# 1 Heuristiken für das Traveling Salesman Problem

Wir betrachten das folgende Problem. Wir wollen einen gegebenen Graphen möglichst schnell so durchlaufen, dass dabei alle Knoten genau einmal besucht werden und wir uns danach wieder am Ausgangspunkt befinden. Als Anwendungsbeispiel kann man sich ein Rechnernetz vorstellen, indem man im Hintergrund einen Prozess laufen lassen möchte, der zyklisch alle Knoten durchläuft und so für einen Informationsaustausch oder eine Synchronisation der Rechner sorgt.
Ein anderes klassisches Anwendungsbeispiel ist das Problem des Handlungsreisenden (engl. *Traveling Salesman Problem*). Ein Vertreter möchte zum Besuch seiner Kunden seine Rundreise möglichst effizient organisieren. Stellt man sich hier die zu besuchenden Städte als Knoten eines Graphen vor und nimmt man als Länge der Kanten zwischen zwei Knoten die Verbindungsdauer um zwischen den zugehörigen Städten zu reisen, so gibt es mindestens eine Rundreise durch alle Städte, bei der keine Stadt mehrfach besucht wird, und bei der die Gesamtdauer der Reise minimiert ist.
Formal definieren wir:

**Definition 1 (Hamiltonkreis)** *Ein Hamiltonkreis in einem Graphen $G = (V, E)$ ist ein Kreis $K$, der alle Knoten von $V$ genau einmal durchläuft.*

Das Traveling Salesman Problem kann nun in der folgenden Weise definiert werden:

**Definition 2 (Traveling Salesman Problem (TSP))** *Gegeben sei ein vollständiger Graph $G = (V, E)$ auf $n$ Knoten, wobei jeder Kante $e \in E$ positive Kosten $c_e$ zugeordnet sind. Gesucht ist ein Hamiltonkreis $K$ in $G$ mit minimalen Kantenkosten.*

Da der Graph vollständig ist, also zwischen je zwei Knoten auch eine Kante existiert, gibt es in jedem Fall einen Hamiltonkreis und man muss sich darauf konzentrieren, einen Hamiltonkreis mit minimalen gesamten Kantenkosten zu finden. In allgemeinen Graphen dagegen ist es sogar bereits schwierig zu entscheiden, ob es überhaupt einen Hamiltonkreis gibt. Genauer hat Richard Karp 1972 bewiesen, dass das Problem *"Gegeben ein Graph $G = (V, E)$, enthält $G$ einen Hamiltonkreis?"* *NP*–vollständig ist (Für die Definition und die Bedeutung des Begriffs *NP*–Vollständigkeit sei auf die Einführungsvorlesungen zur Theoretischen Informatik verwiesen).
Ohne Beweis stellen wir fest, dass auch das Problem des Handlungsreisenden *NP*–vollständig ist[1]. Was ist die Konsequenz: Wir können leider nicht mehr darauf hoffen, einen Algorithmus zu finden, der uns in realistischer Rechenzeit die optimale Lösung ausgeben kann. Wir können nur noch darauf hoffen, Algorithmen zu finden, die uns wenigstens sehr gute Lösungen ausgeben.

[1] Für Kenner der Komplexitätstheorie: Der Beweis erfolgt durch triviale Reduktion auf das Entscheidungsproblem Hamiltonkreis.

## 2 Nearest Neighbour Heuristik (NN) und 2–Opt

Ein einfaches Verfahren zum Auffinden einer Rundreise ist das folgende:

**Nearest Neighbour**

1. Starte mit einem beliebigen Knoten.
2. while (es gibt noch unbesuchte Knoten)
   Wähle unter allen nicht besuchten Knoten denjenigen Knoten $v$,
   der zum aktuellen Knoten die geringste Distanz hat,
   füge ihn in die Rundreise ein und mache den Knoten $v$ zum aktuellen Knoten.
3. Füge den Startknoten an das Ende der Rundreise ein.

Es ist leicht ersichtlich, dass mit einem derartig einfachen Verfahren zwar irgendeine Rundreise, aber im allgemeinen nicht unbedingt eine gute Rundreise gefunden wird. Häufig kann die Rundreise durch Austausch zweier Städte bereits verbessert werden. Angenommen die Knoten $1, ..., n$ werden in der Reihenfolge

$$\pi = \pi_1, ..., \pi_{j-1}, \pi_j, \pi_{j+1}, ..., \pi_{k-1}, \pi_k, \pi_{k+1}, ...\pi_n, \pi_1$$

besucht.
Bei einem so genannten 2–Opt–Schritt werden nun zwei Städte $\pi_j$ und $\pi_k$ ausgewählt und ihre Reihenfolge vertauscht. Die neu entstandene Rundreise lautet dann

$$\pi' = \pi_1, ..., \pi_{j-1}, \pi_k, \pi_{j+1}, ..., \pi_{k-1}, \pi_j, \pi_{k+1}, ...\pi_n, \pi_1$$

Durch wiederholtes Anwenden dieser 2–Opt–Schritte können immer wieder neue Lösungen erzeugt werden und nach Ausführen von "vielen" 2–Opt–Schritten hat man sicherlich eine bedeutend bessere Lösung gefunden, als die durch die einfache Nearest–Neighbour Heuristik erzeugte Lösung.
Ist eine feste Lösung $\pi$ gegeben, so nennt man alle Lösungen, die durch einen 2–Opt–Schritt aus $\pi$ gewonnen werden können, die 2–Opt–*Nachbarschaft* von $\pi$. Analog kann man die $k$–Opt–*Nachbarschaft* von $\pi$ als die Menge aller Rundreisen definieren, die aus $\pi$ durch das simultane Austauschen von $k$ Städten entsteht.
Betrachten wir aber noch einmal einen einzigen 2–Opt–Schritt. Wenn zwei Städte in einer Rundreise $\pi$ vertauscht werden, so kann die resultierende Rundreise kürzer werden, sie kann sich aber auch verlängern. Ist es nun sinnvoll, immer nur diejenigen 2–Opt–Schritte durchzuführen, die die Rundreise tatsächlich verbessern, oder soll man auch mal einen Schritt durchführen, der die Lösung möglicherweise verschlechtert? Dazu betrachten wir einmal das Analogon einer Bergwanderung.
Wir stehen auf der Spitze eines Berges und wollen abwärts ins Tal wandern (vgl. Abb. 1). Wenn wir aber immer nur abwärts gehen würden, dann würden wir möglicherweise

in einem Hochtal enden, ohne in das eigentliche Tal zu gelangen. Wollen wir tatsächlich in das Tal gelangen, so müssen wir möglicherweise so manches mal auch aufwärts gehen, auch wenn wir uns damit zunächst erst einmal verschlechtern sollten.

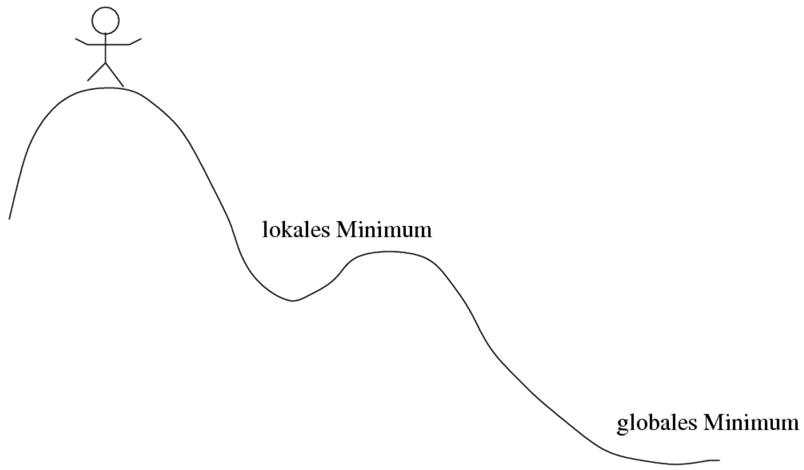


Abbildung 1: Wanderung ins Tal

Ähnlich verhält es sich bei den Optimierungsverfahren. Wenn wir immer nur verbessernde Schritte akzeptieren, so wird irgendwann eine Lösung entstehen, die durch 2–Opt–Schritte nicht mehr verbessert werden kann (lokales Minimum). Das bedeutet aber nicht, dass damit bereits die insgesamt beste Lösung erreicht ist (globales Minimum). Es kann daher durchaus sinnvoll sein, unter gewissen Umständen auch einen 2–Opt–Schritt durchzuführen, der die Lösung verschlechtert, denn die nachfolgenden 2–Opt–Schritte können dann möglicherweise eine Lösung erzeugen, die insgesamt besser ist. Natürlich sollten dabei diejenigen Schritte, die zu einer Verbesserung führen, häufiger ausgeführt werden, als die verschlechternden Schritte. Diese Idee wird beim so genannten *Simulated Annealing* aufgegriffen.

# 3 Simulated Annealing

Unter dem Grundbegriff des Simulated Annealing (SA) ist eine Klasse von Algorithmen gegeben, mit der allgemein Optimierungsprobleme, also auch das TSP angegangen werden können.
Beim SA wird die optimale Lösung innerhalb der Menge aller möglichen Lösungen gesucht, wobei die Wahrscheinlichkeit, in einem lokalen Minimum stecken zubleiben dadurch verringert wird, dass auch Schritte zu schlechteren Lösungen, kontrolliert durch ein randomisiertes Schema, erlaubt werden. Genauer wird ein Schritt von einer Lösung $x$ zu einer um $\Delta$ schlechteren Lösung $x'$ akzeptiert, genau dann, wenn

$$e^{(-\Delta/T)} > R,$$

wobei $T$ ein Kontrollparameter und $R \in [0, 1]$ eine zufällig gezogene Zahl ist. Der Parameter $T$ hat anfangs einen hohen Wert, so dass viele verschlechternde Schritte akzeptiert

werden, und reduziert sich langsam zu einem Wert, bei dem verschlechternde Schritte fast immer ab gelehnt werden.
Das Verfahren heißt übersetzt *simuliertes Ausglühen* und bezieht seinen Namen aus einer Analogie zu einem Prozess in der Thermodynamik. Wird nämlich ein Material über seinen Schmelzpunkt hinaus erwärmt und dann wieder abgekühlt, so dass es sich wieder verfestigt, so hängt die resultierende Struktur von der Abkühlrate ab. Zum Beispiel können große Kristalle durch langsames abkühlen erzeugt werden, wohingegen schnelles Kühlen zu einer hohen Anzahl von Mängeln führt. In der Simulation wird nun die Energieänderung während des Abkühlvorgangs dargestellt, bis das System in einen stabilen Zustand konvergiert. Hier übernimmt dann obiger Kontrollparameter $T$ die Rolle der Temperatur und es wird nun wichtig, ein *Abkühlschema* zu entwickeln, bei dem $T$ verringert wird. Zusätzlich müssen wir Start und Endwerte für $T$ definieren.
Zusammenfassend erhält man den folgenden Algorithmus:

1. Erzeuge eine Startrundreise $x$.
2. Setze $T$ auf eine Initialtemperatur und lege Parameter $\alpha, k, l$ fest.
3. Solange ein Abbruch-Kriterium nicht erfüllt ist,
   (a) Führe einen 2–Opt–Schritt durch und erhalte Lösung $x'$.
   (b) Sei $\Delta$ die Differenz der Längen von $x'$ und $x$.
   Fall 1: $\Delta \leq 0$. Akzeptiere die neue Lösung (setze $x = x'$ ).
   Fall 2: $\Delta > 0$.
      i. Ziehe eine Zufallszahl $R \in [0, 1]$.
      ii. Akzeptiere die neue Lösung, falls $e^{(-\Delta/T)} > R$.
   (c) Sind $k$ 2–Opt–Schritte durchgeführt worden, oder $l$ verbessernde Schritte, so setze $T := \alpha T$ .
4. Gebe die aktuelle Lösung $x$ als Ergebnis aus.

Abbildung 2: Algorithmus Simulated Annealing für TSP

# 4 Wahl der Parameter

Bei der Wahl des Abkühlschemas sind uns viele Freiheiten gelassen und hier liegt die Kunst, solange die einzelnen Parameter einzustellen, bis eine möglichst optimale Lösung resultiert.
Wir betrachten nun die einzelnen einzustellenden Parameter:

**Initialtemperatur ($T$):**

Der Prozess muss in einer Weise starten, dass die meisten – wenn nicht sogar alle – Schritte akzeptiert werden. Das bedeutet, dass die Initialtemperatur hoch sein muss. Bei der Quantifizierung von 'hoch' bedarf es einiger Erfahrung. Hat man diese nicht, so kann man aber einen großen Wert wählen und den Algorithmus kurz laufen lassen und die Akzeptanzrate bestimmen. Ist diese geeignet hoch, dann ist ein guter Startwert gefunden. Was unter geeigneter Akzeptanzrate verstanden wird, hängt vom einzelnen Problem ab, aber es hat sich gezeigt, dass Akzeptanzraten zwischen 40% und 60% gute Richtwerte liefern.

**Abkühlschema ($k$, $l$, $\alpha$):**

Der wohl wichtigste Faktor des Programmes ist das Abkühlschema. Genauer sind die Werte von $k$ und $l$ sowie der Wert $\alpha$ zu bestimmen. Üblicherweise nimmt $\alpha$ einen Wert zwischen 0.9 und 0.99 an. Der Wert $T$ kann aber nicht nur linear durch $T := \alpha T$ verringert werden, sondern auch in anderer Weise, etwa durch setzen von $T := \frac{T}{1+\beta T}$, wobei $\beta$ eine Konstante nahe 0 ist.

**Abbruchkriterium:**

Bei unserem Algorithmus sollen die 2–Opt–Schritte nicht endlos durchgeführt werden, sondern vielmehr irgendwann auch einmal terminieren. Dies kann beispielsweise nach einer vorgegebenen Maximallaufzeit erfolgen, oder wenn eine Folge von $s$ Schritten zu keiner Verbesserung mehr führt, wobei $s$ wiederum ein einzustellender Parameter ist.

Zum Abschluss sei noch bemerkt, dass der 2–Opt oder allgemein der $k$–Opt–Schritt nur eine Möglichkeit ist, aus einer zulässigen Rundreise eine neue Rundreise zu erzeugen. Vielmehr gibt es auch noch eine ganze Reihe weiterer Möglichkeiten, die auch innerhalb des Verfahrens Simulated Annealing verwendet werden können. Auch hier können nur durch viele Experimente die geeigneten Strategien ermittelt werden.