Hochschule Offenburg
University of Applied Sciences

# Algorithmen und Datenstrukturen
# 5. Lokale Suche

Prof. Dr. Klaus Dorer

# Übersicht



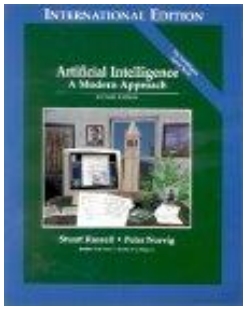
- Lokale Suche
  - Kategorisierung von Umgebungen
  - Beispiel Logistiknetzwerk
  - Hill Climbing
  - Tabu Suche
  - Simulated Annealing

# Ziele

- Gemeinsamkeiten und Unterschiede der Verfahren kennen
- Geeignetes Verfahren für ein Problem auswählen können
- Behandelte Optimierungsverfahren implementieren können

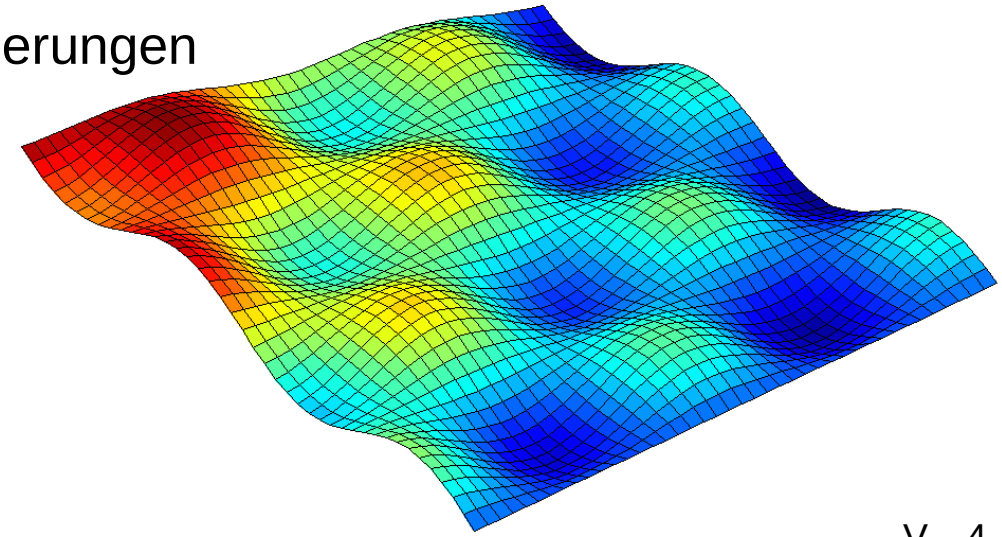
# Quellen

- Russel, Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, ISBN 0137903952, 2002.



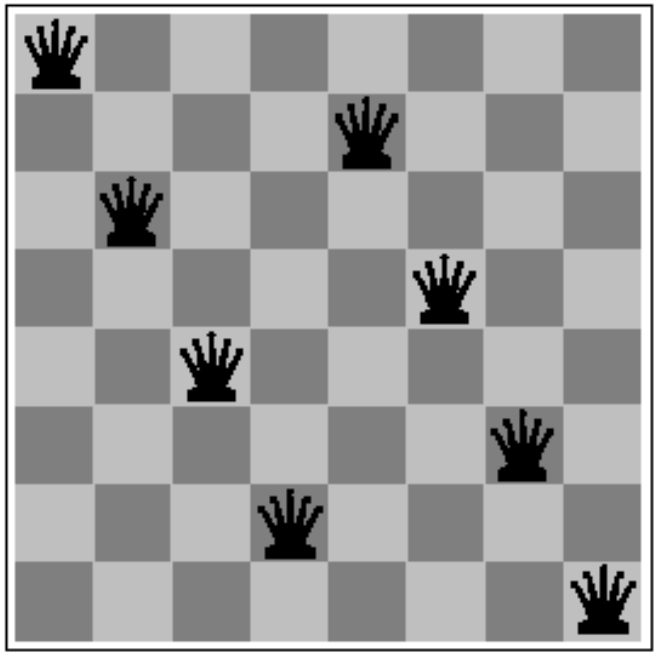
# Lokale Optimierung

- Gesucht wird die optimale Lösung eines Problems
- Durch Variation einer Lösung können ‚benachbarte' Lösungen erreicht werden
- Suche ist wie ein Wandern auf einer Oberfläche, wobei die Position die aktuelle Lösung und deren Höhe die Güte der Lösung widerspiegelt
  - Beginne mit irgendeiner Lösung
  - Mache lokale Verbesserungen



# Beispiel: n Dame Problem

- Auf einem Schachbrett sollen 8 Damen so platziert werden, dass keine Dame eine andere schlagen kann. Erweiterbar auf n Damen
  - Nicht der Weg zur Lösung ist interessant, sondern die Endkonfiguration
- Vorgehensbeispiel
  - Platziere in jede Spalte und Zeile eine Dame
  - Tausche 2 Damen so, dass weniger Konflikte entstehen



# Begriffe

- Suchraum S
  - Menge aller Zustände s des Problems
- Nachbarzustände N von s
  - Menge aller Zustände, die von s aus direkt erreichbar sind
- Operator
  - Führt einen Zustand s in einen Nachbarzustand s' über
- Evaluations-Funktion
  - Bildet einen Zustand auf eine Bewertung ab: e(s)->IR
- Lokales Optimum
  - Güte eines Zustands s ist besser als oder gleich gut wie alle Nachbarzustände: $e(s) >= \max_N(e(s_n))$
- Globales Optimum
  - Zustand s mit bestem (höchsten/niedrigsten) Wert e(s)

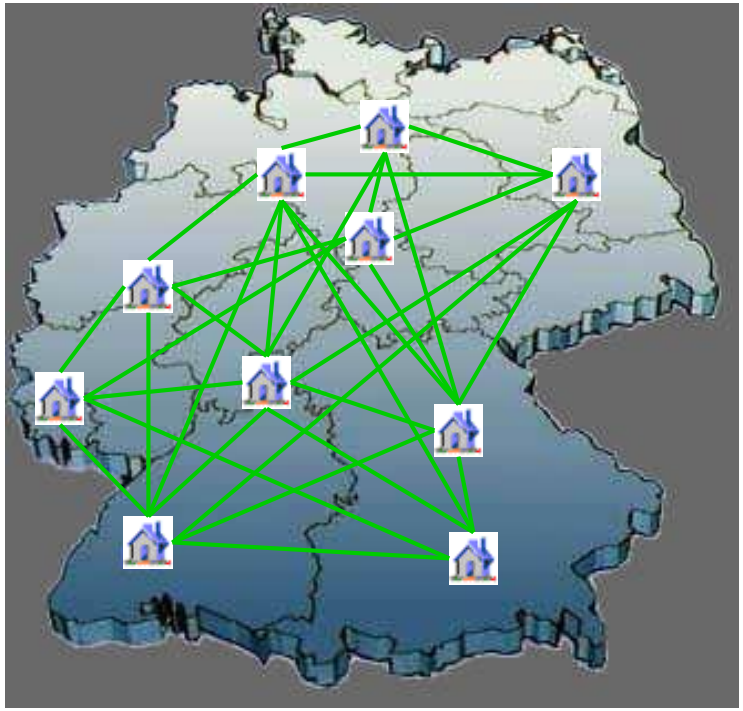
# Repräsentation

- Ein Problem ist definiert durch
  - Anfangszustand $s_a$
  - Operatoren als Funktion f(s,a)->s*, die angibt, welchen Zustand man bei der Ausführung des Operators in der aktuellen Situation erreicht
  - Eine Funktion e(s), die die Güte von Zustand s liefert

- Jeder Zustand ist eine Lösung des Problems
- Gesucht wird eine möglichst gute Lösung in Bezug auf e(s)

# Beispiel: Transport-Netzwerk Optimierung

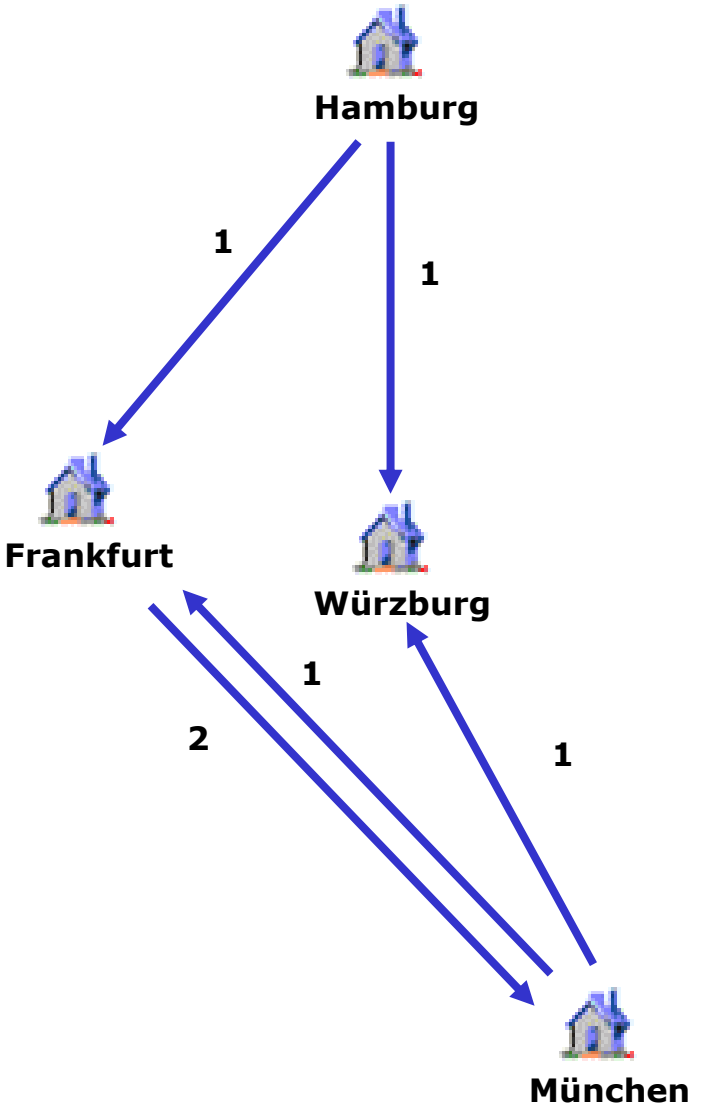
- Aufgabe
  - Generiere Transportplan für alle Relationen zwischen Hubs
- Ziele
  - Kostenreduktion
  - Kilometerreduktion
  - Minimierung der Anzahl LKW
- Randbedingungen
  - Eingeschränkte LKW Kapazität
  - Fahrzeitrestriktionen
  - Früheste Abfahrtzeiten
  - Späteste Ankunftszeiten
  - Lade-, Ablade- und Umladezeiten
  - Öffnungszeiten der Hubs



# Ausgangslage

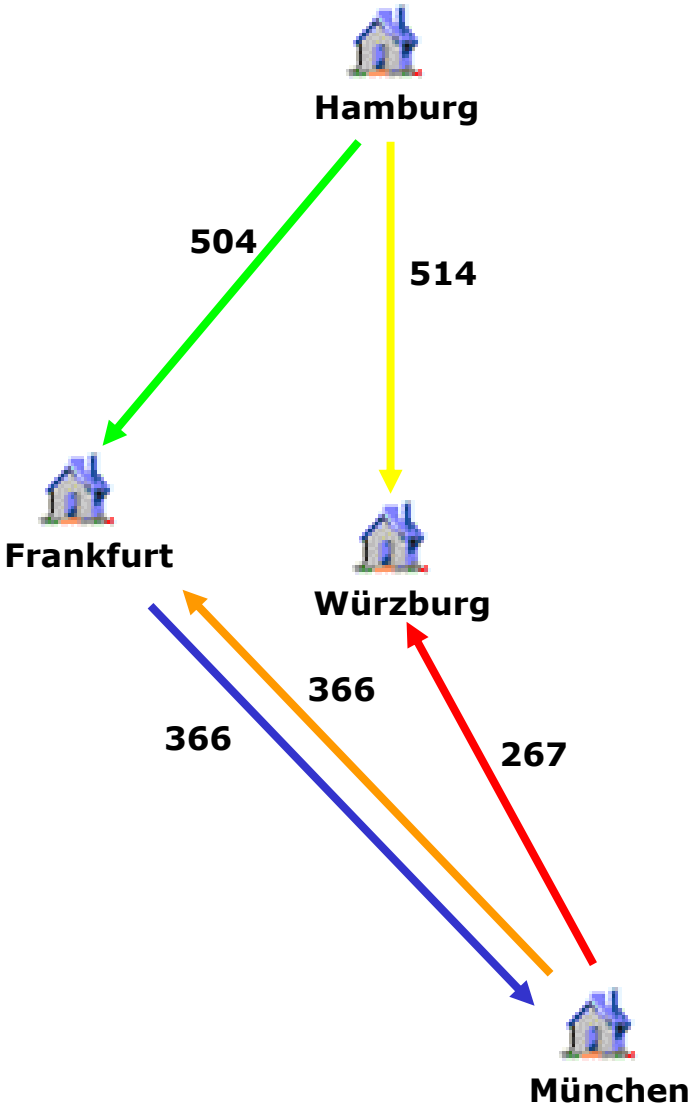
- 5 der 981 Relationen eines Logistik Unternehmens
- 6 Wechselbrücken müssen transportiert werden
- Ein LKW kann 2 Wechselbrücken laden

  - Nicht maßstabsgetreu



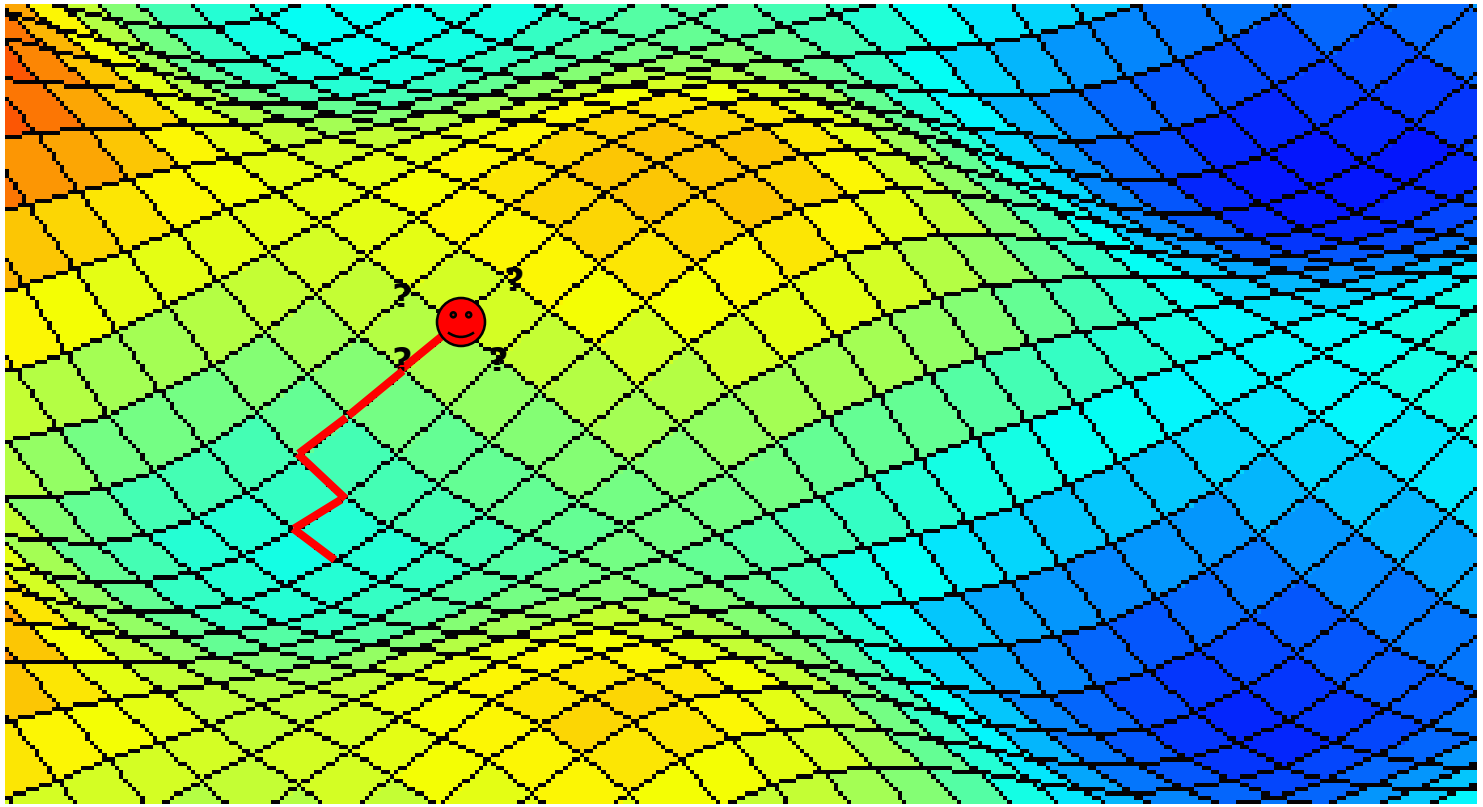
# Erste Lösung: Alles direkt fahren

- Lösungsqualität
  - 2017 km
  - 5 LKW



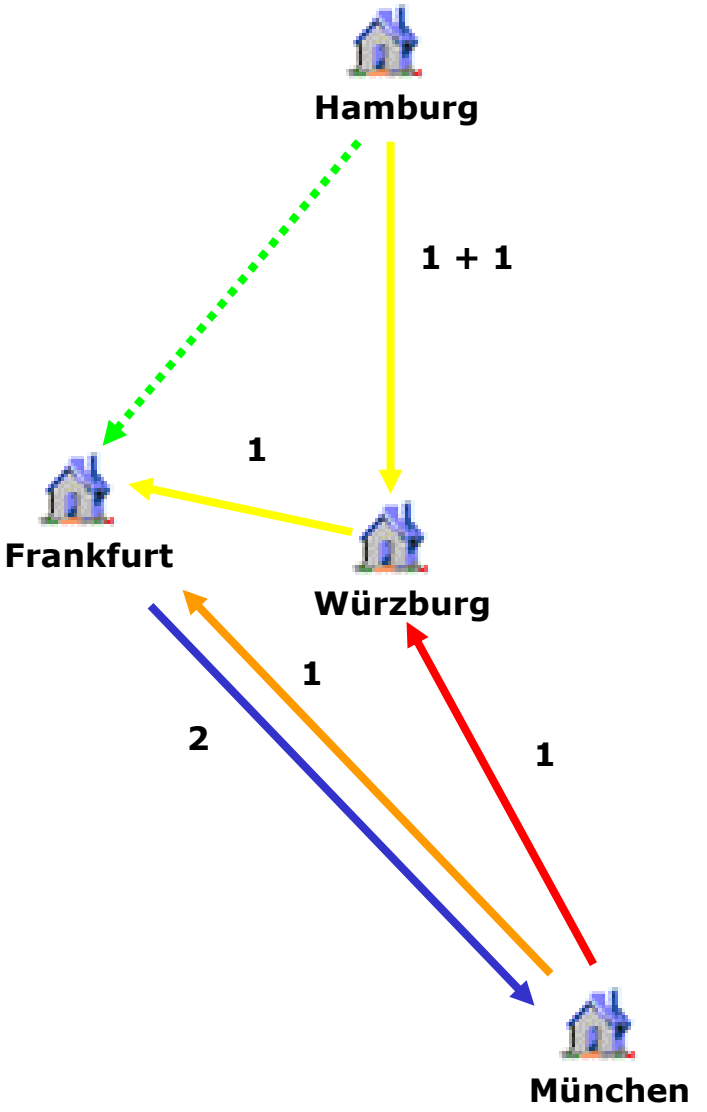
# Hill Climbing

- Auch Gradientenabstieg
- Immer so steil wie möglich bergab gehen



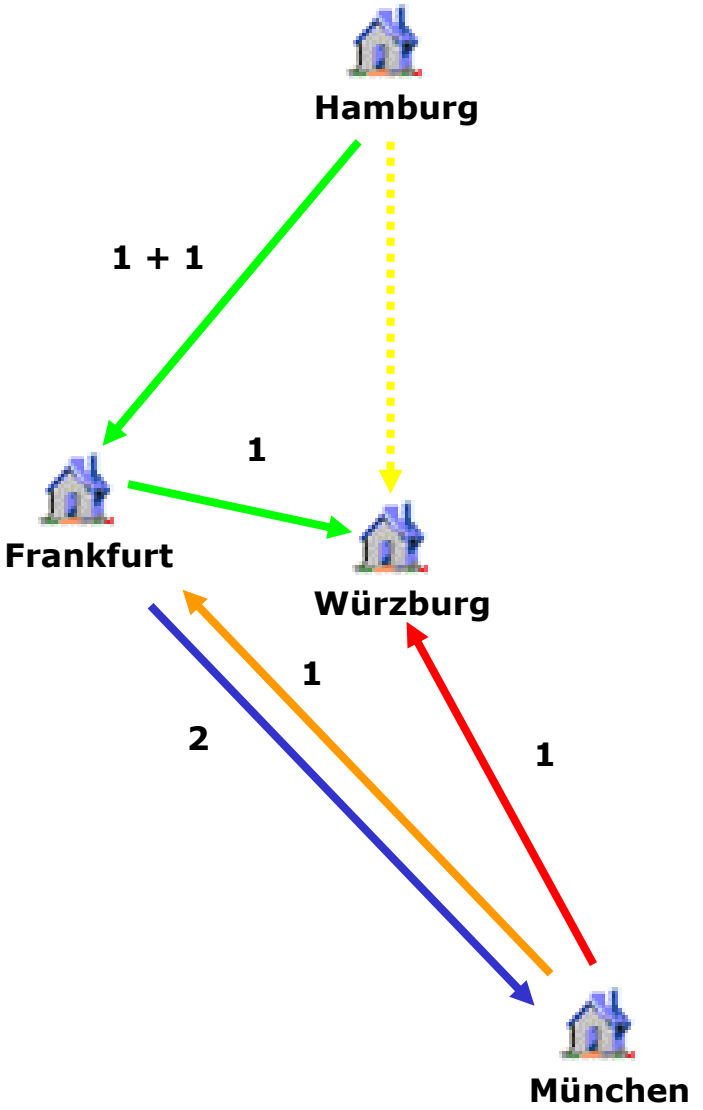
# Verbesserung: über Hub fahren

- Auftrag Hamburg - Frankfurt über Würzburg
- Einsparung
  - 398 km
  - 1 LKW
- Lösungsqualität
  - 1619 km
  - 4 LKW



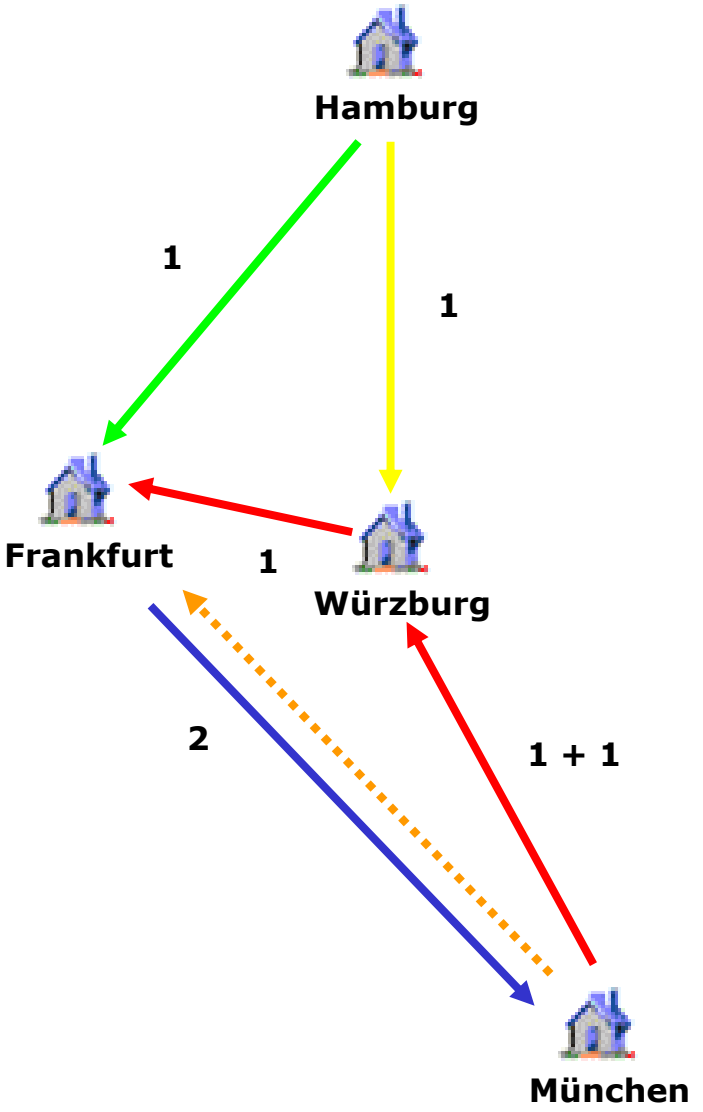
# Verbesserung: über Hub fahren

- Auftrag Hamburg - Würzburg über Frankfurt
- Einsparung
  - 408 km
  - 1 LKW
- Lösungsqualität
  - 1609 km
  - 4 LKW



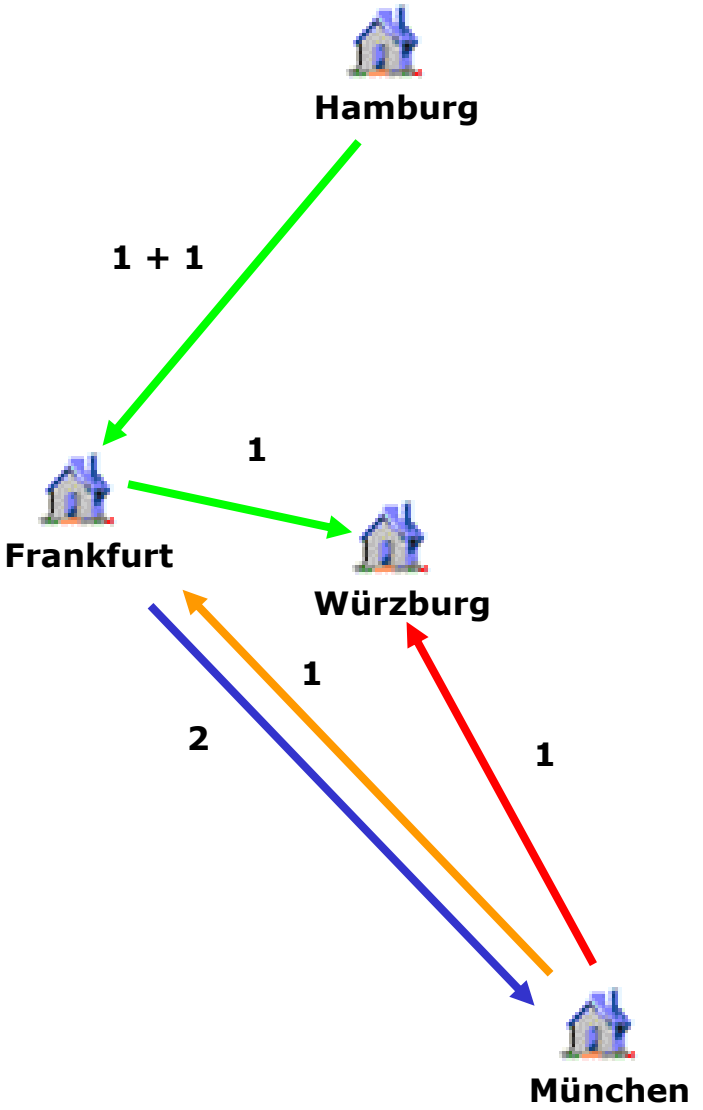
# Verbesserung: über Hub fahren

- München - Frankfurt über Würzburg
- Einsparung
  - 260 km
  - 1 LKW
- Lösungsqualität
  - 1757 km
  - 4 LKW



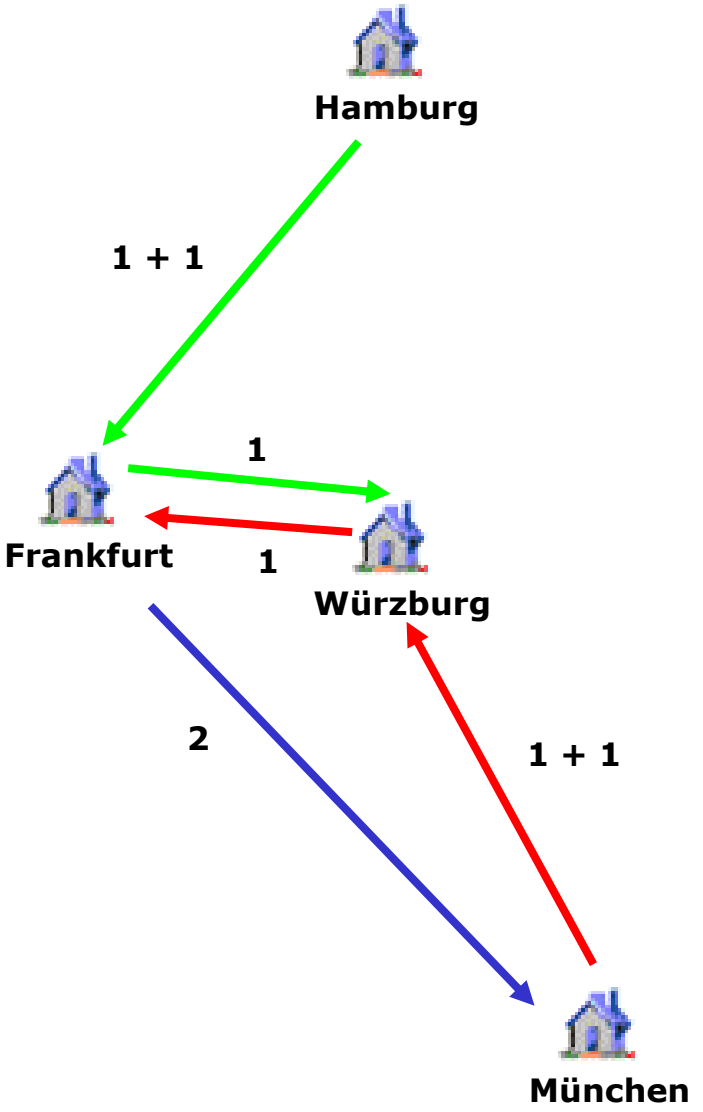
# Verbesserung: über Hub fahren

- Auftrag Hamburg - Würzburg über Frankfurt
- Einsparung
  - 408 km
  - 1 LKW
- Lösungsqualität
  - 1609 km
  - 4 LKW



# Verbesserung: über Hub fahren

- Hamburg - Würzburg über Frankfurt
- München - Frankfurt über Würzburg
- Lösungsqualität
  - 1349 km
  - 3 LKW
- Keine weitere Verbesserung mit Hill Climbing möglich



# Hill Climbing

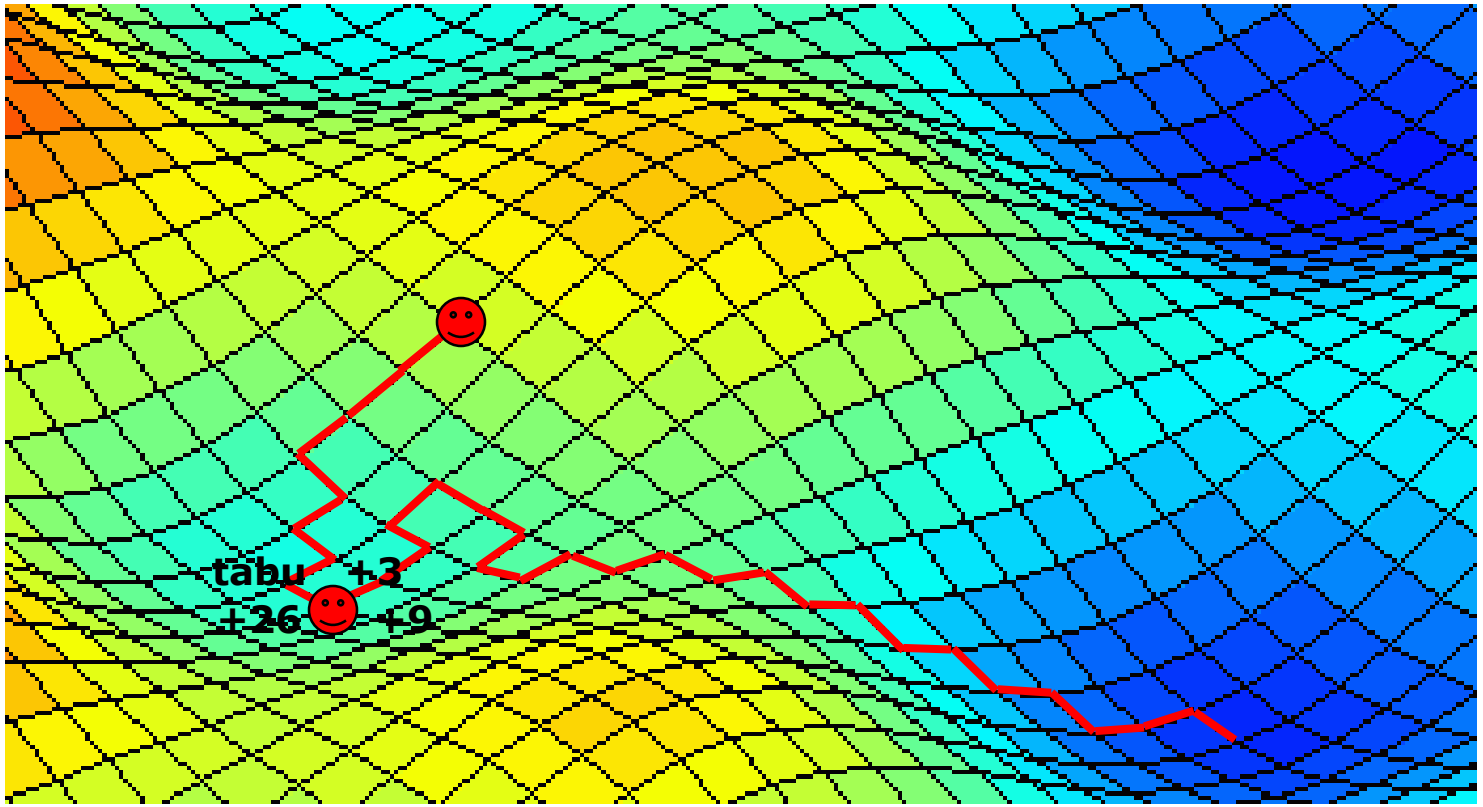
- Vorteile
  - Führt schnell zu Verbesserungen
  - Nur drei Zustände müssen gleichzeitig im Speicher sein
  - Erkennt Suchende selbst
- Nachteile
  - Bleibt in lokalen Optima hängen
  - Ist langsam, wenn es viele Nachbarzustände gibt
- Verbesserungen
  - Restart von einem anderen Ausgangszustand (wenn möglich)
  - Random Walk: im lokalen Optimum gehe n zufällige Schritte und wiederhole m mal.
  - Random Operator Selection: wähle einen Nachbarn zufällig und gehe dahin, wenn eine Verbesserung erzielt wird (Laufzeit)

# Hill Climbing

```
while true
{
     for all operators(currentState)
     {
          getSuccessorState(currentState, operator);
          calculateUtility();
          if (utility > bestSuccessor)
               store bestSuccessor;
     }

     if bestSuccessor <= currentState
          // reached local optimum
          return currentState;

     currentState = bestSuccessor;
}
```

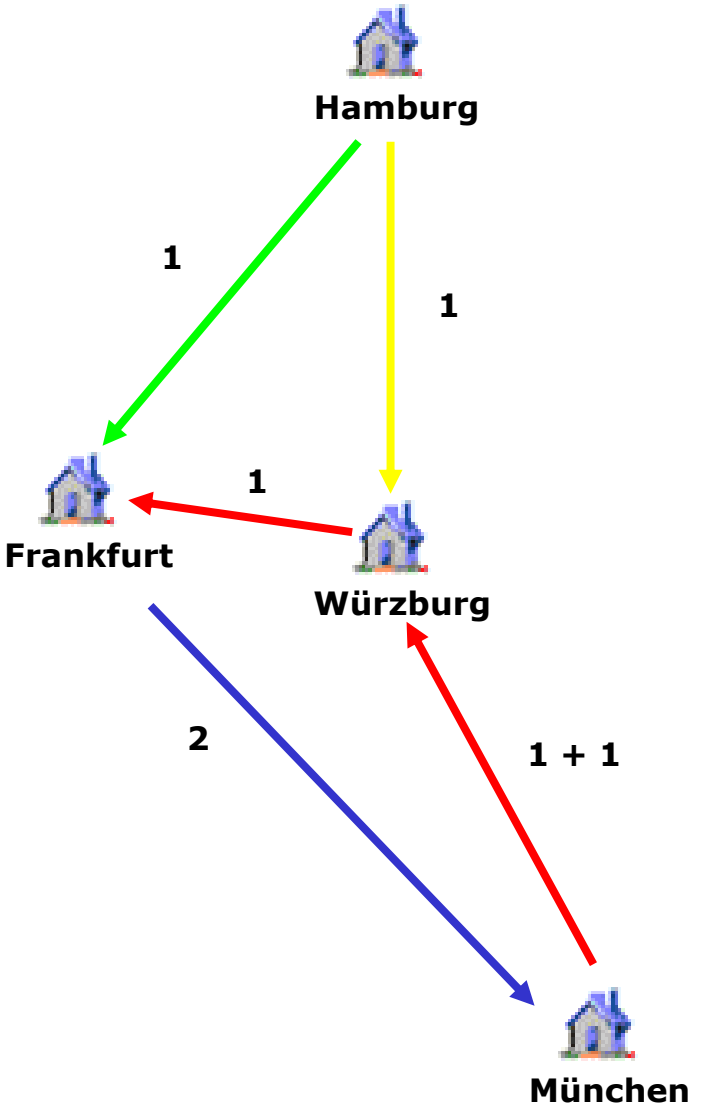
# Tabu Suche

- Immer so steil wie möglich bergab gehen, nur nicht da hin, wo man die letzte Zeit war
- D.h. eventuell auch bergauf gehen



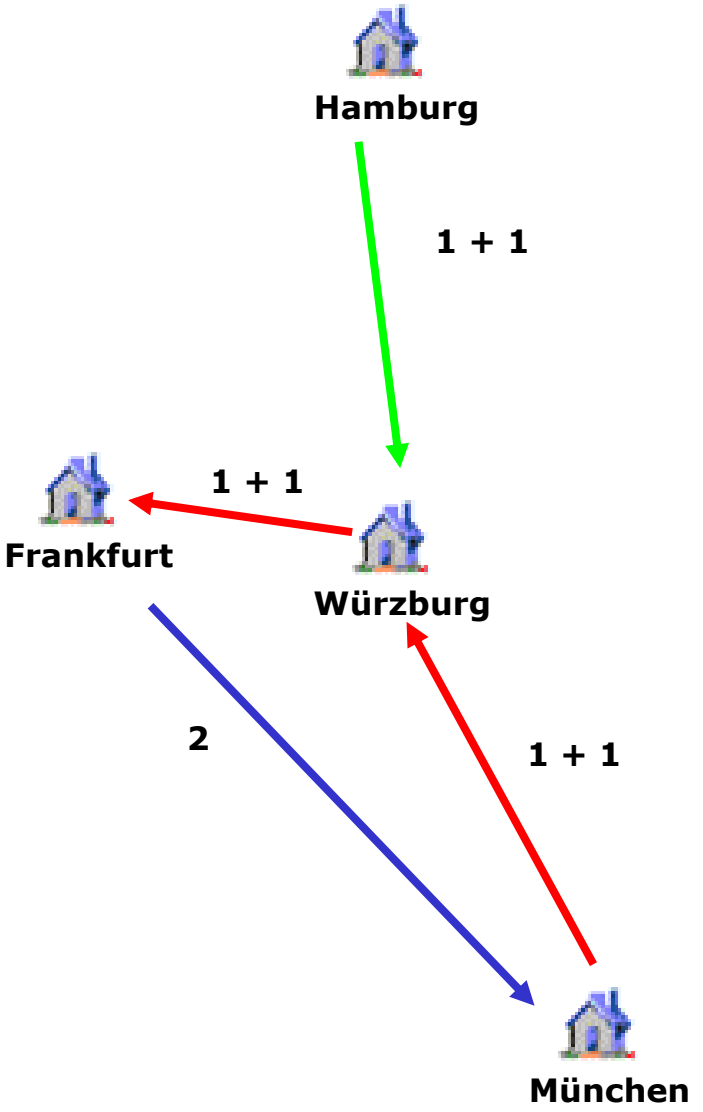
# Verschlechterung: Auftrag direkt fahren

- Hamburg - Würzburg direkt
- Verschlechterung
  - 408 km
  - 1 LKW
- Lösungsqualität
  - 1757 km
  - 4 LKW



# Verbesserung: Auftrag umladen

- Hamburg - Frankfurt über Würzburg und dort umladen
- Verbesserung
  - 504 km
  - 1 LKW
- Lösungsqualität
  - 1253 km
  - 3 LKW
  - Optimale Anzahl Kilometer
  - Optimale Anzahl LKW
  - Kosten noch nicht optimal



# Tabu Suche

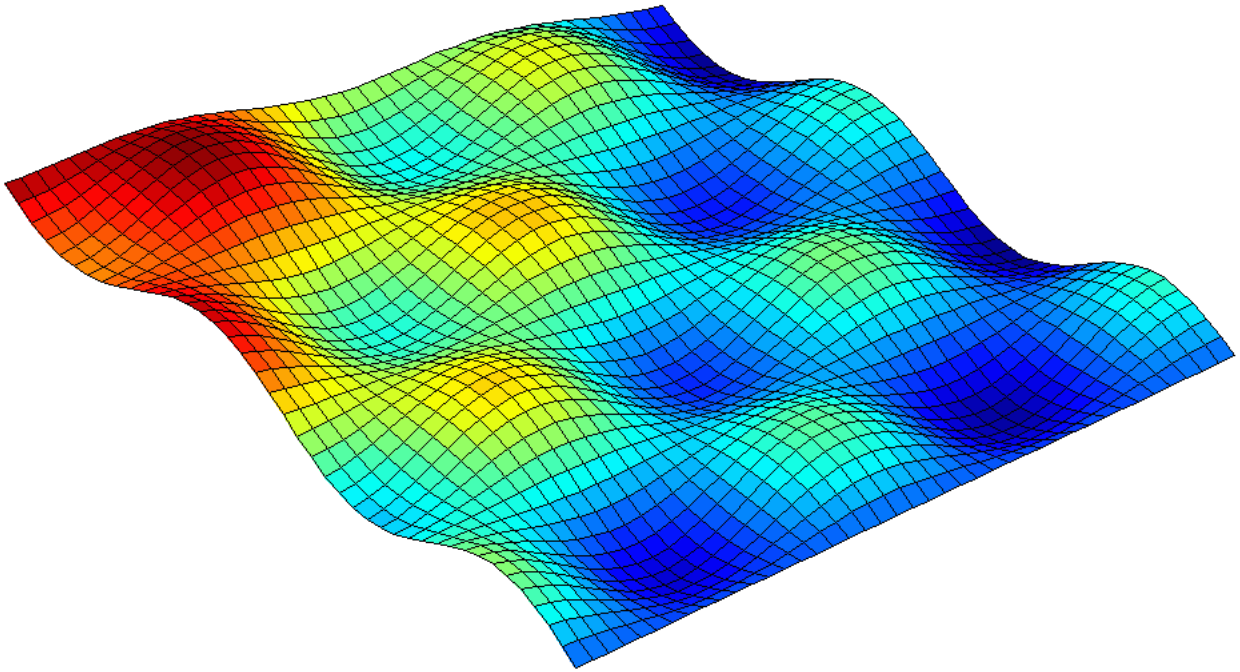
- Vorteile
  - Kann lokale Optima verlassen
  - Führt schnell zu Verbesserungen
- Nachteile
  - Höherer Speicherverbrauch für Tabu Liste
  - Suchende muss durch zusätzliche Bedingung (maximale Zeit, Zyklen) festgelegt werden
  - Ist langsam, wenn es viele Nachbarzustände gibt
  - Es kann lange dauern und eine große Tabu Liste benötigen, um ein lokales Optimum zu verlassen
- Verbesserungen
  - Diversifizierung: ein Langzeitgedächtnis erkennt, wenn lange lokal gesucht wurde und wendet Operatoren zum Verlassen des lokalen Suchraums an

# Tabu Suche

```
while more iterations
{
    for all operators(currentState)
    {
        getSuccessorState(currentState, operator);
        calculateUtility();
        if (best successor && not in tabu list)
            store best successor;
    }
    // no break condition in case of local optimum

    if best state so far
        store best state;

    currentState = best successor;
    addToTabuList(currentState);
}
```

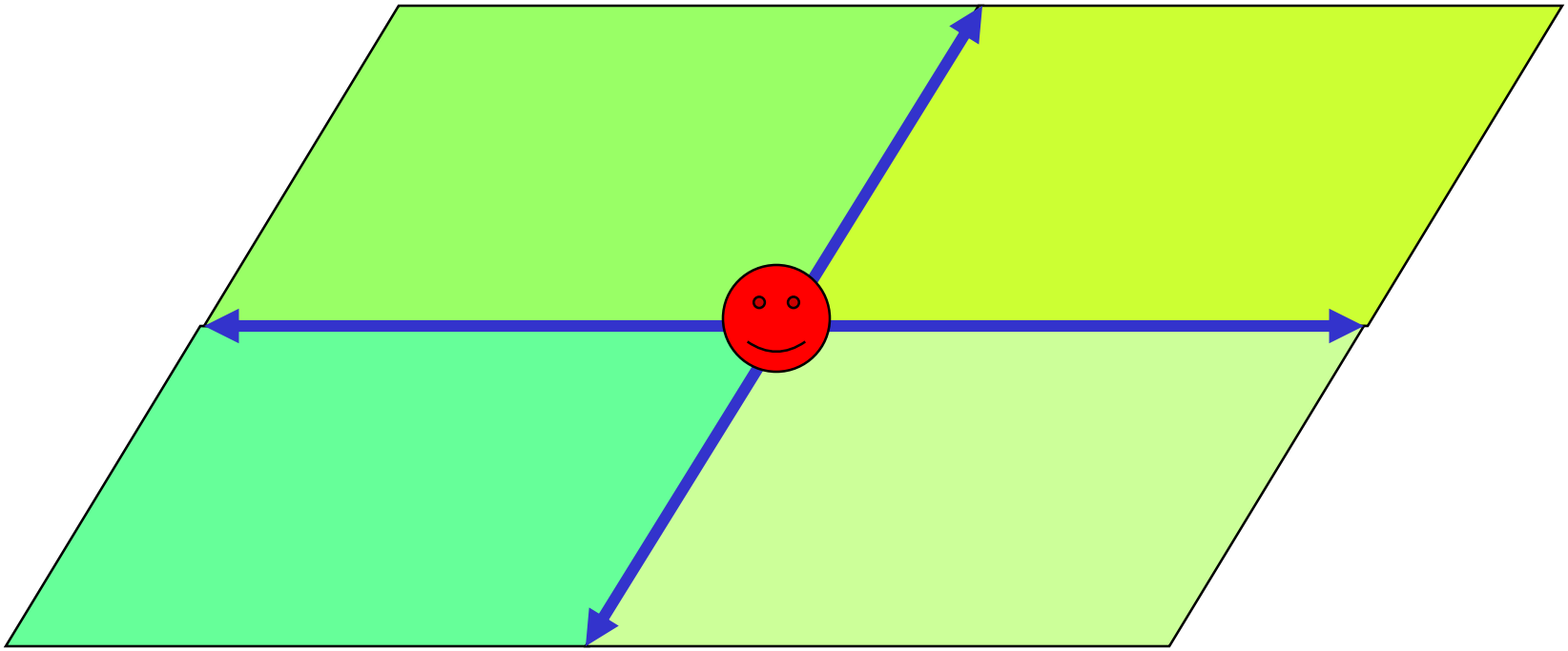
# Schwierigkeiten

- Man sieht doch auf einen Blick, wo die beste Lösung ist



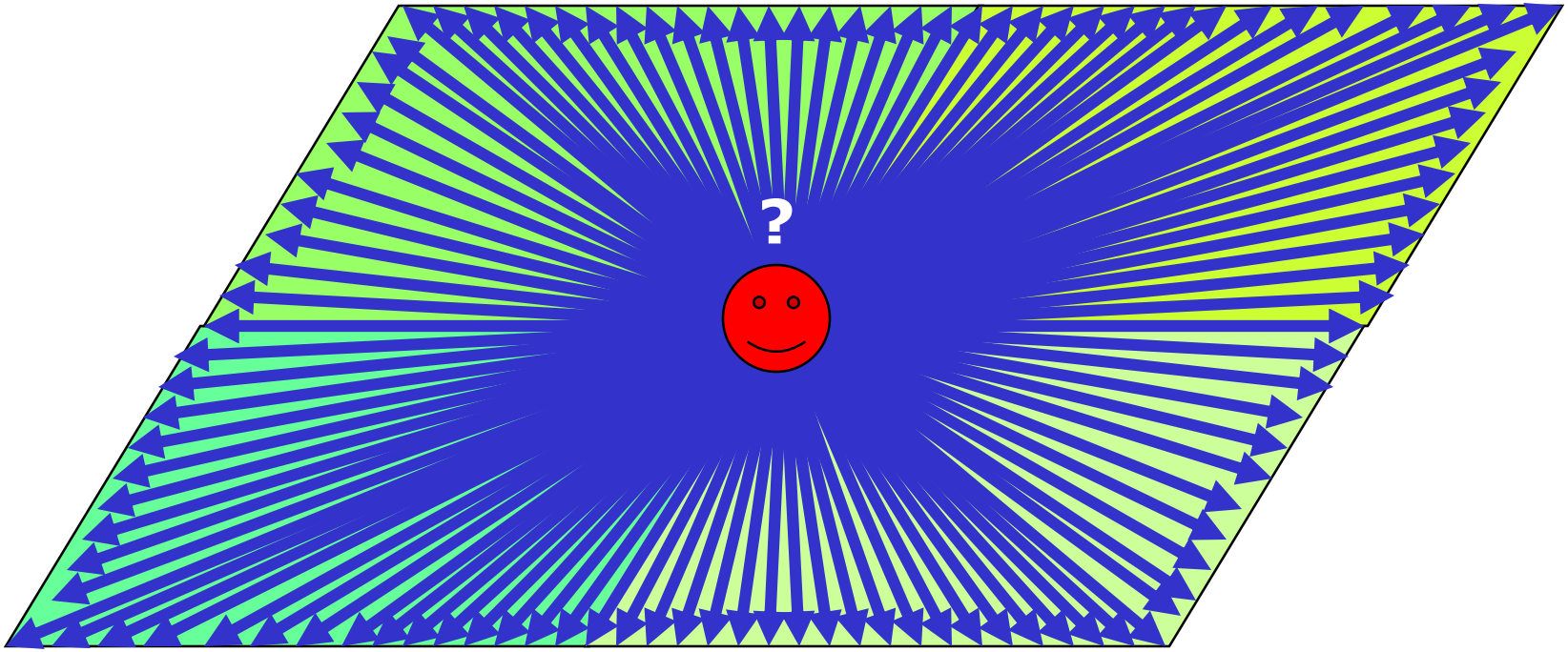
# Schwierigkeiten

- Keine Vogelperspektive



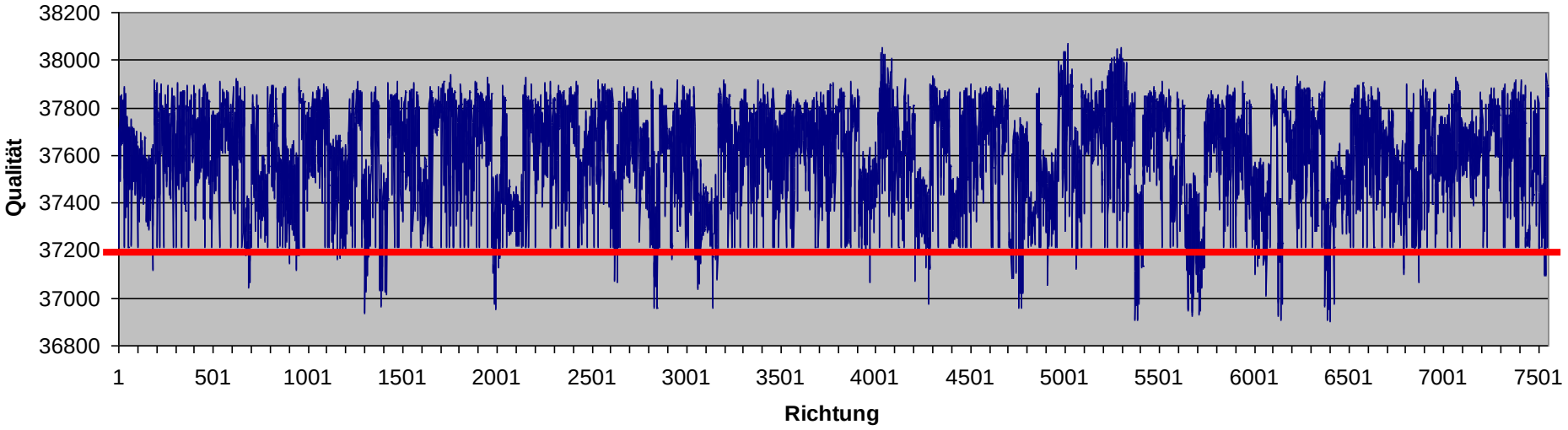
# Schwierigkeiten

- Es gibt nicht 4 mögliche Richtungen sondern ca. 7.500



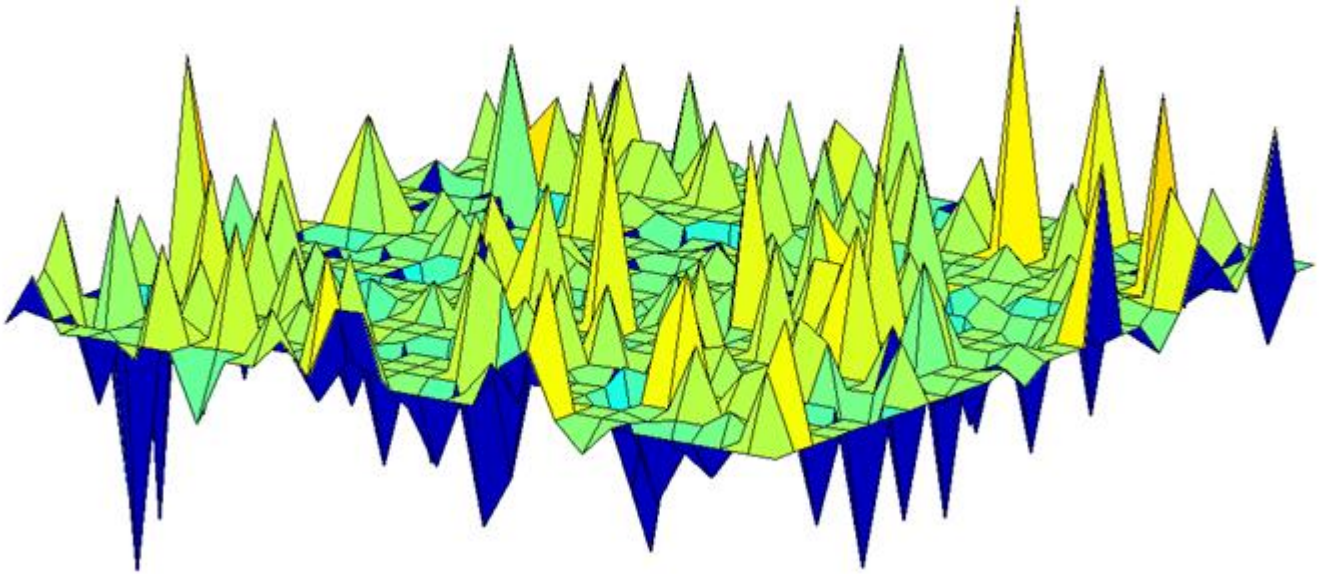
# Schwierigkeiten

- Blick auf die 7553 möglichen Richtungen eines Zustands
- Zustand selbst hat Niveau von 37216
- Wenige Richtungen verbessern Ergebnis
- Viele verschlechtern Ergebnis
- Weg zur optimalen Lösung könnte bergauf gehen



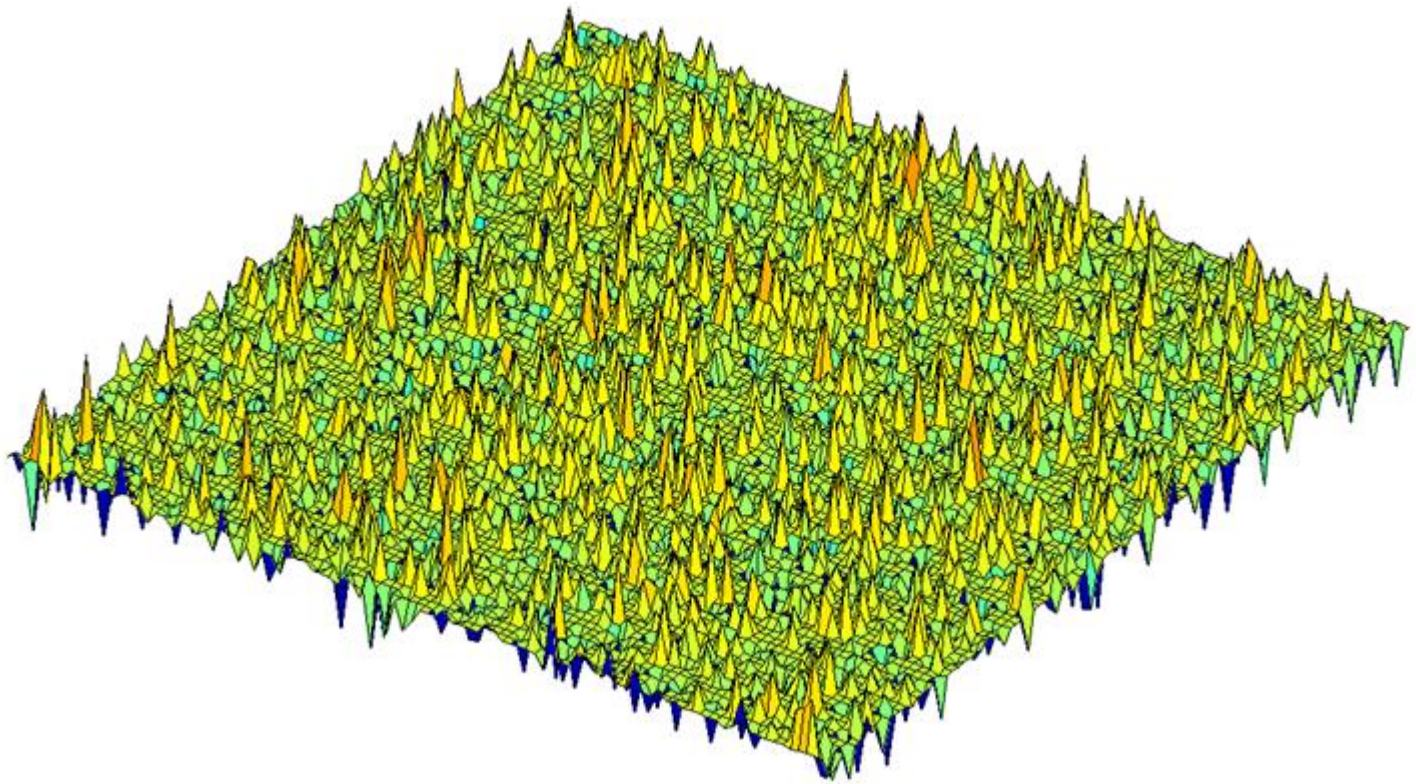
# Schwierigkeiten

- Die Umgebung unseres Beispielproblems ist unebener als die gezeigte Hügellandschaft



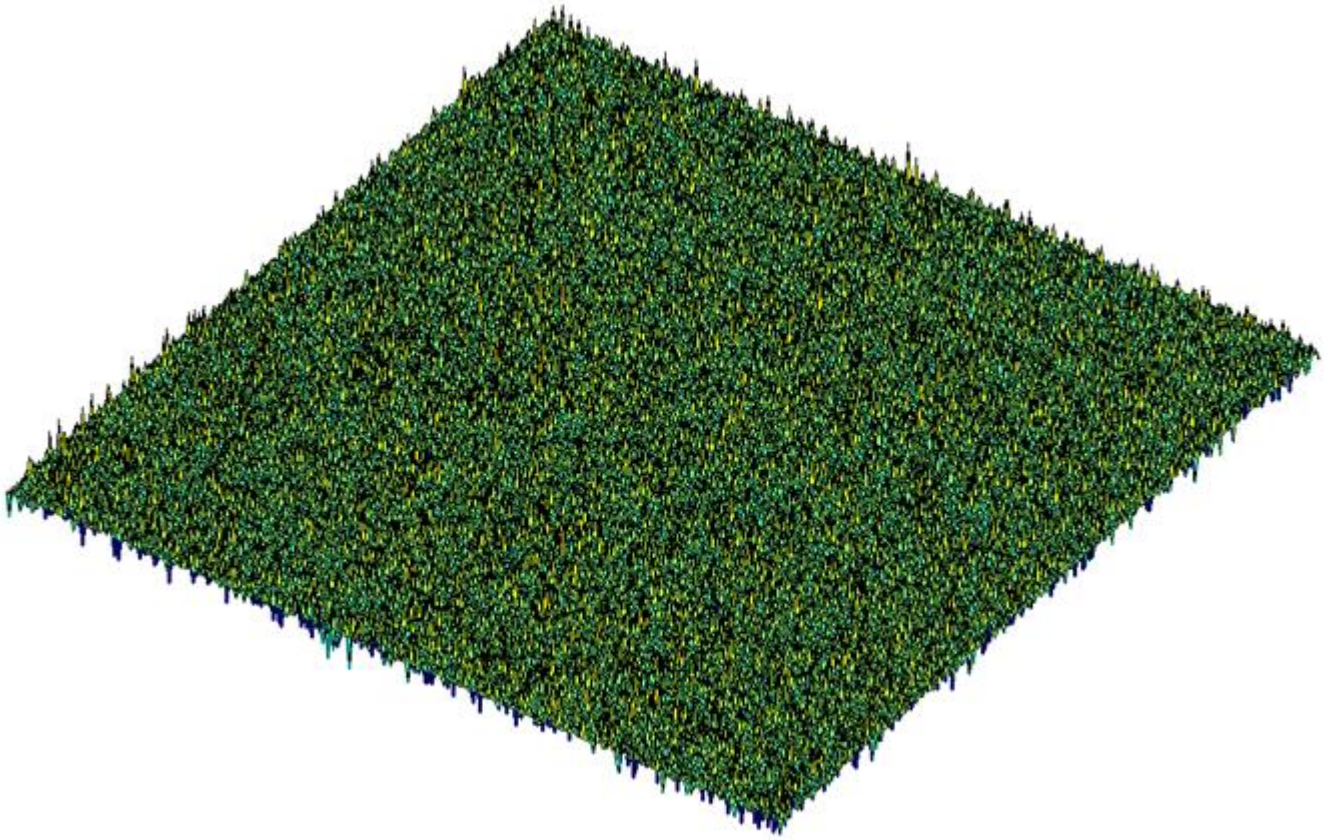
# Schwierigkeiten

- und größer...



# Schwierigkeiten

- ...viel größer...
- Es gibt ca. $16^{175}$ gültige Lösungen



# Simulated Annealing

- Auskühlen
  - Methode, die z.B. in der Industrie angewendet wird, um einen Festkörper (z.B. Metall) hinsichtlich seiner Struktur zu optimieren
  - Festkörper wird erhitzt und langsam abgekühlt
- Erkenntnis aus der Physik/Chemie:
  - Abhängig davon, wie schnell der Festkörper abkühlt, ordnen sich die Atome im Gitter unterschiedlich an
  - Langsames Abkühlen: hohe Qualität
  - Schnelles Abkühlen: niedrige Qualität

# Simulated Annealing

- Algorithmus
  - Wähle eine Richtung zufällig
  - Geht sie bergab, gehe immer zum neuen, besseren Zustand
  - Geht sie bergauf, gehe mit einer gewissen Wahrscheinlichkeit zum neuen, schlechteren Zustand. Wahrscheinlichkeit hängt von der aktuellen Temperatur ab, sowie davon, wie steil bergab gegangen werden soll
  - Reduziere die Wahrscheinlichkeit mit der Zeit (auskühlen)
  - Solange bis Temperatur ‚nahe 0‘
- Wird (unendlich) langsam reduziert, wird das globale Optimum erreicht
- Bei zu schnellem Ausglühen wird (eventuell) ein lokales Optimum gefunden

# Simulated Annealing

- Vorteile
  - Liefert auch in extrem großen Suchräumen gute Ergebnisse
  - Performanz hängt nicht von der Anzahl der Nachbarzustände ab
- Nachteile
  - Lösung verschlechtert sich eventuell am Anfang
  - Nichtdeterministisch: unterschiedliche Ergebnisse bei gleicher Ausgangssituation
  - Konvergiert langsamer
  - Auswahl der Abkühlfunktion muss heuristisch erfolgen
- Verbesserungen
  - Hill Climbing im Anschluss (bei vielen Nachbarzuständen)

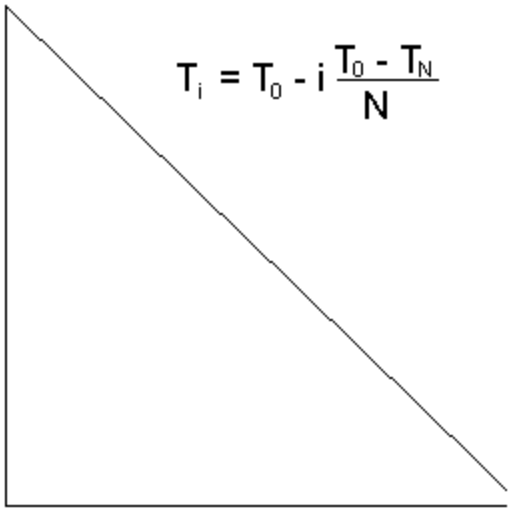
# Simulated Annealing

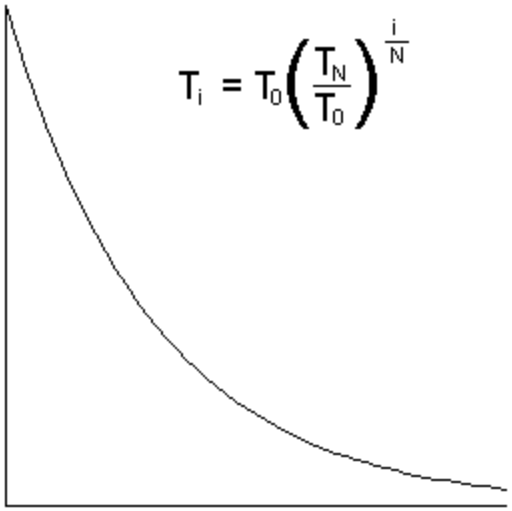
```
while T > 0
{
    reduceTemperature(T, i);
    successorState = randomNeighbor();
    // Zustandsverbesserung neu - alt
    E_delta = E_n - E_a;

    if E_delta > 0
        // besser geht immer
        currentState = successorState;
    else if e^(Edelta/T) > random()
        // schlechter nur mit einer bestimmten
        // Wahrscheinlichkeit abhängig von T und E_delta
        currentState = successorState;
}
```

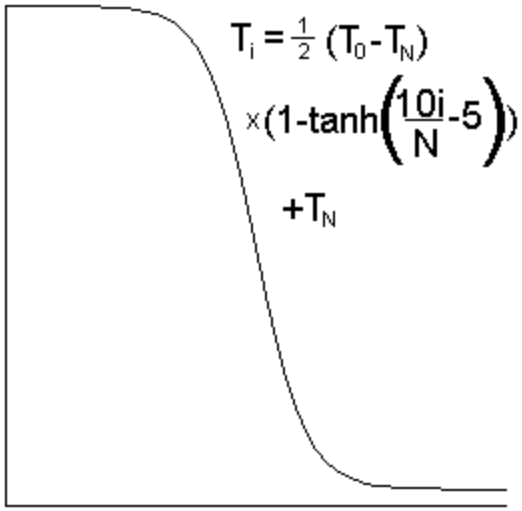
# Simulated Annealing

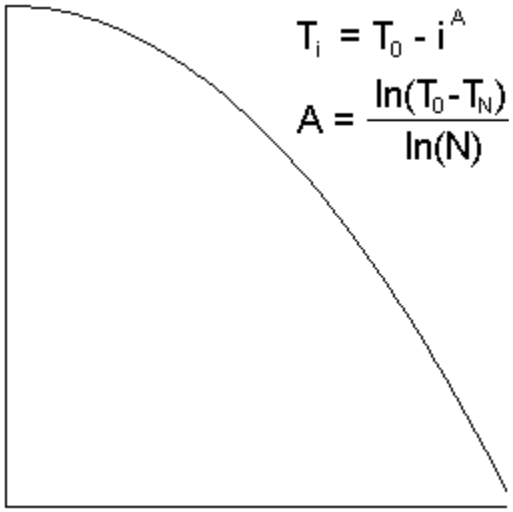
$T_0$: Starttemperatur
$T_N$: Endtemperatur
$T_i$: aktuelle Temperatur
N: Anzahl Zyklen

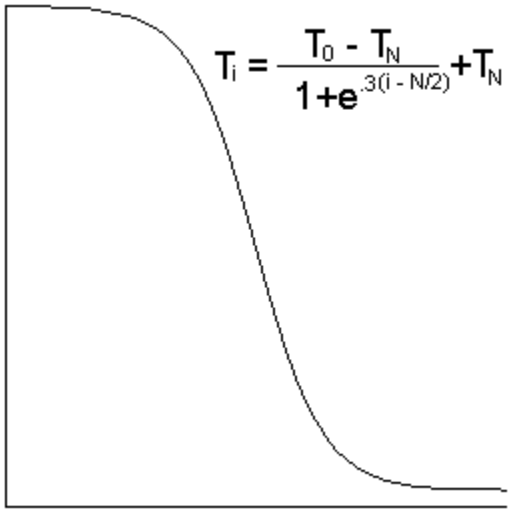
- Temperatur Reduktion

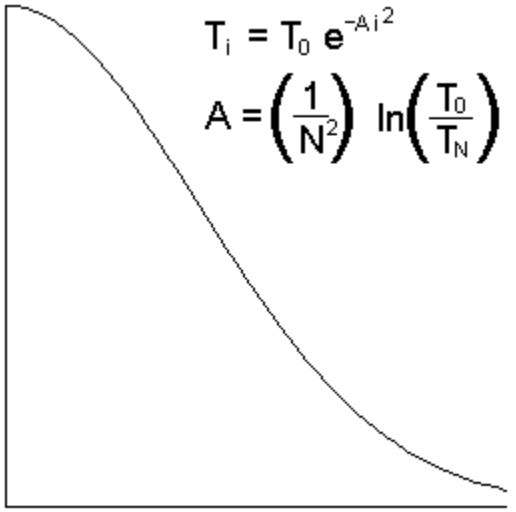


# Ergebnisse

- 175 Aufträge
- 33 Hubs
- Im Mittel 16 Transportpfade je Relation
- 61233 Transportkilometer

| Verfahren | Ergebnis (Mittel) (km) | Ergebnis (Bestes) (km) | Laufzeit (Mittel) (sec) | Ergebnis (Bestbekannt) |
|---|---|---|---|---|
| Hill Climbing | 33.288 | 33.288 | 9 | 27.981 (RW) |
| Tabu Suche | 32.647 | 32.647 | 60 | 32.051 |
| Simulated Annealing | 28.502 | 28.325 | 65 | 27.858 |
| Genetische Algorith. | 30.303 | 29.481 | 61 | 28.110 |

# Zusammenfassung

- Hill climbing ist einfach und robust, bleibt aber in lokalen Minima hängen
- Tabu Suche verbessert Hill climbing durch ein ‚Gedächtnis' und die Möglichkeit ‚bergauf zu gehen'
- Simulated Annealing liefert in sehr großen Problemräumen gute Ergebnisse, ist aber sehr rechenintensiv
- Die Optimierung von Transportnetzwerken ist eine sehr komplexe Aufgabe