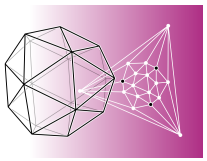
Kulturgut Mathematik und Logistik

Es kann und muss nicht immer alles optimal sein

Christoph Helmberg

TU Chemnitz
Algorithmische und Diskrete Mathematik
(Frank Fischer, Stefan Röhl)

- Einleitung
- Graphenmodelle und kürzeste Wege
- Rundreisen
- Algorithmische Komplexität, ein Milleniums-Problem
- Fluss- und Transportprobleme
- Betriebsplanerstellung
- Schlussbemerkungen



www.ansichtskartenversand.com/ak/100-Ansichtskarten-AK-Geschichten/36-Loktransport-in-Chemnitz

Logistik (Wikipedia 15.10.2024)

Die Logistik ist sowohl eine interdisziplinäre Wissenschaft als auch ein Wirtschaftszweig oder eine betriebliche Funktion in Wirtschaftssubjekten, die sich mit der **Planung, Steuerung, Optimierung** und Durchführung von **Güter-, Informations- und Personenströmen** befasst.

... Zu diesen Strömen zählt das Transportieren, Umschlagen, Lagern (TUL-Prozess), Kommissionieren, Sortieren, Verpacken und Verteilen. Bei Stückgütern spricht man vom Materialfluss. ...

Logistik (Wikipedia 15.10.2024)

Die Logistik ist sowohl eine interdisziplinäre Wissenschaft als auch ein Wirtschaftszweig oder eine betriebliche Funktion in Wirtschaftssubjekten, die sich mit der **Planung, Steuerung, Optimierung** und Durchführung von **Güter-, Informations- und Personenströmen** befasst.

... Zu diesen Strömen zählt das Transportieren, Umschlagen, Lagern (TUL-Prozess), Kommissionieren, Sortieren, Verpacken und Verteilen. Bei Stückgütern spricht man vom Materialfluss. ...

Etymologie

... Das Wort Logistik wurde erstmals 1830 vom französischen Militärtheoretiker Antoine-Henri Jomini verwendet. Er ... leitet ‚(l'art) logistique' (deutsch „die Kunst, Truppen einzuquartieren") vom französischen Wort ‚logis' (‚Unterkunft') ab, das wiederum auf das urgermanische *laubja- (‚Obdach') zurückgeht. Die historische Herleitung des Wortes zeigt den Bezug zum militärischen Nachschubwesen auf, dem die Logistik entspringt. Die Wortbildung verläuft analog und ist homonym zum altgriechischen Wort $\lambda o \gamma \iota \sigma \tau \iota \kappa \acute{\eta}$ (‚praktische Rechenkunst'), geht aber auf eine germanische Wurzel zurück.

Operations Research (Wikipedia 15.10.2024)

Unter Operations Research (US-engl.) oder Operational Research (GB-engl.), kurz OR, im Deutschen selten auch Operationsforschung, Unternehmensplanung oder Optimierungsrechnung, wird allgemein die Entwicklung und der Einsatz quantitativer Modelle und Methoden zur Entscheidungsunterstützung verstanden. Operations Research ist geprägt durch die **Zusammenarbeit von Angewandter Mathematik (insbes. Mathematische Optimierung), Wirtschaftswissenschaften und Informatik**.

Operations Research (Wikipedia 15.10.2024)

Unter Operations Research (US-engl.) oder Operational Research (GB-engl.), kurz OR, im Deutschen selten auch Operationsforschung, Unternehmensplanung oder Optimierungsrechnung, wird allgemein die Entwicklung und der Einsatz quantitativer Modelle und Methoden zur Entscheidungsunterstützung verstanden. Operations Research ist geprägt durch die **Zusammenarbeit von Angewandter Mathematik (insbes. Mathematische Optimierung), Wirtschaftswissenschaften und Informatik**.

Geschichte

Der Begriff Operational Research stammt ursprünglich aus dem Militärwesen ...

Operations Research (Wikipedia 15.10.2024)

Unter Operations Research (US-engl.) oder Operational Research (GB-engl.), kurz OR, im Deutschen selten auch Operationsforschung, Unternehmensplanung oder Optimierungsrechnung, wird allgemein die Entwicklung und der Einsatz quantitativer Modelle und Methoden zur Entscheidungsunterstützung verstanden. Operations Research ist geprägt durch die **Zusammenarbeit von Angewandter Mathematik (insbes. Mathematische Optimierung), Wirtschaftswissenschaften und Informatik**.

Geschichte

Der Begriff Operational Research stammt ursprünglich aus dem Militärwesen . . .

. . . An Kritik wurde vorgebracht, dass Operations Research zu einseitig mathematisch orientiert sei und zu wenig Computeranwendungen erforscht habe. Auch wurden die Modelle als unterkomplex kritisiert.

Reale Probleme sind hoch komplex

Typische Teilprobleme etwa in Zug-/Flugplanung:

- Bedarfsprognosen für Personen-/Güterströme
  (wie viele wollen wann von wo nach wo?)
- Linienplanung
  (welche Routen wie oft, Anschlussbedingungen, . . .)
- Fahrplan-/Betriebsplanerstellung
  (Bahnhöfe/Flughäfen, Umstiegszeiten, Wegebelegung, . . .)
- Umlaufplanung für das Material
  (Zuweisung, Rundläufe, Wartungszeiten, . . .)
- Umlaufplanung für das Personal
  (Zuweisung, Rundläufe, Arbeits-, Ruhezeiten, . . .)

Es gibt starke Wechselwirkungen zwischen den Teilproblemen.
Eine übergreifende „optimale“ Planung ist derzeit unerreichbar.
Keiner der Schritte erfolgt heute ohne mathematische Hilfsmittel.

Übersicht

Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

Übersicht

Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

- Graphenmodelle und kürzeste Wege
- Rundreisen
- Algorithmische Komplexität, ein Milleniums-Problem
- Fluss- und Transportprobleme
- Betriebsplanerstellung
- Schlussbemerkungen

Übersicht

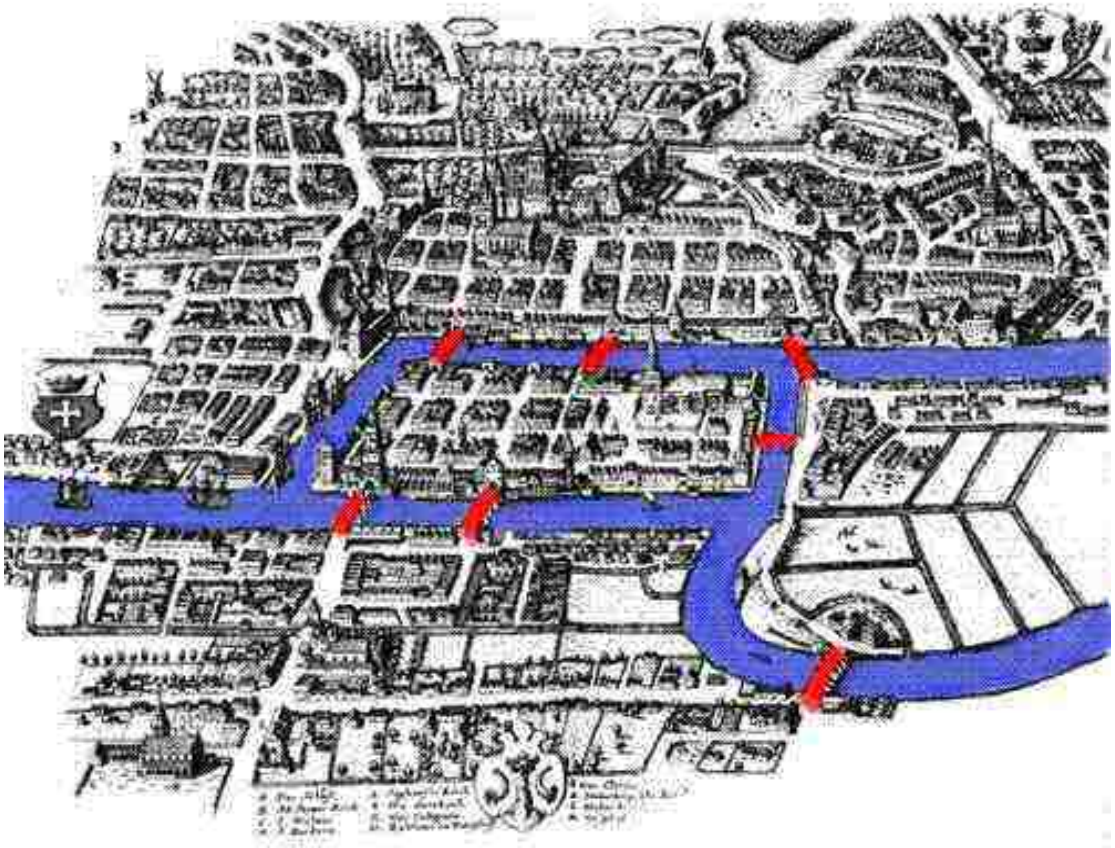
Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

- **Graphenmodelle und kürzeste Wege**
- Rundreisen
- Algorithmische Komplexität, ein Milleniums-Problem
- Fluss- und Transportprobleme
- Betriebsplanerstellung
- Schlussbemerkungen

Das Königsberger Brücken-Problem (Euler 1736)

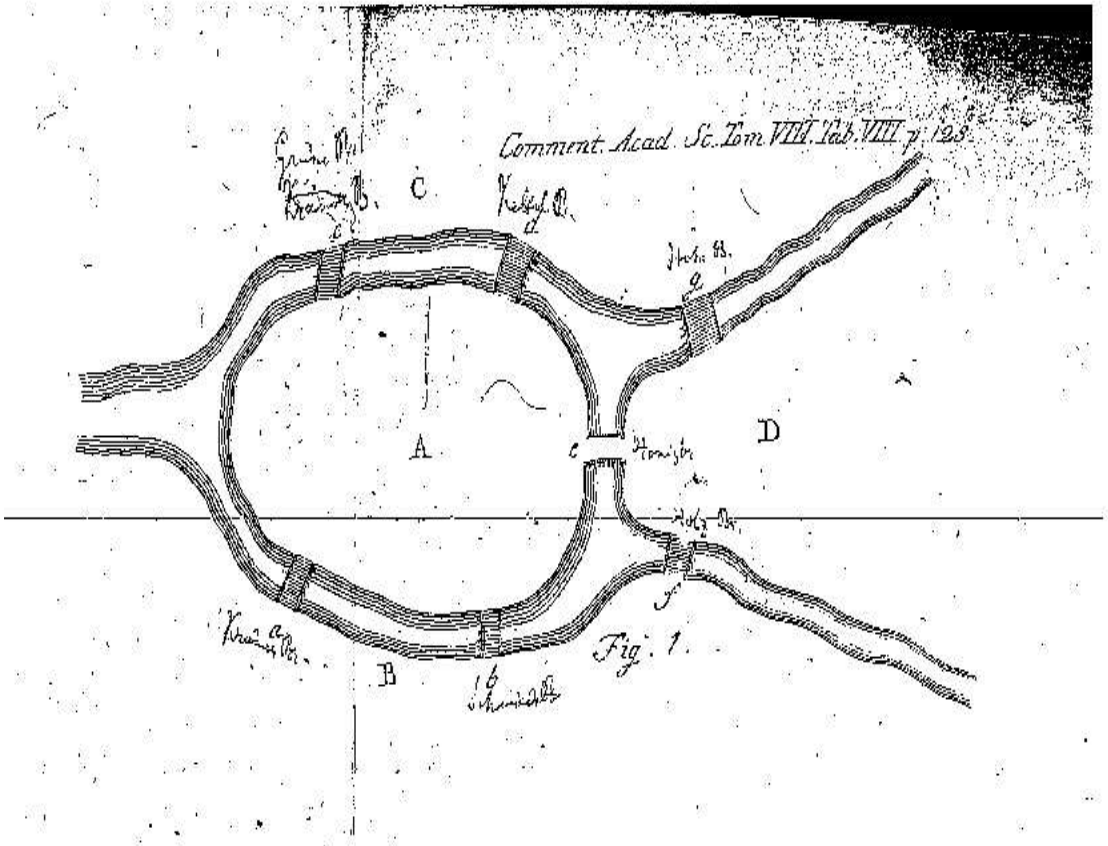


Leonhard Euler

1707 - 1783

Finde einen Spaziergang, der jede Brücke genau einmal quert und am Ausgangsort endet.

Das Königsberger Brücken-Problem (Euler 1736)

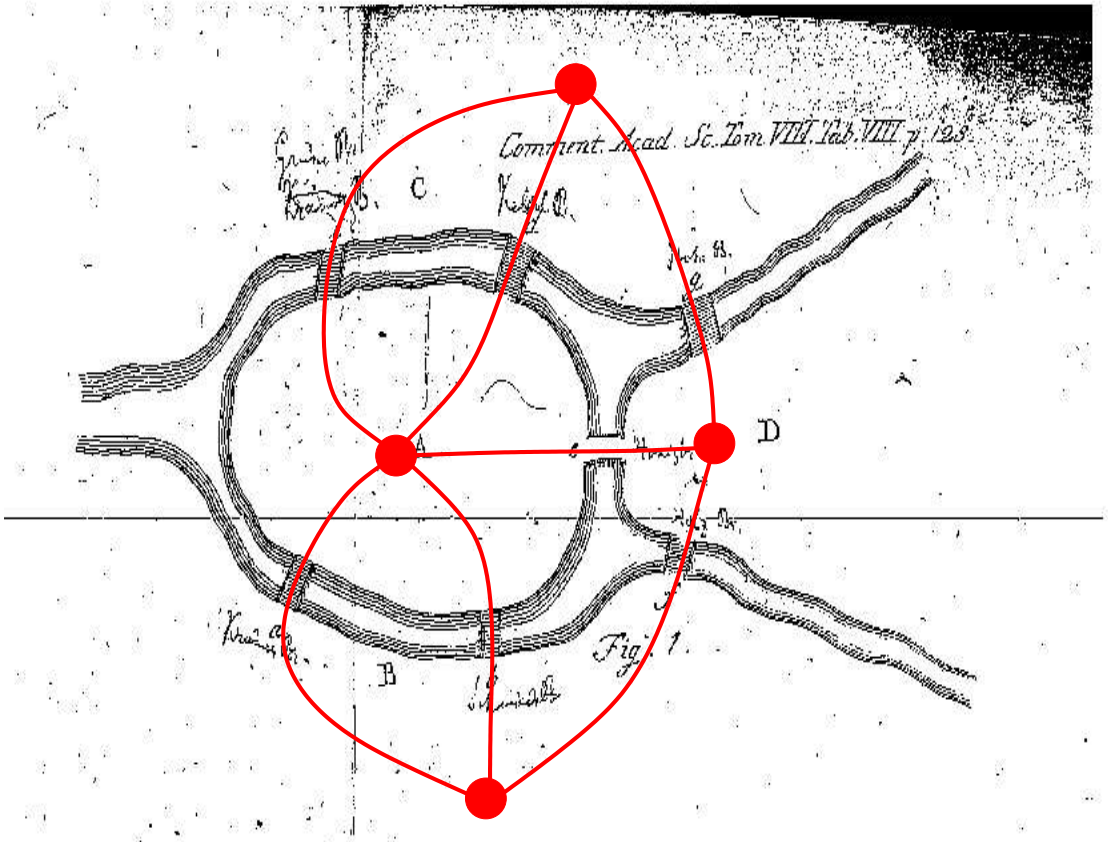


Leonhard Euler

1707 - 1783

Finde einen Spaziergang, der jede Brücke genau einmal quert und am Ausgangsort endet.

Das Königsberger Brücken-Problem (Euler 1736)

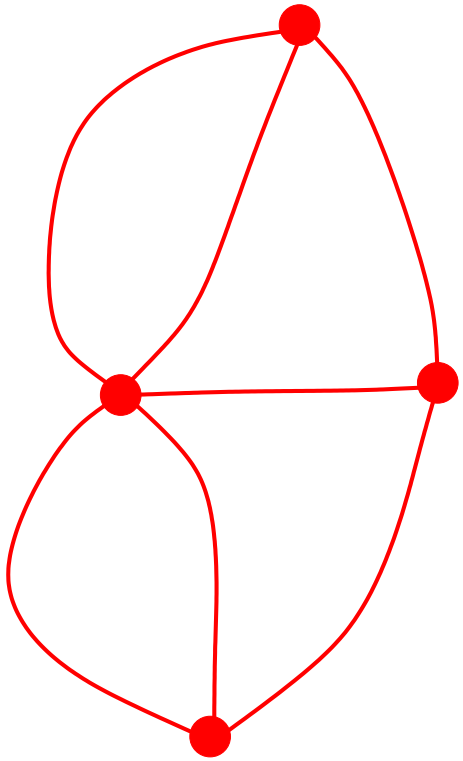


Leonhard Euler

1707 - 1783

Finde einen Spaziergang, der jede Brücke genau einmal quert und am Ausgangsort endet.

Das Königsberger Brücken-Problem (Euler 1736)

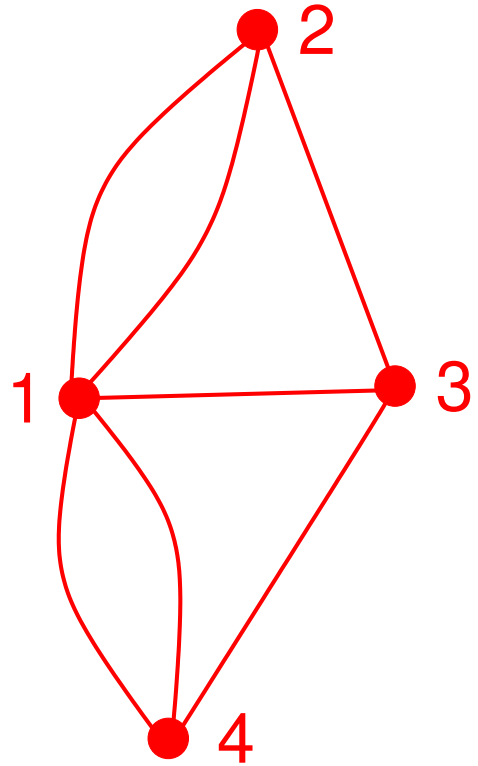


Leonhard Euler

1707 - 1783

Finde einen Spaziergang, der jede Brücke genau einmal quert und am Ausgangsort endet.
Gegeben ein *Graph*, bestehend aus Knoten (•) und Kanten (—), enthält dieser einen Rundweg, der jede Kante genau einmal quert? Ist er „Eulersch“?

Das Königsberger Brücken-Problem (Euler 1736)



Daten:

1 2
1 2
1 3
1 4
1 4
2 3
3 4

Leonhard Euler

1707 - 1783

Finde einen Spaziergang, der jede Brücke genau einmal quert und am Ausgangsort endet.

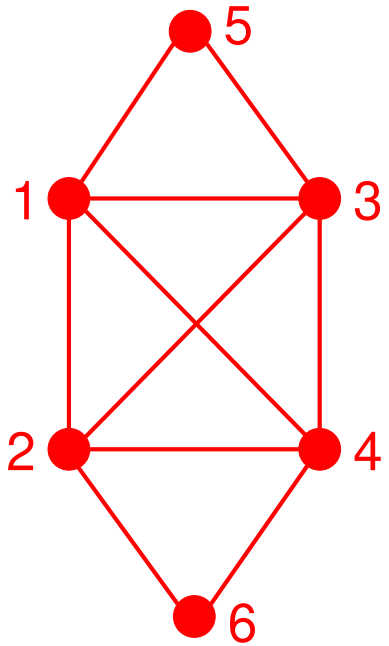
Gegeben ein *Graph*, bestehend aus Knoten (•) und Kanten (—), enthält dieser einen Rundweg, der jede Kante genau einmal quert? Ist er „Eulersch“?

Geometrie auf Paarbeziehungen reduziert!

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

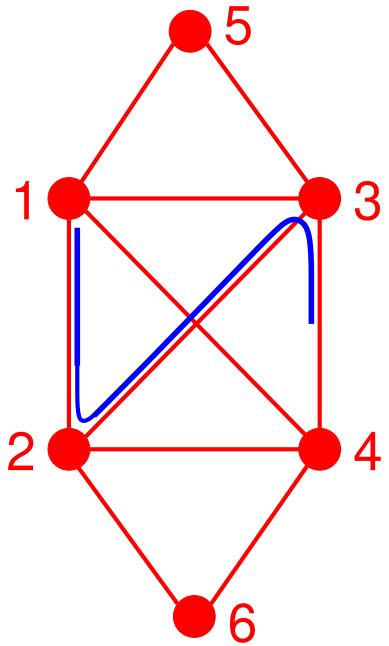
Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.



1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

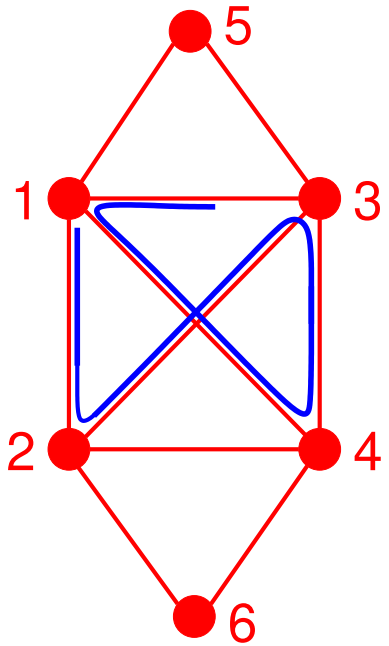
Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.



1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

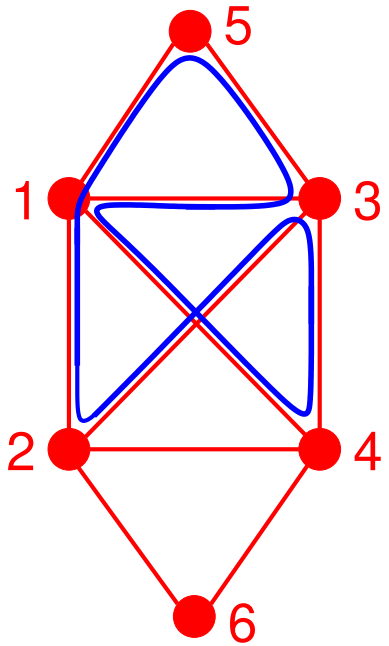
Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.



1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

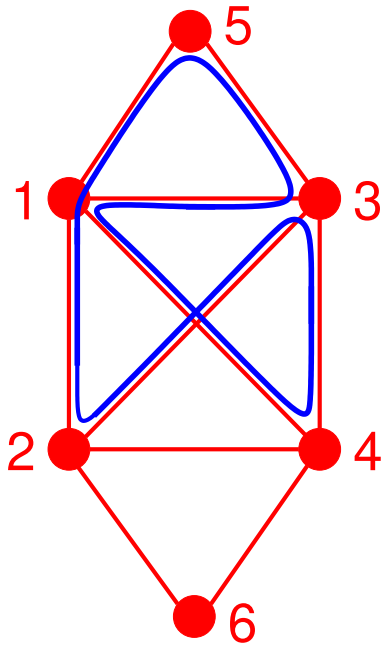
Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.



1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

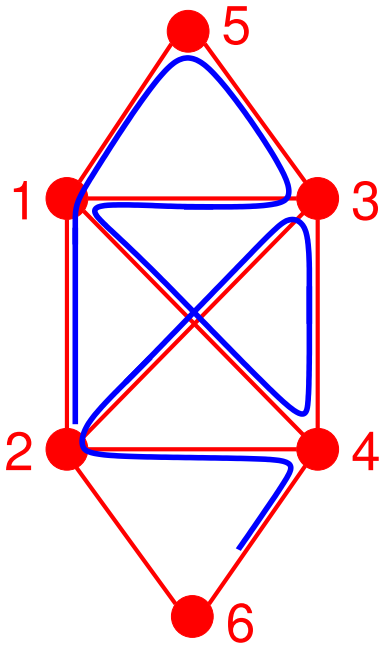
Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.
Sind dann noch nicht alle Kanten überschritten, gibt es einen Knoten auf dem Spaziergang, von dem aus man fortsetzen kann.



1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

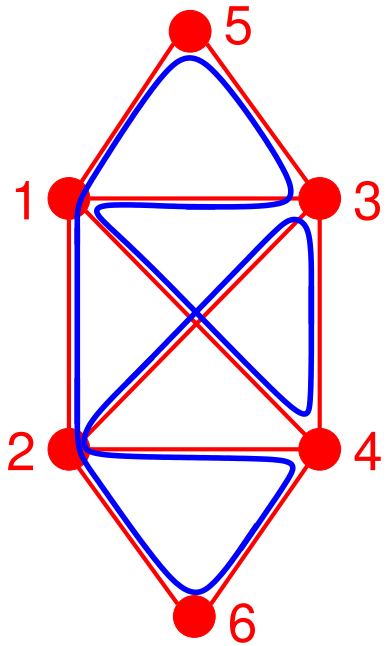
Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.
Sind dann noch nicht alle Kanten überschritten, gibt es einen Knoten auf dem Spaziergang, von dem aus man fortsetzen kann.



1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Satz (Euler 1736) *Ein zusammenhängender Graph ist genau dann Eulersch, wenn jeder Knoten geraden Grad hat.*
(Grad eines Knotens = Anzahl der Kanten, die in ihm enden.)

Einfacher **Algorithmus:**
Starte in einem Knoten und gehe jeweils über eine neue Kante zum nächsten, bis vom neuen Knoten keine neue Kante mehr weiterführt.
Sind dann noch nicht alle Kanten überschritten, gibt es einen Knoten auf dem Spaziergang, von dem aus man fortsetzen kann.

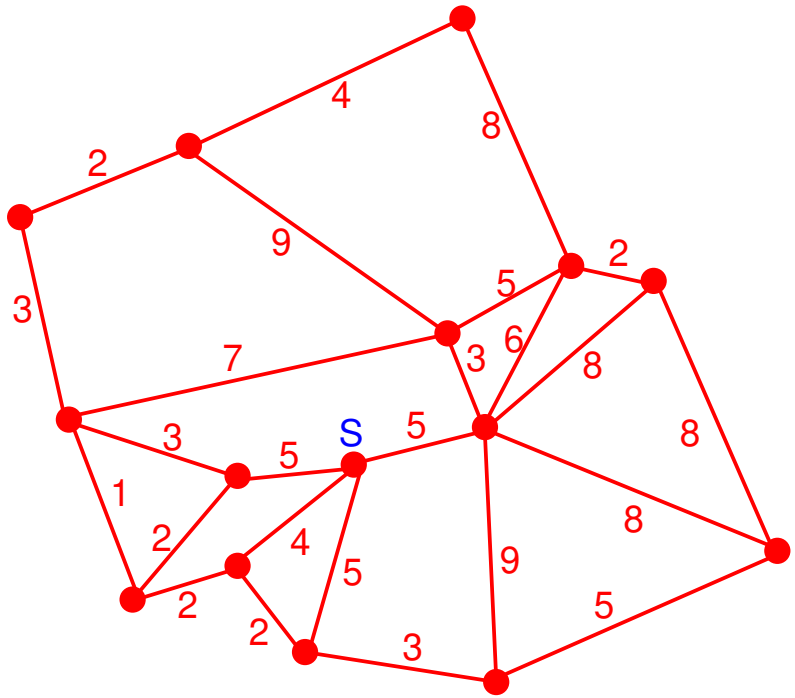


1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 6

Kürzeste Wege in Graphen

Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

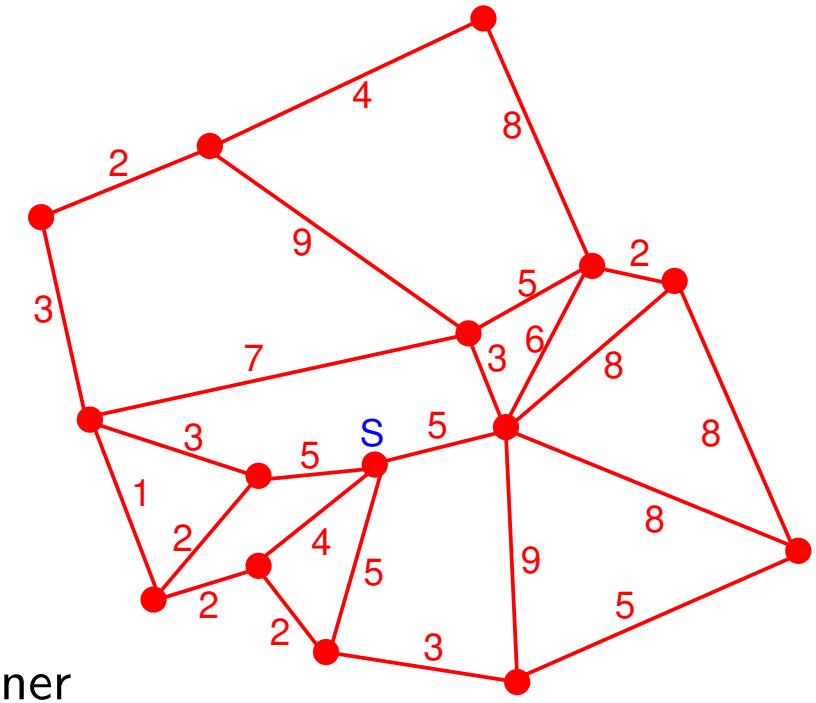
Gesucht: die kürzesten Wege von S zu allen anderen Knoten



Kürzeste Wege in Graphen

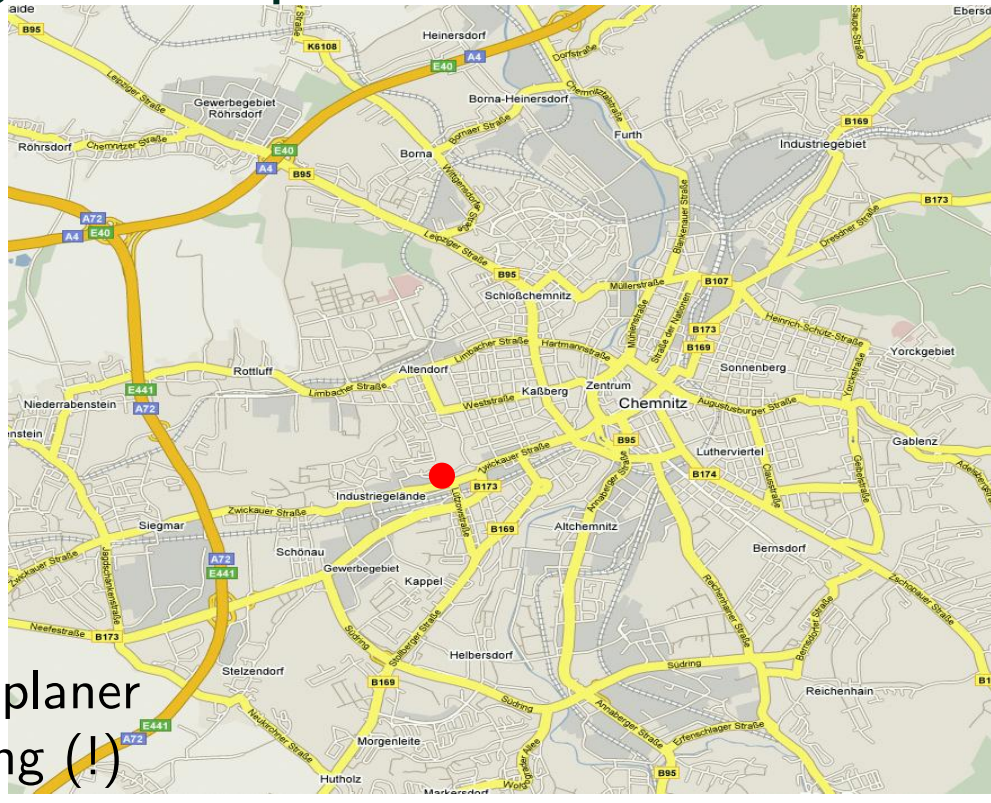
Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

Gesucht: die kürzesten Wege von S zu allen anderen Knoten



Das am öftesten gelöste Problem:
- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

Kürzeste Wege in Graphen



Gegeben ist:

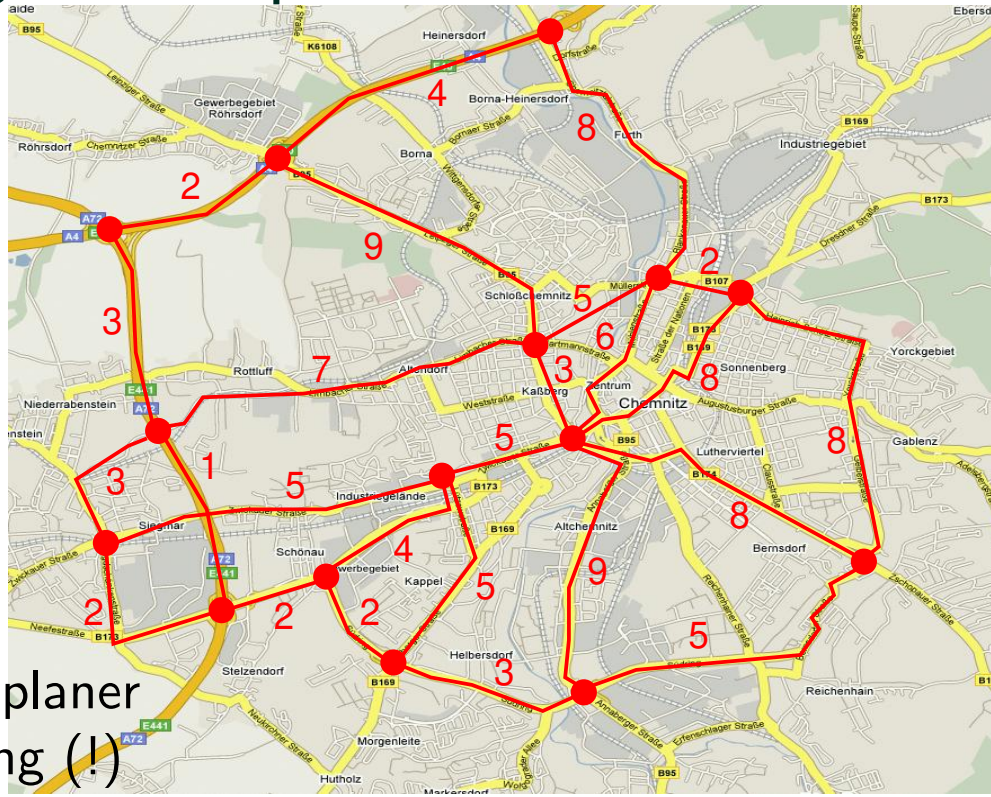
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

Gesucht: die kürzesten Wege von S zu allen anderen Knoten

Das am öftesten gelöste Problem:

- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

Kürzeste Wege in Graphen



Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

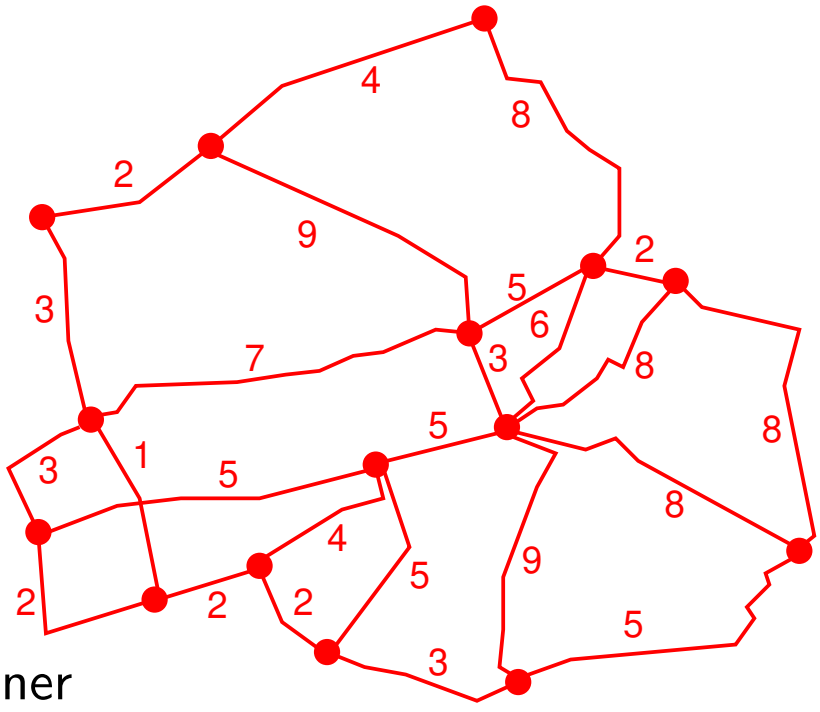
Gesucht: die kürzesten Wege von S zu allen anderen Knoten

Das am öftesten gelöste Problem:
- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

Kürzeste Wege in Graphen

Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

Gesucht: die kürzesten Wege von S zu allen anderen Knoten

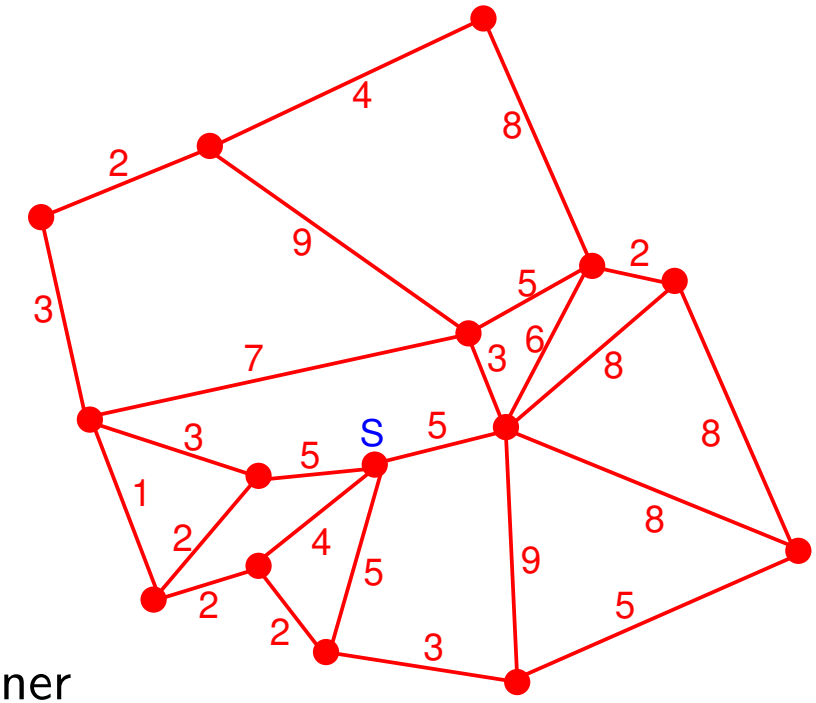


Das am öftesten gelöste Problem:
- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

Kürzeste Wege in Graphen

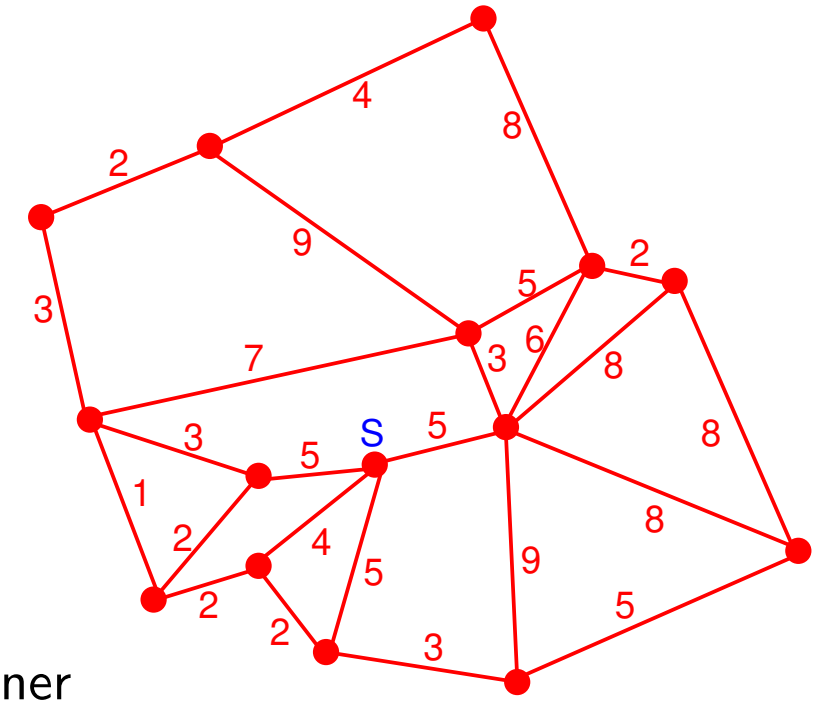
Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

Gesucht: die kürzesten Wege von S zu allen anderen Knoten



Das am öftesten gelöste Problem:
- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

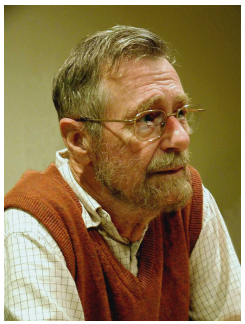
Kürzeste Wege in Graphen



Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

Gesucht: die kürzesten Wege von S zu allen anderen Knoten

Das am öftesten gelöste Problem:
- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

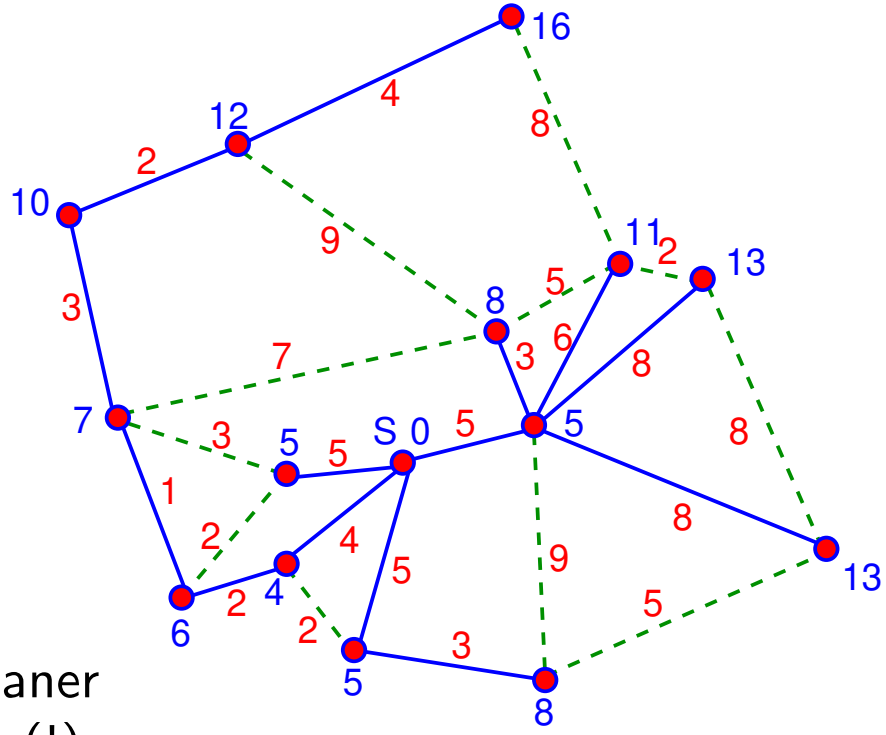


1959, Algorithmus von Dijkstra (1930-2002)

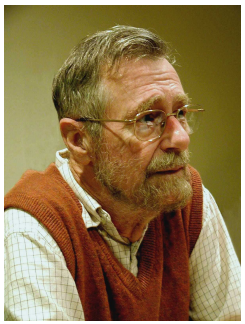
Kürzeste Wege in Graphen

Gegeben ist:
- ein Graph mit Kantenlängen (≥ 0)
- ein Startknoten S

Gesucht: die kürzesten Wege von S zu allen anderen Knoten

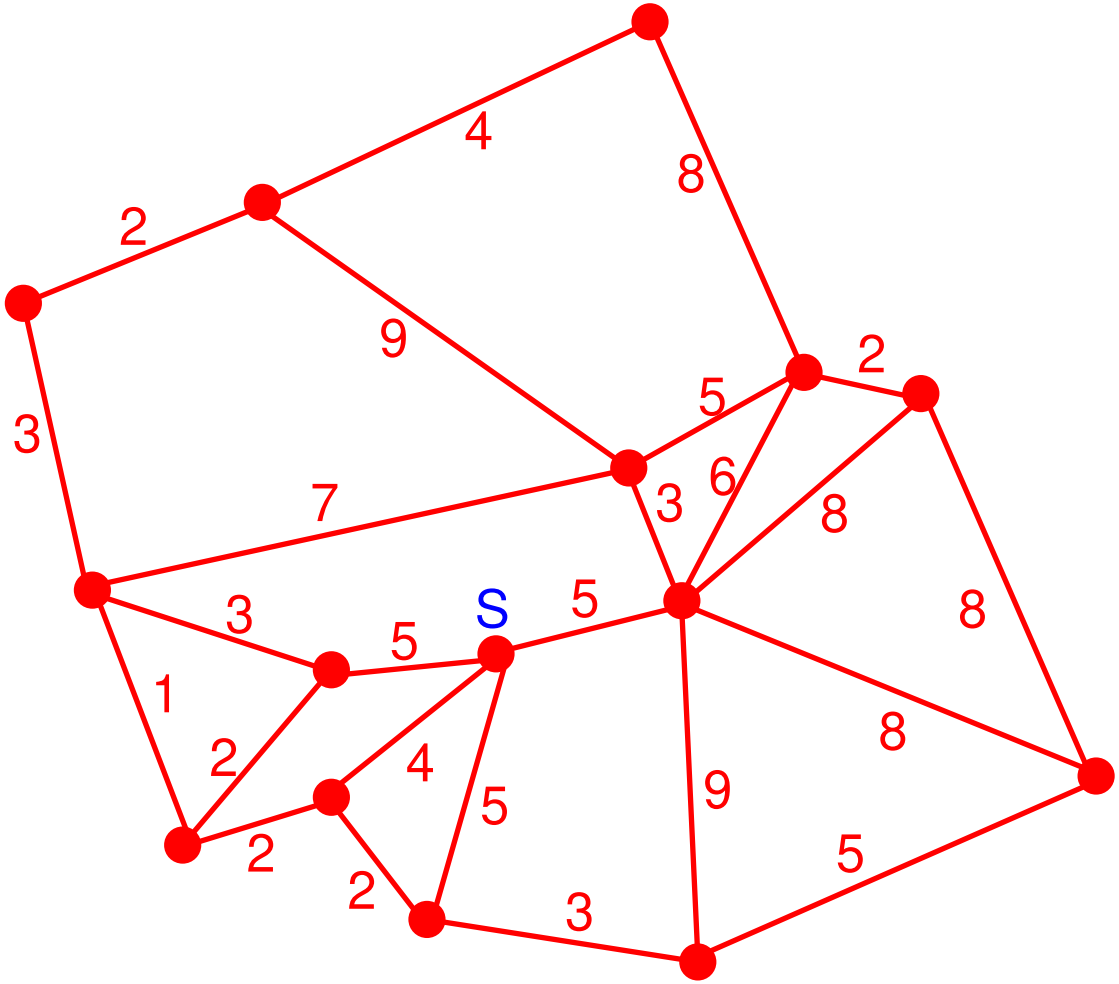


Das am öftesten gelöste Problem:
- in jedem Navigationssystem/Routenplaner
- bei jeder einzelnen Internetverbindung (!)
- als Teilproblem

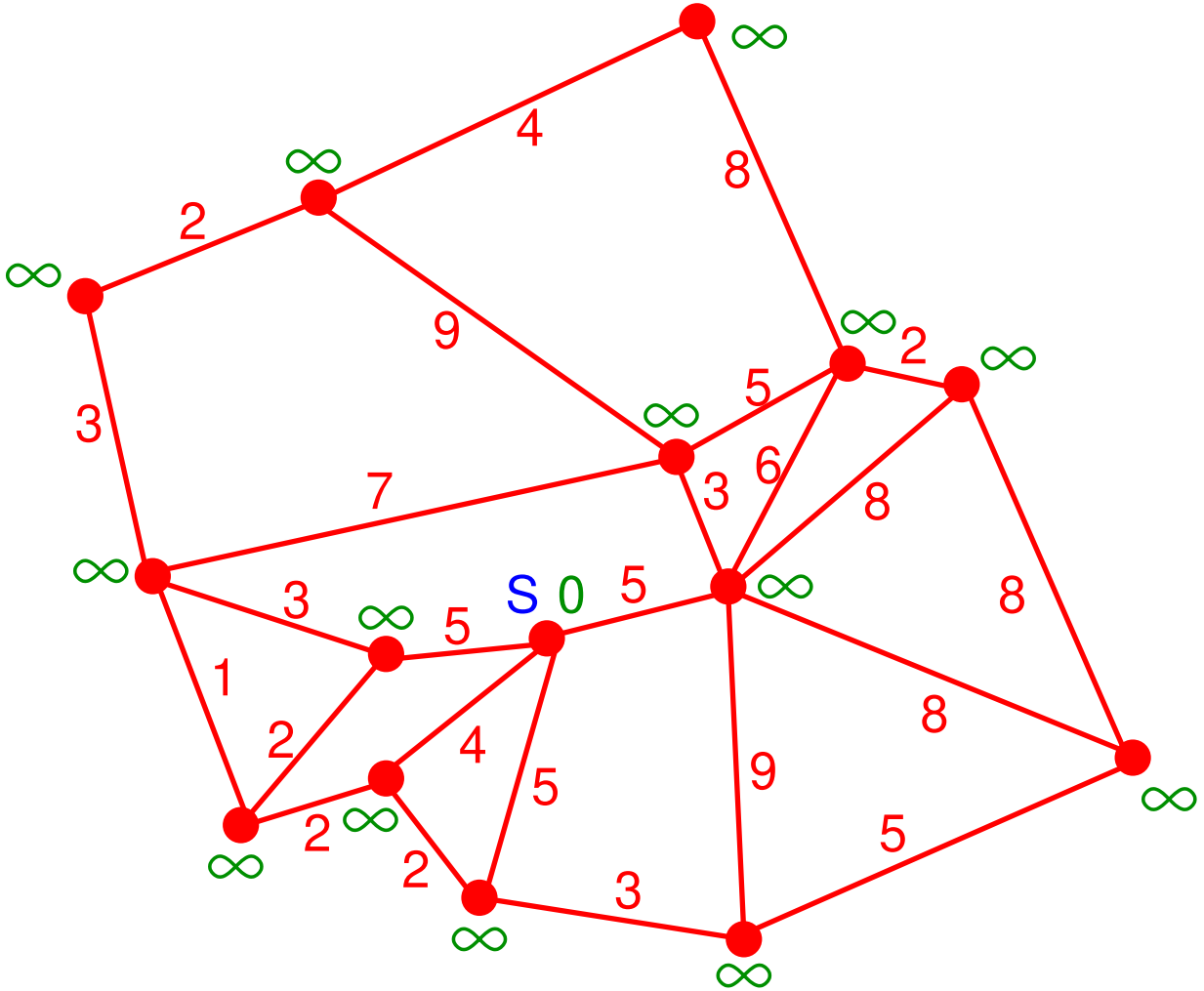


1959, Algorithmus von Dijkstra (1930-2002)

Der Kürzeste-Wege-Algorithmus von Dijkstra

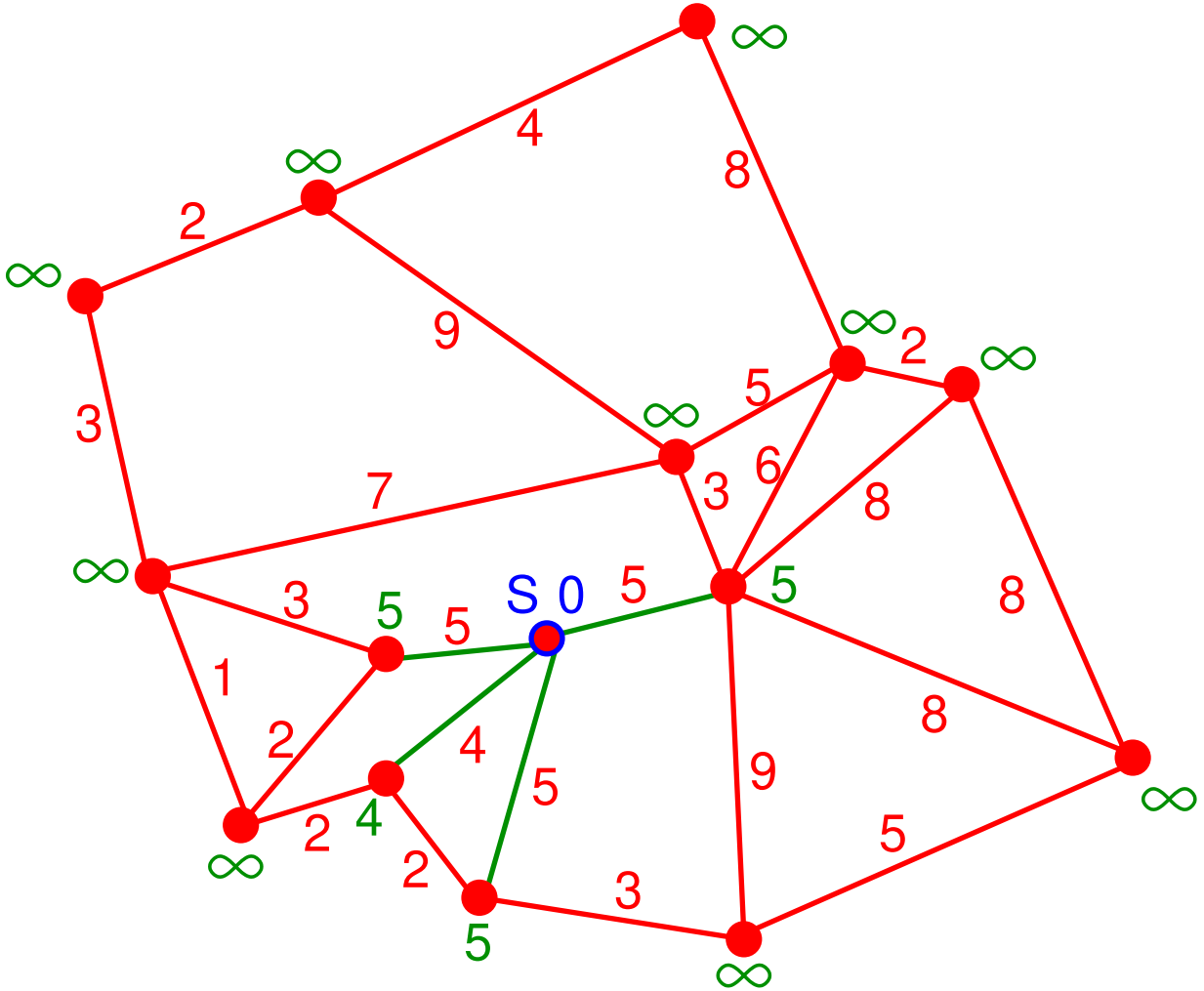


Der Kürzeste-Wege-Algorithmus von Dijkstra



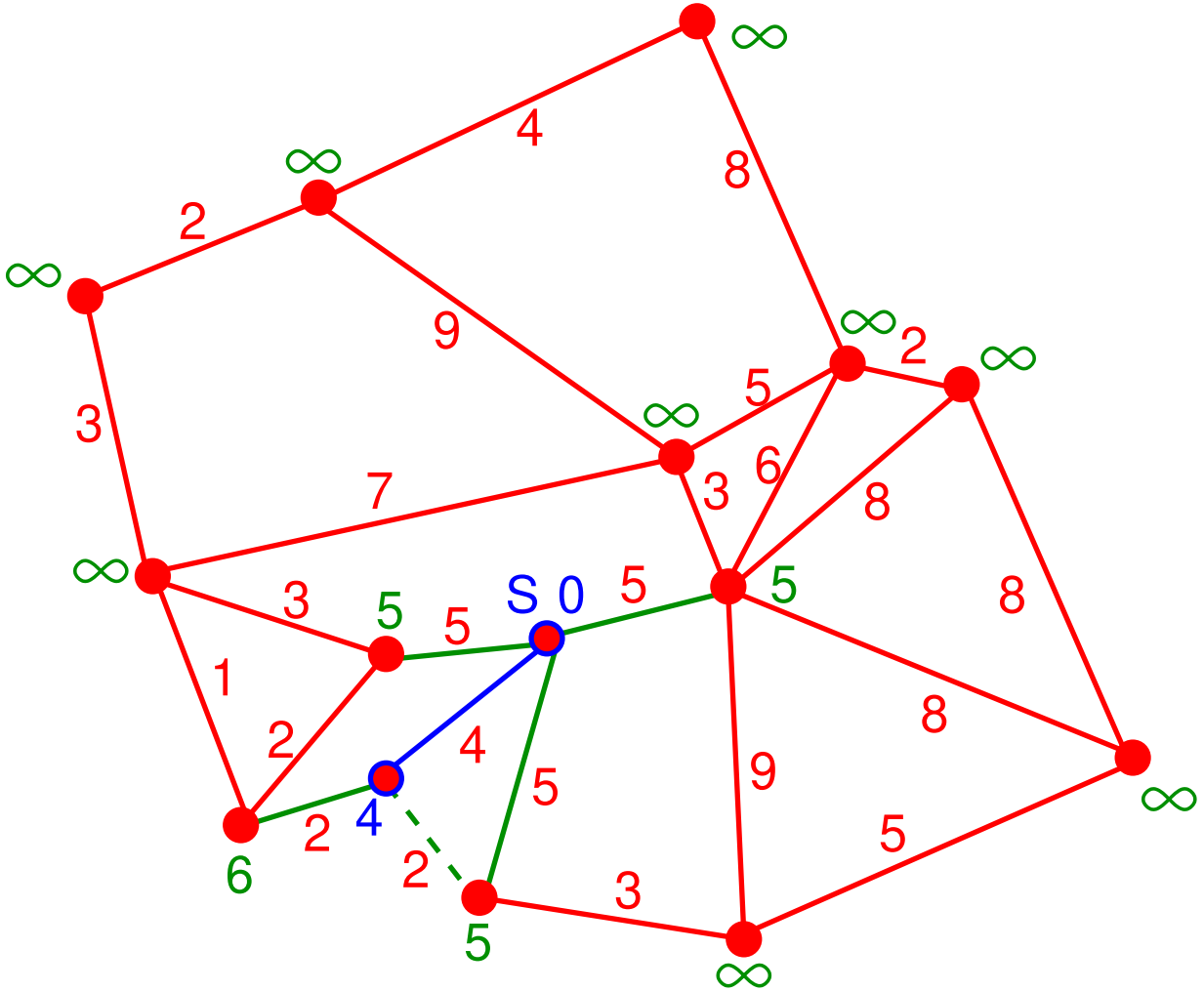
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



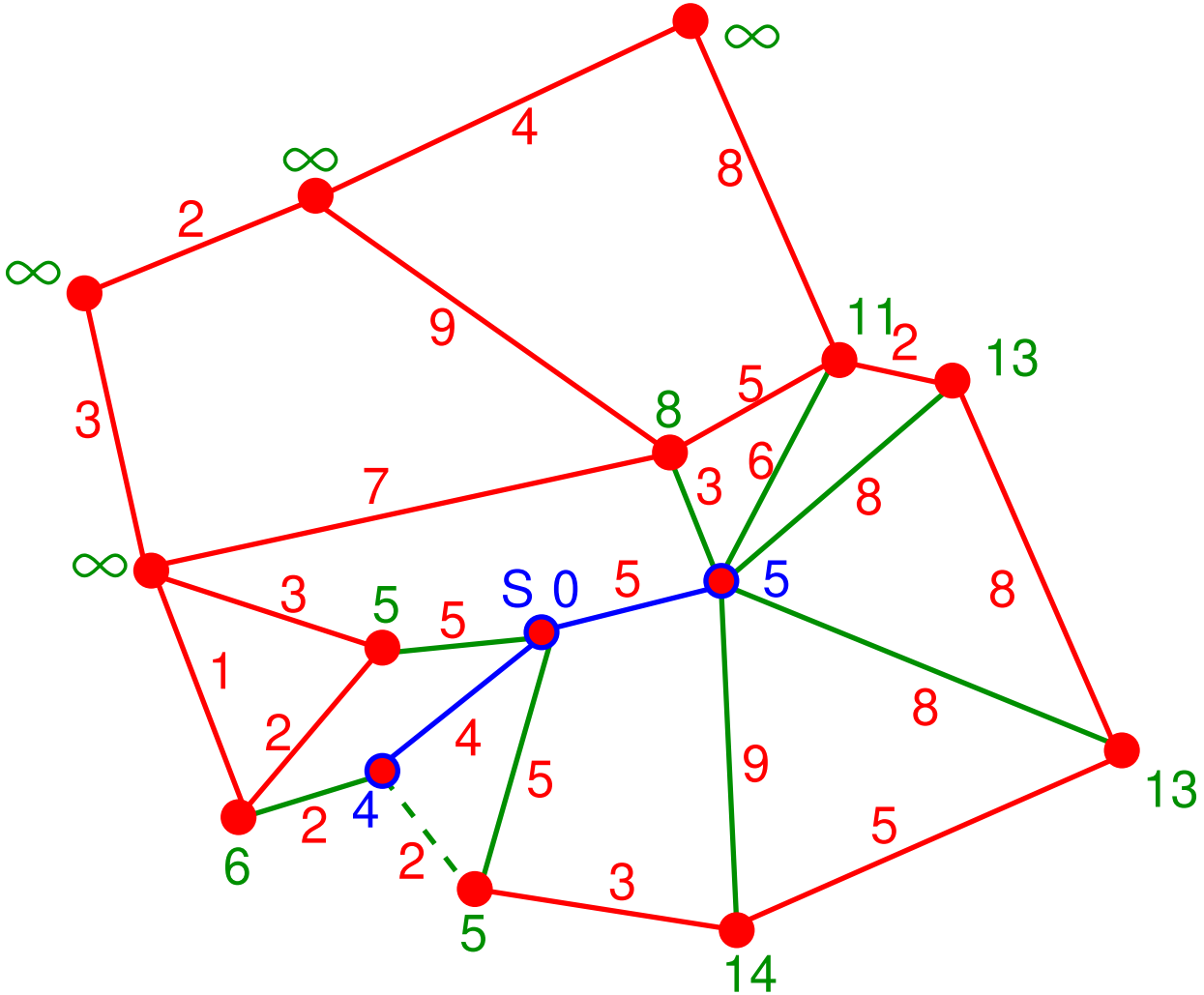
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



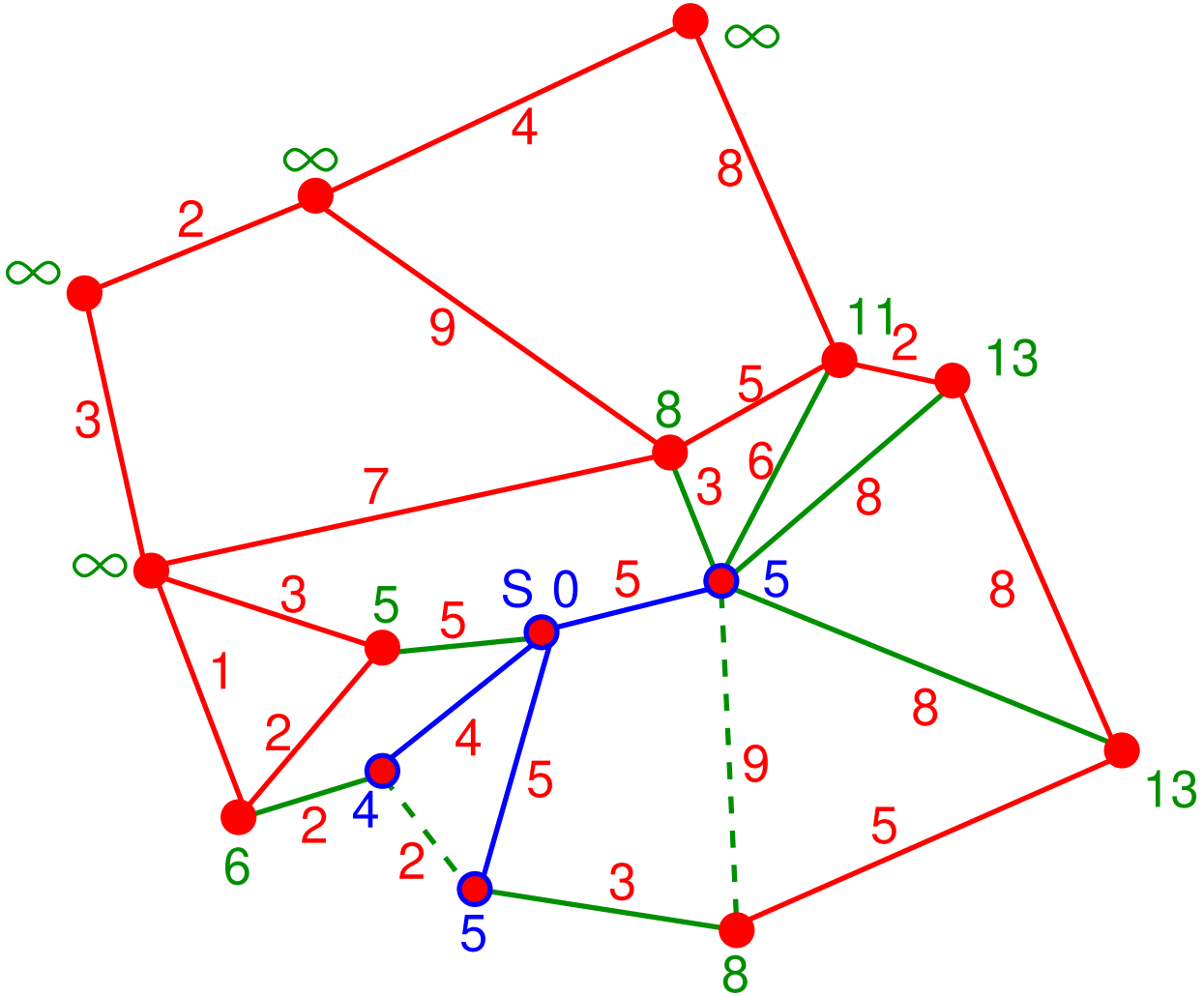
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



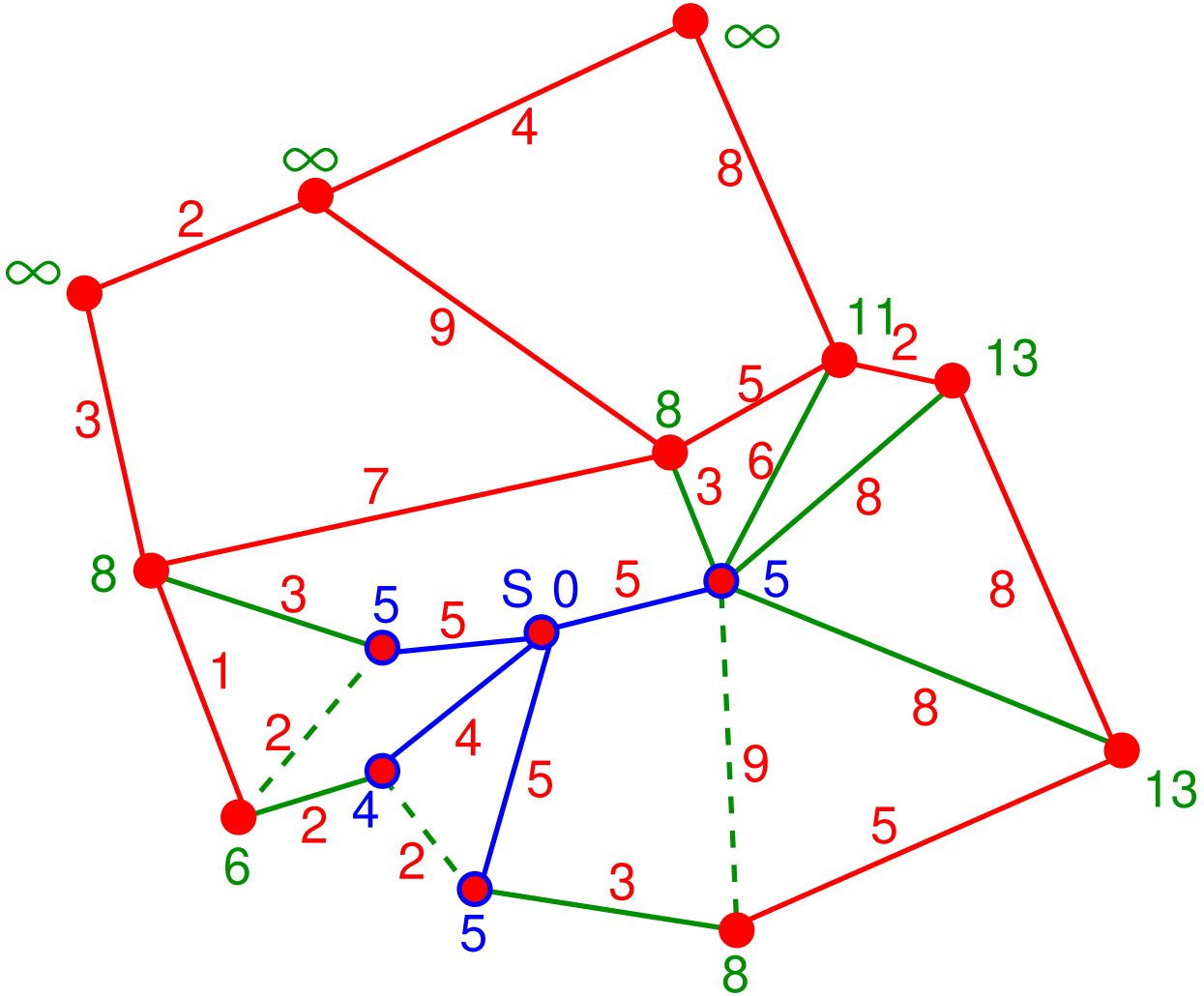
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



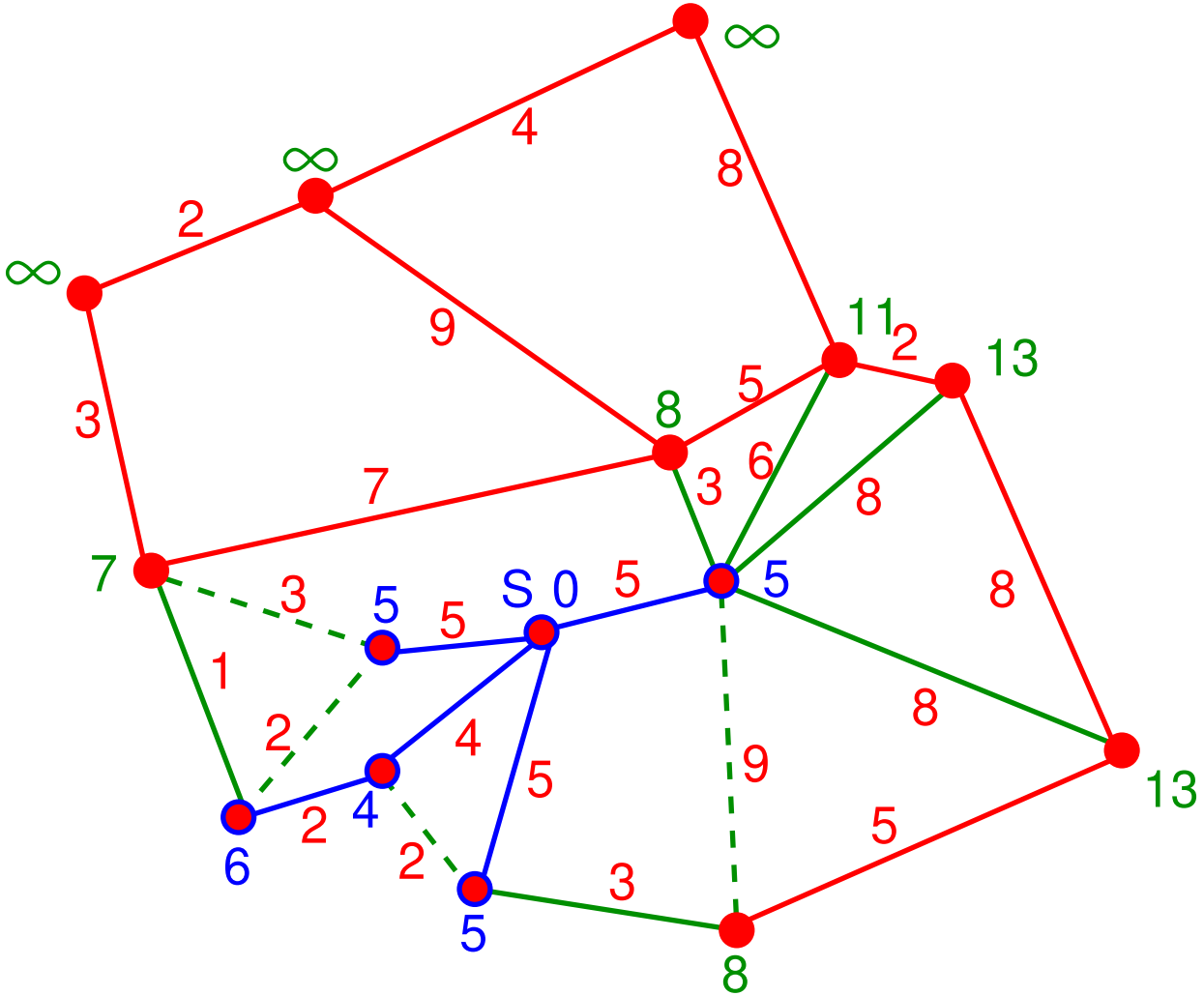
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



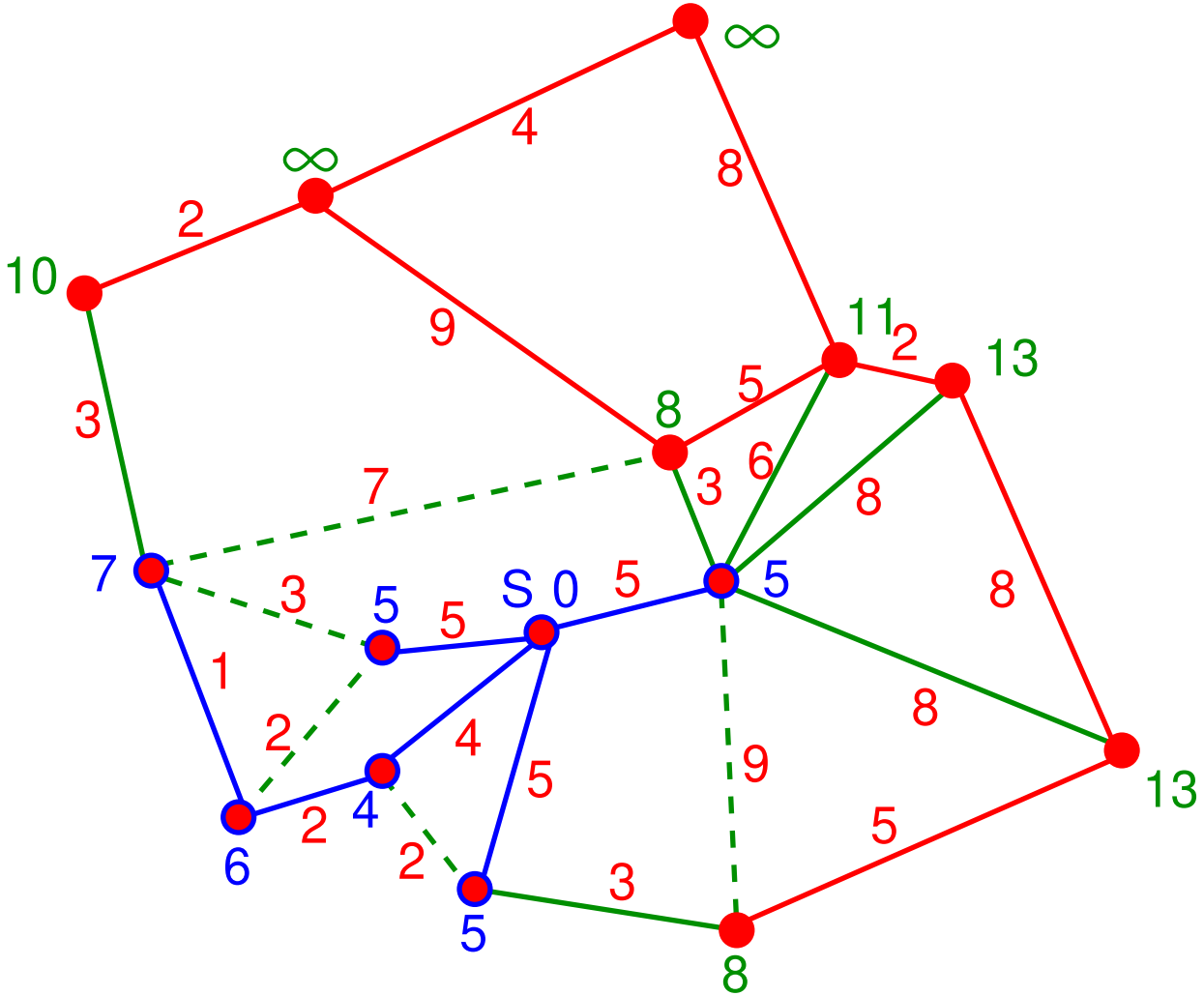
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



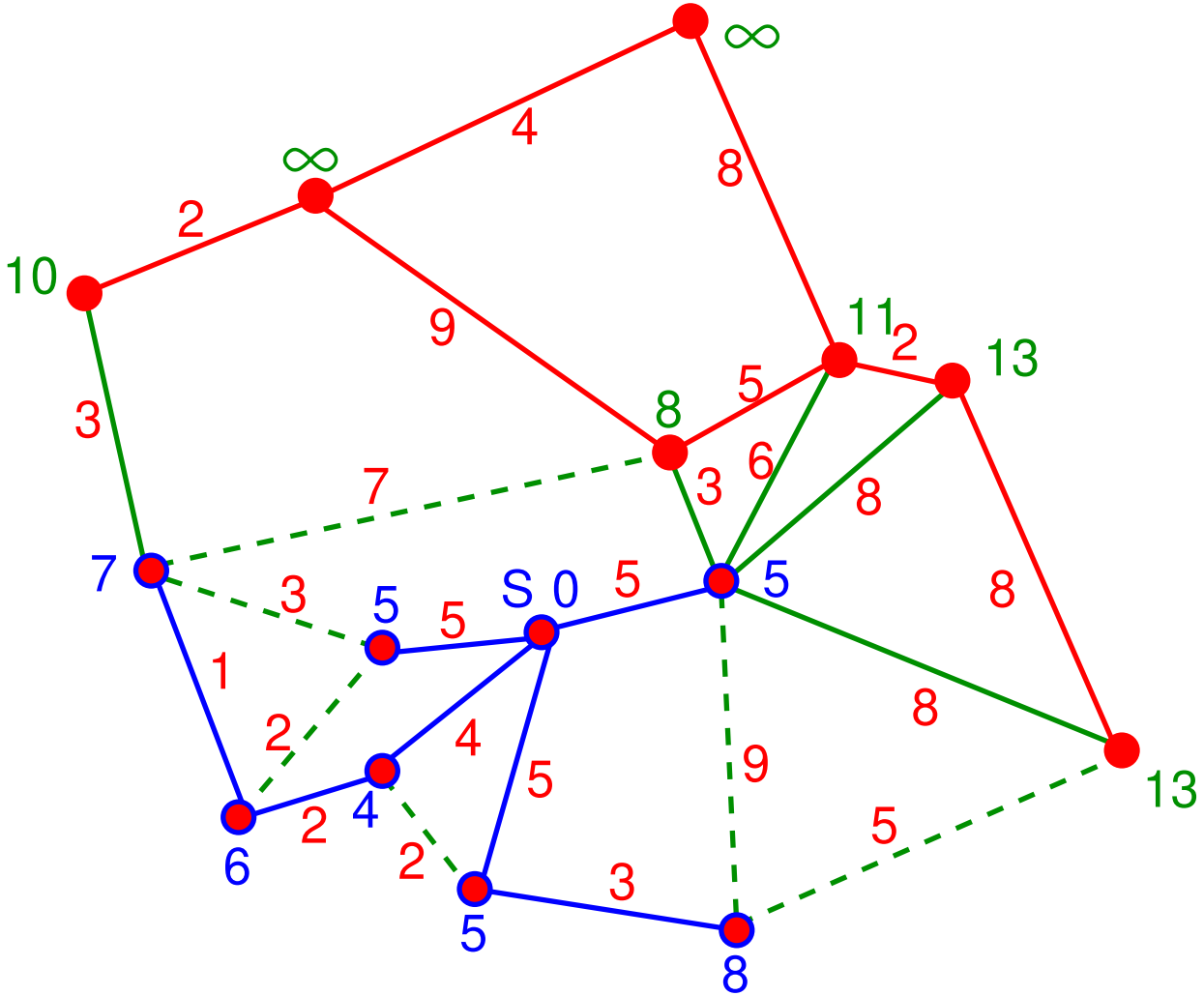
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



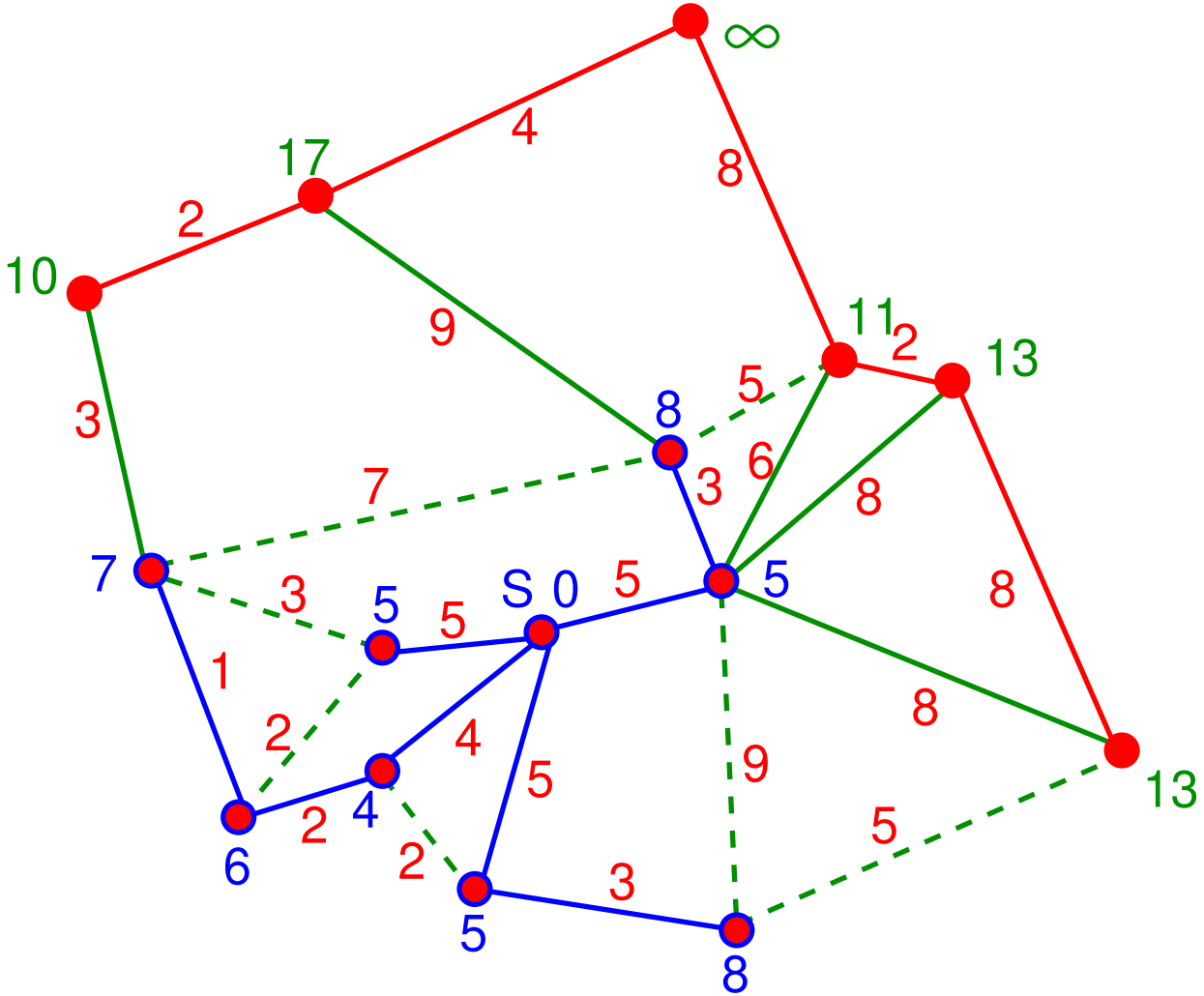
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



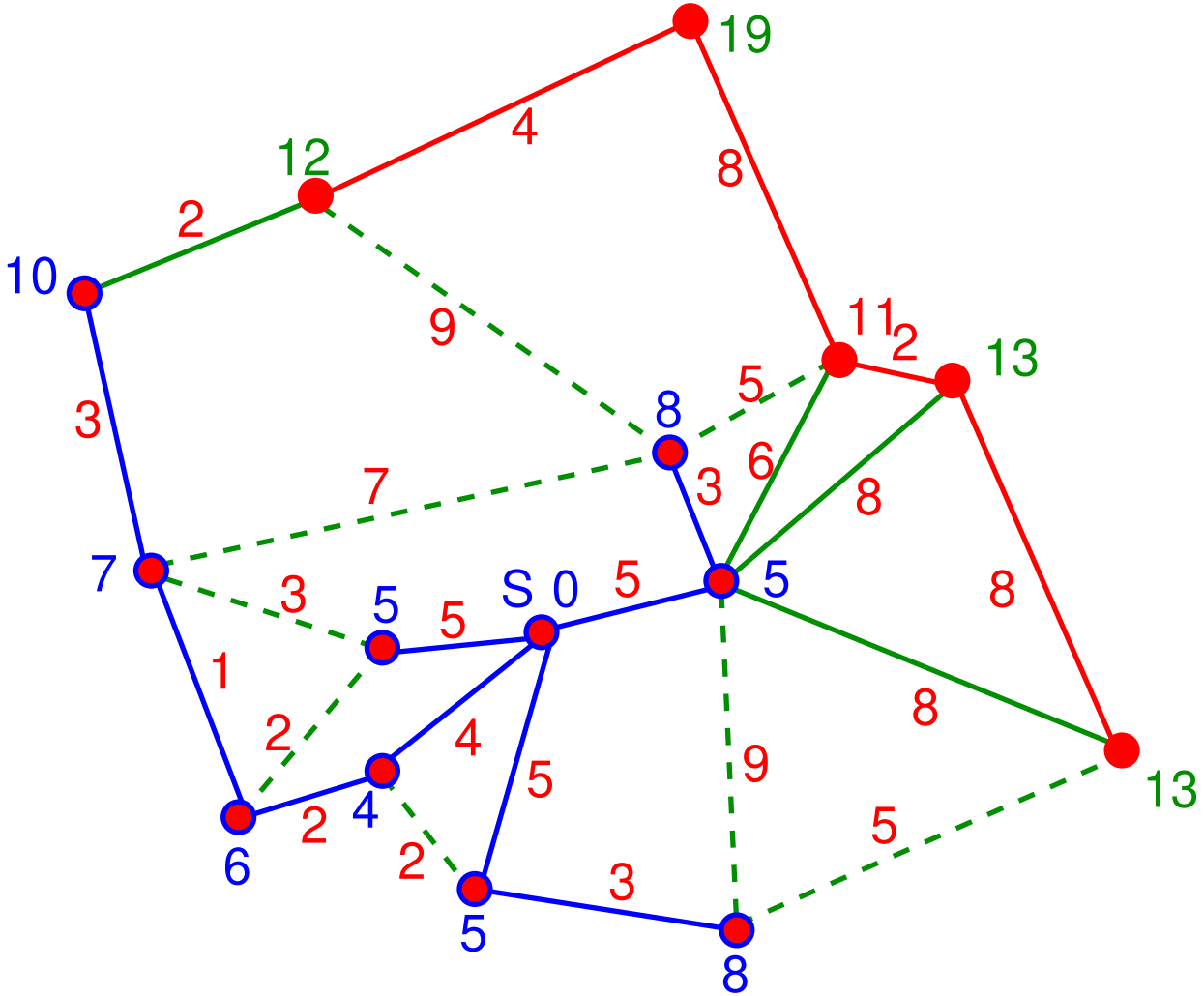
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



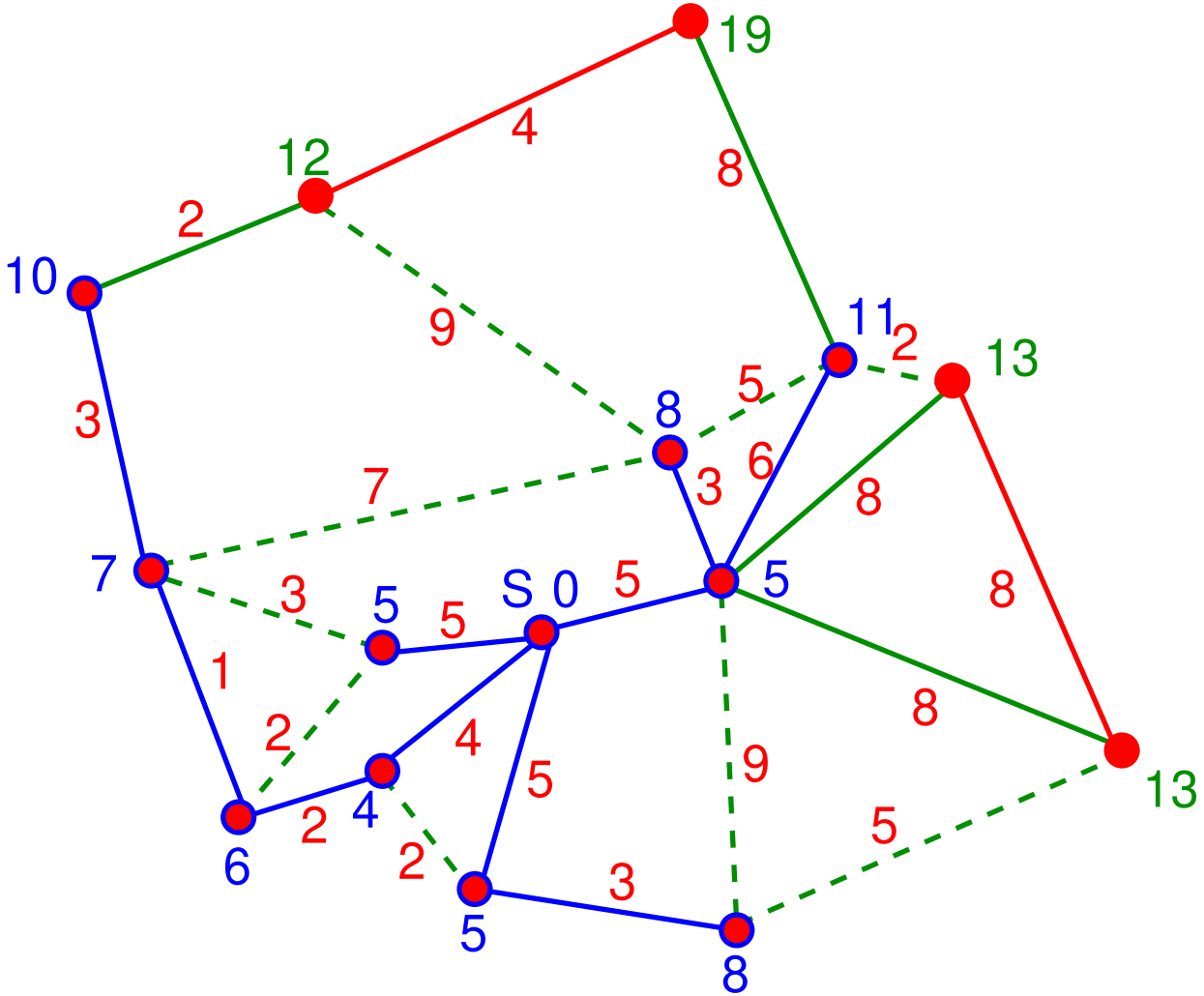
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



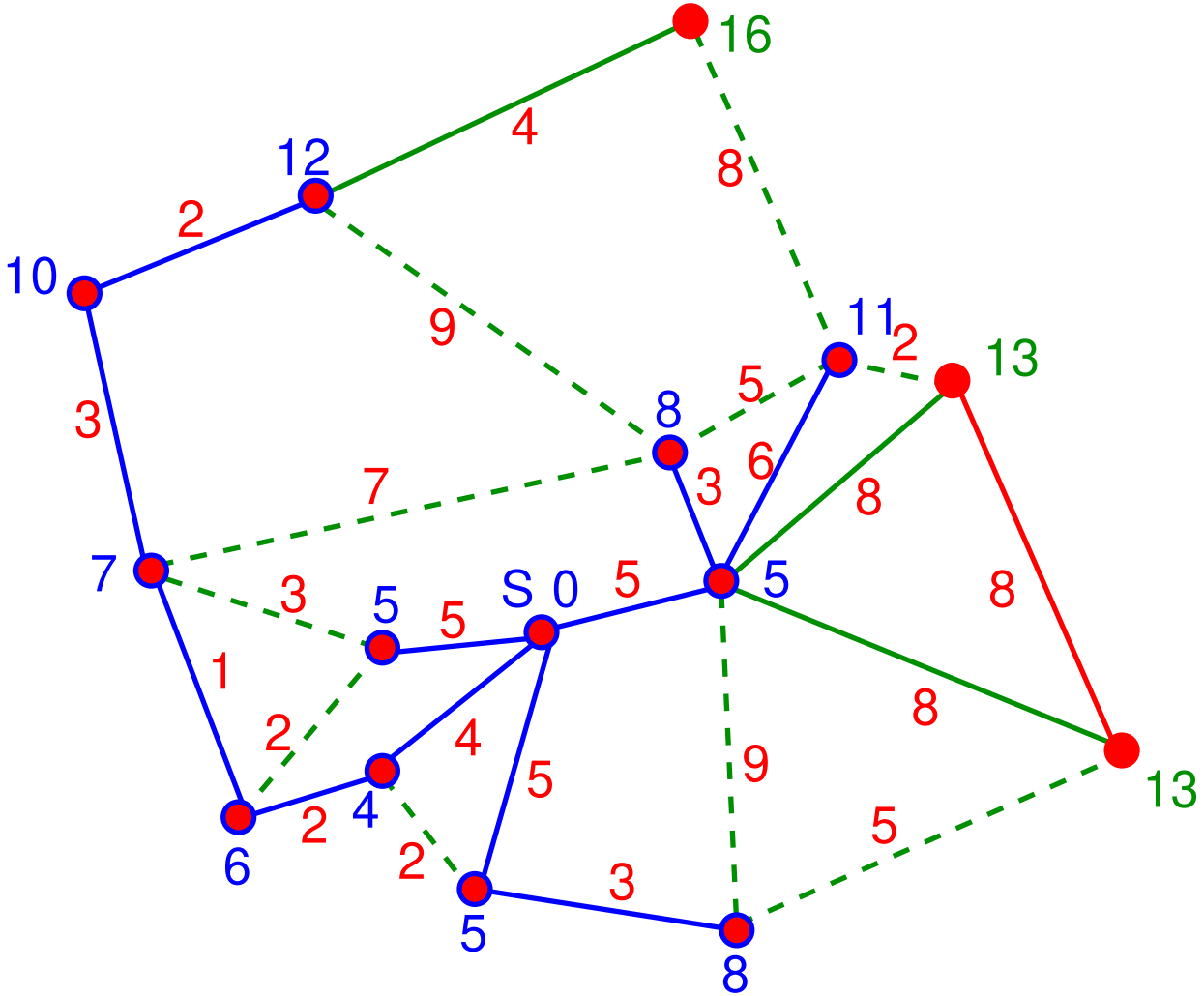
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



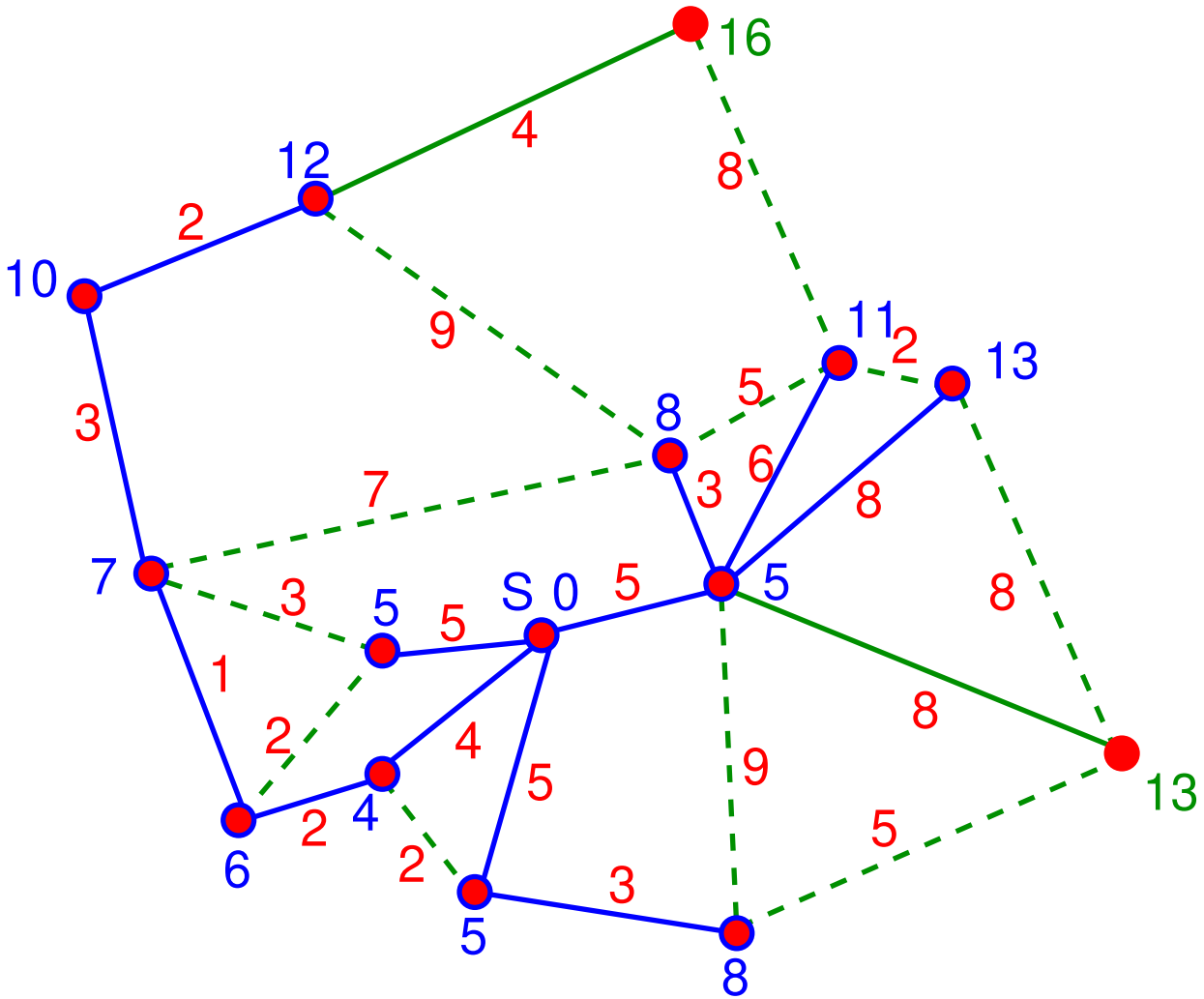
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



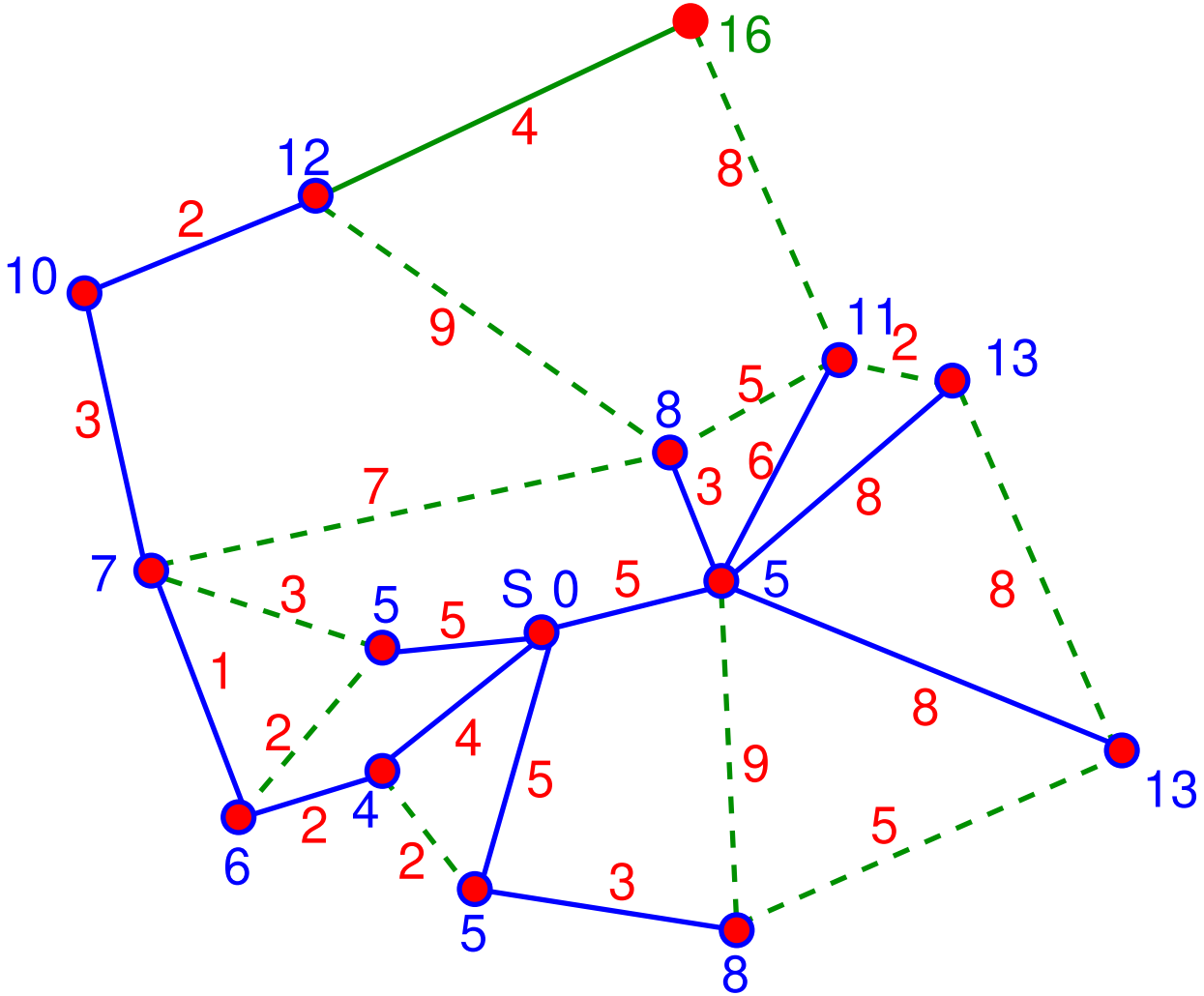
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



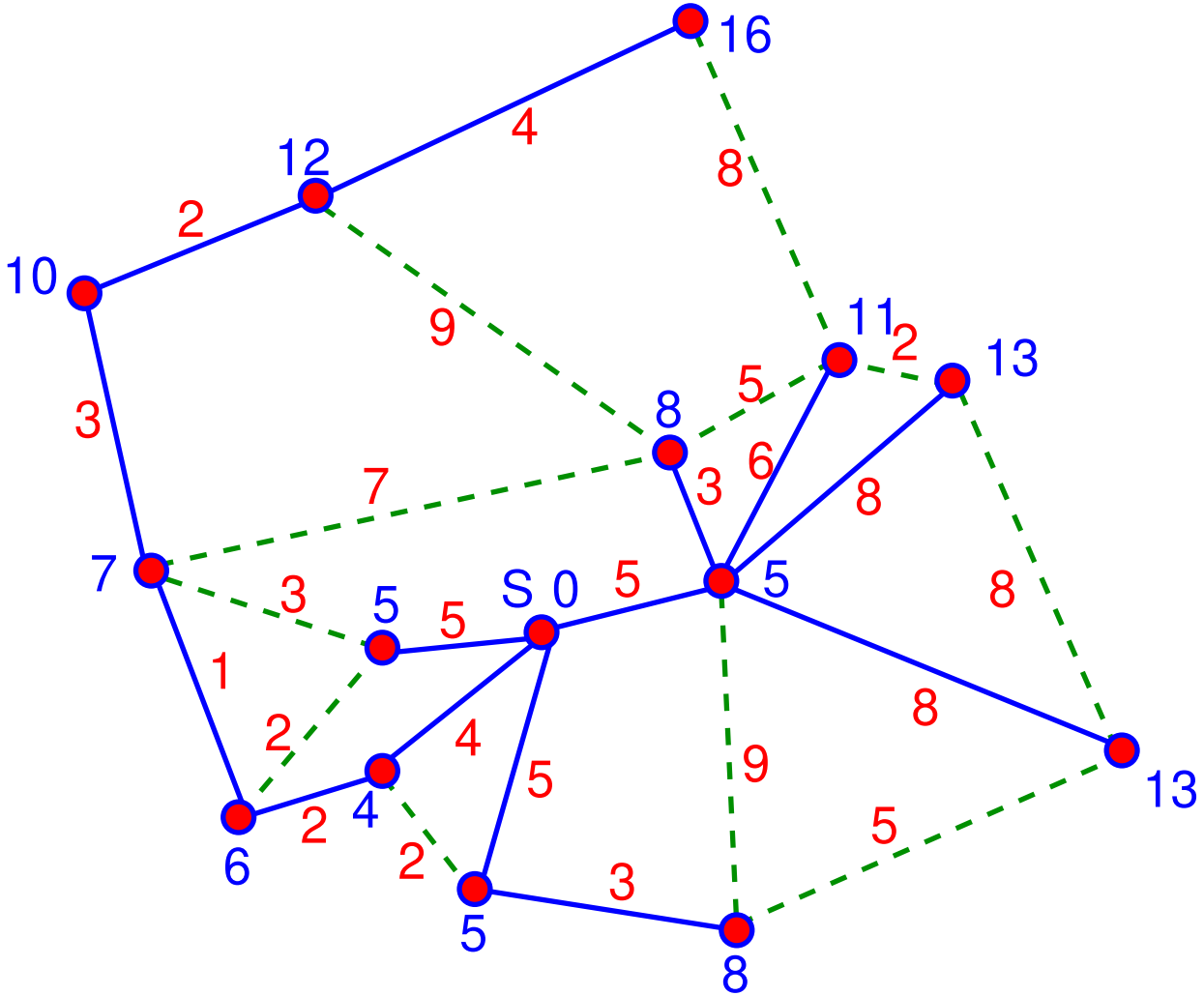
Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Der Kürzeste-Wege-Algorithmus von Dijkstra



Weil alle Kanten nichtnegative Länge haben, kann die kleinste Distanz nicht mehr kürzer werden $\rightarrow$ diese Knotendistanz liegt fest.

Übersicht

Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

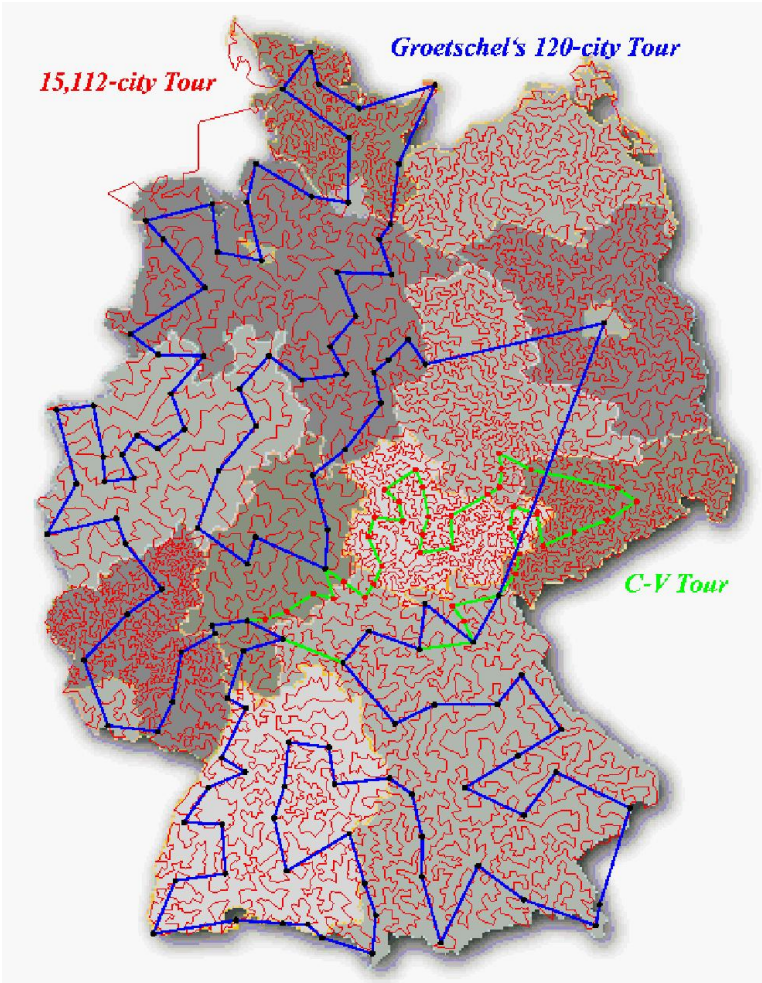
Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

- Graphenmodelle und kürzeste Wege
- **Rundreisen**
- Algorithmische Komplexität, ein Milleniums-Problem
- Fluss- und Transportprobleme
- Betriebsplanerstellung
- Schlussbemerkungen

Das Rundreise- / Traveling Salesman Problem (TSP)

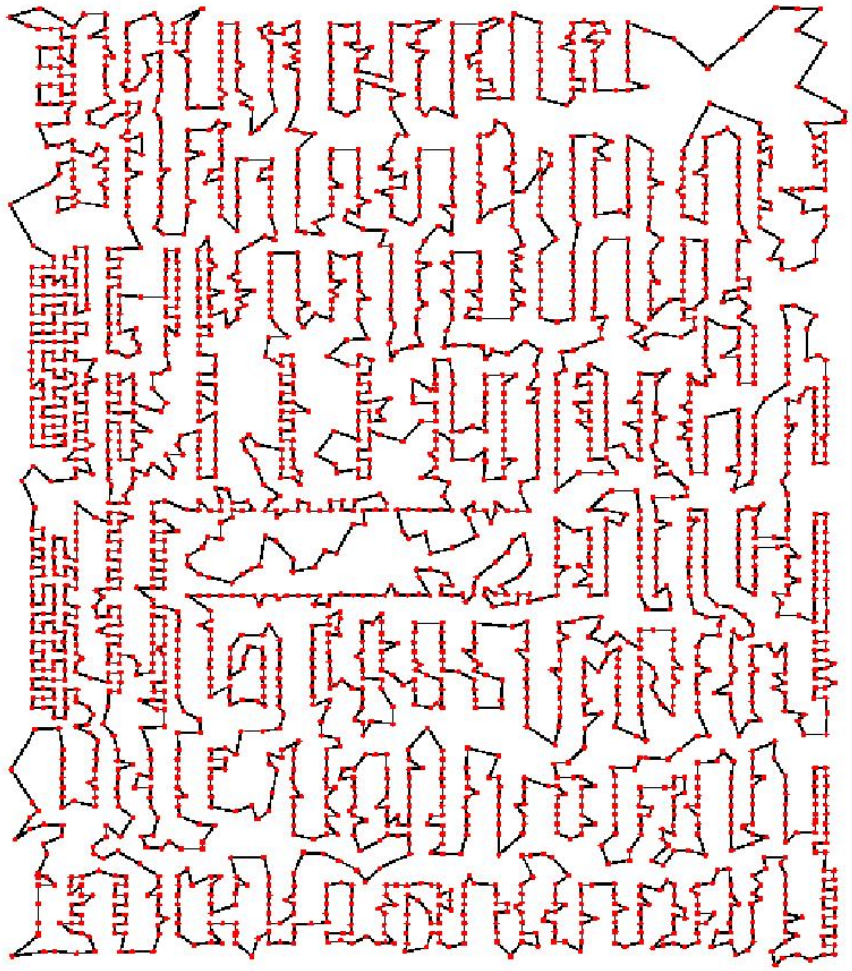
Gegeben sind n Städte mit allen Distanzen.
Gesucht ist die kürzeste Rundreise durch alle n Städte.

Beispiel Deutschland [http://www.math.princeton.edu/tsp]



1832: *Der Handlungsreisende - wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein - Von einem alten Commis-Voyageur*

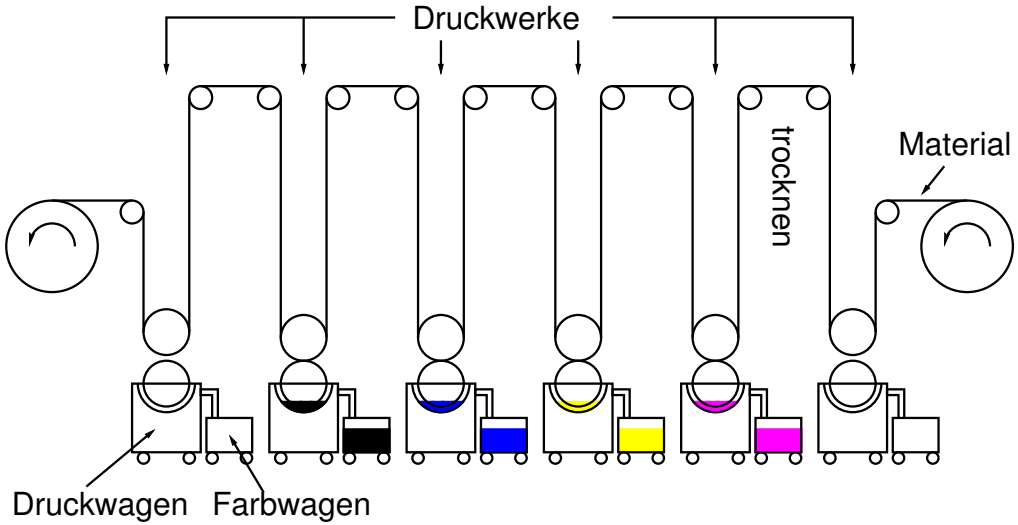
Beispiele: **Löcher in Platinen bohren** [GJR 1991]

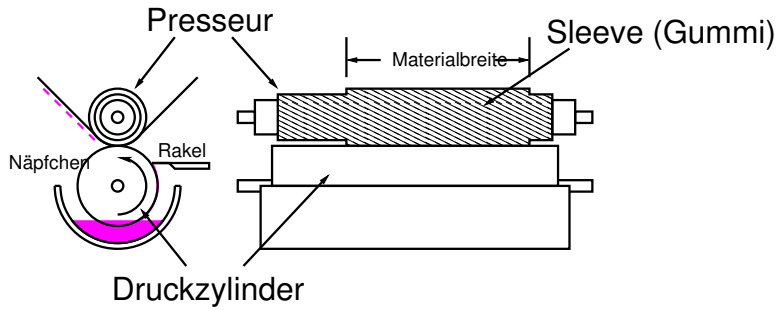


[http://www.math.princeton.edu/tsp]

Reihenfolgeoptimierung bei langen Rüstzeiten

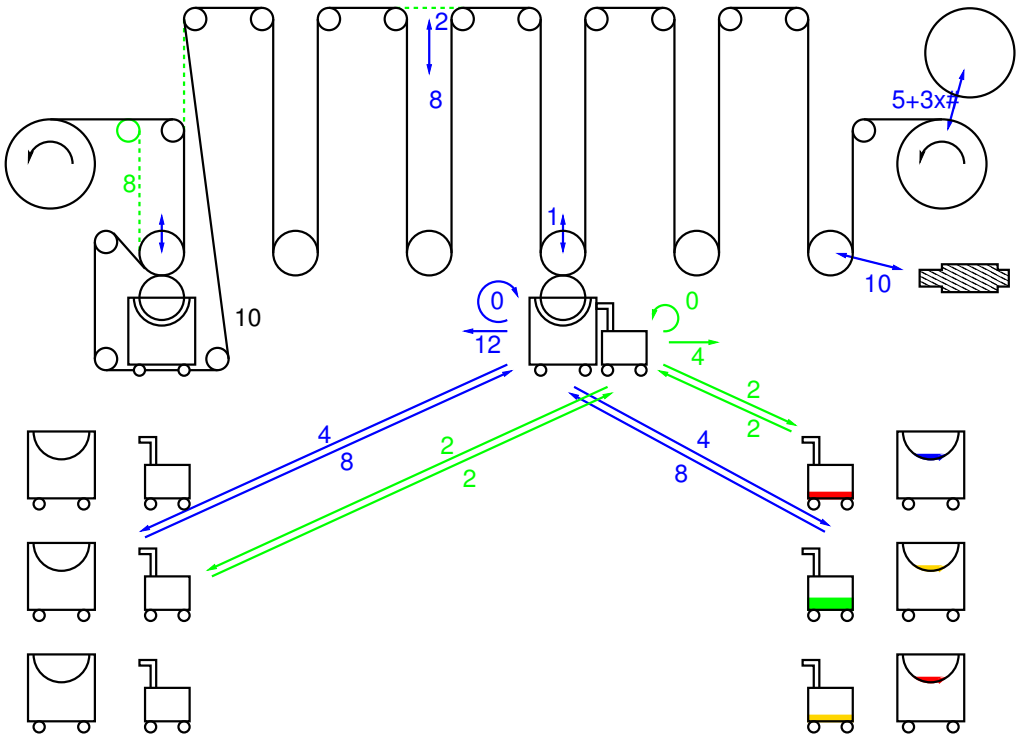
2 Rotationstiefdruckmaschinen zum Drucken von Geschenkpapier





Aufteilung und Reihenfolgeplanung von Druckaufträgen für Maschinen, so dass der letzte Auftrag möglichst früh fertig gestellt wird.

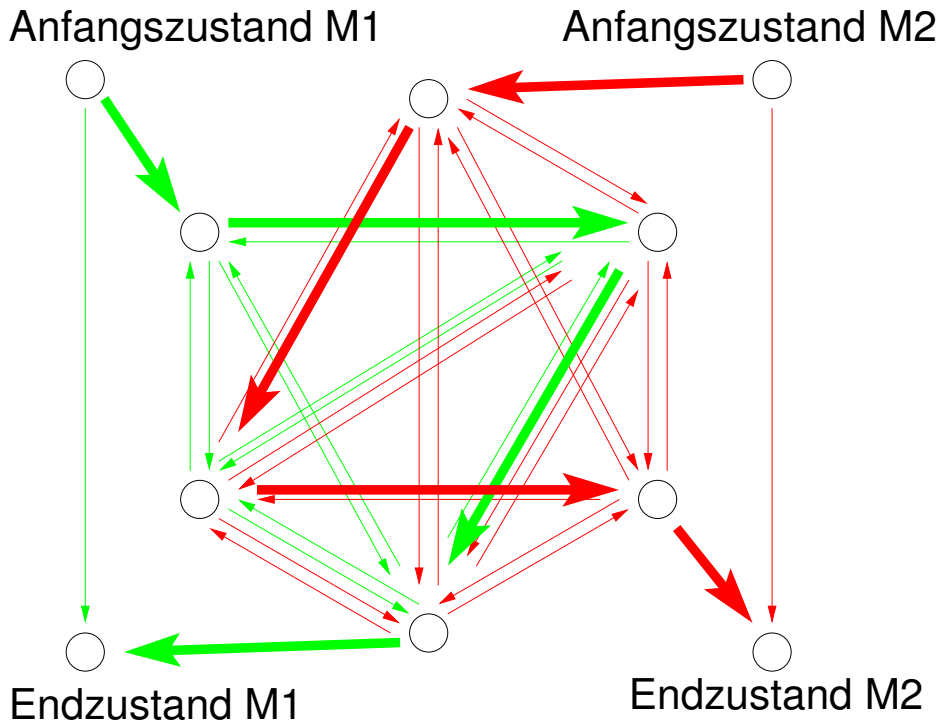
Rüstzeiten:



Passer

Farbanpassung neue Farben: 20

Druckfolgeoptimierung als Rundreiseproblem mit 2 „Fahrzeugen“



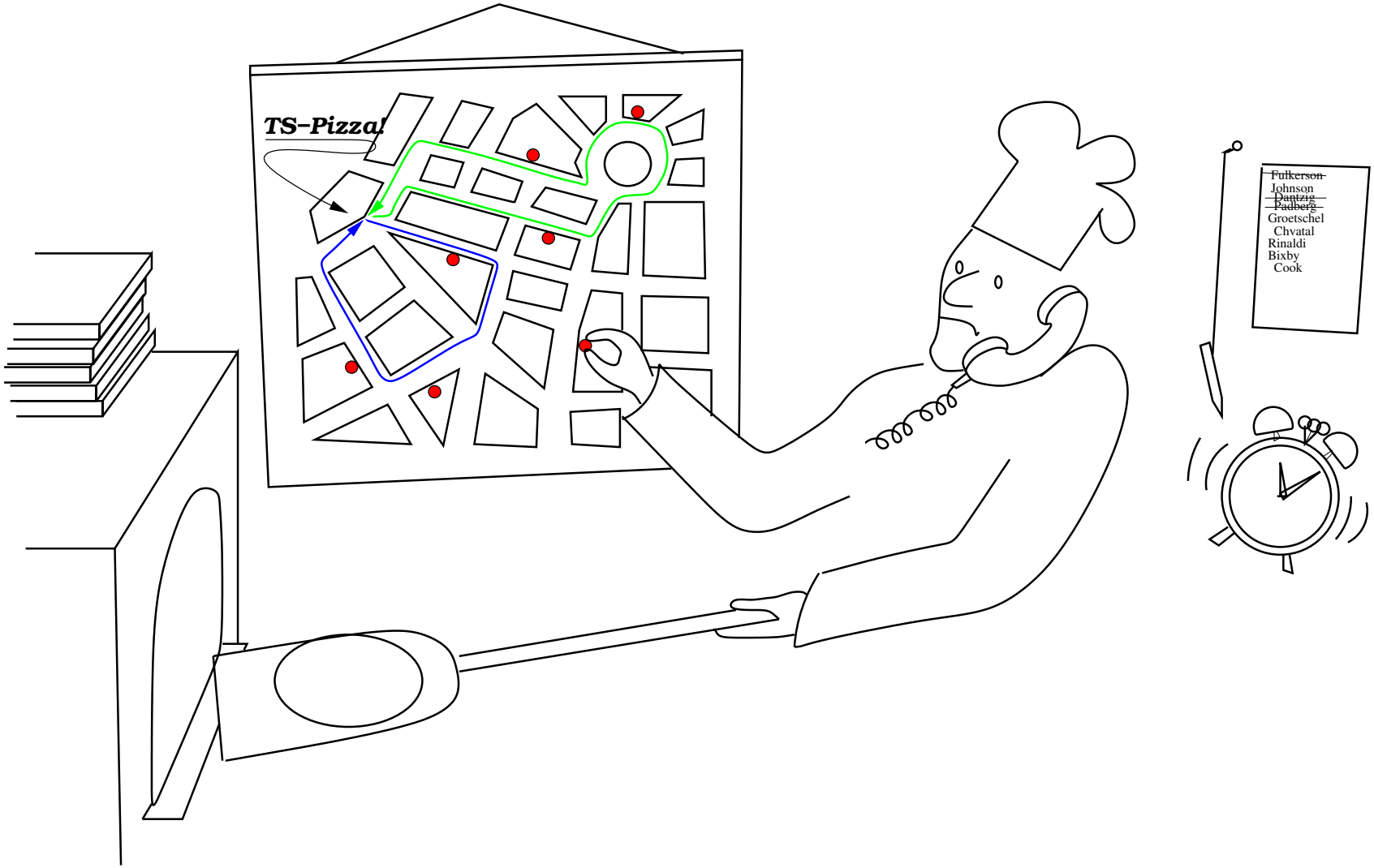
Knoten entsprechen den zu druckenden Geschenkpapierarten.

Kanten sind mit Bearbeitungszeit (Umrüsten und Drucken) gewichtet.

Suche zwei Wege so, dass alle Knoten besucht werden und die größere der Summen der Kantengewichte möglichst klein ist.

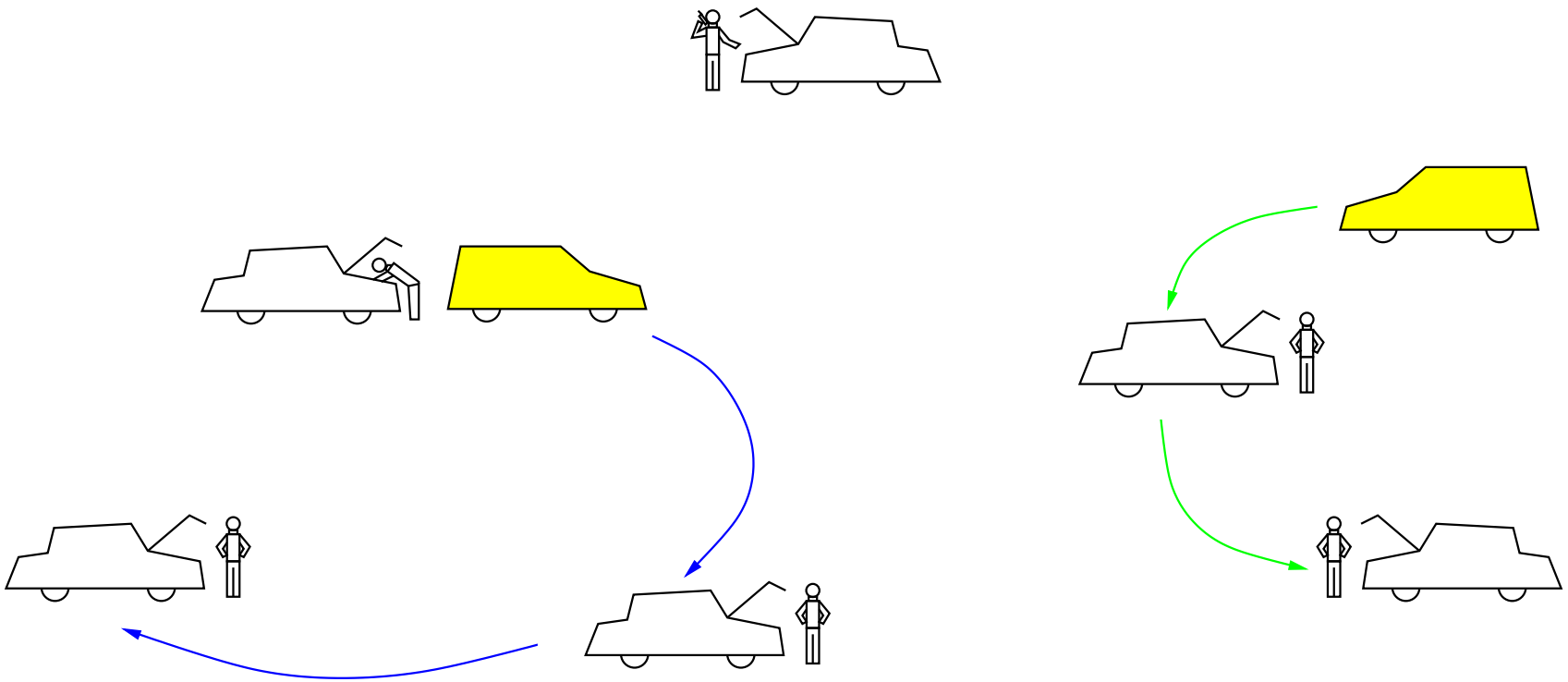
Das Zustelldienst-Problem

Fahrzeugflotte, mit der alle Anfragen in Zeitfenstern zu bedienen sind



Pannenhilfe

[ADAC, ZIB]

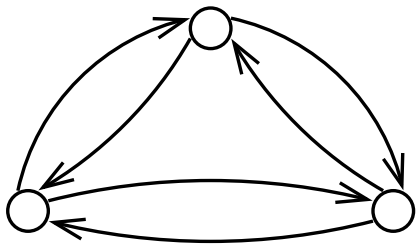


Bestimme Autozuweisung und Reihenfolge für jedes Pannenfahrzeug so, dass bereits versprochene Wartezeiten nicht überschritten werden.

Wie schwer sind die Probleme zu lösen?

„Nur“ endlich viele Möglichkeiten, aber dafür enorm viele!

Wieviele Rundreisen gibt es durch n Städte?
$n = 3$:



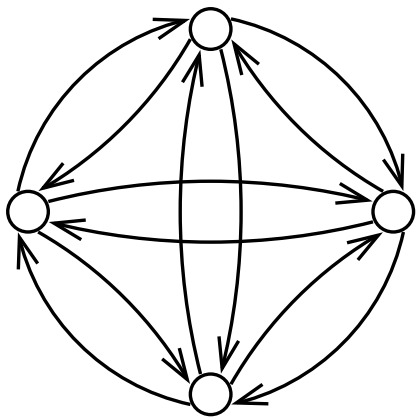
Wie schwer sind die Probleme zu lösen?

„Nur“ endlich viele Möglichkeiten, aber dafür enorm viele!

Wieviele Rundreisen gibt es durch n Städte?

$n =$ 3: 2

$n =$ 4:



Wie schwer sind die Probleme zu lösen?

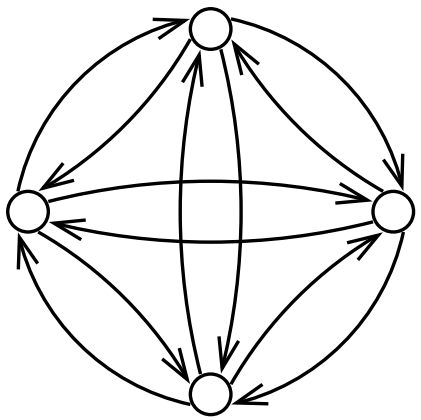
„Nur“ endlich viele Möglichkeiten, aber dafür enorm viele!

Wieviele Rundreisen gibt es durch n Städte?

$n = 3$: 2

$n = 4$: $3 \cdot 2 = 6$

$n = 5$:



Wie schwer sind die Probleme zu lösen?

„Nur“ endlich viele Möglichkeiten, aber dafür enorm viele!

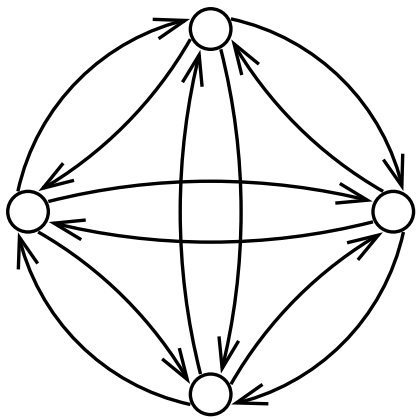
Wieviele Rundreisen gibt es durch n Städte?

$n = 3$: 2

$n = 4$: $3 \cdot 2 = 6$

$n = 5$: $4 \cdot 3 \cdot 2 = 24$

$n = 10$:

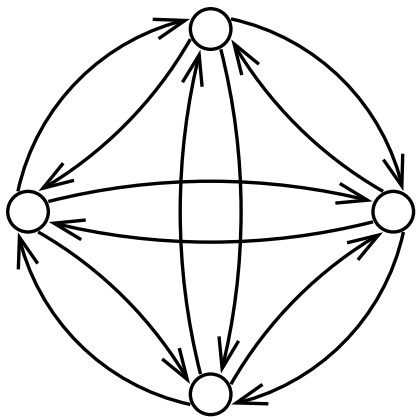


Wie schwer sind die Probleme zu lösen?

„Nur“ endlich viele Möglichkeiten, aber dafür enorm viele!

Wieviele Rundreisen gibt es durch n Städte?

$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}



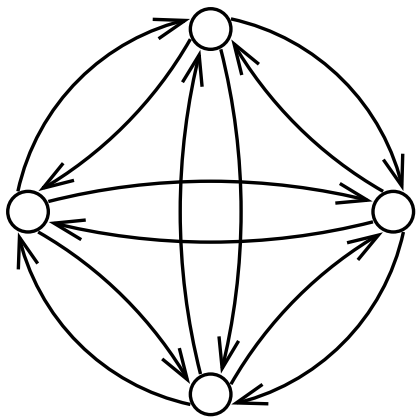
wächst exponentiell in $n \Rightarrow$ alle ausprobieren funktioniert nicht!

Wie schwer sind die Probleme zu lösen?

„Nur“ endlich viele Möglichkeiten, aber dafür enorm viele!

Wieviele Rundreisen gibt es durch n Städte?

$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}



wächst exponentiell in $n \Rightarrow$ alle ausprobieren funktioniert nicht!

Es liegt aber nicht nur an der Anzahl der Möglichkeiten.

Das Rundreiseproblem gehört zu einer Klasse von Problemen, die alle beweisbar gleich schwer zu lösen sind und für die keine „effizienten“ Verfahren bekannt sind! (NP-schwere Probleme)

"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

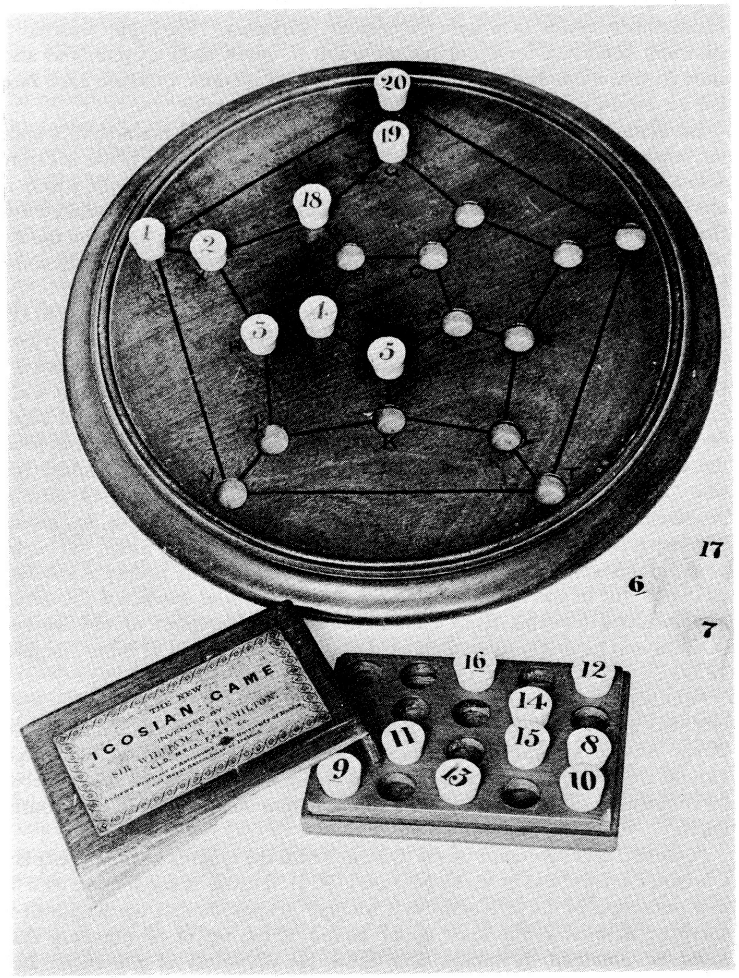


Figure 1.1 The Icosian Game (Photographed by Terri Russell. Copyright © 1983 by the Royal Irish Academy. Reprinted with permission.)

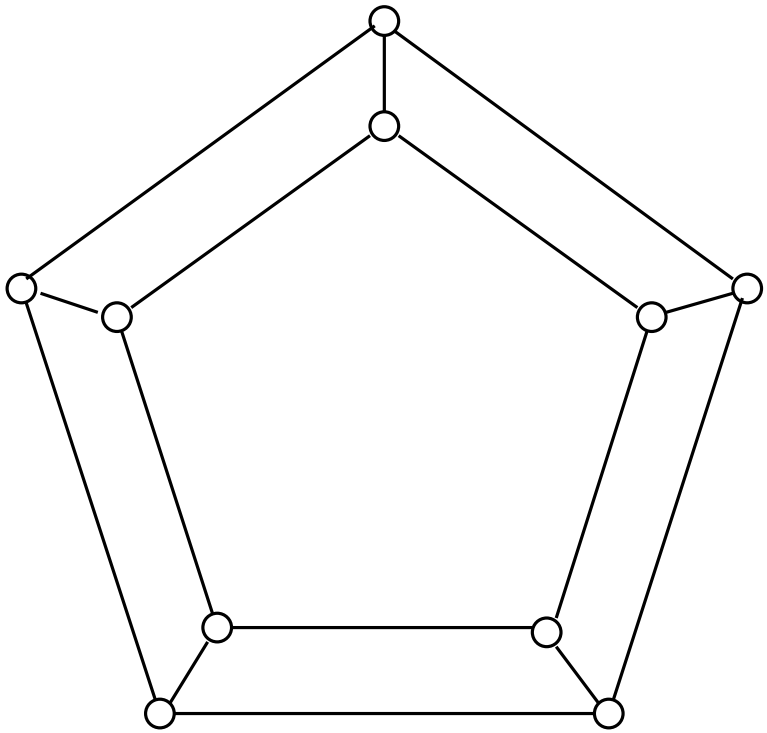
Sir William R. Hamilton

1805-1865

"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

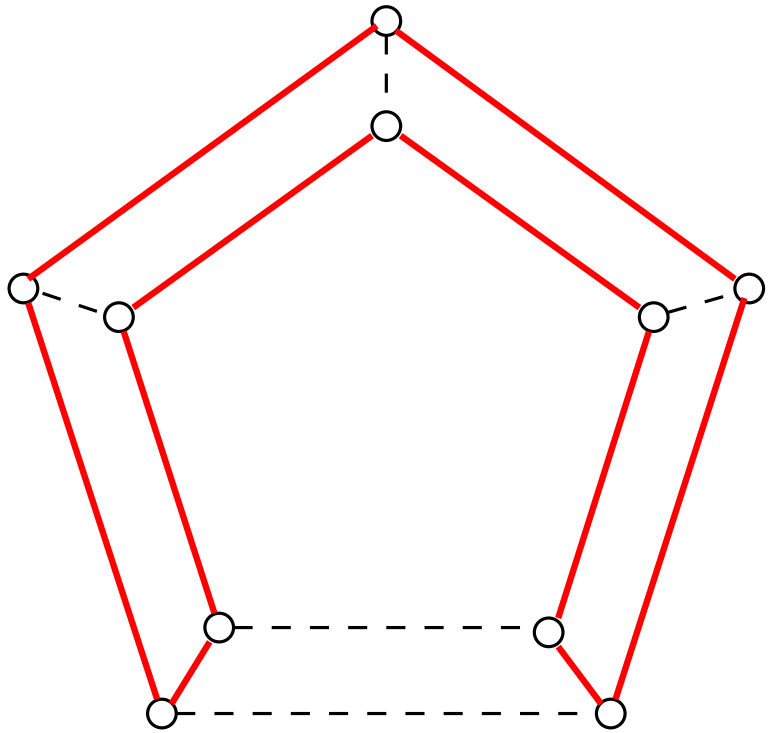
Mathematisches Problem:
Gibt es im jeweils gegebenen Graphen einen Hamiltonschen Kreis?



"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

Mathematisches Problem:
Gibt es im jeweils gegebenen Graphen einen Hamiltonschen Kreis?

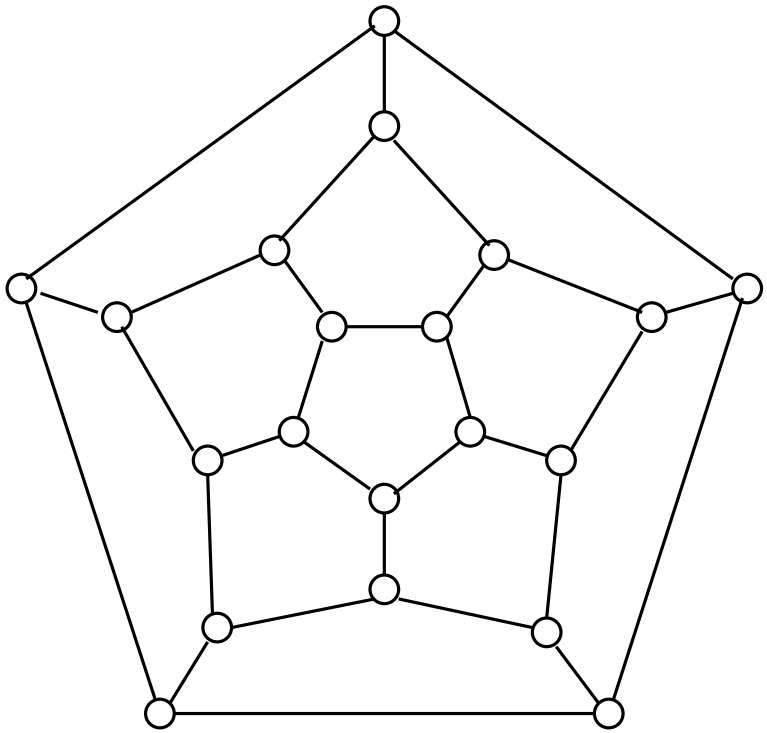


Ja und die Richtigkeit einer Lösung ist leicht zu überprüfen.

"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

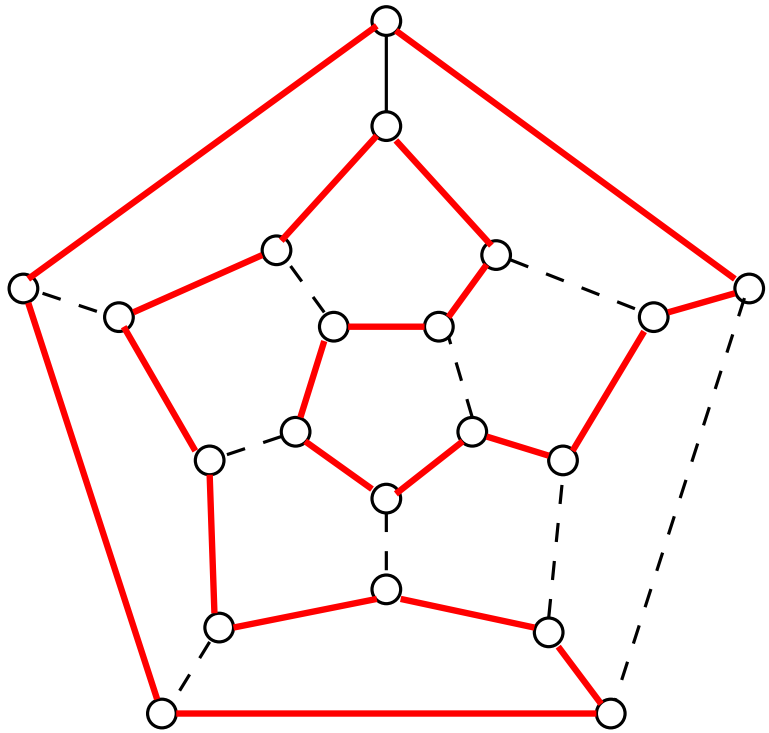
Mathematisches Problem:
Gibt es im jeweils gegebenen Graphen einen Hamiltonschen Kreis?



"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

Mathematisches Problem:
Gibt es im jeweils gegebenen Graphen einen Hamiltonschen Kreis?

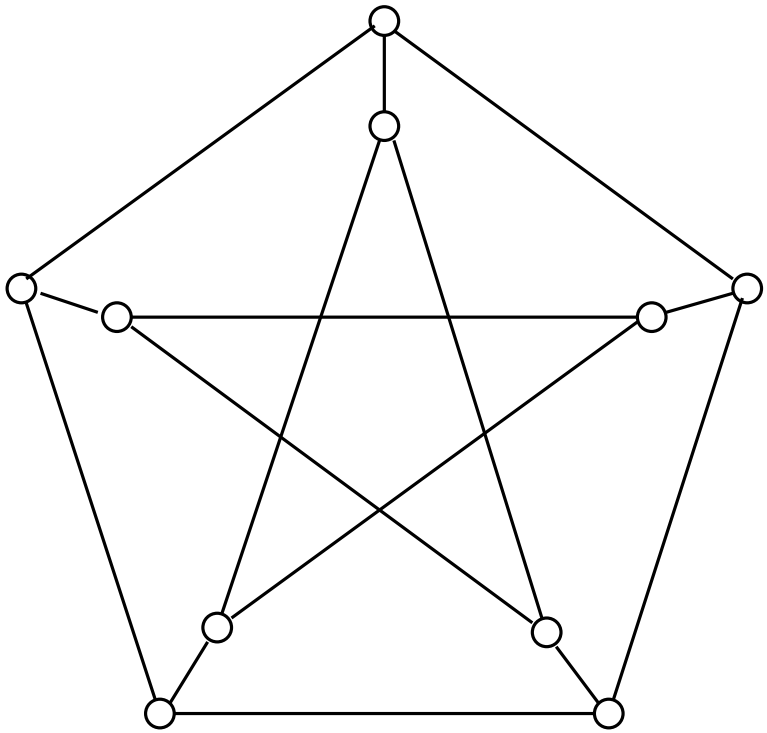


Ja und die Richtigkeit einer Lösung ist leicht zu überprüfen.

"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

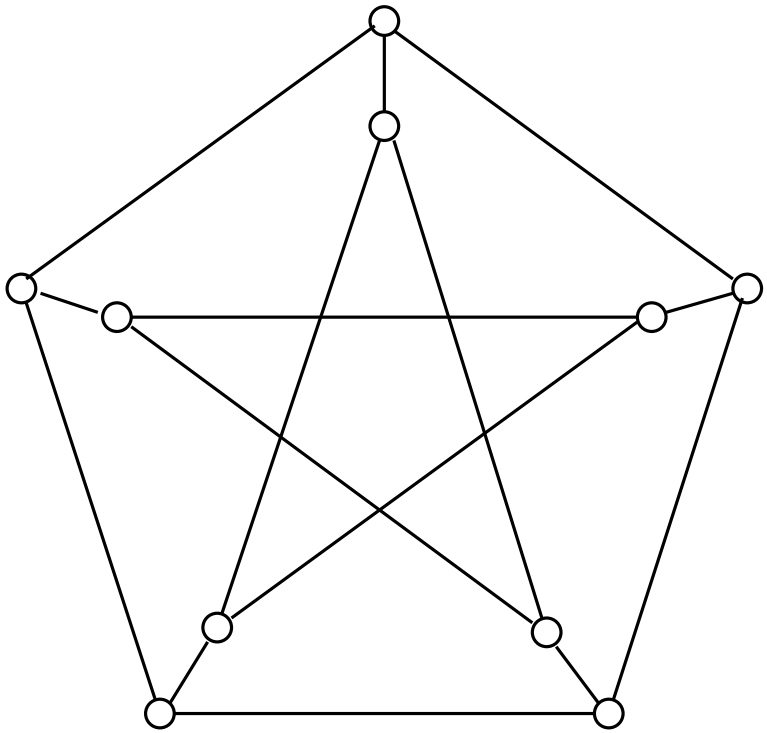
Mathematisches Problem:
Gibt es im jeweils gegebenen Graphen einen Hamiltonschen Kreis?



"Vereinfachtes" Rundreiseproblem: Hamiltonscher Kreis

Eine Rundreise in einem nicht notwendig vollständigen Graphen, die jeden Knoten genau einmal besucht, heißt **Hamiltonscher Kreis**.

Mathematisches Problem:
Gibt es im jeweils gegebenen Graphen einen Hamiltonschen Kreis?



Dass es keine Lösung gibt, ist anscheinend nur schwer nachzuweisen.

Übersicht

Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

- Graphenmodelle und kürzeste Wege
- Rundreisen
- **Algorithmische Komplexität, ein Milleniums-Problem**
- Fluss- und Transportprobleme
- Betriebsplanerstellung
- Schlussbemerkungen

Ausflug in die Algorithmische Komplexität

Eine **Instanz** eines Problems ist eine konkrete Realisierung des Problems,

z.B. ein TSP: $n = 3$, $d_{12} = d_{23} = d_{31} = 1$, $d_{21} = d_{32} = d_{13} = 2$

Ausflug in die Algorithmische Komplexität

Eine **Instanz** eines Problems ist eine konkrete Realisierung des Problems,

z.B. ein TSP: $n = 3$, $d_{12} = d_{23} = d_{31} = 1$, $d_{21} = d_{32} = d_{13} = 2$

Die (Kodierungs-)**Länge** einer Instanz ist die Anzahl der Zeichen/Bits, die benötigt werden, um alle Daten der Instanz zu beschreiben,

TSP-Bsp: „`3;1 2 1;1 3 2;2 1 2;2 3 1;3 1 1;3 2 2`“= 38 Zeichen

Ausflug in die Algorithmische Komplexität

Eine **Instanz** eines Problems ist eine konkrete Realisierung des Problems,

z.B. ein TSP: $n = 3$, $d_{12} = d_{23} = d_{31} = 1$, $d_{21} = d_{32} = d_{13} = 2$

Die (Kodierungs-)**Länge** einer Instanz ist die Anzahl der Zeichen/Bits, die benötigt werden, um alle Daten der Instanz zu beschreiben,

TSP-Bsp: „`3;1 2 1;1 3 2;2 1 2;2 3 1;3 1 1;3 2 2`“= 38 Zeichen

Ein Problem heißt **polynomial („effizient“) lösbar**, wenn es einen (deterministischen) Algorithmus und ein Polynom p gibt mit der Eigenschaft, dass der Algorithmus für jede Eingabe-Instanz nach höchstens p(Länge der Instanz) vielen Operationen eine passende Antwort liefert.

Ausflug in die Algorithmische Komplexität

Eine **Instanz** eines Problems ist eine konkrete Realisierung des Problems,

z.B. ein TSP: $n = 3$, $d_{12} = d_{23} = d_{31} = 1$, $d_{21} = d_{32} = d_{13} = 2$

Die (Kodierungs-)**Länge** einer Instanz ist die Anzahl der Zeichen/Bits, die benötigt werden, um alle Daten der Instanz zu beschreiben,

TSP-Bsp: „`3;1 2 1;1 3 2;2 1 2;2 3 1;3 1 1;3 2 2`“= 38 Zeichen

Ein Problem heißt **polynomial („effizient“) lösbar**, wenn es einen (deterministischen) Algorithmus und ein Polynom p gibt mit der Eigenschaft, dass der Algorithmus für jede Eingabe-Instanz nach höchstens p(Länge der Instanz) vielen Operationen eine passende Antwort liefert.

Die **Klasse P** umfasst alle polynomial lösbaren Probleme.

(z.B. Euler-Touren, kürzeste Wege, Lineare Optimierung, etc.)

Ausflug in die Algorithmische Komplexität

Eine **Instanz** eines Problems ist eine konkrete Realisierung des Problems,

z.B. ein TSP: $n = 3$, $d_{12} = d_{23} = d_{31} = 1$, $d_{21} = d_{32} = d_{13} = 2$

Die (Kodierungs-)**Länge** einer Instanz ist die Anzahl der Zeichen/Bits, die benötigt werden, um alle Daten der Instanz zu beschreiben,

TSP-Bsp: „`3;1 2 1;1 3 2;2 1 2;2 3 1;3 1 1;3 2 2`“= 38 Zeichen

Ein Problem heißt **polynomial („effizient“) lösbar**, wenn es einen (deterministischen) Algorithmus und ein Polynom p gibt mit der Eigenschaft, dass der Algorithmus für jede Eingabe-Instanz nach höchstens p(Länge der Instanz) vielen Operationen eine passende Antwort liefert.

Die **Klasse P** umfasst alle polynomial lösbaren Probleme.

(z.B. Euler-Touren, kürzeste Wege, Lineare Optimierung, etc.)

Die **Klasse NP** umfasst alle Probleme, für die eine als „Lösung“ angegebene Antwort auf eine Instanz in polynomialer Zeit auf ihre Richtigkeit überprüft werden kann.

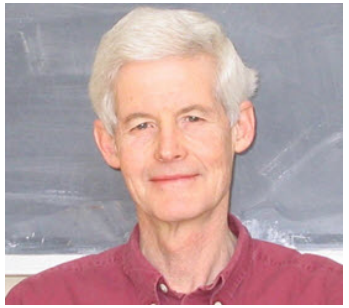
NP steht für **nichtdeterministisch polynomial**.

Es gilt $\mathrm{P} \subseteq \mathrm{NP}$

Erstaunlich:
Es gibt „schwerste“ Probleme in NP.

Kann man diese polynomial lösen, dann auch alle anderen!

Man nennt solche Probleme **NP-vollständig**.

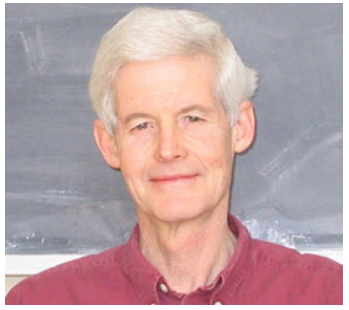


Stephen Cook, 1971

Erstaunlich:
Es gibt „schwerste“ Probleme in NP.

Kann man diese polynomial lösen, dann auch alle anderen!

Man nennt solche Probleme **NP-vollständig**.



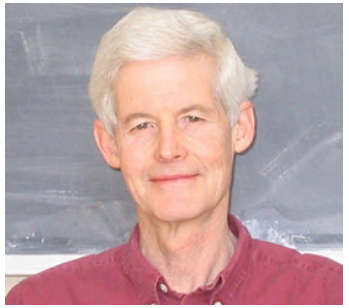
Stephen Cook, 1971

Noch erstaunlicher:
Das Hamiltonscher-Kreis-Problem ist NP-vollständig. [Karp 1972]

Erstaunlich:
Es gibt „schwerste“ Probleme in NP.

Kann man diese polynomial lösen, dann auch alle anderen!

Man nennt solche Probleme **NP-vollständig**.



Stephen Cook, 1971

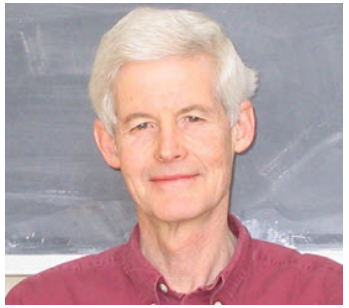
Noch erstaunlicher:
Das Hamiltonscher-Kreis-Problem ist NP-vollständig. [Karp 1972]

TSP in der Form „Gibt es eine Rundreise der Länge K?“ ist gleich schwer, denn man kann damit das Hamiltonscher-Kreis-Problem lösen.

Erstaunlich:
Es gibt „schwerste“ Probleme in NP.

Kann man diese polynomial lösen, dann auch alle anderen!

Man nennt solche Probleme **NP-vollständig**.



Stephen Cook, 1971

Noch erstaunlicher:
Das Hamiltonscher-Kreis-Problem ist NP-vollständig. [Karp 1972]

TSP in der Form „Gibt es eine Rundreise der Länge K?“ ist gleich schwer, denn man kann damit das Hamiltonscher-Kreis-Problem lösen.

Gibt es einen polynomialen Algorithmus für TSP?
Ist P=NP?

Clay Mathematics Institute http://www.claymath.org/millennium/P_vs_NP/

Clay Mathematics Institute

Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

P vs NP Problem

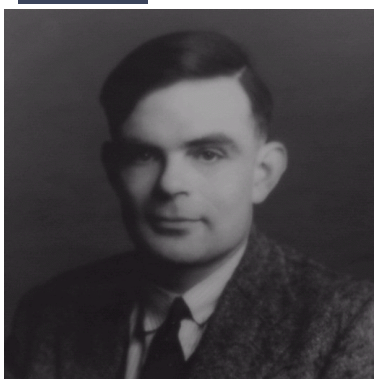
Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

The Millennium Problems

Official Problem Description — Stephen Cook

Lecture by Vijaya Ramachandran at University of Texas (video)

Minesweeper



Übersicht

Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

- Graphenmodelle und kürzeste Wege
- Rundreisen
- Algorithmische Komplexität, ein Milleniums-Problem
- **Fluss- und Transportprobleme**
- Betriebsplanerstellung
- Schlussbemerkungen

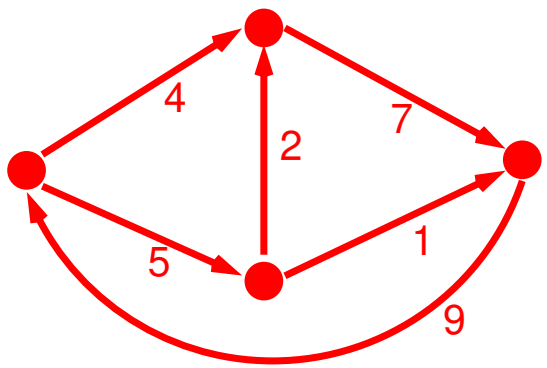
Netzwerke und Flüsse

Modellierungswerkzeug: Transportprobleme, Evakuierungspläne, Auftrags-, Verkehrsplanung, Kommunikationsnetzwerke, ...

Netzwerke und Flüsse

Modellierungswerkzeug: Transportprobleme, Evakuierungspläne, Auftrags-, Verkehrsplanung, Kommunikationsnetzwerke, ...

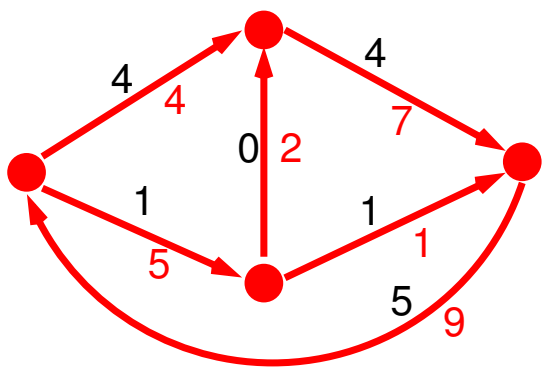
Ein **Netzwerk** besteht aus einem Graphen mit gerichteten Kanten (kurz Digraph) und (Kanten-)**Kapazitäten**.



Netzwerke und Flüsse

Modellierungswerkzeug: Transportprobleme, Evakuierungspläne, Auftrags-, Verkehrsplanung, Kommunikationsnetzwerke, ...

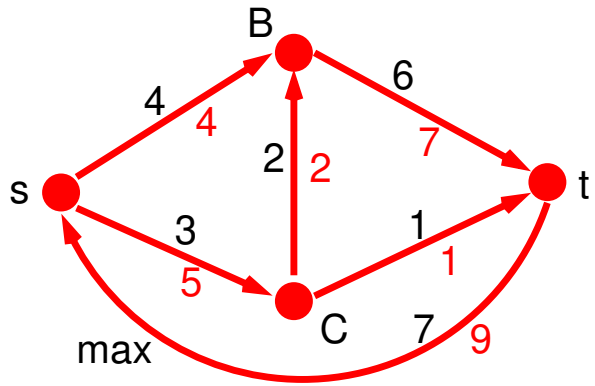
Ein **Netzwerk** besteht aus einem Graphen mit gerichteten Kanten (kurz Digraph) und (Kanten-)**Kapazitäten**.



Eine Wertzuweisung an dessen Kanten innerhalb dieser Kapazitäten heißt (zulässiger) **Fluss**, falls in jedem Knoten die **Flusserhaltungsgleichungen** (es fließt gleichviel hinein wie hinaus) eingehalten werden.

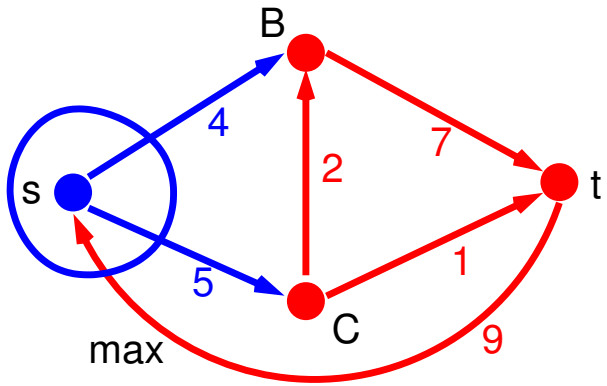
Maximale s-t-Flüsse, Minimale s-t-Schnitte

Ein maximaler s-t-Fluss hat zwei ausgezeichnete Knoten s (die **Quelle**) und t (die **Senke**) und maximiert den **Flusswert** auf der Kante (t, s) zurück von t nach s.



Maximale s-t-Flüsse, Minimale s-t-Schnitte

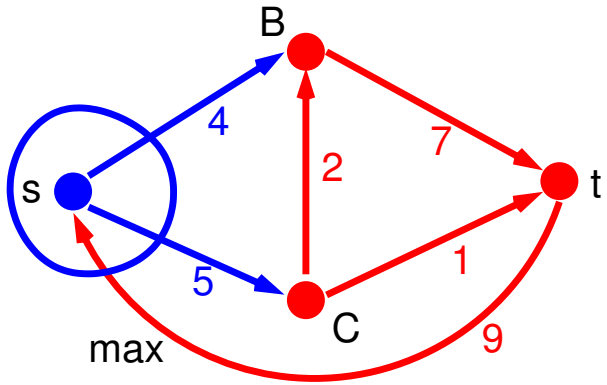
Ein maximaler s-t-Fluss hat zwei ausgezeichnete Knoten s (die **Quelle**) und t (die **Senke**) und maximiert den **Flusswert** auf der Kante (t, s) zurück von t nach s.



Jede Teilmenge der Knoten, die die Quelle aber nicht die Senke enthält, definiert einen s-t-**Schnitt**: die Kanten, die aus der Teilmenge führen.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Ein maximaler s-t-Fluss hat zwei ausgezeichnete Knoten s (die **Quelle**) und t (die **Senke**) und maximiert den **Flusswert** auf der Kante (t, s) zurück von t nach s.

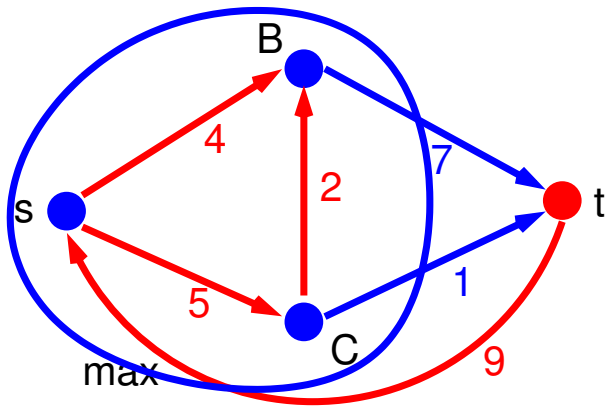


Jede Teilmenge der Knoten, die die Quelle aber nicht die Senke enthält, definiert einen s-t-**Schnitt**: die Kanten, die aus der Teilmenge führen.

Über diese fließt höchstens die Summe ihrer Kapazitäten, der **Wert** des s-t-Schnitts $\rightarrow$ obere Schranke für jeden s-t-Fluss.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Ein maximaler s-t-Fluss hat zwei ausgezeichnete Knoten s (die **Quelle**) und t (die **Senke**) und maximiert den **Flusswert** auf der Kante (t, s) zurück von t nach s.

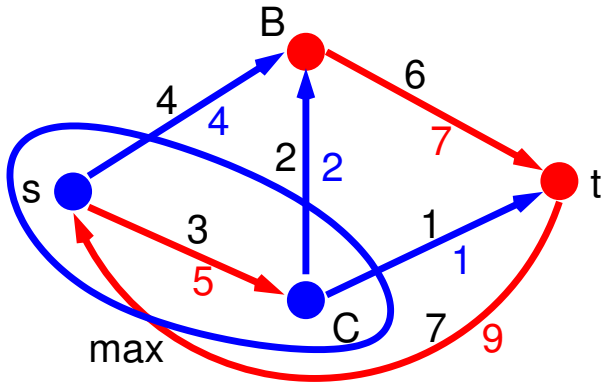


Jede Teilmenge der Knoten, die die Quelle aber nicht die Senke enthält, definiert einen s-t-**Schnitt**: die Kanten, die aus der Teilmenge führen.

Über diese fließt höchstens die Summe ihrer Kapazitäten, der **Wert** des s-t-Schnitts $\rightarrow$ obere Schranke für jeden s-t-Fluss.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Ein maximaler s-t-Fluss hat zwei ausgezeichnete Knoten s (die **Quelle**) und t (die **Senke**) und maximiert den **Flusswert** auf der Kante (t, s) zurück von t nach s.

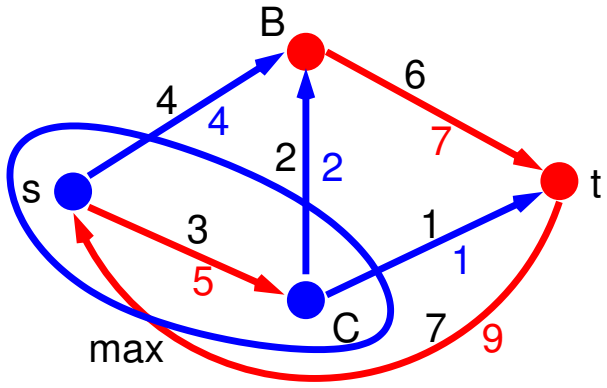


Jede Teilmenge der Knoten, die die Quelle aber nicht die Senke enthält, definiert einen s-t-**Schnitt**: die Kanten, die aus der Teilmenge führen.

Über diese fließt höchstens die Summe ihrer Kapazitäten, der **Wert** des s-t-Schnitts $\rightarrow$ obere Schranke für jeden s-t-Fluss.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Ein maximaler s-t-Fluss hat zwei ausgezeichnete Knoten s (die **Quelle**) und t (die **Senke**) und maximiert den **Flusswert** auf der Kante (t, s) zurück von t nach s.



Jede Teilmenge der Knoten, die die Quelle aber nicht die Senke enthält, definiert einen s-t-**Schnitt**: die Kanten, die aus der Teilmenge führen.

Über diese fließt höchstens die Summe ihrer Kapazitäten, der **Wert** des s-t-Schnitts $\rightarrow$ obere Schranke für jeden s-t-Fluss.

Satz (Max-Flow Min-Cut Theorem von Ford & Fulkerson 1954)
Der maximale Wert eines s-t-Flusses ist gleich dem minimalen Wert eines s-t-Schnitts.

Anstoß: der Harris-Ross Report 1955 [Schrijver 2003]

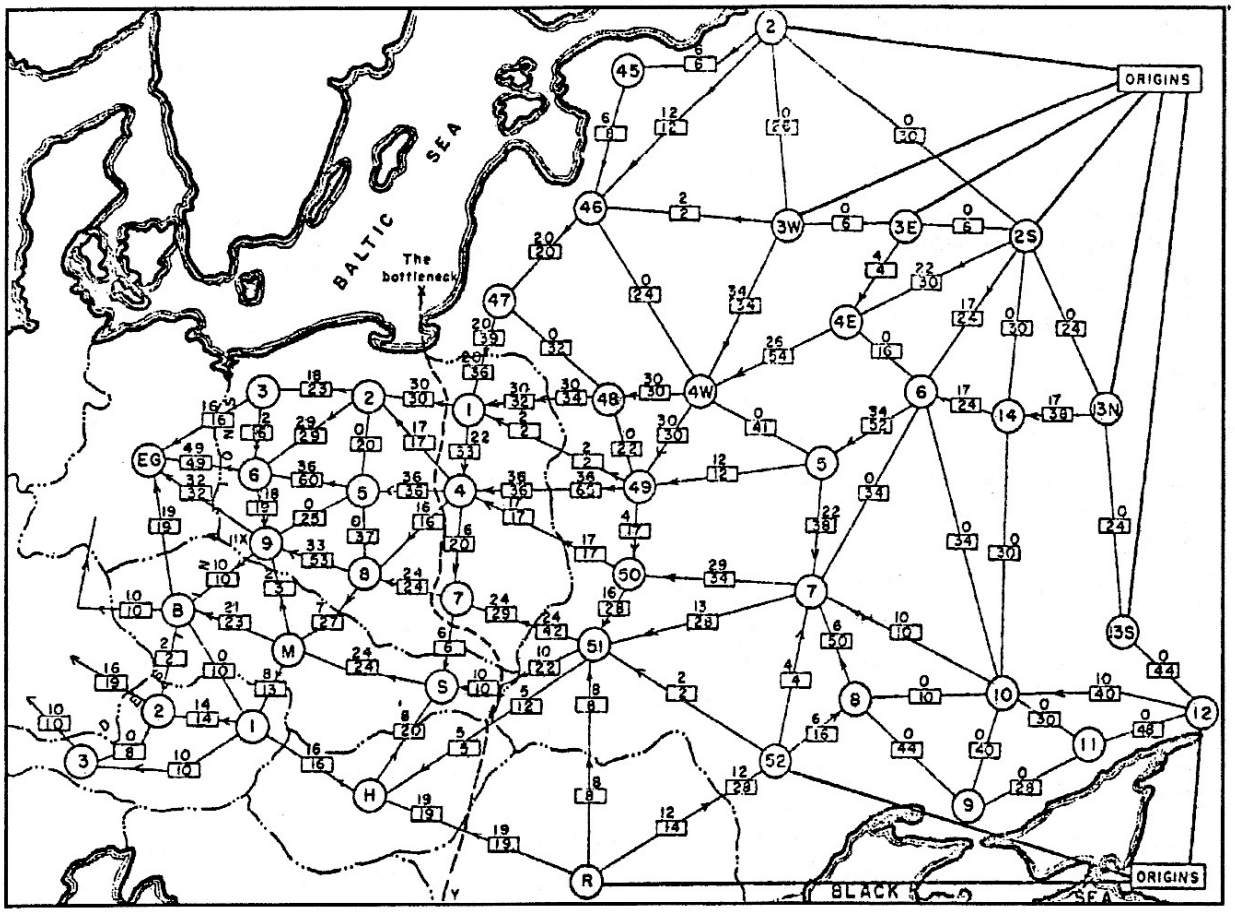
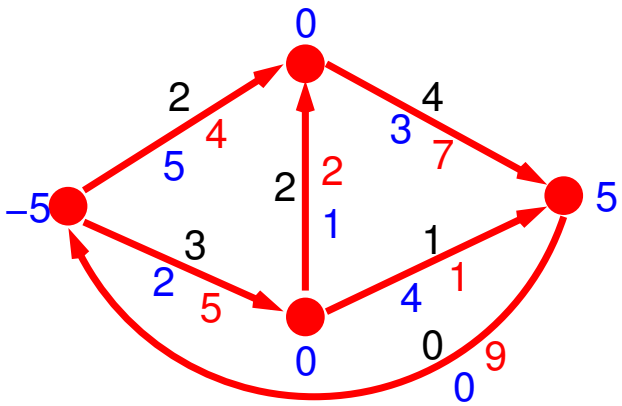


Figure 10.2

From Harris and Ross [1955]: Schematic diagram of the railway network of the Western Soviet Union and East European countries, with a maximum flow of value 163,000 tons from Russia to Eastern Europe, and a cut of capacity 163,000 tons indicated as 'The bottleneck'.

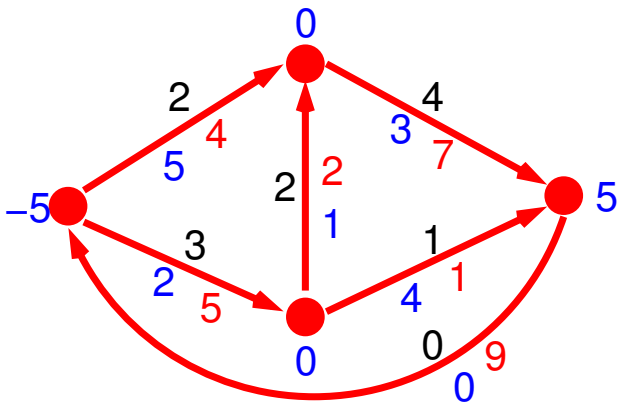
Minimale-Kosten-Flüsse (Min-Cost-Flow)

Die Flussmenge wird durch Balancen (mit Summe 0) auf den Knoten vorgegeben, pro Flusseinheit fallen Kantenkosten an.
Finde den günstigsten Fluss.



Minimale-Kosten-Flüsse (Min-Cost-Flow)

Die Flussmenge wird durch Balancen (mit Summe 0) auf den Knoten vorgegeben, pro Flusseinheit fallen Kantenkosten an.
Finde den günstigsten Fluss.

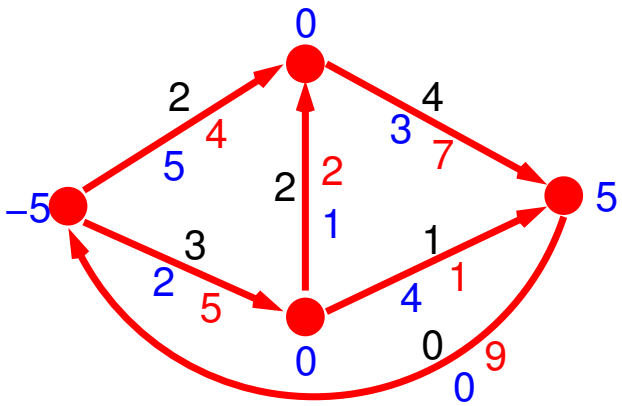


$$\begin{aligned} \min \quad & \begin{bmatrix} 5 & 2 & 1 & 3 & 4 & 0 \end{bmatrix} x \\ \text{s.t.} \quad & \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} x = \begin{bmatrix} -5 \\ 0 \\ 0 \\ 5 \end{bmatrix} \\ & 0 \leq x \leq w \end{aligned}$$

Lässt sich effizient als Lineares Optimierungsproblem schreiben und lösen.

Minimale-Kosten-Flüsse (Min-Cost-Flow)

Die Flussmenge wird durch Balancen (mit Summe 0) auf den Knoten vorgegeben, pro Flusseinheit fallen Kantenkosten an.
Finde den günstigsten Fluss.



$$\begin{aligned} \min \quad & \begin{bmatrix} 5 & 2 & 1 & 3 & 4 & 0 \end{bmatrix} x \\ \text{s.t.} \quad & \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} x = \begin{bmatrix} -5 \\ 0 \\ 0 \\ 5 \end{bmatrix} \\ & 0 \leq x \leq w \end{aligned}$$

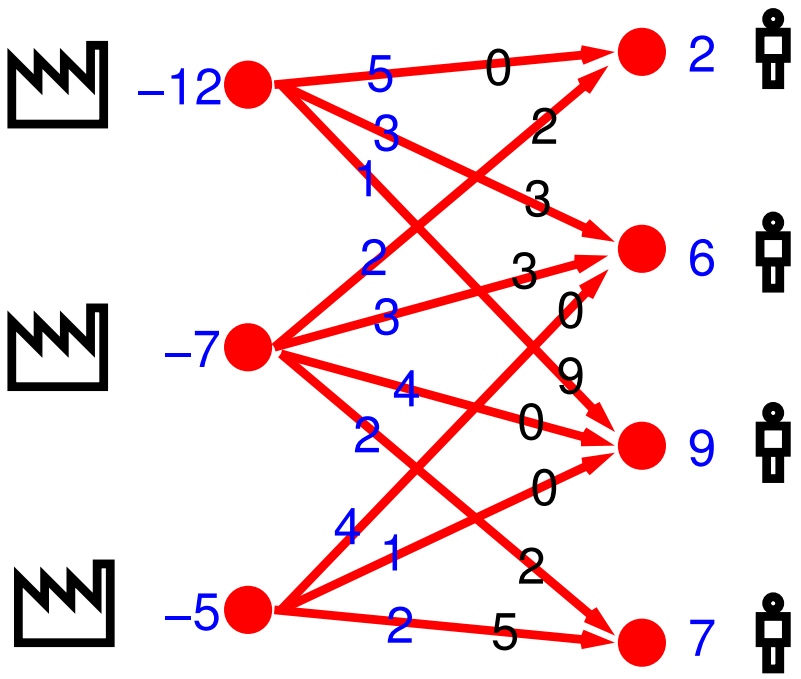
Lässt sich effizient als Lineares Optimierungsproblem schreiben und lösen.

Extrem breite Anwendungsmöglichkeiten, beliebtes Modellierungswerkzeug

Beispiel: Transportproblem

Eine Firma mit mehreren Produktionsstandorten hat mehrere Kunden zu beliefern. Wie geschieht dies am günstigsten unter Berücksichtigung der Transportkosten?

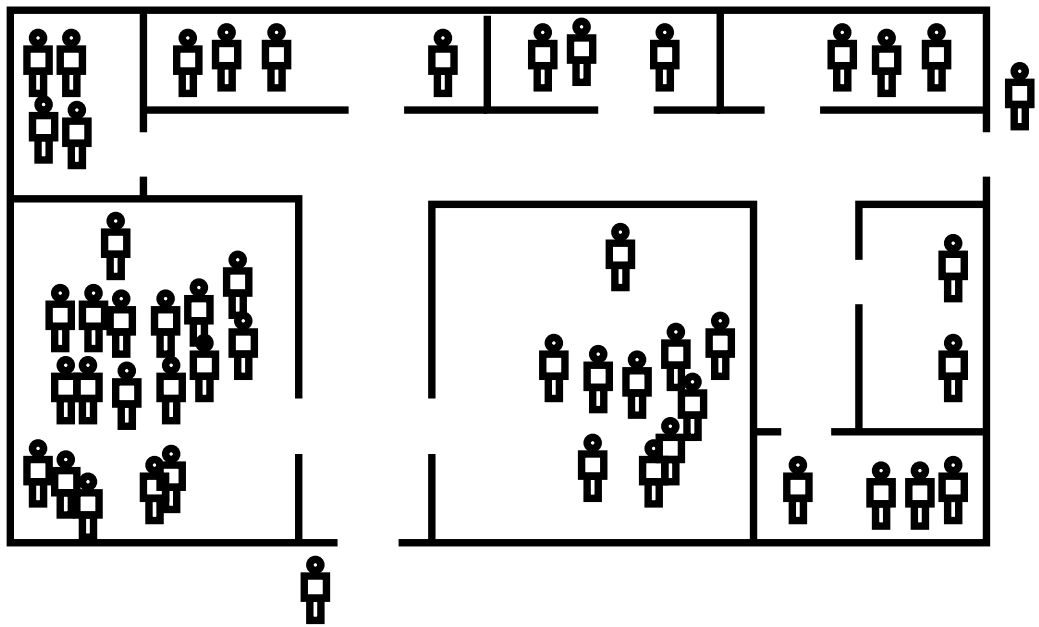
Gaspard Monge
1746-1818 (1781)



Beachte: Nur ein Produkttyp!

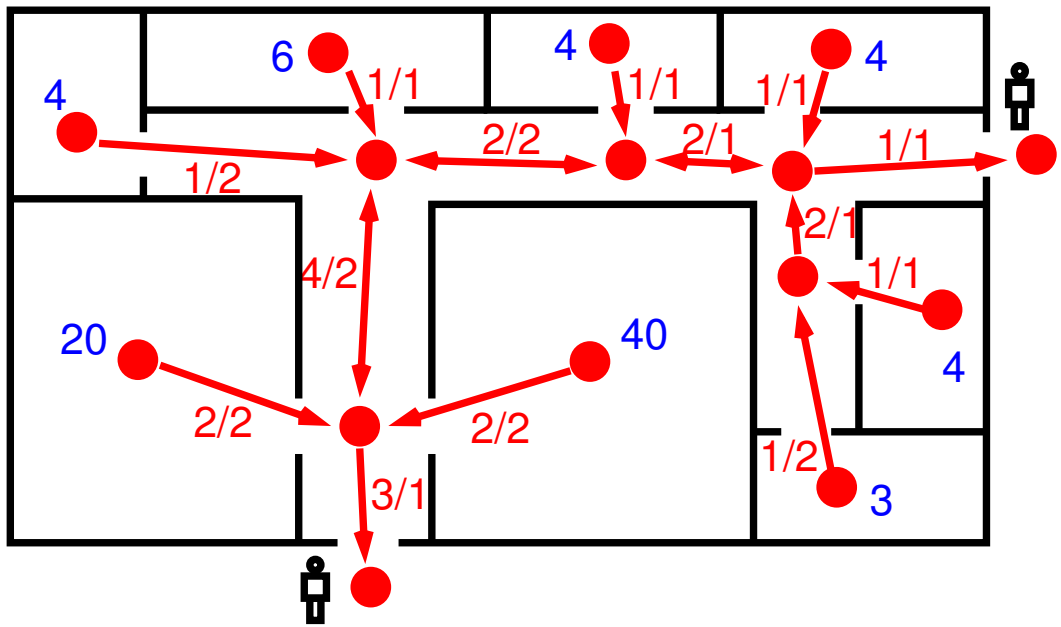
Beispiel: Evakuierungsplanung

Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Beispiel: Evakuierungsplanung

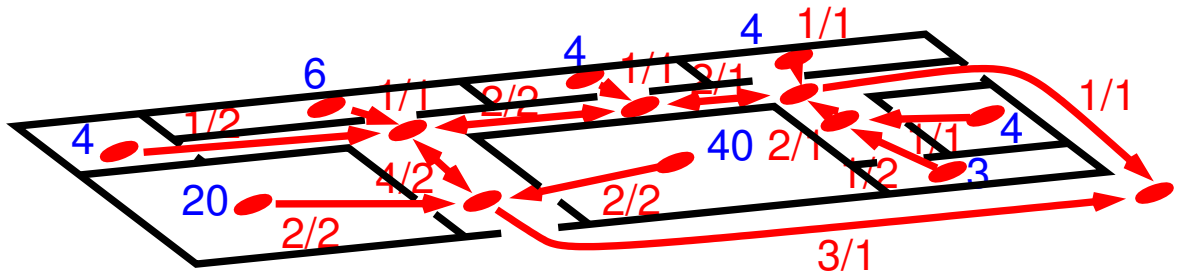
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Kanten mit Kapazität und Durchquerungszeit

Beispiel: Evakuierungsplanung

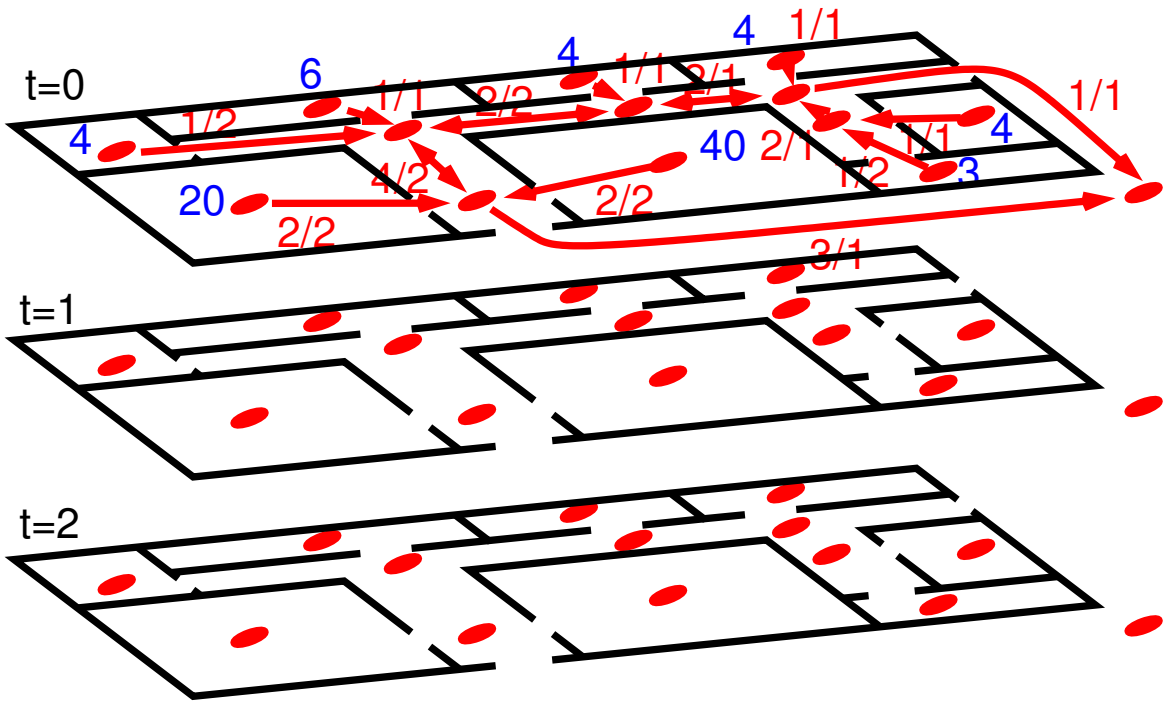
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Flusswerte ändern sich mit der Zeit

Beispiel: Evakuierungsplanung

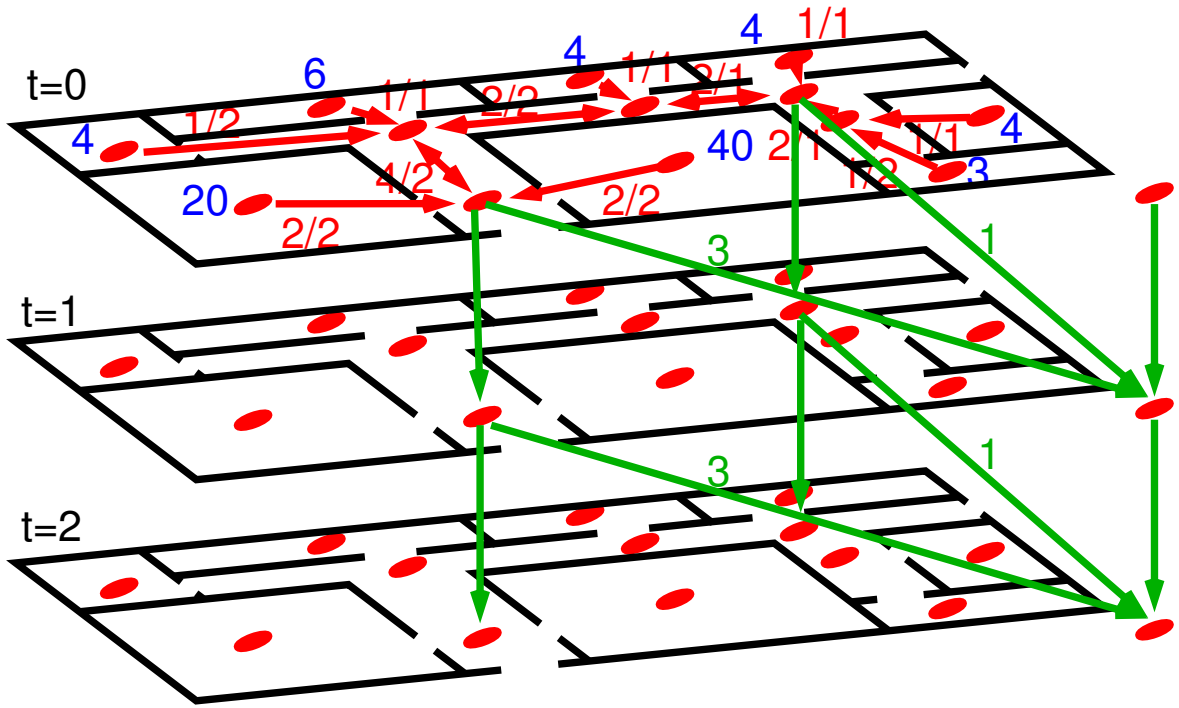
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Diskretisierung der Zeit, ein Niveau pro Zeiteinheit

Beispiel: Evakuierungsplanung

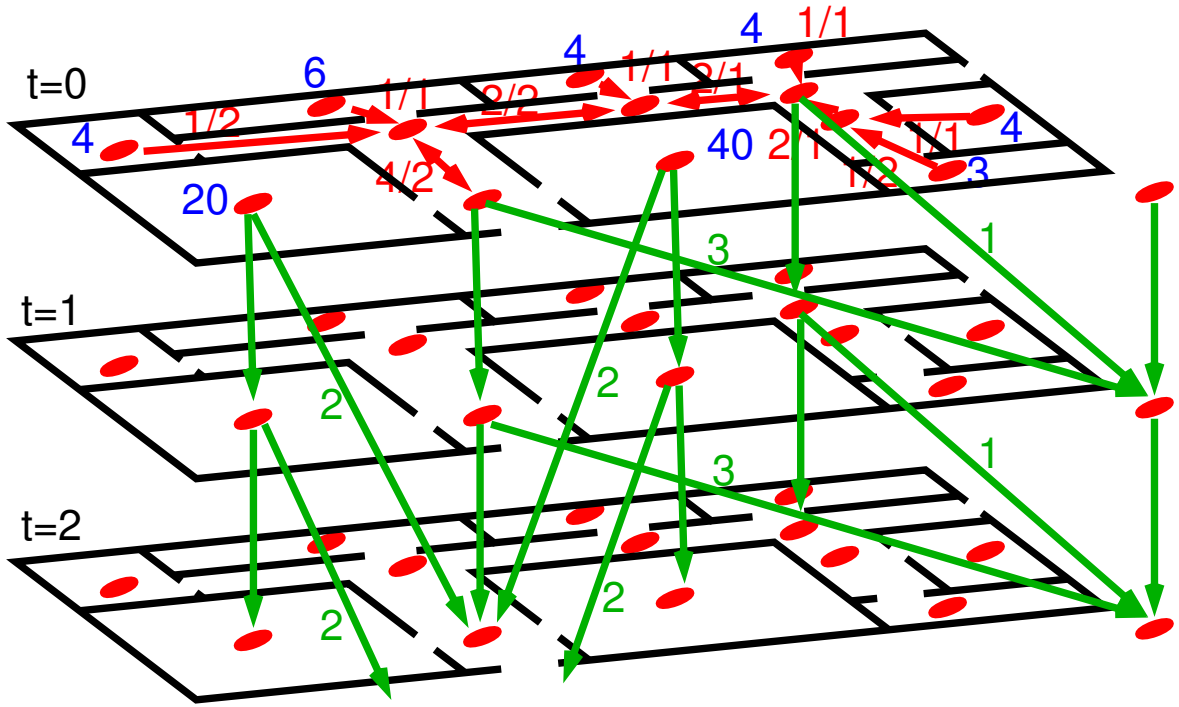
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Durchquerungszeit verbindet Niveaus, Kapazität bleibt

Beispiel: Evakuierungsplanung

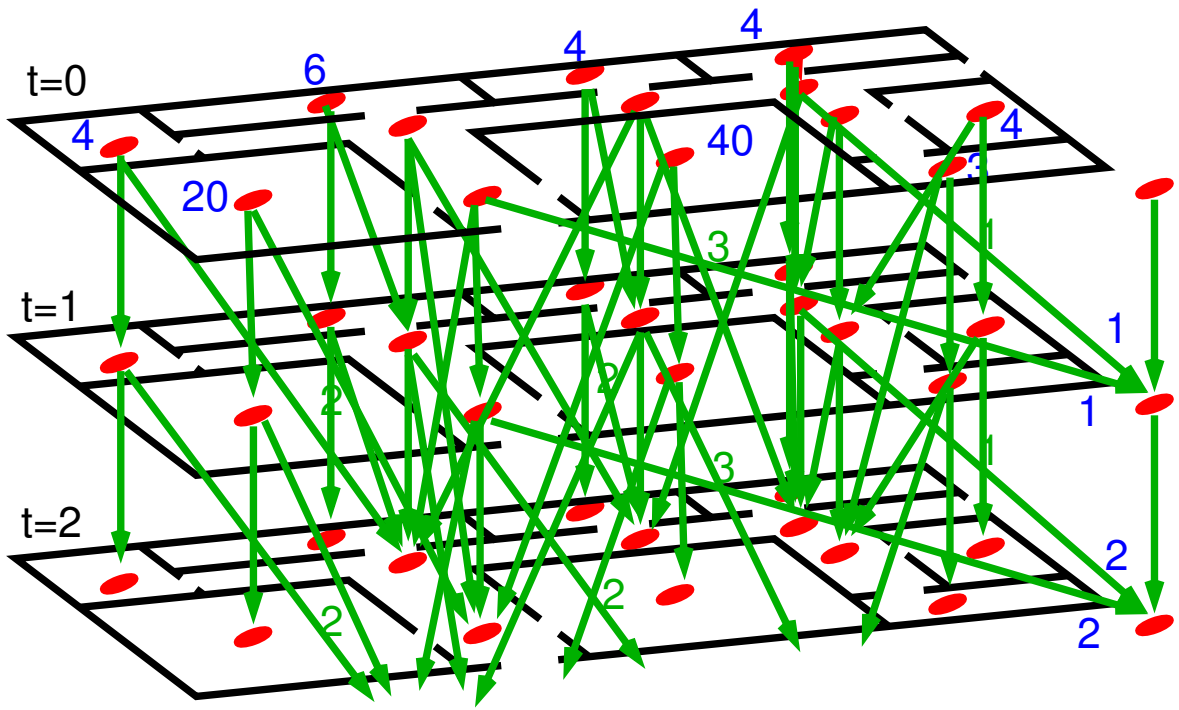
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Durchquerungszeit verbindet Niveaus, Kapazität bleibt

Beispiel: Evakuierungsplanung

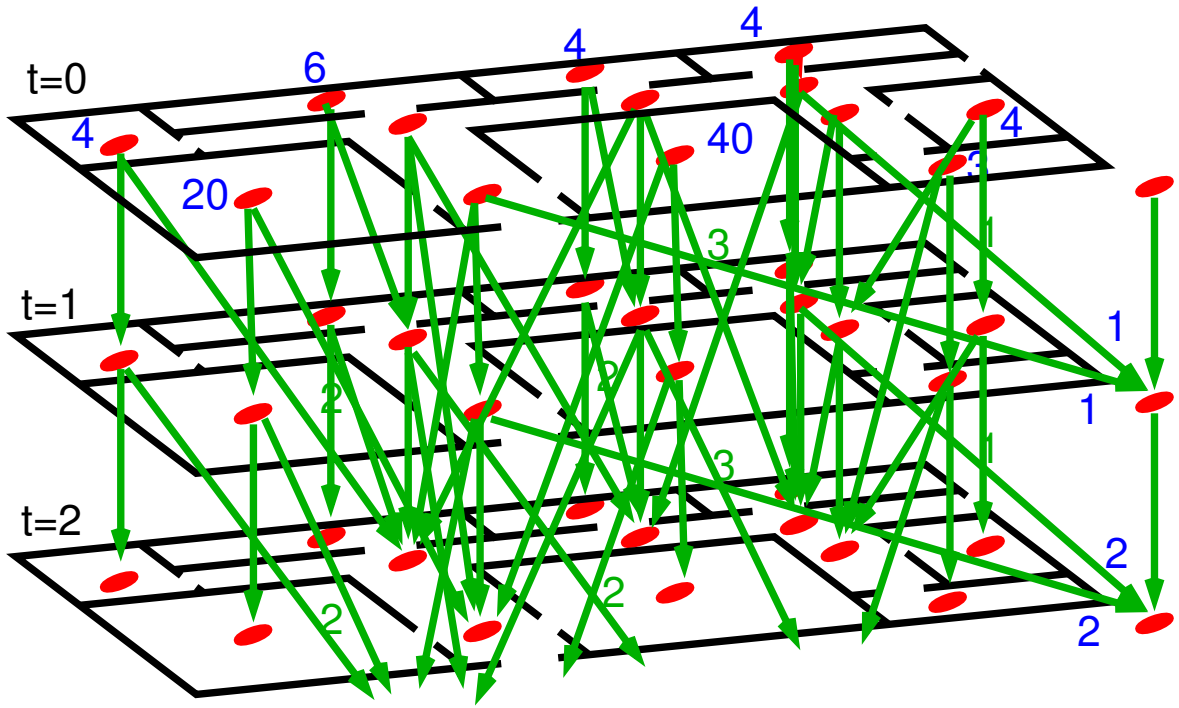
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Steigende Kosten auf den Ausgangskanten für rasches Verlassen

Beispiel: Evakuierungsplanung

Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.

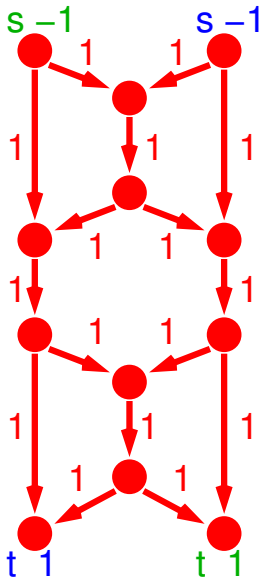


Steigende Kosten auf den Ausgangskanten für rasches Verlassen

Ansatz nur ok, wenn Personen nicht unterschieden werden müssen!

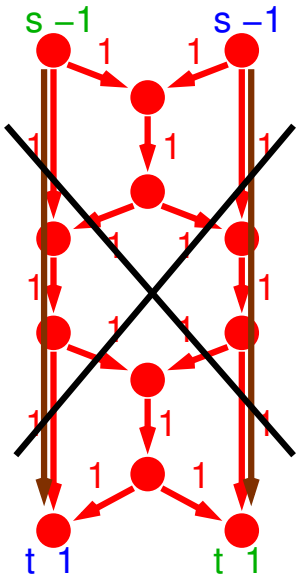
Mehrgüterflussprobleme (Multicommodity Flow)

In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



Mehrgüterflussprobleme (Multicommodity Flow)

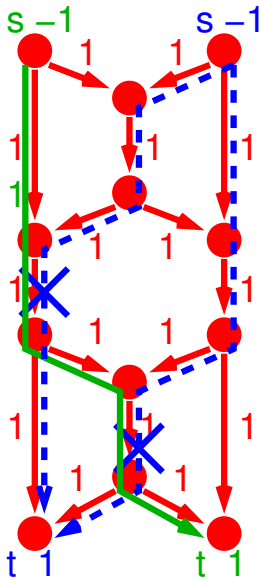
In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



Mischen verboten!

Mehrgüterflussprobleme (Multicommodity Flow)

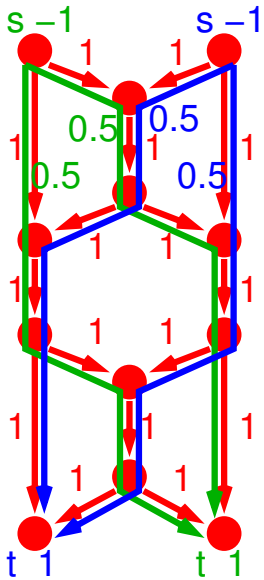
In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



ganzz. geht nicht

Mehrgüterflussprobleme (Multicommodity Flow)

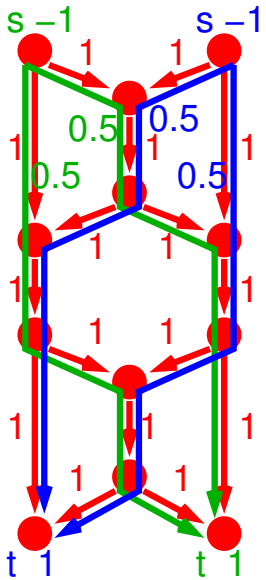
In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



gebrochen geht

Mehrgüterflussprobleme (Multicommodity Flow)

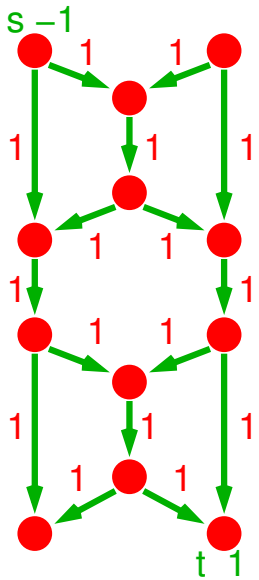
In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



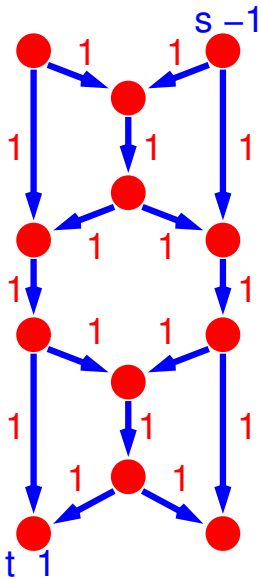
gebrochen geht

Umsetzung

$\longrightarrow$



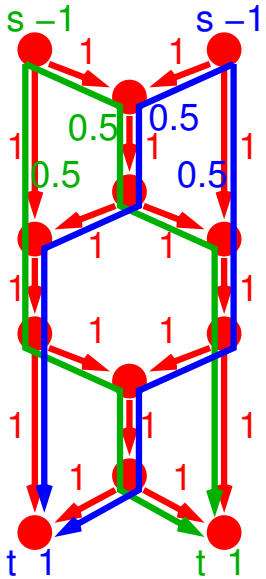
Kopie 1



Kopie 2

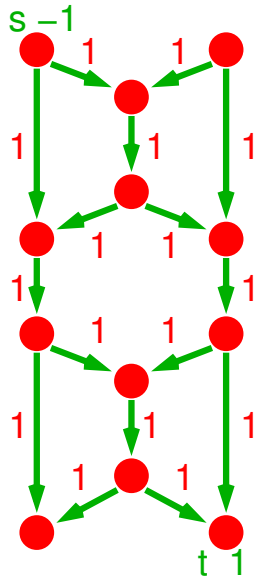
Mehrgüterflussprobleme (Multicommodity Flow)

In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.

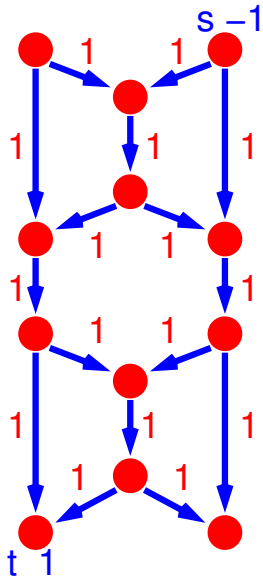


gebrochen geht

Umsetzung $\longrightarrow$



Kopie 1

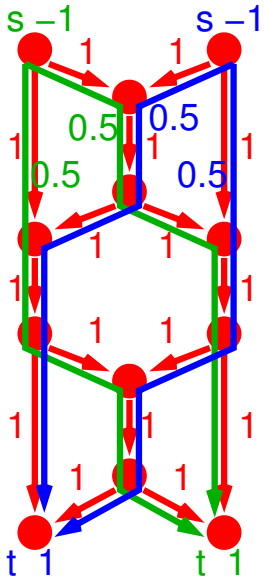


Kopie 2

$$
\begin{array}{lccccc}
\min & c^{(1)\,T} x^{(1)} & + & c^{(2)\,T} x^{(2)} & & \\
\text{s.t.} & A x^{(1)} & & & = & b^{(1)} \\
 & & & A x^{(2)} & = & b^{(2)} \\
 & I x^{(1)} & + & I x^{(2)} & \leq & w \\
 & x^{(1)} \geq 0, & & x^{(2)} \geq 0. & &
\end{array}
$$

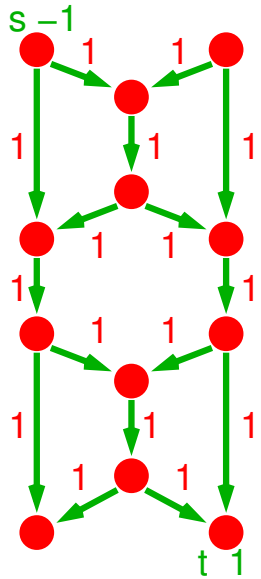
Mehrgüterflussprobleme (Multicommodity Flow)

In einem Netzwerk sind für unterschiedliche Güter mit Quellen/Senken (s_i, t_i) und Flusswerten f_i zulässige Flüsse $x^{(i)}$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.

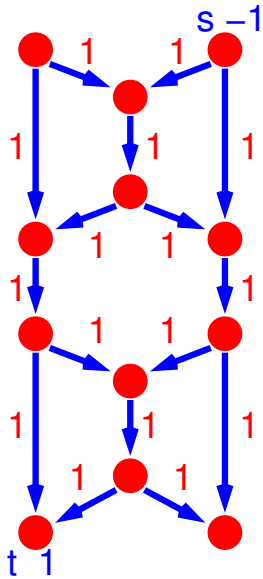


gebrochen geht

Umsetzung $\longrightarrow$



Kopie 1



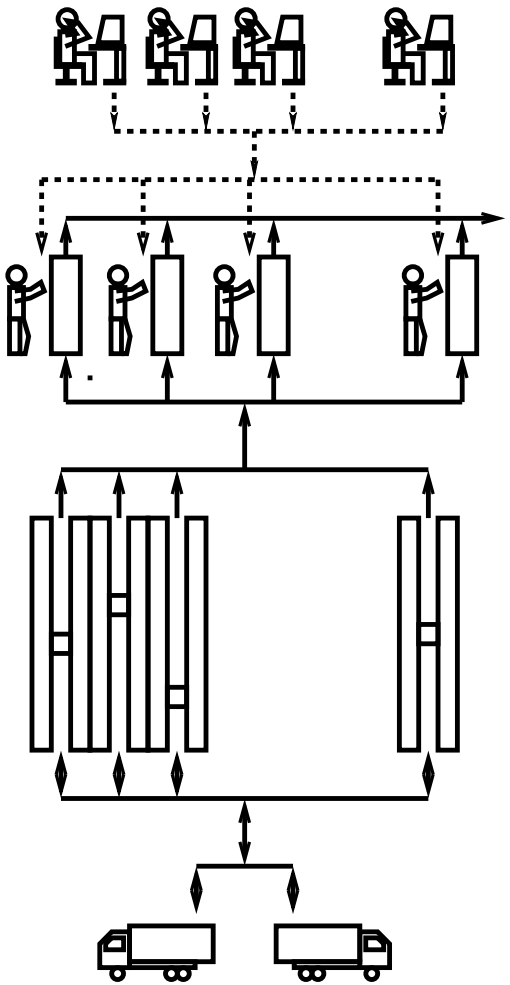
Kopie 2

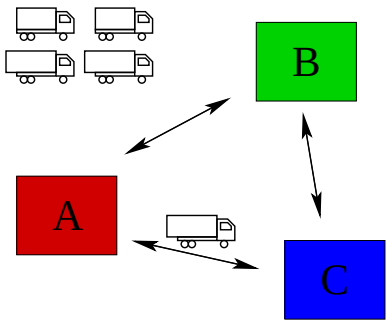
Gebrochen gut lösbar (in P), ganzzahlig SEHR schwer (NP-vollständig)

$$
\begin{array}{lrcrcl}
\min & {c^{(1)}}^T x^{(1)} & + & {c^{(2)}}^T x^{(2)} & & \\
\text{s.t.} & Ax^{(1)} & & & = & b^{(1)} \\
 & & & Ax^{(2)} & = & b^{(2)} \\
 & Ix^{(1)} & + & Ix^{(2)} & \leq & w \\
 & x^{(1)} \geq 0, & & x^{(2)} \geq 0. & &
\end{array}
$$

Beispiel: Transport-Logistik

Paletten sind bedarfsgerecht mit LKWs zwischen Lagern zu verschieben

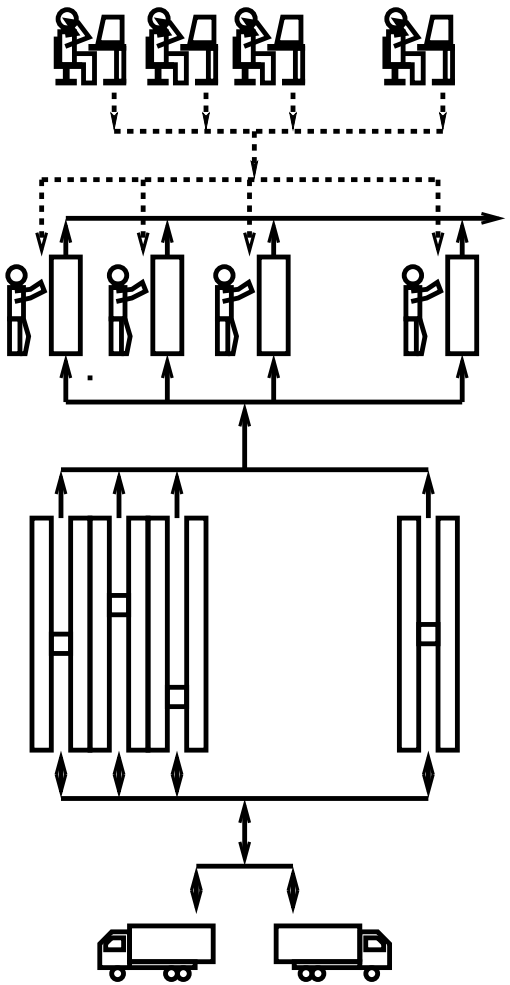


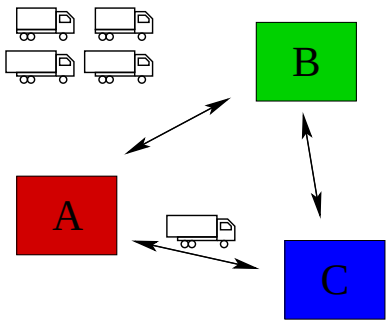



DEMAG
DEMAG
DEMAG

Beispiel: Transport-Logistik

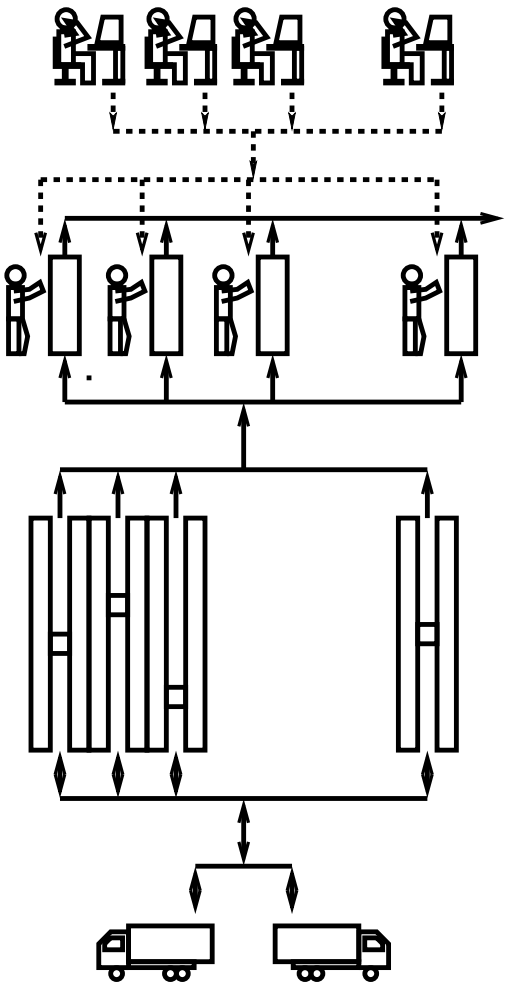
Paletten sind bedarfsgerecht mit LKWs zwischen Lagern zu verschieben

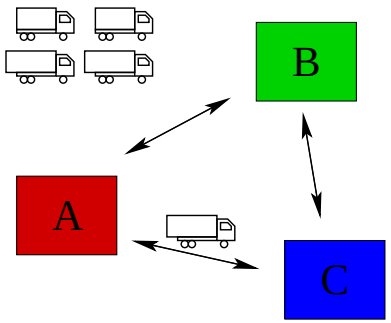




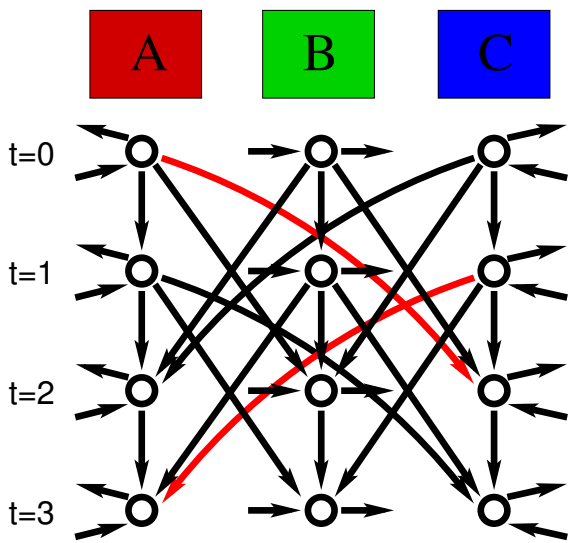
Beispiel: Transport-Logistik

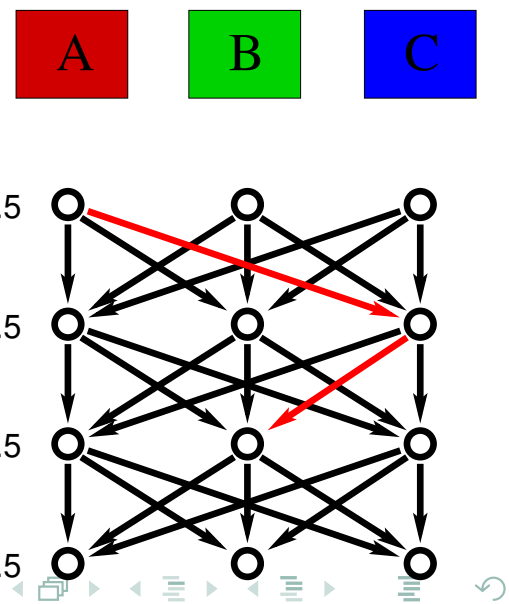
Paletten sind bedarfsgerecht mit LKWs zwischen Lagern zu verschieben





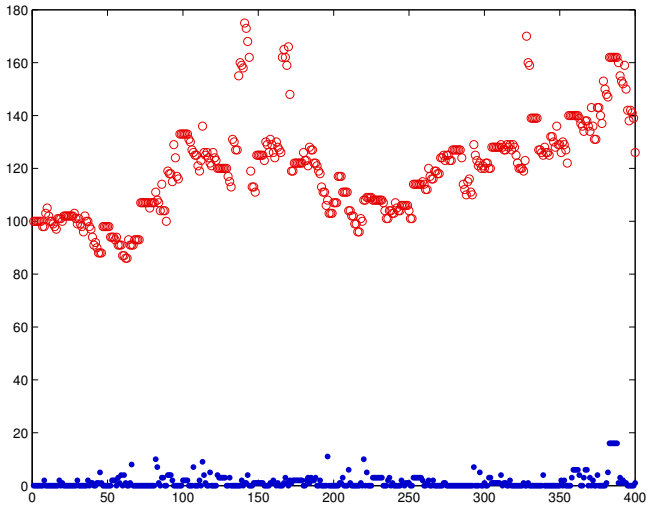
pro Artikel ein Palettengraph



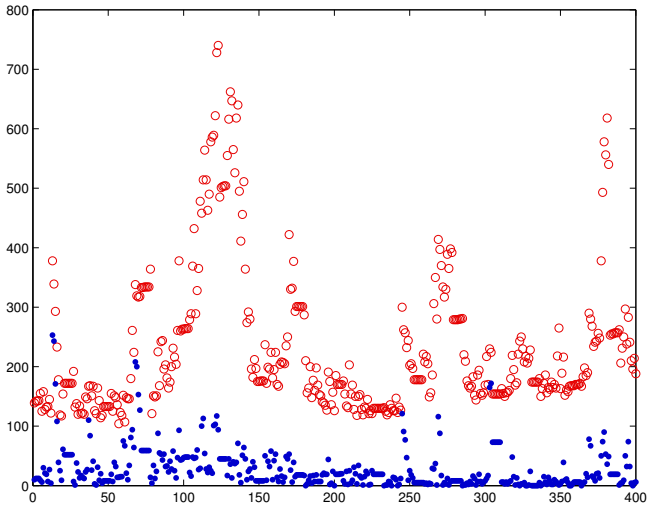


Vergleich Realität zu Simulation für 2 Lagerhäuser

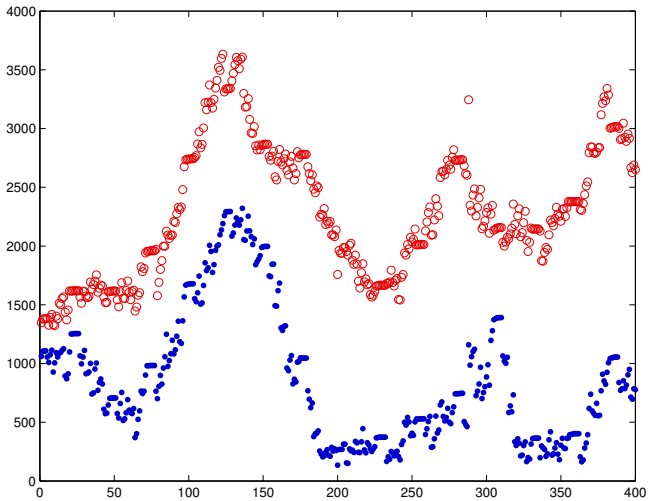
fehlende Palletten



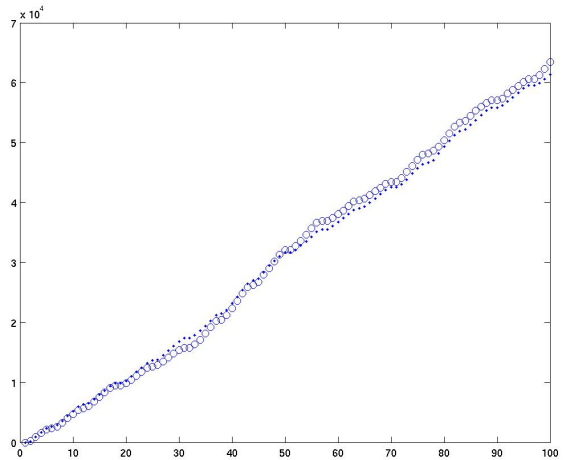
erforderliche Palletten



erforderlich für 90% Sicherheit



Anzahl Transportfahrten



Übersicht

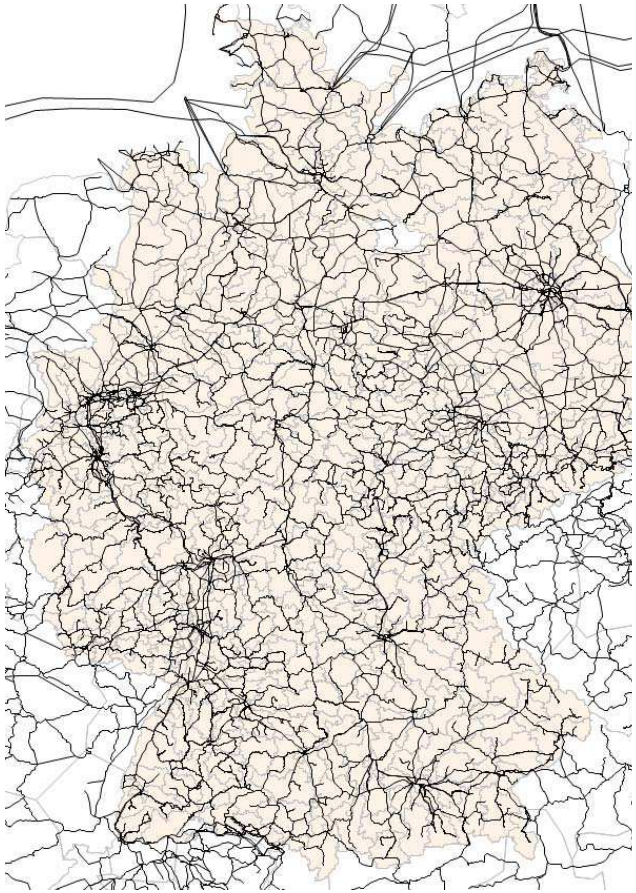
Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

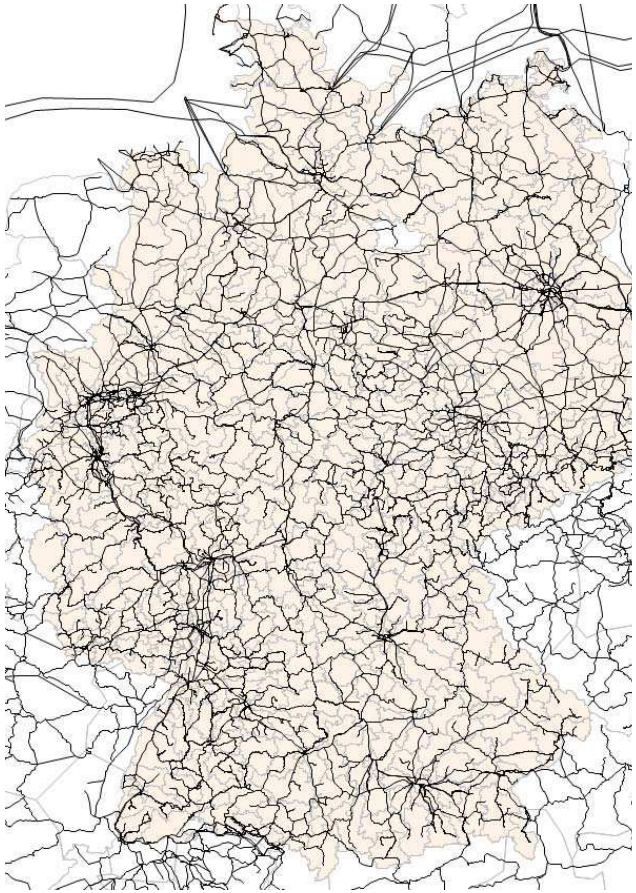
- Graphenmodelle und kürzeste Wege
- Rundreisen
- Algorithmische Komplexität, ein Milleniums-Problem
- Fluss- und Transportprobleme
- **Betriebsplanerstellung**
- Schlussbemerkungen

Beispiel Betriebsplanerstellung im deutschen Bahnnetz



Daten:

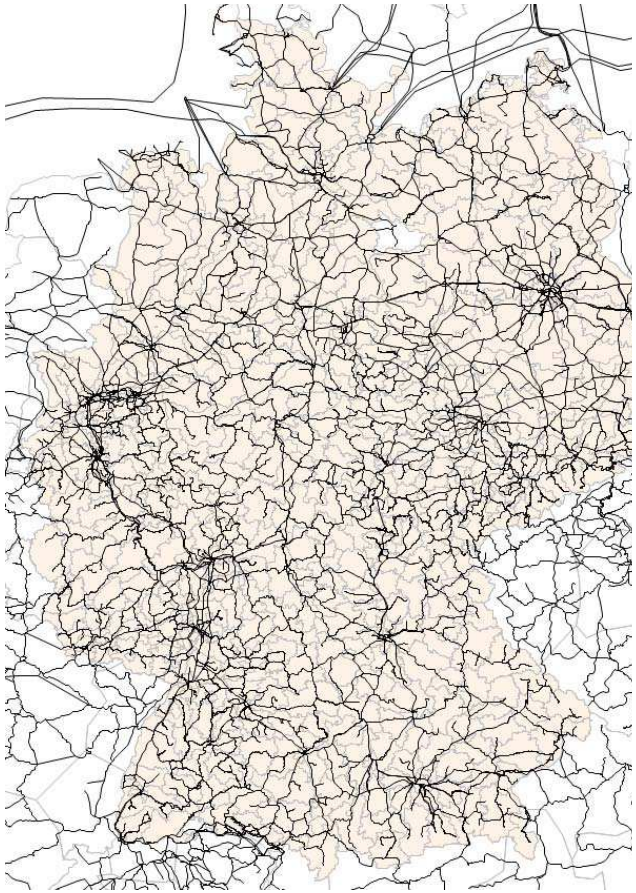
Beispiel Betriebsplanerstellung im deutschen Bahnnetz



Daten:

- Infrastrukturnetz
  - Knoten (Bahnhöfe, Kreuzungen, . . .)
  - Kanten (Gleise).

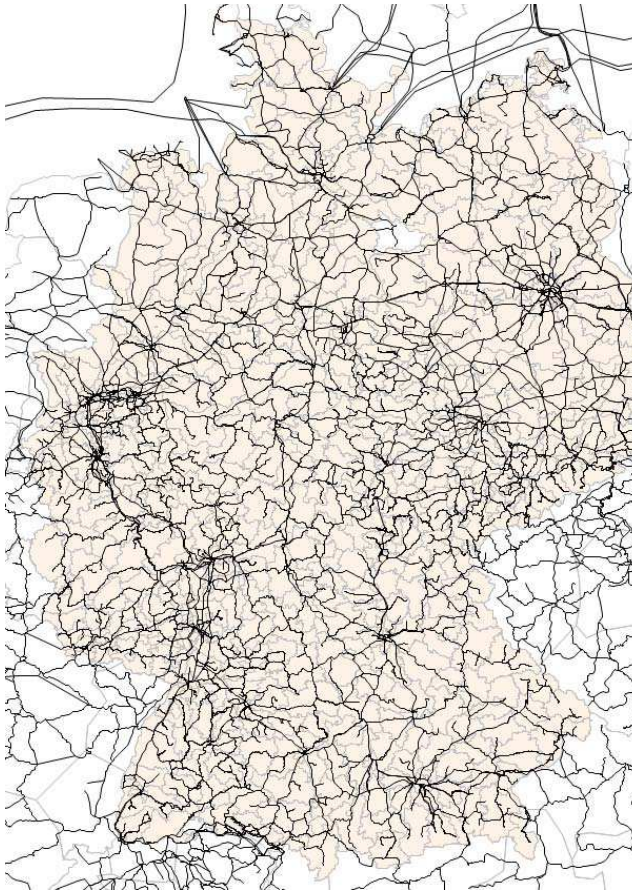
Beispiel Betriebsplanerstellung im deutschen Bahnnetz



Daten:

- Infrastrukturnetz
  - Knoten (Bahnhöfe, Kreuzungen, . . .)
  - Kanten (Gleise).
- Personen- *und* Güterzüge
  - Zugtyp (ICE, RB, Güterzug, . . .)
  - Fahrtstrecke mit Bschleu.-Profil

Beispiel Betriebsplanerstellung im deutschen Bahnnetz

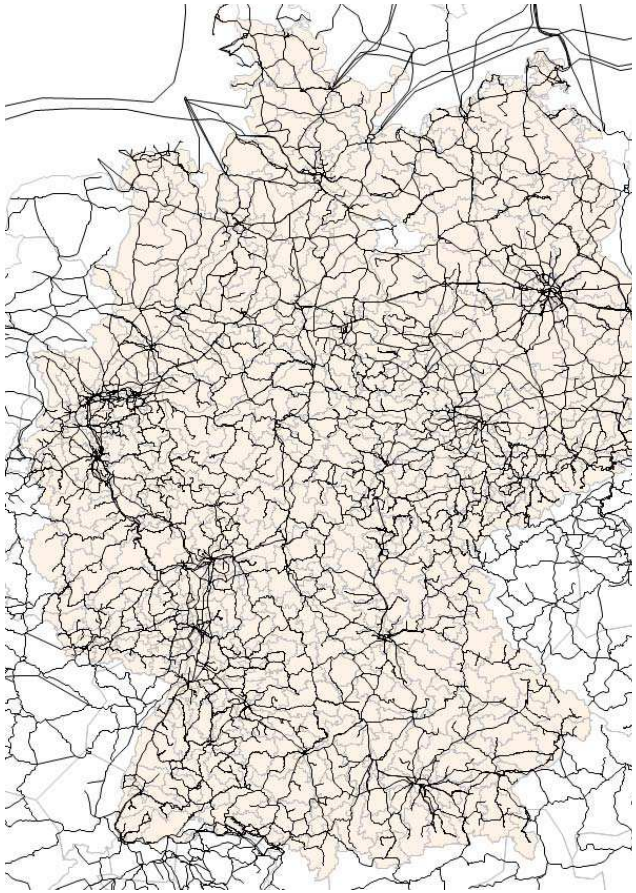


Daten:

- Infrastrukturnetz
  - Knoten (Bahnhöfe, Kreuzungen, . . .)
  - Kanten (Gleise).
- Personen- *und* Güterzüge
  - Zugtyp (ICE, RB, Güterzug, . . .)
  - Fahrtstrecke mit Bschleu.-Profil

Nebenbedingungen:

Beispiel Betriebsplanerstellung im deutschen Bahnnetz



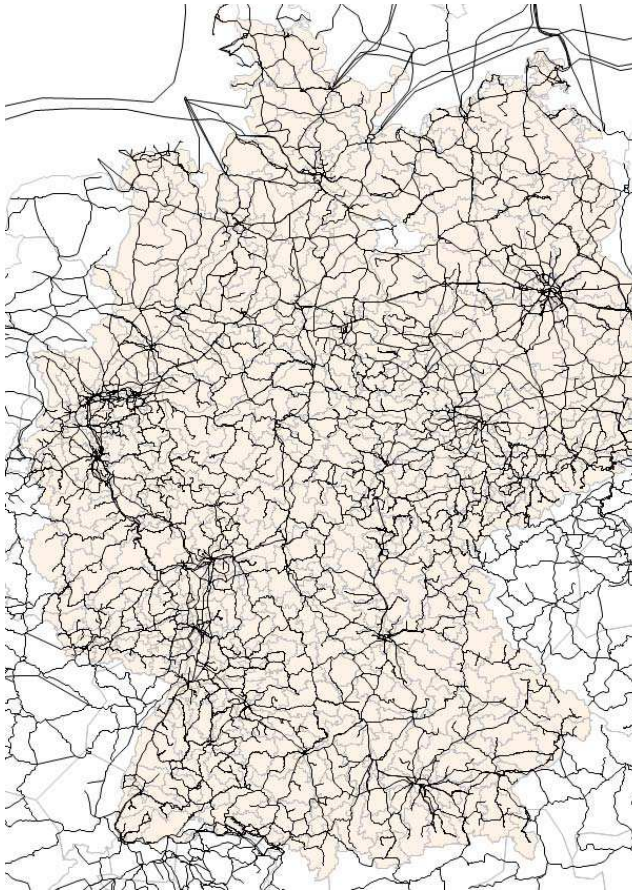
Daten:

- Infrastrukturnetz
  - Knoten (Bahnhöfe, Kreuzungen, . . .)
  - Kanten (Gleise).
- Personen- *und* Güterzüge
  - Zugtyp (ICE, RB, Güterzug, . . .)
  - Fahrtstrecke mit Bschleu.-Profil

Nebenbedingungen:

- Fahrzeiten, Mindestzugfolgezeiten,
- Haltepunktkapazitäten,
- Rahmenfahrplan für Personenzüge.

Beispiel Betriebsplanerstellung im deutschen Bahnnetz



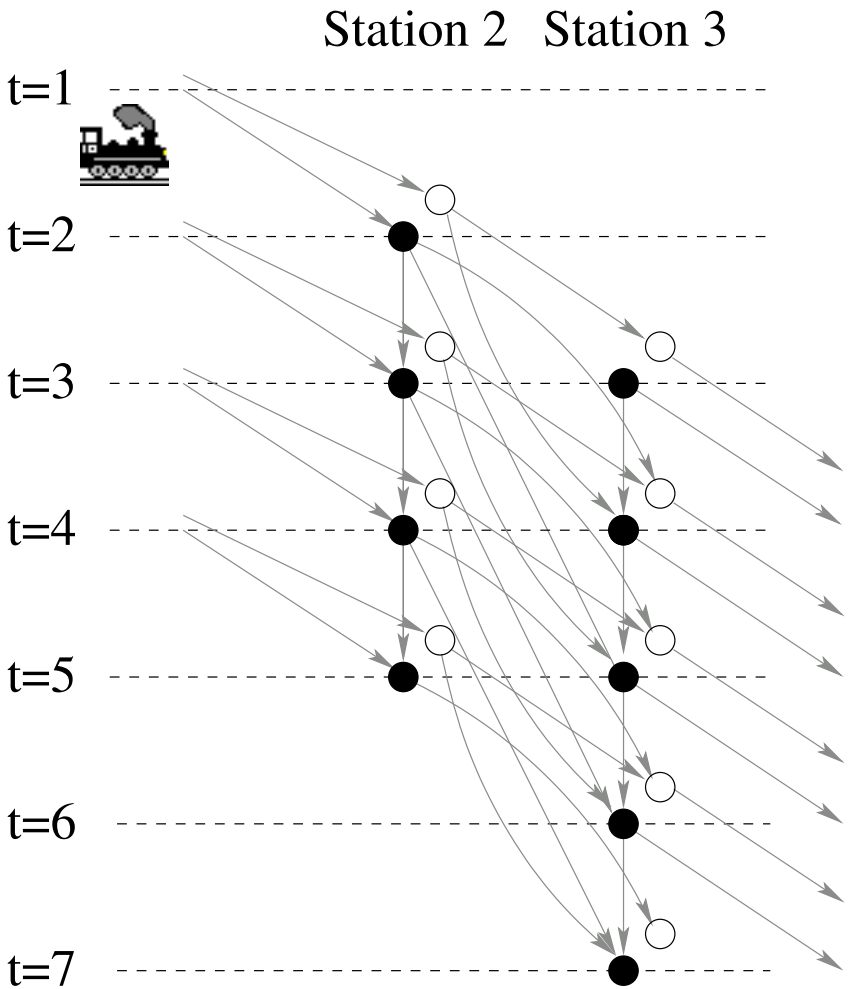
Daten:

- Infrastrukturnetz
  - Knoten (Bahnhöfe, Kreuzungen, . . .)
  - Kanten (Gleise).
- Personen- *und* Güterzüge
  - Zugtyp (ICE, RB, Güterzug, . . .)
  - Fahrtstrecke mit Bschleu.-Profil

Nebenbedingungen:

- **Fahrzeiten, Mindestzugfolgezeiten**,
- Haltepunktkapazitäten,
- Rahmenfahrplan für Personenzüge.

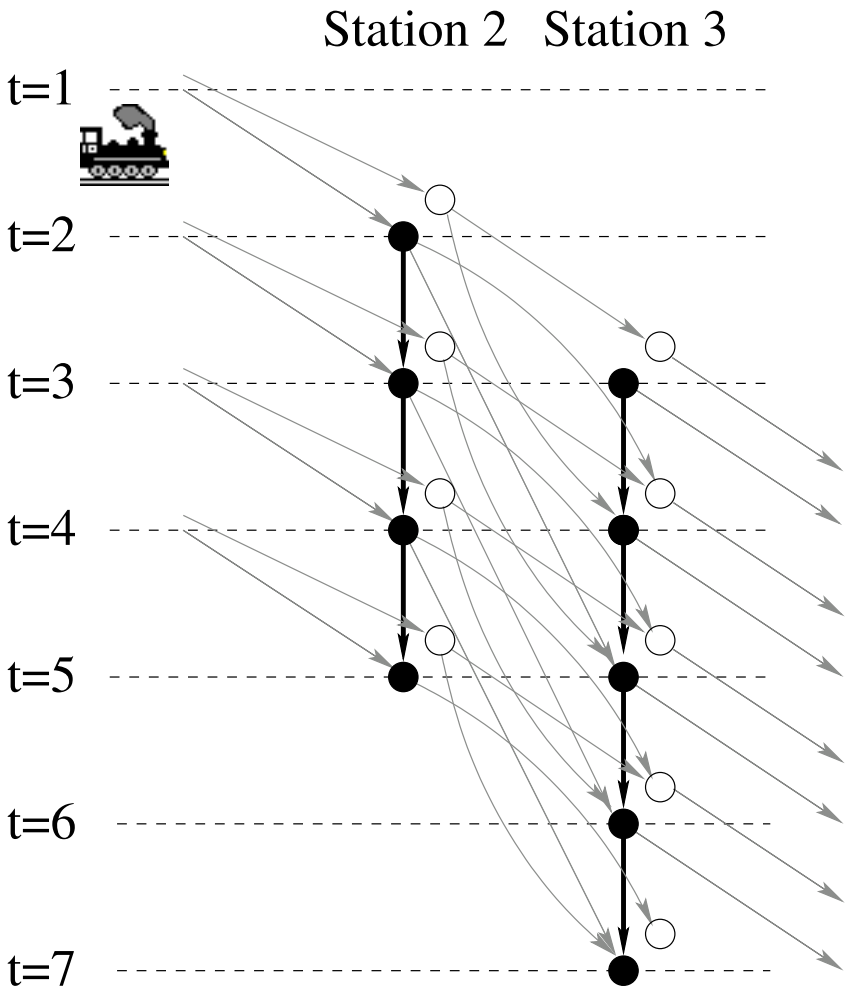
Pro Zug ein Fahrgraph



Pro Haltepunkt und Zeitschritt:

- ○ Durchfahrtknoten
- • Halteknoten

Pro Zug ein Fahrgraph

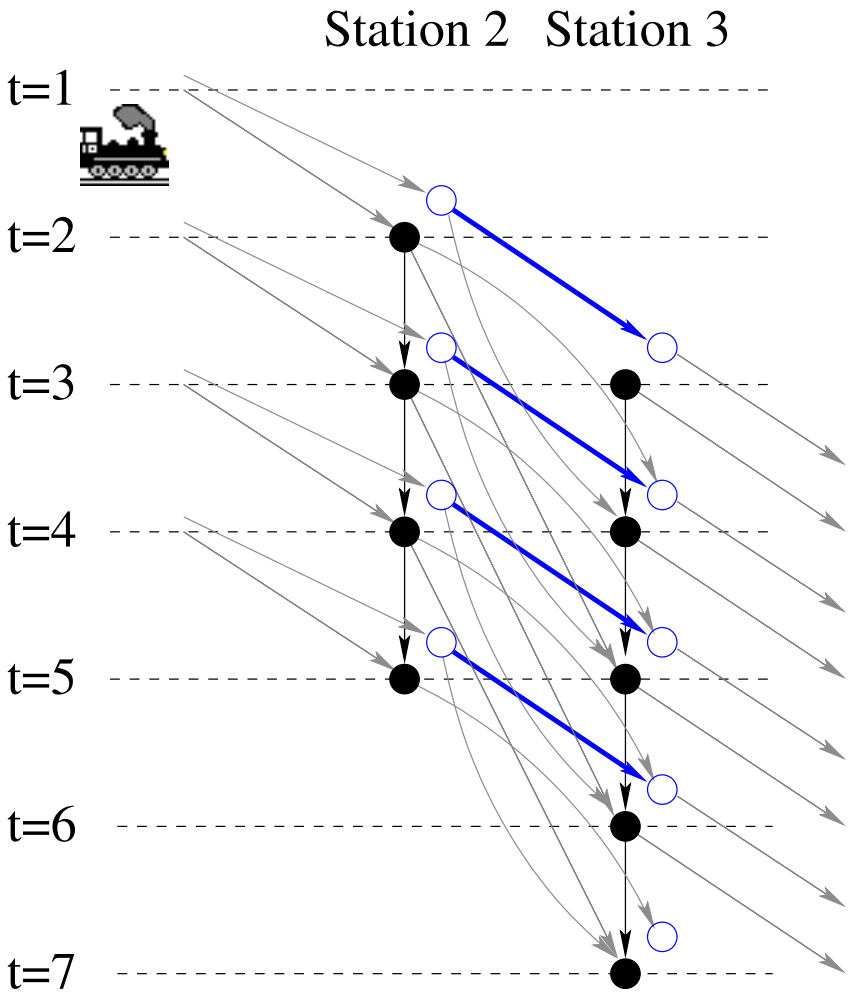


Pro Haltepunkt und Zeitschritt:

- ○ Durchfahrtknoten
- • Halteknoten

Wartekanten

Pro Zug ein Fahrgraph



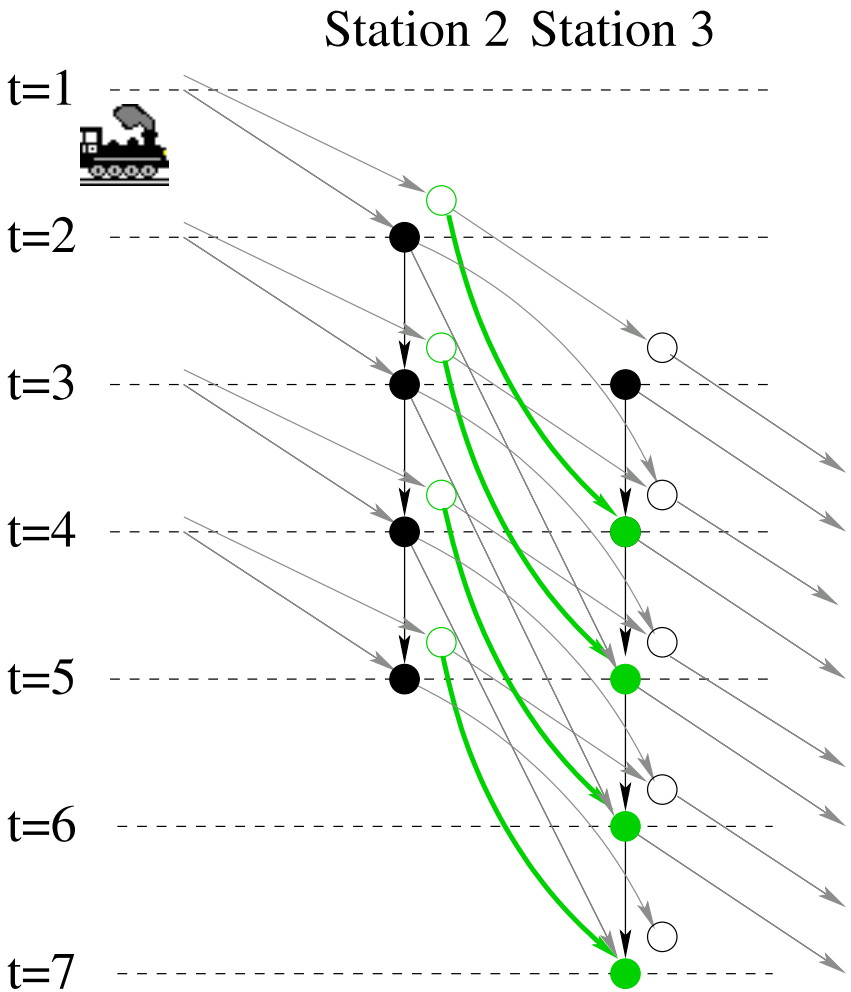
Pro Haltepunkt und Zeitschritt:

- ○ Durchfahrtknoten
- • Halteknoten

Wartekanten

Durchfahrt-Durchfahrt-Kanten

Pro Zug ein Fahrgraph



Pro Haltepunkt und Zeitschritt:

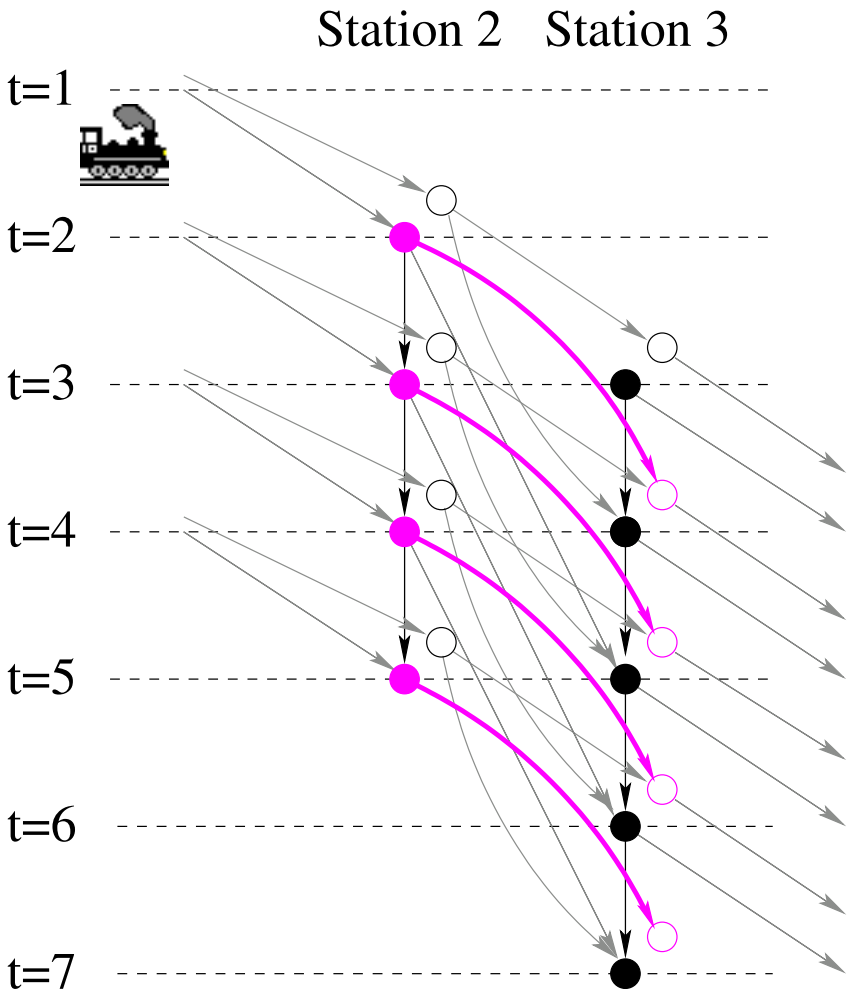
- ○ Durchfahrtknoten
- • Halteknoten

Wartekanten

Durchfahrt-Durchfahrt-Kanten

Durchfahrt-Halt-Kanten

Pro Zug ein Fahrgraph



Pro Haltepunkt und Zeitschritt:

- ○ Durchfahrtknoten
- • Halteknoten

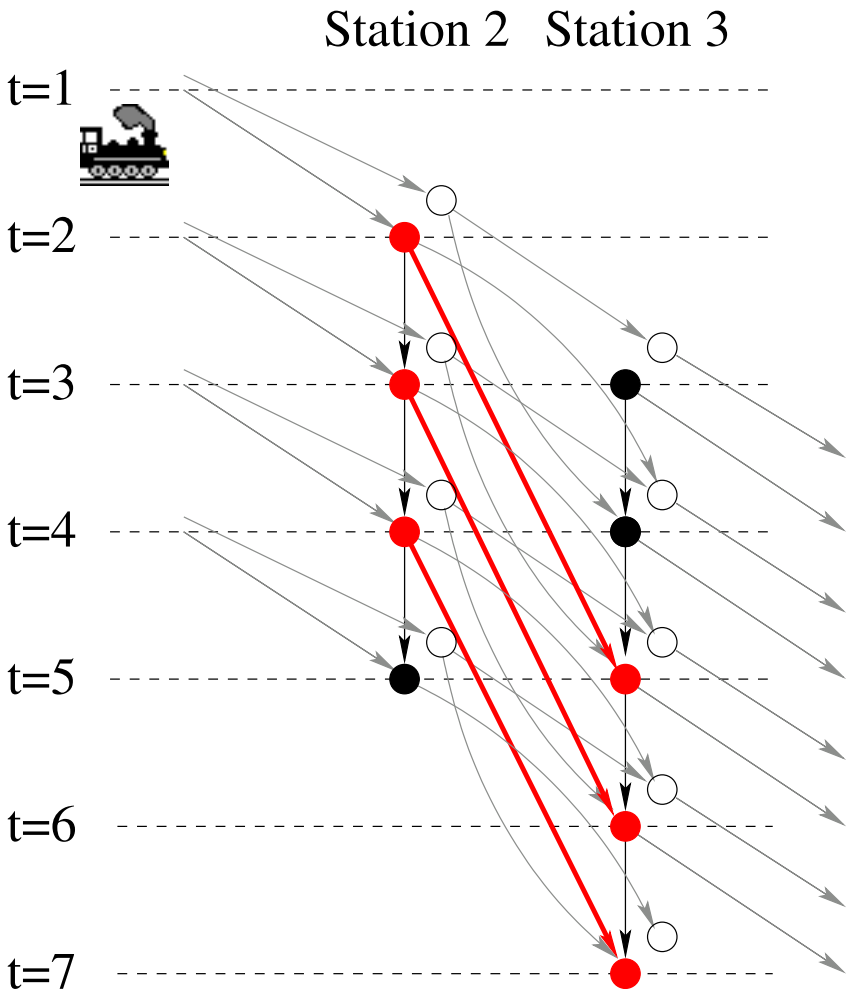
Wartekanten

Durchfahrt-Durchfahrt-Kanten

Durchfahrt-Halt-Kanten

Halt-Durchfahrt-Kanten

Pro Zug ein Fahrgraph



Pro Haltepunkt und Zeitschritt:

- ○ Durchfahrtknoten
- ● Halteknoten

Wartekanten

Durchfahrt-Durchfahrt-Kanten

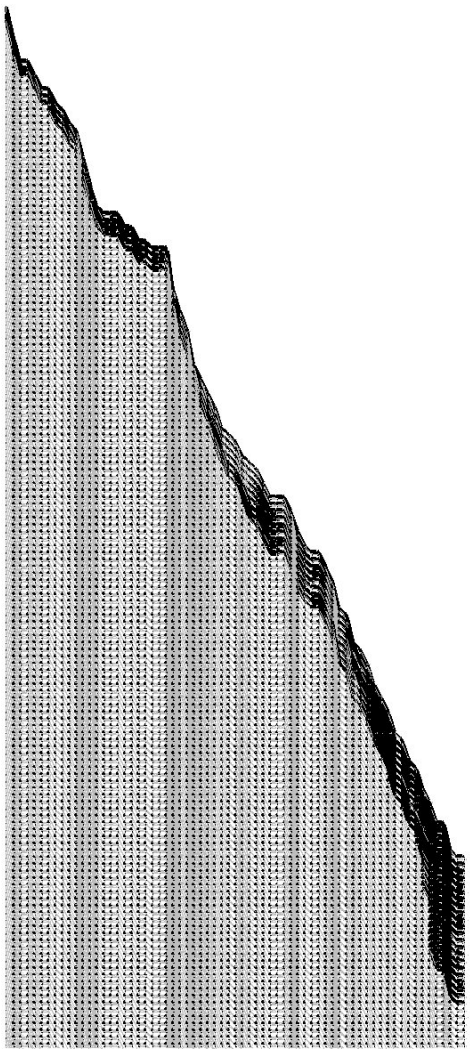
Durchfahrt-Halt-Kanten

Halt-Durchfahrt-Kanten

Halt-Halt-Kanten

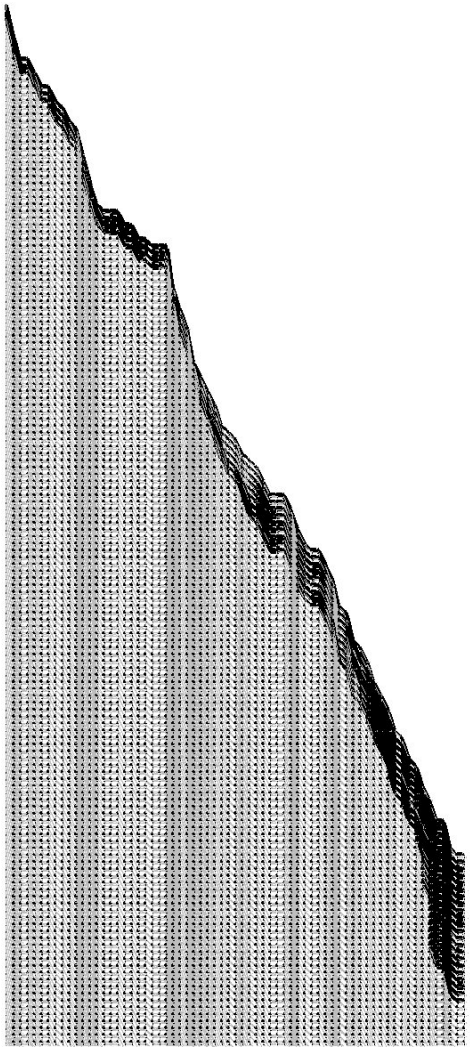
Güterzüge ohne Zeitvorgabe: riesige Graphen

zeitdiskretisierter Graph

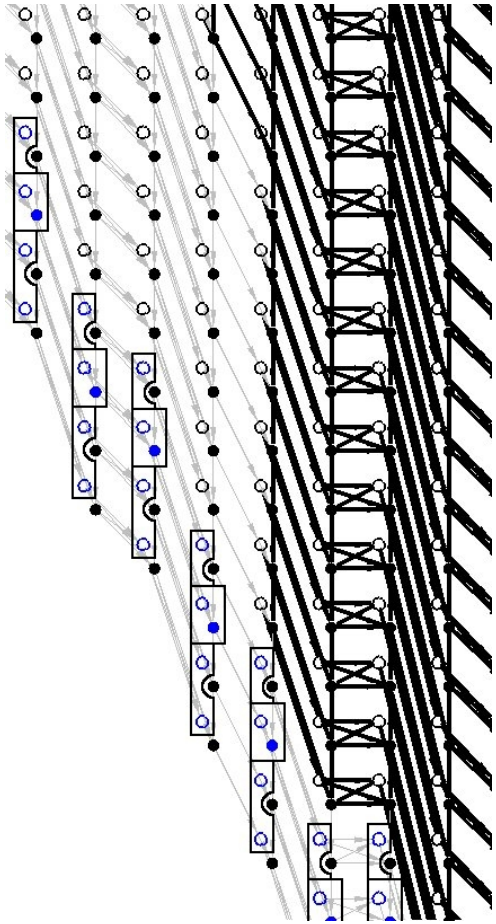


Güterzüge ohne Zeitvorgabe: riesige Graphen

zeitdiskretisierter Graph

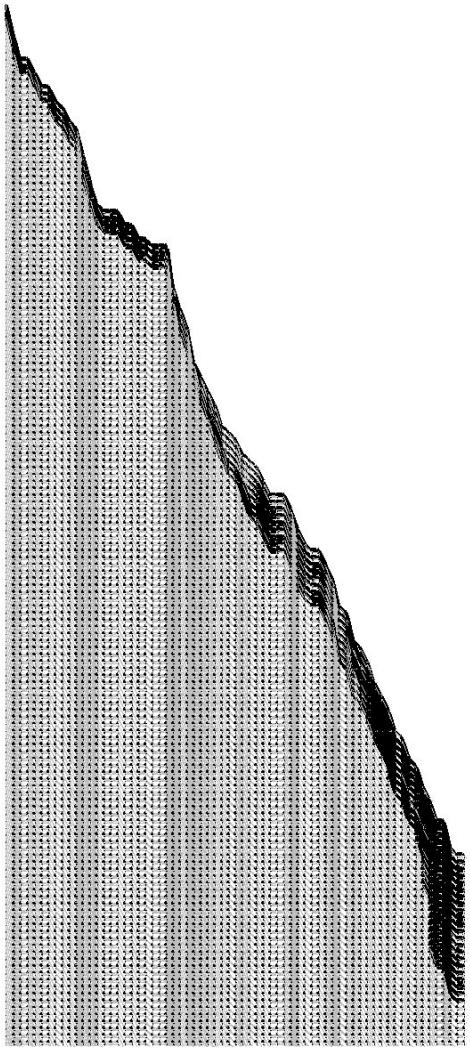


dynamische Graphenerzeugung

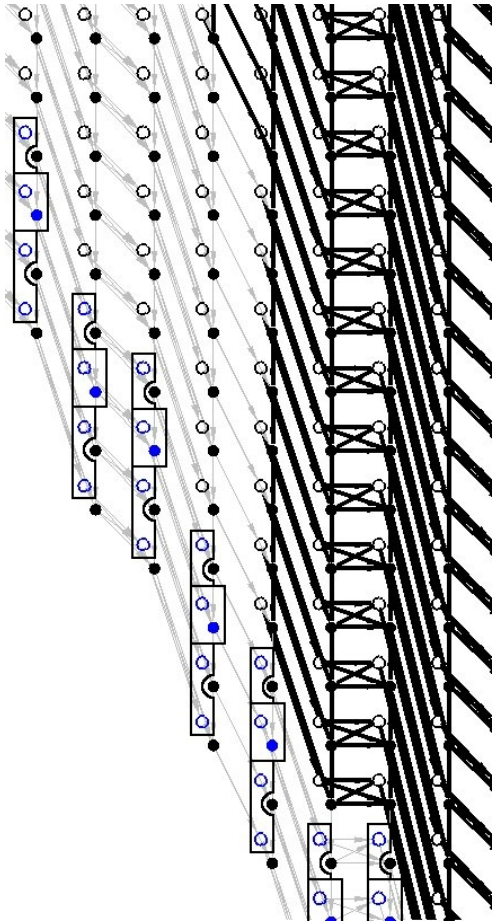


Güterzüge ohne Zeitvorgabe: riesige Graphen

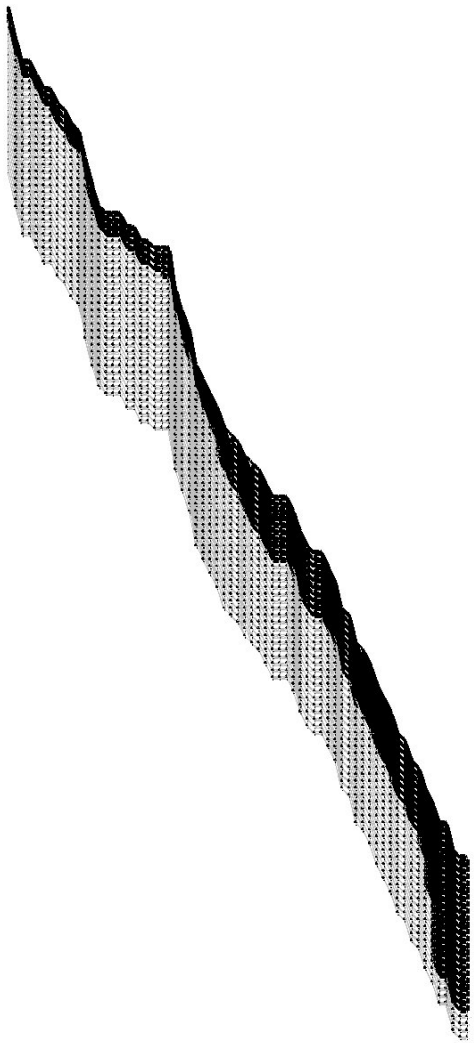
zeitdiskretisierter Graph



dynamische Graphenerzeugung

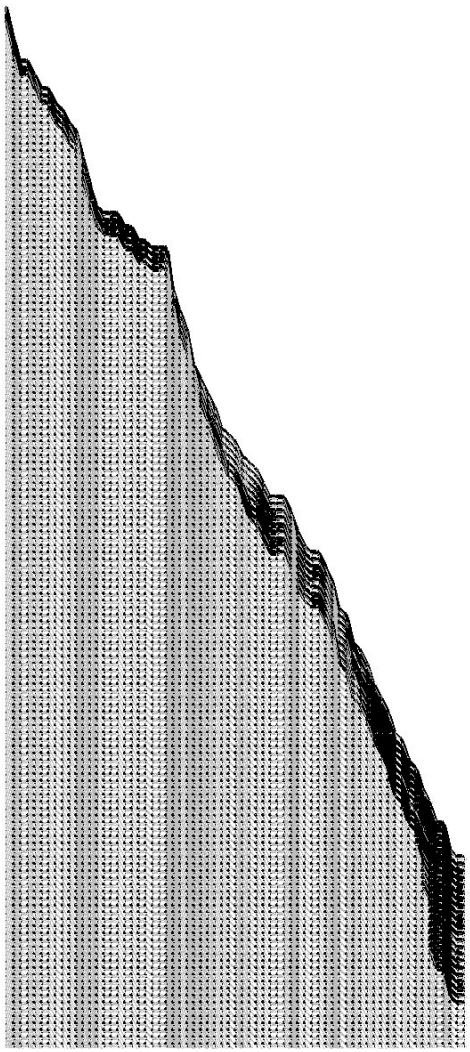


reduzierter Graph

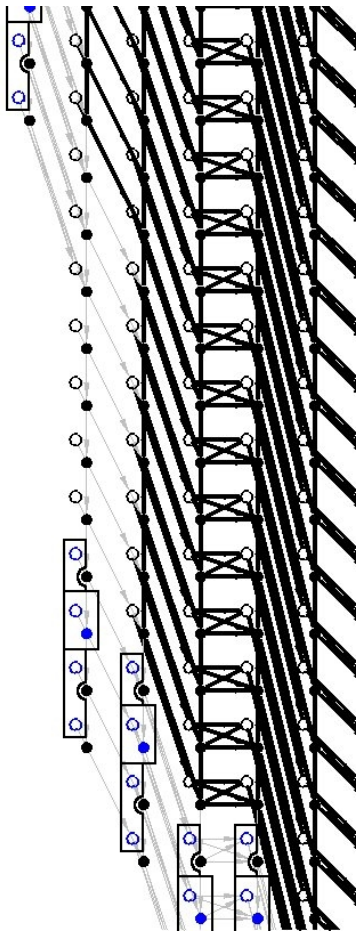


Güterzüge ohne Zeitvorgabe: riesige Graphen

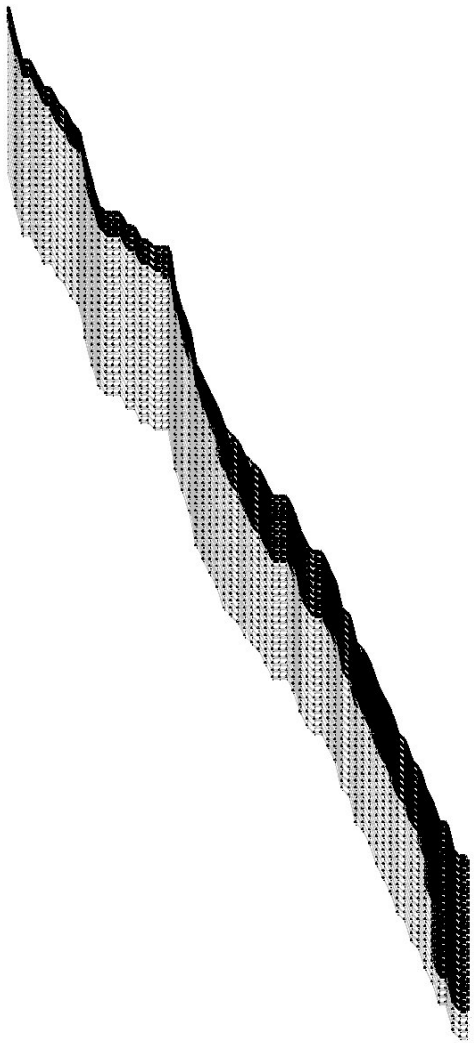
zeitdiskretisierter Graph



dynamische Graphenerzeugung 2

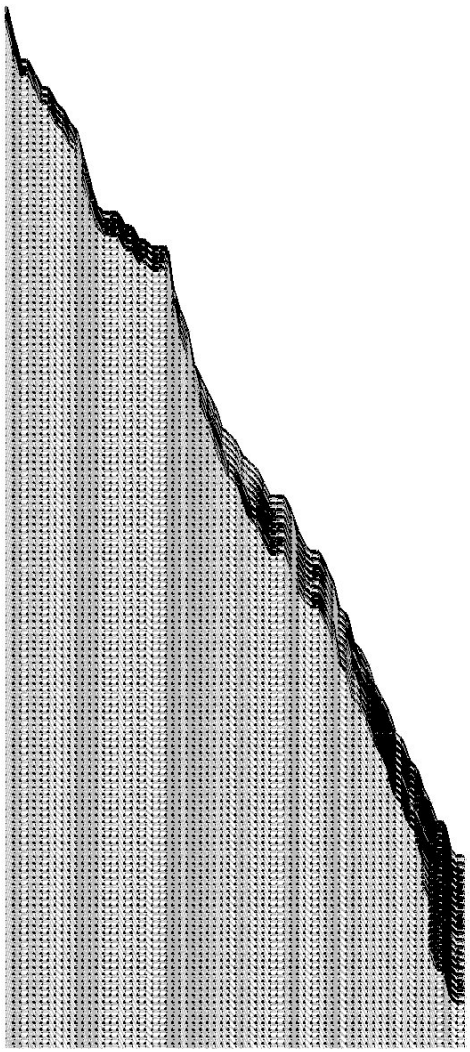


reduzierter Graph

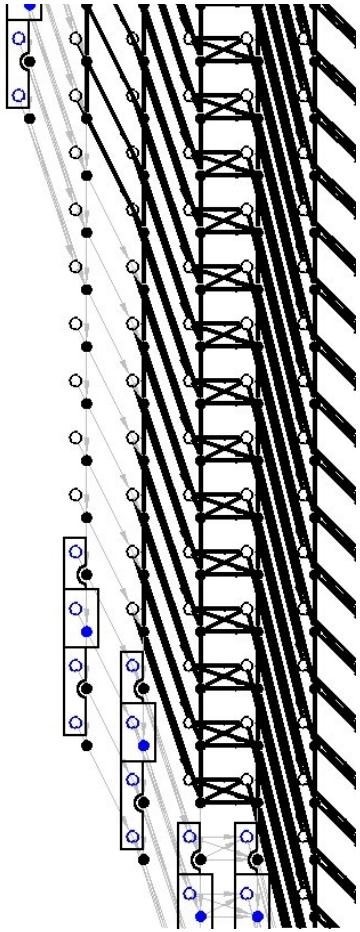


Güterzüge ohne Zeitvorgabe: riesige Graphen

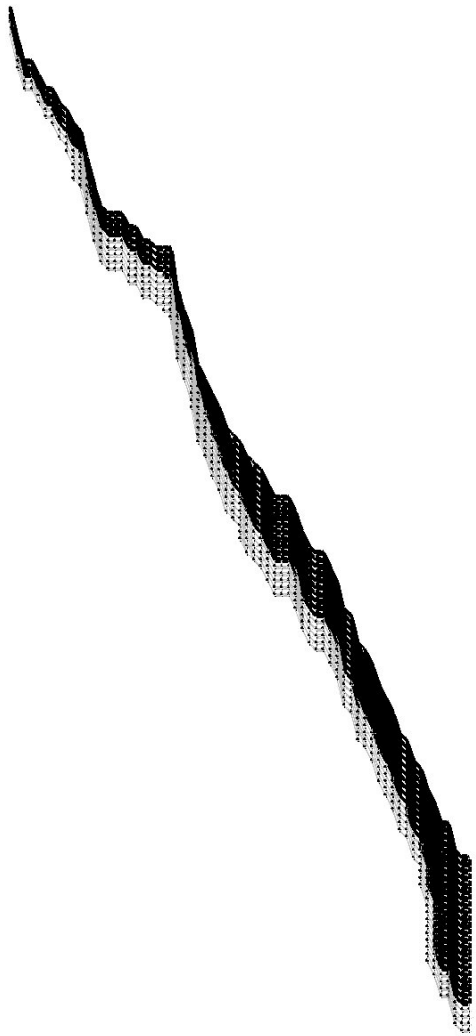
zeitdiskretisierter Graph



dynamische Graphenerzeugung 2

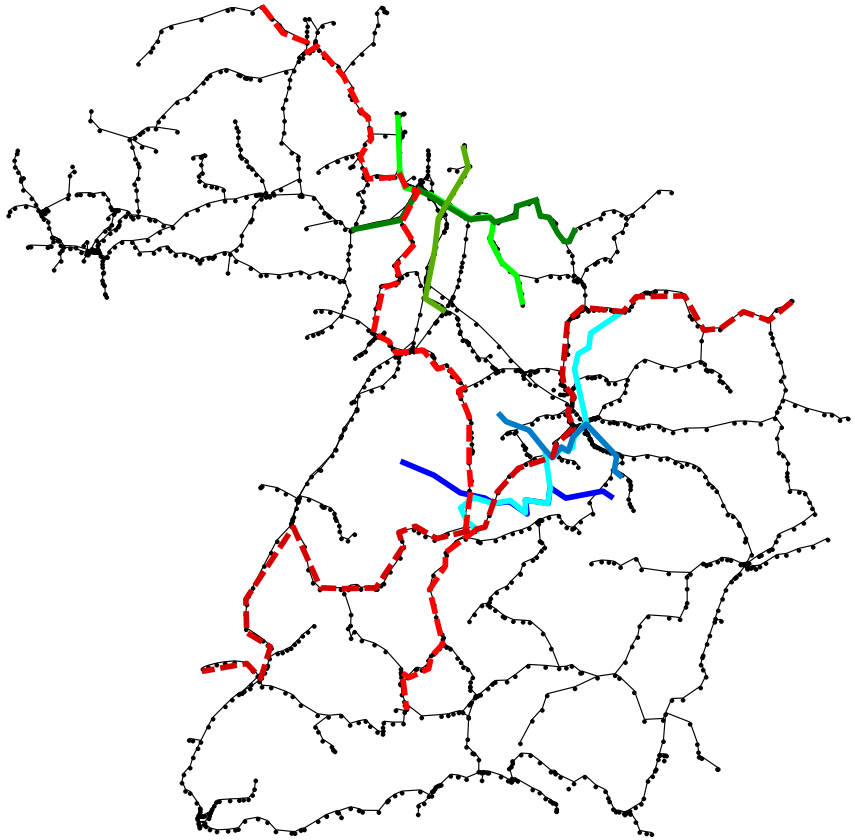


reduzierter Graph 2



Mindestzugfolgezeiten

Züge können nicht zur selben Zeit dieselbe Strecke befahren.

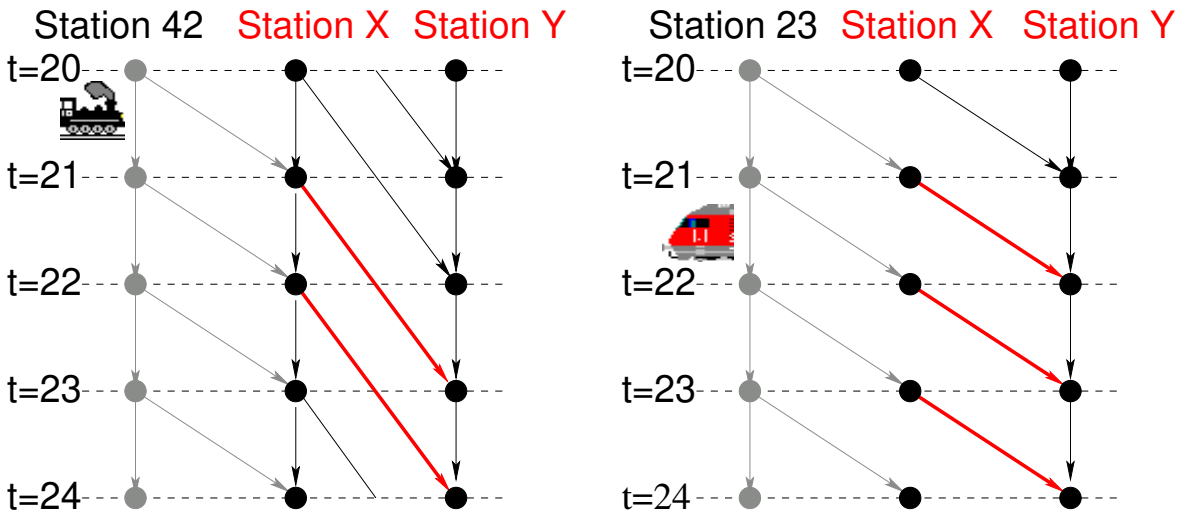


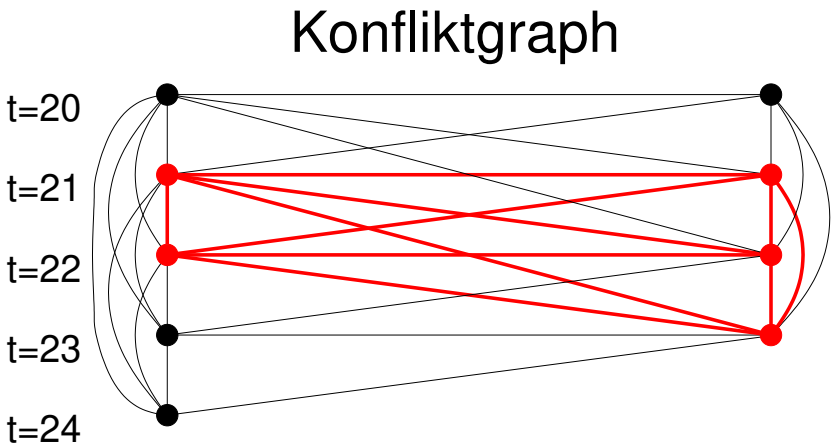
Nach schnellen Zügen dürfen langsame bald fahren,
nach langsamen Zügen müssen schnelle lange warten.

$\rightarrow$ weitreichende Wechselwirkungen auch zwischen konfliktfreien Zügen!

Mindestzugfolgezeiten, Variante Konfliktgraph

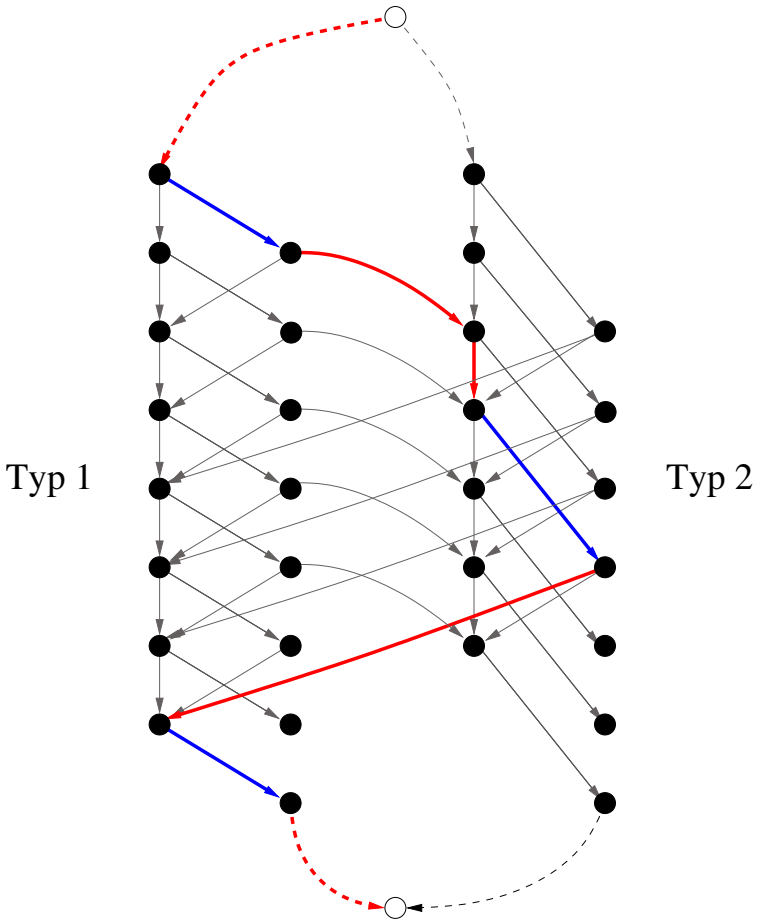
Aus gewissen Teilmengen von Kanten darf höchstens eine genutzt werden.

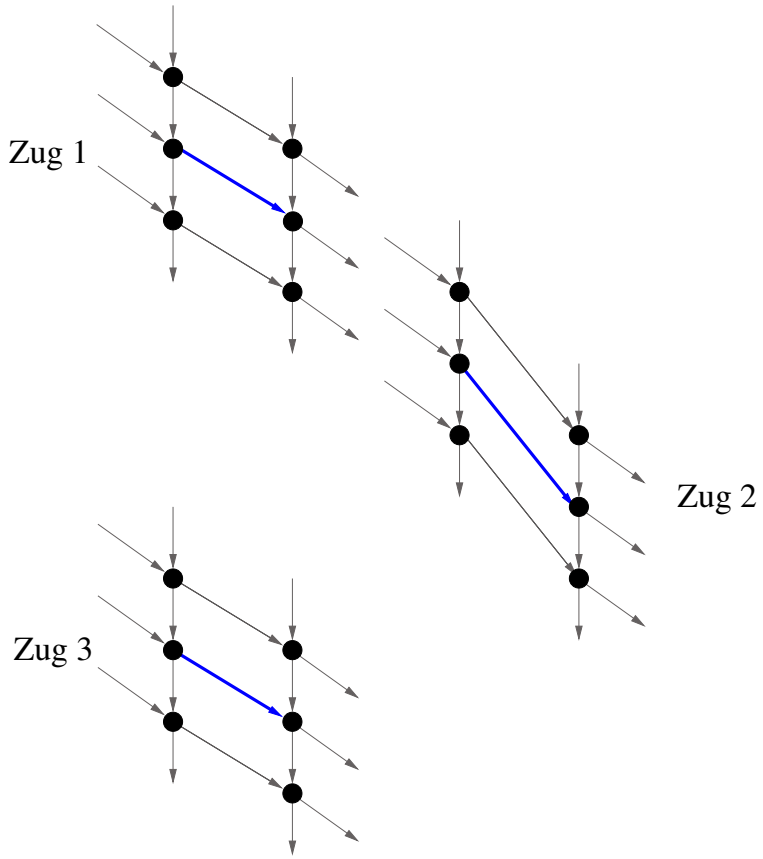




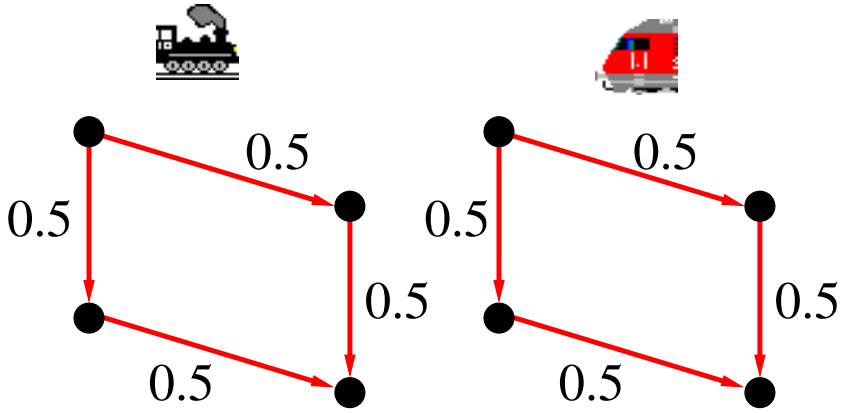
Mindestzugfolgezeiten, Variante Konfigurationsgraph

Pro mehrfach genutzter Strecke ein Reihefolgegraph für Zugtypen



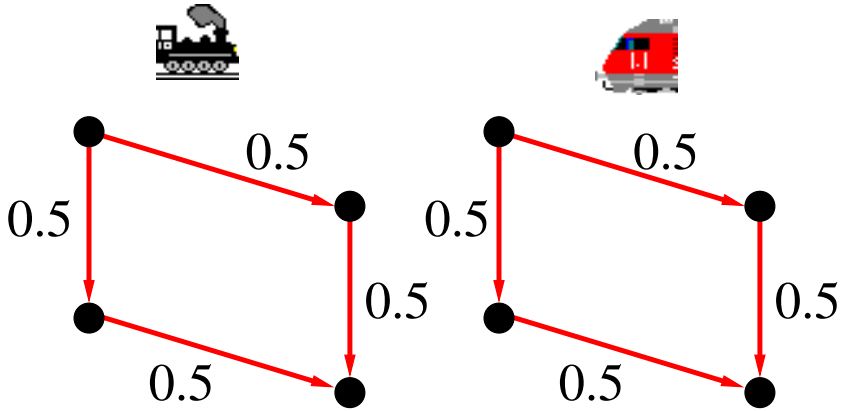


Dennoch bleibt die Lösung des Mehrgüterflussproblems gebrochen / zu gut,



Mit sukzessivem Runden / Festlegen kann man gute Lösungen finden.

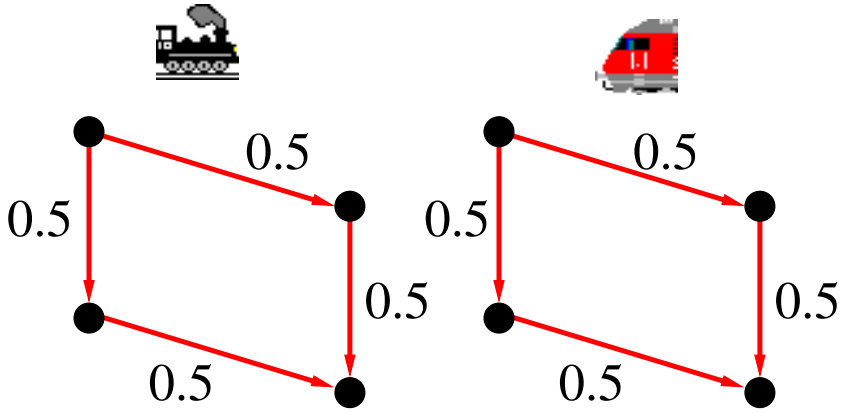
Dennoch bleibt die Lösung des Mehrgüterflussproblems gebrochen / zu gut,



Mit sukzessivem Runden / Festlegen kann man gute Lösungen finden.

Für kleinere Netze als Deutschland gibt es besser geeignete Modelle.

Dennoch bleibt die Lösung des Mehrgüterflussproblems gebrochen / zu gut,



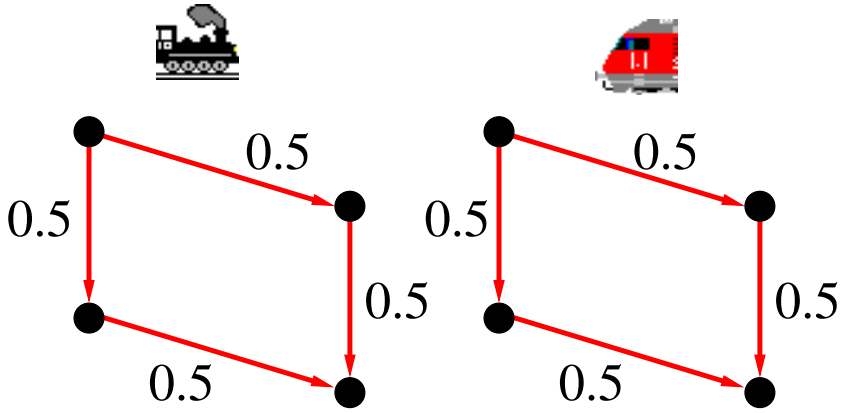
Mit sukzessivem Runden / Festlegen kann man gute Lösungen finden.

Für kleinere Netze als Deutschland gibt es besser geeignete Modelle.

Viele weitere Aspekte wären in der Betriebsplanung relevant, z.B.

- möglichst periodische Abläufe bei Personenzügen
- Robustheit gegenüber Verzögerungen (Bahnhöfe/Fahrzeit)
- schnelle Nachberechnung bei Abweichungen/Zugausfällen
- Umleitungsentscheidungen bei Streckenausbau/-wartung/-ausfall

Dennoch bleibt die Lösung des Mehrgüterflussproblems gebrochen / zu gut,



Mit sukzessivem Runden / Festlegen kann man gute Lösungen finden.

Für kleinere Netze als Deutschland gibt es besser geeignete Modelle.

Viele weitere Aspekte wären in der Betriebsplanung relevant, z.B.

- möglichst periodische Abläufe bei Personenzügen
- Robustheit gegenüber Verzögerungen (Bahnhöfe/Fahrzeit)
- schnelle Nachberechnung bei Abweichungen/Zugausfällen
- Umleitungsentscheidungen bei Streckenausbau/-wartung/-ausfall

---

Die Betriebsplanung ist nur ein Teilproblem des Gesamtplanungsproblems!

Übersicht

Ziel ist, eine grobe Idee zu vermitteln zu:

Wie geht man überhaupt vor und warum ist es so schwierig?

Drei Beispielprobleme: kürzeste Wege, Rundreisen, Flüsse in Netzwerken

- Graphenmodelle und kürzeste Wege
- Rundreisen
- Algorithmische Komplexität, ein Milleniums-Problem
- Fluss- und Transportprobleme
- Betriebsplanerstellung
- **Schlussbemerkungen**

Schlussbemerkungen

- In fast allen Bereichen des täglichen Lebens nutzen wir heute explizit oder implizit Mathematik, sie ist wesentlicher Kultur- und Erfolgsfaktor.
- Ohne mathematische Methoden sind viele Planungsprobleme nicht mehr handhabbar.
- Kaum ein praxisrelevantes Problem ist „optimal“ lösbar.
- In der Praxis sind exakte Optimallösungen oft unbrauchbar, wichtiger sind gute robuste Lösungen $\rightarrow$ hochaktueller Forschungsbereich.
- Der Nachteil mathematischer Ansätze: Die Modellierung ist zeitaufwendig, schwer anzupassen, erfordert viel Sachverstand und Erfahrung $\rightarrow$ bisher eher grobe, allgemein nutzbare Modelle
- Der Vorteil mathematischer Ansätze: sie erlauben gleichzeitig gute Lösungen (mit globaler Sicht) und globale Schranken zu finden.

Schlussbemerkungen

- In fast allen Bereichen des täglichen Lebens nutzen wir heute explizit oder implizit Mathematik, sie ist wesentlicher Kultur- und Erfolgsfaktor.
- Ohne mathematische Methoden sind viele Planungsprobleme nicht mehr handhabbar.
- Kaum ein praxisrelevantes Problem ist „optimal“ lösbar.
- In der Praxis sind exakte Optimallösungen oft unbrauchbar, wichtiger sind gute robuste Lösungen $\rightarrow$ hochaktueller Forschungsbereich.
- Der Nachteil mathematischer Ansätze: Die Modellierung ist zeitaufwendig, schwer anzupassen, erfordert viel Sachverstand und Erfahrung $\rightarrow$ bisher eher grobe, allgemein nutzbare Modelle
- Der Vorteil mathematischer Ansätze: sie erlauben gleichzeitig gute Lösungen (mit globaler Sicht) und globale Schranken zu finden.

Danke für Ihre Aufmerksamkeit!