# Evolutionäre Algorithmen: Lösung eines bestimmten Network-Flow-Problems

## Das Single Source Uncapacitated Concave Minimum-Cost Network Flow Problem

Das Single Source Uncapacitated Concave Minimum-Cost Network Flow Problem (SSUC - MCNFP) ist ein Problem bei dem aus einer Quelle mehrere sog. Customer mit Waren versorgt werden müssen. Diese Versorgung geschieht über einen gerichten Graphen bestehend aus Kanten, die über eine mit verschiedenen Kosten versehen sind. Das Ziel ist es, möglichst geringe Gesamt-Kosten zu verursachen während weiterhin alle Customer mit ihren benötigten Waren (Demand) versorgt werden. Dementsprechend ist es ein Minimierungsproblem.

Mathematisch lässt sich das Problem folgendermaßen darstellen:

$$min : \sum_{(i,j)\in A} g_{ij}(x_{ij}, y_{ij})$$

$$s.t. : demand_j \sum_{i|(i,j)\in A} x_{ij}, y_{ij} - \sum_{(j,k)\in A} x_{jk}$$

wobei g die Kostenfunktion ist.

### OR-Library Testdaten

Die Test-daten sind von der Seite "OR-Library" von J. E. Beasley, unter http://people.brunel.ac.uk/~mastjjb/jeb/orlib/netflowccinfo.html ist es möglich eine Datensatz-Sammlung verschieden großer Datensätze herunterzuladen.

Die Daten werden als einfache, unterteilte Liste bereitgestellt und enthalten die Anzahl an Knoten, einen Demand der Knoten und anschließend verschiedene Kosten für die Kanten. Jedoch können die Daten über Funktionen in drei Matrizen umgewandelt werden. Der letzte Knoten hat keinen Demand, dementsprechend kann er als Quelle angeshen werden.

Die Kosten für die Kanten sind in drei verschiedene Komponenten unterteilt, die Komponente ist A, welche variabel ist und für eine nicht-vorhandene Kante den Wert 0 hat. Die zweite Komponente ist B, die ebenfalls variabel ist, aber für eine nicht-vorhandene Kante den Wert 50000000 annimmt. Die letzte Komponente ist C, die unabhängig von dem sog. Flow ist.

Zusätzlich werden drei verschiedene polynomiale konkave Funktionen bereitgestellt, die als Teil einer Güte-Funktion genutzt werden können.

## Hillclimbing im Netzwerk

Der Hillclimbing-Algorithmus basiert auf der Annahme, dass in der Nachbarschaft eines Individuums ein besseres Individuum gefunden werden kann. Zusätzlich basiert er auf der Annahme, dass ausgehend von diesem Individuum ein weiteres gefunden werden kann. In diesem Fall ist das Individuum ein sog. Flow, also wieviele Waren über welche Kante transportiert werden. Dabei wird es in diesem Fall im Programm durch eine Matrix dargestellt.

Das Individuum ist ein struct, also ein Objekt, das einerseits die Flow-Matrix des Individuums beinhaltet, aber außerdem noch den Demand der einzelnen Knoten sowie eine aktuelle Verteilung der waren auf die Knoten, sowie den aktuellen Güte-Wert.

Die Flow-Matrix des Individuums wird durch eine Funktion initialisiert, die verschiedene Sachen wie das Vorhandensein von Kanten, den sog. Demand und auch die Quellenkapazität berücksichtigt. Die Quellenkapazität wird dabei über die Summe der Demands berechnet, da sie nur den Demand befriedigen muss. Die Position der Kanten wird über die Kosten-Matrizen berechnet. Der Demand wird als negatives Lager berücksichtigt, der Überschuss dient anschließend als Obergrenze für eine Zufallsfunktion, die die restlichen Waren weiterverteilt. Anschließend wird der Güte-Wert berechnet.

Der Nachbar wird über eine Funtkion gefunden, die genau die gleichen Faktoren berücksichtigt. Jedoch ist der Anfangspunkt nicht die Quelle, sondern eine zufällig ausgewählte Kante. Anschließend werden alle folgenden Punkte über eine Zufallsfunktion mit dem Überschuss als Obergrenze neu verteilt. Dadurch sind zwar Nachbarn relativ weit voneinander entfernt, aber die Chance tatsächliche Verbesserungen zu erreichen ist deutlich erhöht. Statt Güte-Werten, die konstant im neun- bis zehn-stelligen Bereich sind, wurden dadurch auch Werte im mittleren sieben-stelligen Bereich gefunden.

Wie es beim Hillclimbing üblich ist, werden die Gütewerte der Individuen verglichen und für das Individuum mit dem besseren, also kleineren, Gütewert wird ein neuer Nachbar gefunden.

## Evolution im Netzwerk