**Netzwerkflussproblem**

Von Alexander M. Westphal, Klaus Riedl, Robert Taglauer

Inhaltsverzeichnis

[Einleitung (Motivation/Aufgabenstellung, Ziele, Aufbau, Umfeld) 3](#_Toc11862363)

[Theoretische Grundlagen 3](#_Toc11862364)

[Was ist die Graphentheorie 3](#_Toc11862365)

[Was ist das Netzwerk 4](#_Toc11862366)

[Problembeschreibung, z.B. Projektrisiko oder Fallstudie 5](#_Toc11862367)

[Konzept/Entwurf 5](#_Toc11862368)

[Praktischer Teil 5](#_Toc11862369)

[Zusammenfassung + Fazit + Ausblick 5](#_Toc11862370)

[Literaturverzeichnis 5](#_Toc11862371)

[Bücher und Journals 5](#_Toc11862372)

[Internetquellen 5](#_Toc11862373)

# Einleitung (Motivation/Aufgabenstellung, Ziele, Aufbau, Umfeld)

Man stelle sich folgendes Scenario vor: Sie sind als Logistiker in einem Unternehmen angestellt, der Netz AG. Die Netz AG produziert Waren in ihren Fertigungshallen. Nun besteht ihre Aufgabe als Logistiker darin einen geeigneten Weg von A nach B zu finden. Zwischen Punkt A und B befinden sich jedoch noch weitere Punkte an denen sie halt machen müssen, da zum Beispiel das Transportmittel gewechselt wird. Nun ist es so das verschiedene Transportmittel unterschiedlich viel Kosten verursachen.

Womit eine weitere Größe Einzug in unser Problem gefunden hat und es wird deutlicher welchen Schwierigkeiten sich ergeben. So wird weiterhin offensichtlicher das wir hier ein Optimierungsproblem lösen wollen. Da der Transport einer Ware von A nach B entweder zu maximaler Menge oder minimalen Kosten führen sollte. Diese Art von Problem kann auf verschiedenen Art und Weisen beschrieben werden. In letzter Konsequenz reduziert es sich dennoch immer auf die optimale Auslastung beziehungsweise Minimierung der Kosten oder Maximierung der transportieren Menge. Deshalb werden wir im Folgenden näher betrachten wie es möglich ist eine optimale Verteilung beziehungsweise Auslastung unseres Netzwerks zu finden. Hierzu wird im Folgenden eine Vertiefung des zuvor beschriebenen Problems stattfinden sowie eine mathematische Modellierung. Ebenfalls wird ein Beispiel Netzwerk in Python implementiert und näher erläutert.

# Theoretische Grundlagen

## Was ist die Graphentheorie

Um sich dem Problem des Netzwerkflusses anzunähern macht es Sinn zu verdeutlichen in welchem Bereich der Mathematik Netzwerke und Flüsse zu finden sind. Somit sollte zunächst einmal die Definition der Wortes Graphentheorie näher Betrachtet werden. Unter dem Terminus Graphentheorie versteht man ein Teilbereich der Mathematik welcher sich die Eigenschaften sowie die Beziehung von Graphen näher erschließt. Besonders hervorzuheben ist hierbei noch das jenes Themengebiet nicht nur in der Mathematik anzutreffen ist. Besonders in der Informatik findet sich das Thema Graphen beziehungsweise Algorithmen in Verbindung mit Graphen wieder und hat somit einen hohen Bezug zur Praxis.

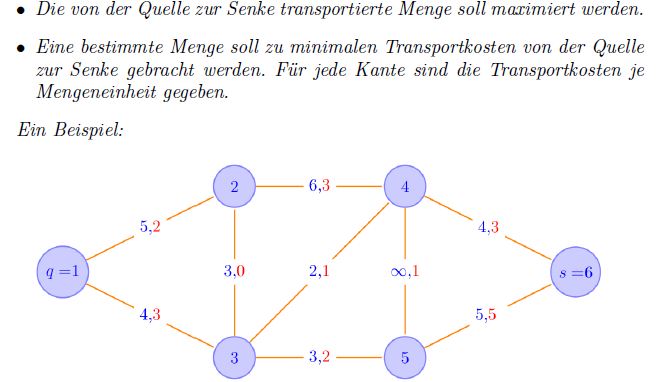
## Was ist ein Netzwerk

Nun nach dem zuvor verdeutlicht wurde was ein Graph ist sollte ebenfalls

# Problembeschreibung, z.B. Projektrisiko oder Fallstudie

# Konzept/Entwurf

# Praktischer Teil

Gegeben:

1.0 Mathematische Bedingungen:

* Transportproblem:
  + Menge, die durch Rohre geschickt wird, kann nicht kleiner als 0 sein.
  + Menge, kann nicht größer sein als die maximale Transport Kapazität.
  + Die Menge die ein Punkt bekommt muss gleich sein der Menge die er verschickt (Ausnahme q,s)

**2.0 Transportproblem: mit Kapazitätsbeschrenkung**

* Bildung der Inzidenzmatrix

# **Definition: Eine Inzidenzmatrix eines Graphen ist eine Matrix welche die Beziehungen zwischen den Knoten und Kanten Speichert. Da wir es hier mit einen Gerichteten Graphen zutun haben haben wir 3 mögliche eintragungspunkte**

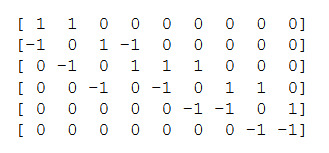
# **1 -> Anfangsknoten**

# **0 - > weder Anfangsknoten noch Endknoten**

# **-1 -> Endknoten**

Daraus ergibt sich dann die folgende Matrix

Die Reihen stehen für die Knoten und die Spalten für die Beziehungen zwischen zwei Knoten. Zum Beispiel Row[1],Column[1] ist der Knoten 1 ein Anfangsknoten bei der Beziehung zwischen Knoten 1 und Knoten 2 ? -> Ja ist er daher ist der Eintrag 1.

 x1,2 x1,3 x2,4 x3,2 x3,4 x3,5 x4,5 x4,6 x5,6

1

2

3

4

5

6

* Bildung Nebenbedingungen:

Unsere Nebenbedingungen ergeben Sich aus 1.0

* + 0 ≤ x
  + x ≥ u |u= maximale Transport Kapazität
  + Ax=b |b=Control Vektor erster Eintrag Zielmenge und letzter Eintrag negative

Zielmenge somit geben wir an was bei Punkt 1 Reinkommt und bei

Punkt 6 Ankommen soll

* Minimierung:

Wir wollen ja die Niedrigsten kosten für eine bestimmte Menge erfahren die wir von der Quelle zur Sekante senden.

∑ xji\*cji ji Element aus allen verfügbaren Kanten

Nun sehen wir das wir ein Netzwerkflussproblem in Form von

haben.

Lineare Optimierung:

Nun als letzten Schritt müssen wir die von uns Aufgestellten Gleichung noch lösen.

Hierzu verwendeten wir SciPy (Scientific Computing Tools for Python), da es hier die Funktion LinProg() gibt die wir bereits aus der Optimization Toolbox aus Mathlab kennen.

Linprog:

scipy.optimize.linprog(c, A\_ub=None, b\_ub=None, A\_eq=None, b\_eq=None, bounds=None)

hierbei können wir der Funktion einfach unsere Werte mitgeben und erhalten ein Optimiertes Ergebnis.

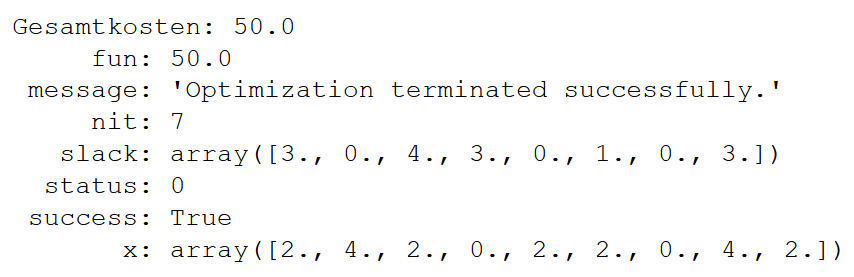
In unseren Fall haben wir ja folgendes gegeben:

c = Vektor von unseren Kosten die wir Optimieren wollen

A\_eq=Inzidenzmatrix

b\_eq=Control Vektor

bounds = lowerbound -> nullvektor, upperbound->Vektor mit den Kapazitäten

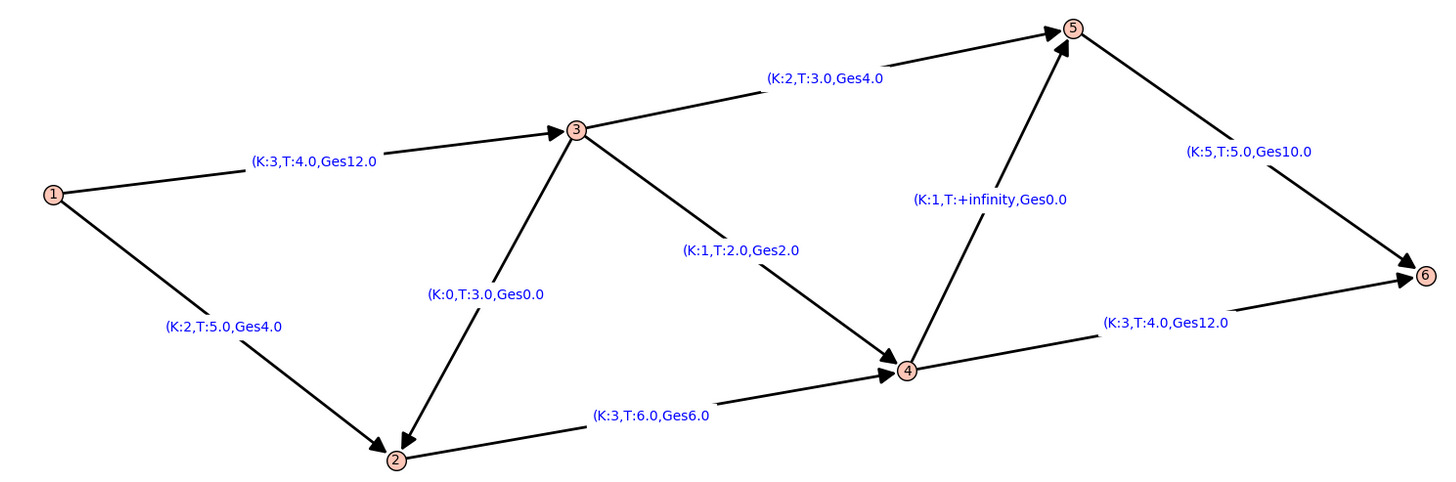
Als Return Wert bekommen wir nun:

# 

Lösung:

Wenn wir eine Menge von 6 durch unseren Netzwerkfluss schicken wollen und auch wieder 6 an der Sekante ankommen sollen haben, ist der günstigste Weg insgesamt 50 Kosten teuer mit der folgenden Lösung.

x1,2=2|x1,3=4|x2,4=2|x3,2=0|x3,4=2|x3,5=2|x4,5=0|x4,6=4|x5,6=2



**2.1 Transportproblem: Ohne Kapazitätsbeschrenkung**

Ohne Kapazitätsbeschrenkung suchen wir den Günstigsten Pfad. Hierzu haben wir uns überlegt das wir diesen Pfad bekommen indem wir einfach davon ausgehen das wir in Unseren gegebenen Netzwerkfluss nur die Menge 1 von der quelle zur Sekante transportieren und dies wie in 2.0 beschrieben auflösen mit der Änderung das man alle Einträge des Kapazitätsvektors mit einer 1 ersetzt und somit sicherstellt das die Menge 1 theoretisch über jede Kante kann.

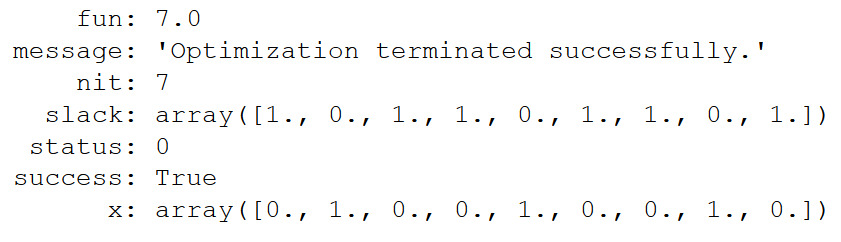
scipy.optimize.linprog(c, A\_ub=None, b\_ub=None, A\_eq=None, b\_eq=None, bounds=None)

c = Vektor von unseren Kosten die wir Optimieren wollen

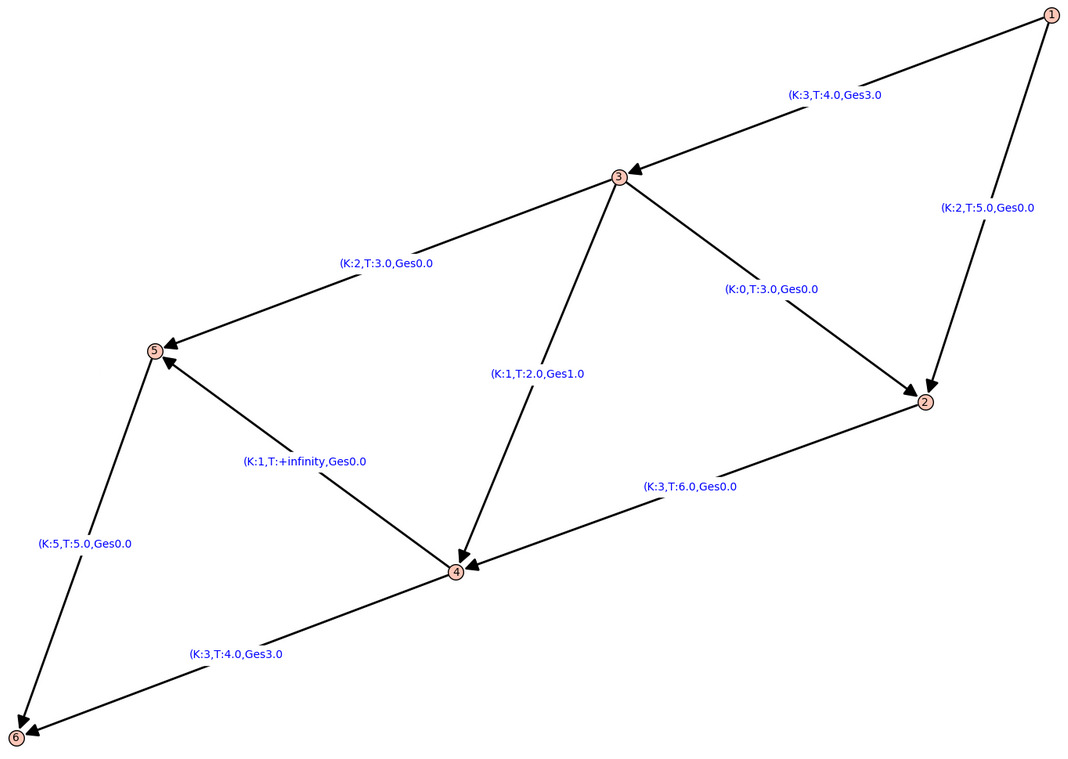
A\_eq=Inzidenzmatrix

b\_eq=Control Vektor (Hier immer ersten Eintrag 1 und letzten Eintrag -1)

bounds = lowerbound -> nullvektor, upperbound->Vektor voll mit einsern



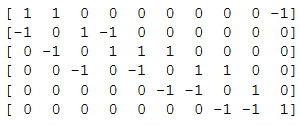
Der günstige Weg verursacht pro Menge Kosten von 7 dank der folgenden Lösung.

x1,2=0|x1,3=1|x2,4=0|x3,2=0|x3,4=1|x3,5=0|x4,5=0|x4,6=1|x5,6=0

**3.0 Maximalerfluss:**

Überlegung ob man ein Maxfluss Problem auch als Kostenminimaler Fluss Problem umschreiben kann.

Als erstes haben wir und überlegt einen Kante von Senke zu Quelle zu machen damit man einen unendlichen Input von der Quelle hat. Hierbei muss man nun auch die Inzidenzmatrix anpassen. Die Kosten werden auf null gesetzt.



Normal hätte man nun als Bedingung das wir den x6,1 maximieren, da wir dann jedoch versuchen wollen unser Problem umzuschreiben dachten wir das wir jetzt pro Mengen Übertragung nicht ins positive sondern ins negative gehen. Somit können wir den Wert der von der Sekante zum Quellen Input kommt und zurück zur Sekante damit Maximieren wenn wir –(x6,1) Minimieren.

3.1 Mathematische Bedingungen:

* Transportproblem:
  + Menge, die durch Rohre geschickt wird, kann nicht kleiner als 0 sein.
  + Menge, kann nicht größer sein als die maximale Transport Kapazität.
  + Menge, die von der Sekante zur Quelle kommt muss gleich sein wie die von der Quelle zur Sekante
* Bildung Nebenbedingungen:

Unsere Nebenbedingungen ergeben Sich aus 3.1

* + 0 ≤ x
  + x ≥ u |u= maximale Transport Kapazität
  + Ax=b |b= nullvektor um sicher zu stellen das auch das was von der Sekante zur

Quelle fließt auch wieder bei der Sekante ankommt

Nun haben wir wieder die gleiche Form wie beim Transportkostenproblem.



Nun können wir wieder die linprog Funktion verwenden.

scipy.optimize.linprog(c, A\_ub=None, b\_ub=None, A\_eq=None, b\_eq=None, bounds=None)

c = nullvektor bei dem der letzte Eintrag -1 ist (x6,1)

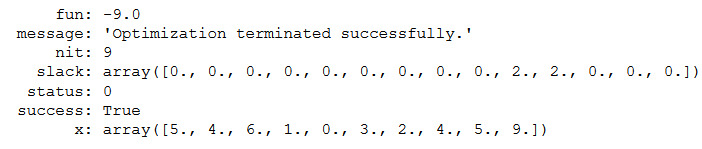
A\_eq=Inzidenzmatrix

b\_eq=Control Vektor (hier jetzt ein nullvektor-> da wir keinen festen input haben )

bounds =

lowerbound -> nullvektor mit letzten Eintrag von -np.inf (wg. Sekante zu Quelle )

upperbound->Vektor den Transportkapazitäten



Hierbei ist unser Ergebnis negativ da wir nicht nach x6,1 Maximiert haben sondern nach –(x6,1) Minimiert um nun das Korrekte Vorzeichen zu erhalten müssen wir fun\*-1 rechnen.

Den Maximalen Fluss von 9 erhalten wir mit der folgenden Lösung.

x1,2=5|x1,3=4|x2,4=6|x3,2=1|x3,4=0|x3,5=3|x4,5=2|x4,6=4|x5,6=5|

# Zusammenfassung + Fazit + Ausblick

# Literaturverzeichnis

## Bücher und Journals

## Internetquellen