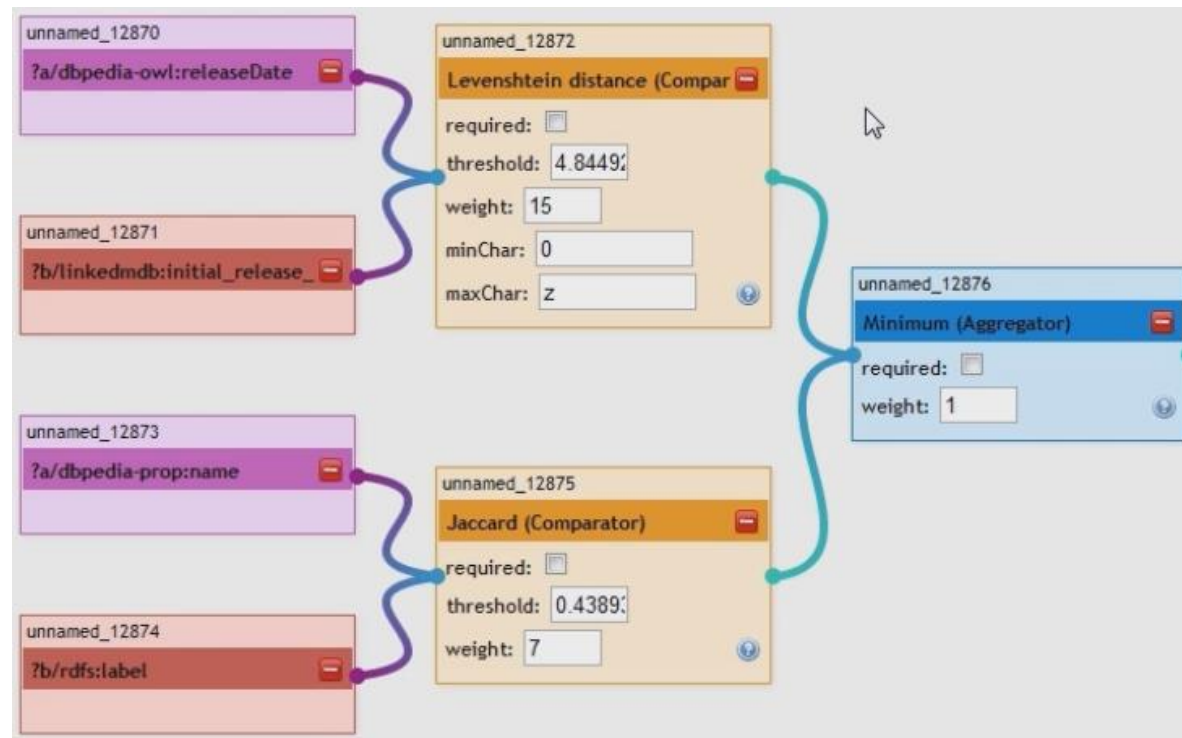
- Jedes Individuum wird durch eine Zeichenkette aus einem endlichen Alphabet dargestellt
  - Beispiel 8-Damen Problem: Eine Belegung des Bretts als 8 Ziffern von 1-8
  - 24748552 bedeutet z.B. in erster Spalte steht eine Dame auf Reihe 2, in der zweiten Spalte auf Reihe 4 usw.
- Kombination durch Auswahl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



# 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 Individuen zufällig gegeneinander antreten
  - Turnier findet nach jeder Iteration statt
- Anzahl der Siege entscheidet, ob Individuum (Heuristik) im Genpool bleibt



# 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 gefundenen Lösung vom Zustandsraum abhängig
  - Güte einer gefundenen 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 Zwischenergebnissen‘

# So geht's weiter

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