Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER MARTIN GRÖTSCHEL ANDREAS LÖBEL

Der Schnellste Weg zum Ziel

ZIB-Report 99-32 (September 1999, Revidiert Juni 2000)

Der Schnellste Weg zum Ziel

Ralf Borndörfer Martin Grötschel Andreas Löbel

Konrad-Zuse-Zentrum Berlin – Takustr. 7 – 14195 Berlin
`http://www.zib.de`

Zusammenfassung. Wir geben eine Einführung in die Mathematik von und mit Wegen. Nicht auf dem kürzesten, aber auf einem hoffentlich kurzweiligen Weg!

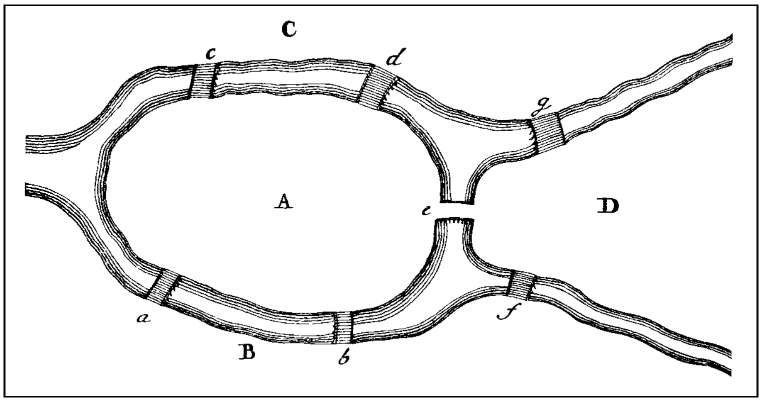
Schlüsselworte. Kürzeste Wege, Kombinatorische Optimierung, Operations Research

Mathematics Subject Classification (1991). 90C10

1 Historische Ouvertüre

Das Thema "Wege" weckt Assoziationen zu Straßen, Transport, Verkehr ... und Mathematik! Hier sind vier Beispiele.

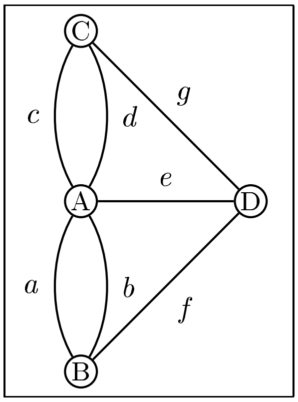
Das Königsberger Brückenproblem. Der Mathematiker, Physiker und Astronom Leonhard Euler (1707–1783) studierte im Jahr 1736 diese einfache und doch geheimnisvolle Skizze:



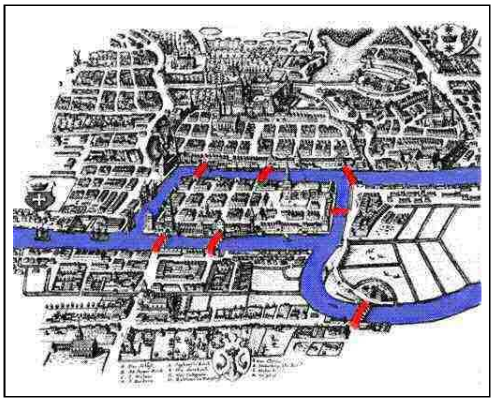
Figur 1: Das Königsberger Brückenproblem.

Das Problem, das ziemlich bekannt sein soll, war folgendes: Zu Königsberg in Preußen ist eine Insel A, genannt „der Kneiphof“, und der Fluß, der sie umfließt, teilt sich in zwei Arme, wie dies aus Fig. 1 ersichtlich ist. Über die Arme dieses Flußes führen sieben Brücken a, b, c, d, e, f und g. Nun wurde gefragt, ob jemand seinen Spaziergang so einrichten könne, daß er jede dieser Brücken einmal und nicht mehr als einmal überschreite. Es wurde mir gesagt, daß einige diese Möglichkeit verneinen, andere daran zweifeln, daß aber niemand sie erhärte. Hieraus bildete ich mir folgendes höchst allgemeine Problem: Wie auch die Gestalt des Flußes und seine Verteilung in Arme, sowie die Anzahl der Brücken ist, zu finden, ob es möglich sei, jede Brücke genau einmal zu überschreiten oder nicht.

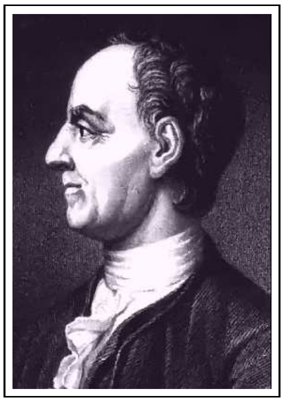
Das war das *Königsberger Brückenproblem.* Das Problem war höchst ungewöhnlich! Es war zweifellos mathematischer Natur, und doch so beschaffen, daß es weder die Bestimmung einer Größe erforderte, noch eine Lösung mit Hilfe des Größenkalküls gestattete. *Geometria situs*, *Geometrie der Lage*, nennt Euler (der den Begriff einem Brief von Leibniz aus dem Jahre 1679 entnimmt) diese neue Mathematik ohne Zahlen, in der Größe und Form keine Rolle spielen, sondern nur die Struktur der Anordnung.



Figur 2: Der 1. Graph.



Figur 3: Königsberg 1736.



Figur 4: L. Euler.

Eulers erster Schritt in das unbekannte Gebiet war eine geniale Abstraktion. Aus der Karte in Fig. 1 hatte er das Diagramm in Fig. 2 gebildet. In diesem *Graphen* hatten sich die Inseln und Brücken in formlose Knoten und Kanten verwandelt. Bei der Betrachtung dieser Darstellung bemerkte Euler, daß die Anzahl der Kanten, die in einen Knoten führen, der *Grad*, eine wichtige Rolle spielte. Diese Größe war der Schlüssel zur Lösung nicht nur des Königsberger, sondern überhaupt aller Brückenprobleme durch zwei wunderbare Resultate:

1.1 Theorem *Die Anzahl der Knoten mit ungeradem Grad ist gerade.*

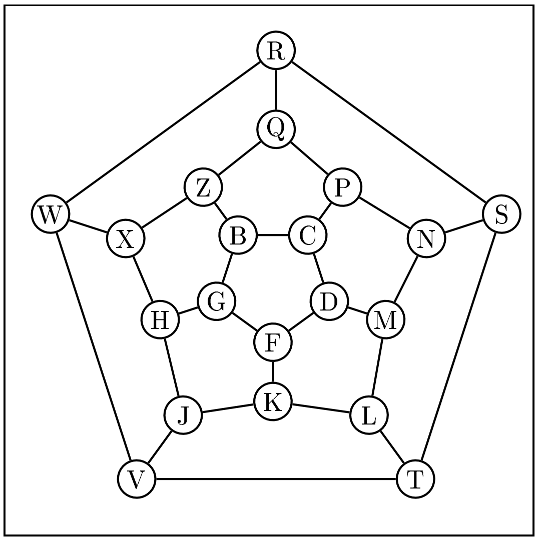
1.2 Theorem *Einen Spaziergang, der jede Kante genau einmal benutzt, gibt es genau dann, wenn höchstens zwei Knoten ungeraden Grad haben.*

Diese Sätze waren die ersten Beiträge zur *Topologie*, wie die Geometrie der Lage heute heißt, genauer gesagt zur (topologischen) *Graphentheorie.*

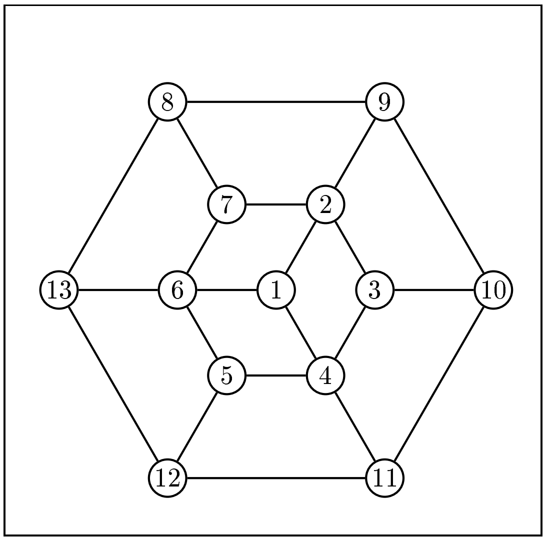
Die höchste Ehre, die es in der Mathematik gibt, ist die Benennung einer Entdeckung nach einer Person. Euler ist wahrscheinlich der in dieser Hinsicht berühmteste Mathematiker aller Zeiten. Die Bezeichung *Eulertour* für Spaziergänge in Graphen, die jede Kante genau einmal verwenden, ist eine dieser Referenzen, mit denen die Mathematiker noch heute die Arbeit Eulers würdigen.

Es bleibt die Frage an Sie, lieber Leser: Gibt es nun eine Eulertour über die Königsberger Brücken oder nicht? Mit Eulers Hilfe gelingt Ihnen die Antwort sicherlich!

Das Hamiltonsche Kreisproblem. Genau 120 Jahre später erfand der irische Mathematiker Sir William Rowan Hamilton (1805–1865) das auf den ersten Blick sehr ähnliche “Ikosaederspiel” in Fig. 7. In dem Graphen in Fig. 5, dessen Knoten Orte wie Brüssel, Canton, Delhi bis Zanzibar darstellten, sollte der Spieler eine (zyklische) *Rundreise* bestimmen, die jeden Ort genau einmal besucht. Versuchen Sie es einmal! Probieren Sie auch die “Bienenwabe” in Fig. 6 daneben, die etwa gleichzeitig von Hamiltons englischem Kollegen Thomas Penyngton Kirkman (1806–1895) erfunden wurde. Sie werden einen Unterschied feststellen!



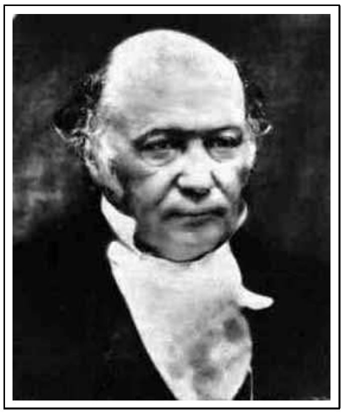
Figur 5: Graph zum Ikosaederspiel.



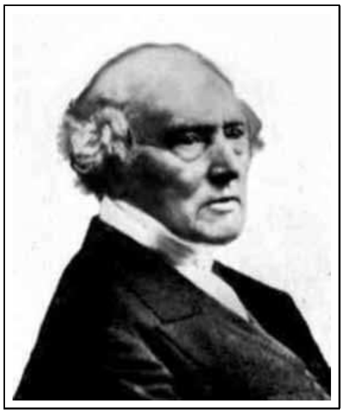
Figur 6: Die "Bienenwabe".

Rundreisen in Graphen nennt man *Hamiltonkreise*, und die Frage, ob ein Graph eine solche zuläßt, ist das *Hamiltonsche Kreisproblem* (Hamiltonian Circuit Problem, kurz: HCP). Das Erstaunliche am HCP ist Folgendes: Obwohl das Problem beinahe wie das Brückenproblem aussieht, gibt es keine vergleichbare Lösungsmethode. Alle bekannten Verfahren müssen sich im schlimmsten Fall damit retten, alle Möglichkeiten durch *Enumeration* auszuprobieren. Der verblüffende Grund dafür ist, daß es höchstwahrscheinlich nicht besser geht! Die *Komplexitätstheorie* zeigt, daß das HCP zur Klasse der *$\mathcal{NP}$-vollständigen Probleme* gehört, die in einem mathematisch präzisierbaren Sinne schwierig sind. Ein endgültiger Beweis für die Unvermeidbarkeit der Enumeration bei $\mathcal{NP}$-vollständigen Problemen ist allerdings trotz großer Forschungsanstrengungen bisher nicht erbracht.

Figur 7: Das Ikosaederspiel.

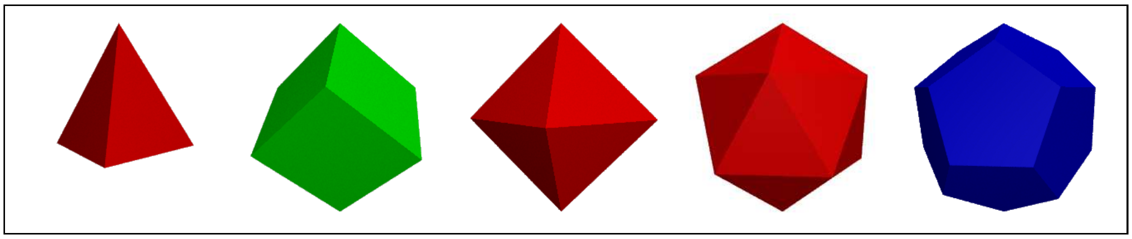


Figur 8: W.R. Hamilton.



Figur 9: T.P. Kirkman.

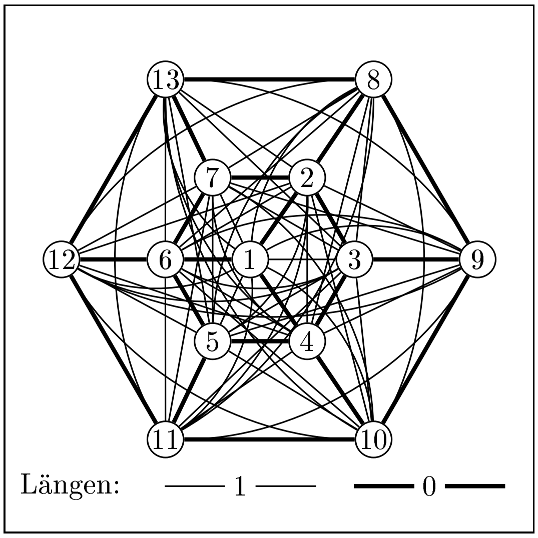
Für spezielle Graphen kann man mehr tun und besondere Algorithmen entwickeln. Hamilton wußte für sein Ikosaederspiel eine Lösungsmethode, bei der er sogar die Anfangsstädte vorschreiben konnte. Trotzdem war sein Spiel ein Ladenhüter. Das war allerdings kein Wunder. Allein der Beginn der Anleitung, in der Hamilton erklärt, warum sein *Ikosaeder*spiel auf einem *Dodekaeder* stattfindet (siehe Fig. 10), kann einem die Spielfreude verderben. Hamilton verstand etwas von Mathematik, aber offensichtlich nichts von Marketing!



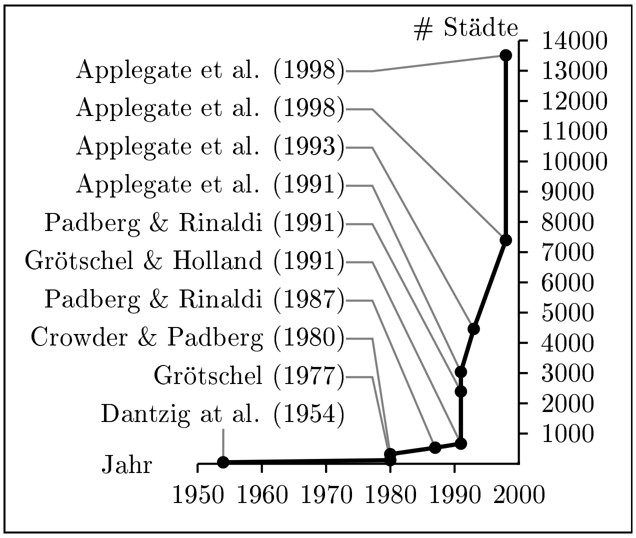
Figur 10: Die Platonischen Körper: Tetraeder, Kubus, Oktaeder, Ikosaeder, Dodekaeder.

Das Problem des Handlungsreisenden. Eine Optimierungsvariante des HCPs ist das *Problem des Handlungsreisenden* (Travelling Salesman Problem, kurz: TSP). In einem *vollständigen Graphen* (Graph mit allen denkbaren Kanten), dessen Kanten *Längen* haben, ist die kürzeste Rundreise gefragt. Nützlich für Handlungsreisende und Bohrautomaten, doch am wichtigsten ist das TSP als *das Benchmarkproblem* der kombinatorischen Optimierung.

Im Gegensatz zum HCP liegt die Schwierigkeit beim TSP nicht im Finden einer Rundreise: In einem vollständigen Graphen mit n Knoten stehen $\frac{1}{2}(n-1)!$ Rundreisen zur Auswahl. Fig. 11 deutet die wahren Probleme an. Man kann jedes HCP (hier die Bienenwabe) in ein TSP verwandeln: Eine Rundreise der Länge 0 gibt es genau dann, wenn das Ausgangs-HCP einen Hamiltonkreis zuläßt. Diese *Problemtransformation* zeigt, daß das TSP $\mathcal{NP}$*-schwer* ist. ($\mathcal{NP}$-vollständige Optimierungsprobleme nennt man $\mathcal{NP}$-schwer.)



Figur 11: Das HCP als TSP.



Figur 12: TSP Weltrekorde.

$\mathcal{NP}$-schwer heißt trotzdem nicht, daß Trivialenumeration die einzige Lösungsmöglichkeit ist. Eine Idee ist, Enumeration mit empirisch effizienten Techniken zu kombinieren, die in der Praxis bei Problemen bis zu einer gewissen Größe zu erträglichen Laufzeiten führen. *Branch & Bound*, *Branch & Cut* und *Branch & Price* sind die Hauptvertreter dieser Linie von Algorithmen. "Branch" steht für Enumeration, "Bound", "Cut" und "Price" für verschiedene Beschleunigungstechniken. Nur getunte Enumeration? Komplexitätstheoretisch ja, und doch unendlich wertvoll im *konkreten Einzelfall.* Fig. 12 zeigt die Entwicklung beim TSP. 1991 war die 666-Städte-Reise in Fig. 13 die Grenze des Möglichen. Heute kann man dieses Problem in 260 Sekunden lösen, und der Weltrekord, gehalten von dem vierköpfigen Team David Applegate, Bob Bixby, Bill Cook und Vasĕk Chvátal, steht bei 13.509 Städten! Gegenüber dem 666-Städte-Problem sind das 20× mehr Städte, 412× mehr Kanten und $10^{48342}\times$ mehr Rundreisen! Der algorithmische Fortschritt (bereinigt um die 40-fache Steigerung der Rechenleistung, die sich nach dem *Mooreschen Gesetz* seit 15 Jahren alle 18 Monate verdoppelt) ist natürlich schwer zu quantifizieren. Eins läßt sich aber nicht abstreiten: *Branch & Co* ist mehr als Trivialenumeration.

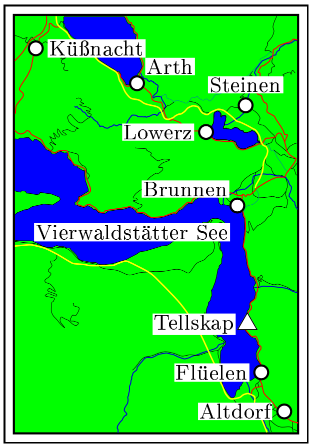
Eine Alternative zu diesen exakten Verfahren ist, statt der besten auch "gute" Lösungen zu akzeptieren, die vom (unbekannten!) Optimum nicht mehr als einen vorher festgelegten Prozentsatz abweichen. Gibt es schnelle *approximative Algorithmen* mit solchen *Gütegarantien* für $\mathcal{NP}$-vollständige Probleme? Manchmal ja! Für *euklidische TSPs* mit geographischen Kantenlängen (ein $\mathcal{NP}$-schweres Problem) hat der in Berkeley lehrende indische Mathematiker Sanjeev Arora im Jahre 1996 eine solche Methode entdeckt, bei der sich die Güte sogar beliebig einstellen läßt. Für allgemeine (nicht-euklidische) TSPs kann man beweisen, daß es ein solches Verfahren nicht geben kann.

Figur 13: Die kürzeste Reise um die Welt.

Das Kürzeste-Wege-Problem. Das älteste uns bekannte Wegeproblem kommt aus einer klassischen Quelle: Friedrich Schillers (1759–1805) Schauspiel "Wilhelm Tell". Denn dieser konnte bereits 1291 nicht nur gut schießen, sondern auch optimieren. Und nur mit dieser Kombination konnte er die Schweiz befreien! Tell befindet sich nach dem Apfelschuß am Ufer des Vierwaldstätter Sees unweit des Ortes Altdorf. Unbedingt muß er vor dem Reichsvogt Hermann Geßler die Hohle Gasse in Küßnacht erreichen, siehe Fig. 15. Schiller berichtet:

Tell. Nennt mir den nächsten Weg nach Arth und Küßnacht.
Fischer. Die offne Straße zieht sich über Steinen,
Doch einen kürzern Weg und heimlichern
Kann Euch mein Knabe über Lowerz führen.
Tell (gibt ihm die Hand). Gott lohn Euch Eure Guttat. Lebet wohl.

Figur 14: F. Schiller.



Figur 15: Vierwaldstätter See.

Figur 16: W. Tell.

Tell löst in dieser Szene offensichtlich ein graphentheoretisches Optimierungsproblem. In einem Graphen (Wegenetz am Vierwaldstätter See) mit Kantenlängen (Reisezeit) soll der kürzeste Weg zwischen zwei vorgegebenen Punkten (Altdorf und Küßnacht) bestimmt werden. Das ist das *Kürzeste-Wege-Problem* (Single Source Shortest Path [Problem], kurz: SSSP, das zweite P wird weggelassen). Tell behandelt sogar eine kompliziertere Variante mit einer zusätzlichen *Nebenbedingung*: Die Summe von "Verhaftungskoeffizienten" muß unterhalb eines sicheren Grenzwertes bleiben. Literatur und Mathematik — keineswegs ein Gegensatz!

*

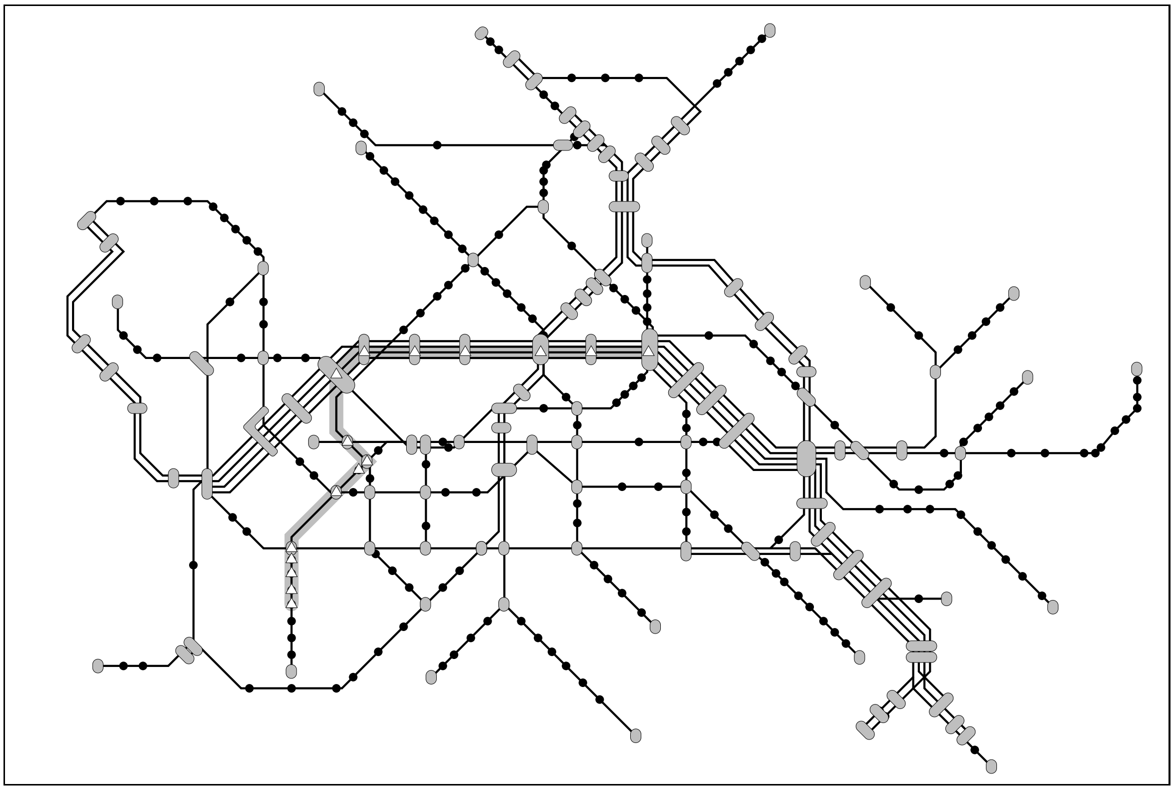
Wegeprobleme treten überall im Zusammenhang mit Netzwerken auf: Bei der Routen- und Personalplanung, in der Logistik, im Projektmanagement, beim Entwurf von integrierten Schaltungen und in der Telekommunikation beim Netzentwurf, beim Routing von Telefongesprächen und Daten etc. Mathematik kann helfen, Kosten zu verringern, Qualität zu erhöhen und Planung zu beschleunigen.

Einige Fragen, Methoden und Möglichkeiten der mathematischen Behandlung von Wegeproblemen wollen wir in diesem Artikel beleuchten. Die Beispiele der Fahrgastinformation im öffentlichen Nahverkehr und der Fahrzeugeinsatzplanung in Anrufsammeltaxisystemen bilden die Grundlage unserer Darstellung. Den Auftakt macht das fundamentale Kürzeste-Wege-Problem und seine Behandlung. Zum gleichzeitigen Planen von mehreren Wegen in der Fahrzeugeinsatzplanung muß schweres Branch & Price-Geschütz aufgefahren werden. Kürzeste Wege spielen dort eine wichtige Rolle. Am Ende stehen ein Blick auf das planerische Umfeld, Auflösungen im Text gestellter Fragen und Hinweise auf weiterführende Literatur. Viel Vergnügen auf Ihrem Weg durch die Mathematik der Wege!

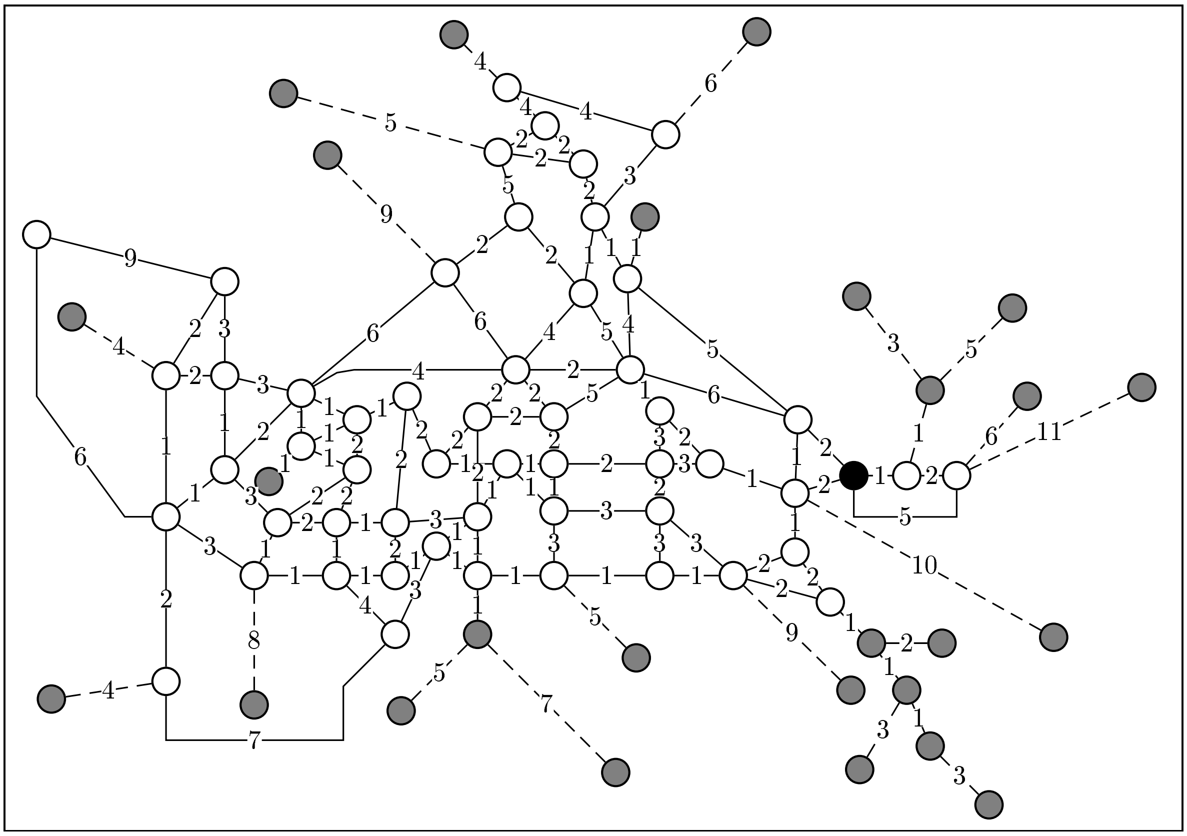
2 Kombinatorik der Kürzesten Wege

2.1 Nahverkehr und Graphentheorie

Ein Kürzeste-Wege-Problem, das jeder kennt, ist die Wahl des Fahrtweges im Nahverkehr. Fig. 17 zeigt das Beispiel des Berliner Schnellbahnnetzes (U- und S-Bahn). 306 Stationen und 445 Strecken machen die Entscheidung nicht immer leicht.



Figur 17: Das Schnellbahnnetz von Berlin.



Figur 18: Reduziertes Schnellbahnnetz.

Was soll optimiert werden: Zeit, Strecke, Umsteigen, Preis? Wir betrachten ein *Modell.*

2.1 Definition (Kürzeste-Wege Problem (SSSP))

Gegeben: *Graph $G = (V, E)$ (Knotenmenge V, Kantenmenge E)*
Nichtnegative Längen oder Gewichte w_{uv} für alle Kanten $uv \in E$
Zwei Knoten s und t

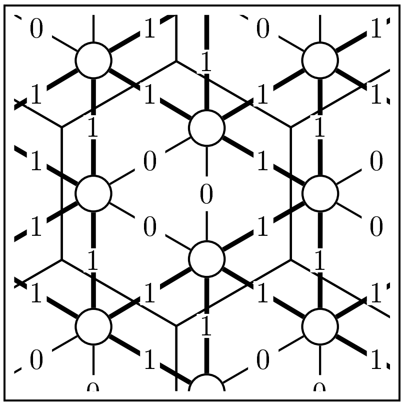
Gesucht: *Ein kürzester Weg von s nach t in G*

Der Vorteil dieser *Abstraktion* ist, daß die Problemdaten eine beliebige Interpretation zulassen. Die Kantenlängen können Preise, Strecken, Zeiten etc. sein, der Mathematik ist das einerlei. Die Vielseitigkeit des Modells geht aber noch sehr viel weiter. Durch geschickte Variation der Längen und Manipulation der Struktur lassen sich nämlich auch viele Problemvarianten auf das Grundmodell zurückführen. Wir geben drei Beispiele für solche *Modellierungstricks.*

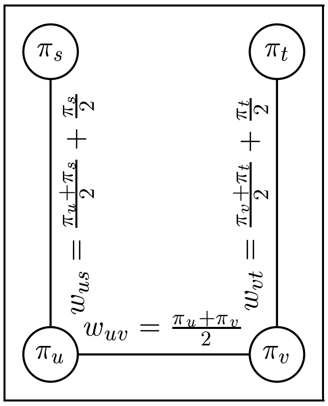
Fahrpreisminimierung in Wabensystemen. Viele Verkehrsbetriebe verwenden Tarife, bei denen in jeder neuen Wabe eine Zuzahlung fällig wird. Fig. 19 zeigt eine Kantenbewertung für konstante (immer gleiche) Aufzahlungen. Praxisnäher sind sicher abnehmende Grenzpreise; auf diesen Fall werden wir in Abschnitt 2.3 eingehen.

Knotengewichte. Gewichte (Kosten, Preise) an den Knoten kann man auf die Kanten "umlegen". Die Formeln dazu sind in Fig 20 angedeutet (mit Spezialbehandlung für Start und Ziel).

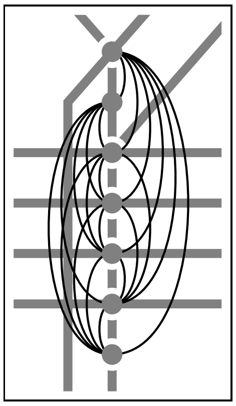
Umsteigen. Fig. 21 zeigt als Beispiel für eine strukturelle Transformation eine Behandlung von Umsteigezeiten auf Bahnhöfen mit sich kreuzenden Linien.



Figur 19: Wabentarif.



Figur 20: Knotengewichte.



Figur 21: Umsteigen.

Bei aller Begeisterung für solche Techniken gilt: Beim Modellieren ist weniger oft mehr. Es ist weder sinnvoll noch notwendig, jedes unbedeutende Detail auszu-x-en. Oft setzt die Verfügbarkeit von Daten engere Grenzen als die Leistungskraft der Algorithmen!

Wir bleiben beim Einfachen und "vervollständigen" das Schnellbahnbeispiel, indem wir jeder Kante in Fig. 17 die Länge 1 zuweisen. Die Länge eines Weges ist dann die Anzahl der durchfahrenen Strecken. Der durch markierte Weg in Fig. 17 vom Alexanderplatz nach Dahlem-Dorf (Standort des Konrad-Zuse-Zentrums) hat z. B. eine Länge von 15 (Kanten).

Solche Weglängenbestimmungen kann man mit geringem Aufwand sehr vereinfachen. Fig. 18 zeigt, wie man durch Zusammenlegen von Streckenfolgen ohne Umsteigemöglichkeit zu Kanten mit entsprechender Länge (eine neue Kante der Länge k entspricht k alten Kanten) eine wesentliche Reduktion der Problemgröße auf 80 Knoten und 122 Kanten erzielt.

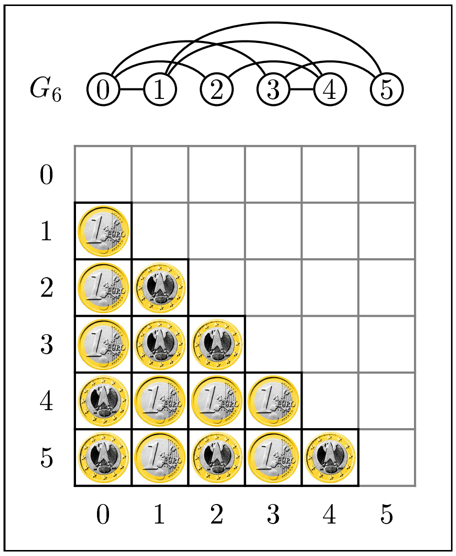
Der Preis für dieses *Preprocessing* ist, daß man kürzeste Wege nur noch zwischen den End- und Kreuzungspunkten von Linien direkt berechnen kann. Daraus ergeben sich aber leicht alle kürzesten Wege mit Hilfe einiger Fallunterscheidungen. (Der Weg von und zu einem Knoten zwischen zwei Umsteige-/ Endknoten A und B verläuft entweder über A oder über B. Das gleiche gilt für das andere Ende mit Knoten C und D. Es ergeben sich vier Fälle AC, AD, BC und BD.)

Die Reduktion läßt sich noch weitertreiben. Man kann die grau/gestrichelt gezeichneten *Bäume* in Fig. 18 eliminieren, in denen man keine Wahl hat. Man kann den Graphen an dem schwarzen *Artikulationsknoten* auseinandernehmen. Usw. Irgendwo beginnt schließlich der Overkill, bei dem der Aufwand für Rechnung und Implementation(!) den Nutzen übersteigt. Trotzdem gilt: Die richtige Dosis Preprocessing ist ein Muß, wenn man praktische Optimierungsprobleme lösen will.

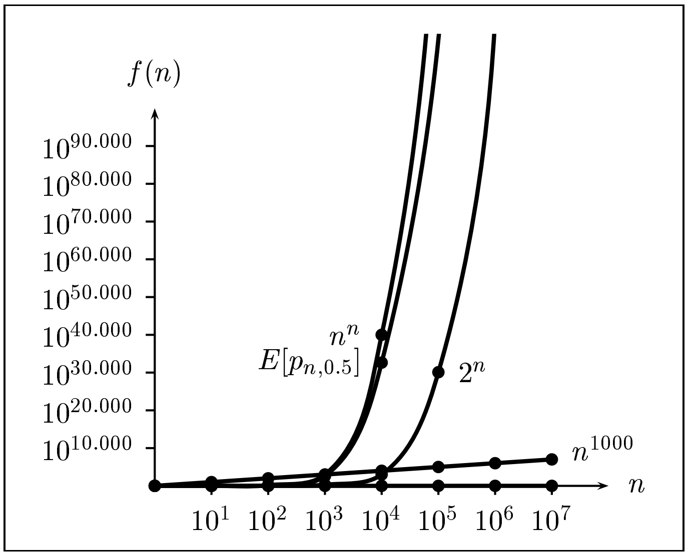
2.2 Auf den Spuren des Zufalls

Der einfachste Ansatz zur Bestimmung eines kürzesten Weges ist Enumeration. Doch welcher Verkehrsbetrieb würde seinen Fahrgästen diesen Rat geben! Die Anzahl der Wege ist zu groß. Wie groß? Das wollen wir nun herausfinden. Eine Formel für die genaue Wegezahl in einem beliebigen Graphen gibt es leider nicht. Diese Frage ist zu komplex. Erstaunlicherweise gibt es aber eine Formel für die "ungefähre Anzahl der Wege in einem durchschnittliche Graphen mit n Knoten".

Eine einfache Methode, einen "durchschnittlichen" Graphen durch Münzwürfe zu erzeugen, zeigt Fig. 22: Für jede der $(n^2-n)/2$ möglichen Kanten wird ein Euro geworfen. Zahl zeichnet die Kante, Adler verwirft sie. (Wir werfen "deutsche Euros" mit Adlern.) Es entsteht eine *Realisierung* des *Zufallsgraphen* G_n auf n Knoten. Jede Realisierung enthält zwar andere Kanten, aber im Durchschnitt weiß man über diese Graphen eine Menge. Z. B. hat G_n eine *erwartete Anzahl* von $(n^2-n)/4$ Kanten, die Hälfte der möglichen, denn jede einzelne existiert mit Wahrscheinlichkeit 1/2.



Figur 22: Ein Zufallsgraph entsteht.



Figur 23: Kombinatorische Explosion.

Ähnlich kann man die *erwartete Anzahl an Wegen* in G_n zwischen zwei zufällig gewählten Knoten s und t berechnen. Betrachten wir einmal nur die längstmöglichen (s,t)-Wege. Diese haben $n-1$ Kanten. Es gibt $(n-2)!$ solche Wege, einen für jede Anordnung der $n-2$ "inneren" Knoten. Doch nicht alle diese Wege existieren in jeder Realisierung des Zufallsgraphen G_n. Die erste Kante existiert in 1/2 der Fälle, die erste und die zweite in 1/4 der Fälle, usw. Alle $n-1$ Kanten existieren in $1/2^{n-1}$ der Fälle. Im Durchschnitt kann man

$$\frac{(n-2)!}{2^{n-1}} = \frac{n!}{2^{n-1}} \cdot \frac{1}{n(n-1)} \approx \frac{n^n e^{-n}}{2^{n-1}} \cdot \frac{\sqrt{2\pi n}}{n(n-1)} = \left(\frac{n}{2e}\right)^{n-2} \cdot \frac{n^2\sqrt{2\pi n}}{2n(n-1)e^2} \geq \left(\frac{n}{2e}\right)^{n-2} \cdot \frac{\sqrt{n}}{e^2}$$

(s,t)-Wege der Länge $n-1$ erwarten, wobei der relative Fehler der Approximation durch die *Stirlingformel*

$$n! \approx n^n e^{-n}\sqrt{2\pi n}$$

für $n \geq 10$ kleiner als 1% ist. Mit etwas mehr Algebra kann man die erwartete Anzahl $E[p_{n,\rho}]$ aller (s,t)-Wege in G_n wie folgt abschätzen:

$$E[p_{n,\rho}] \approx \left(\frac{n\rho}{e}\right)^{n-2} \cdot e^{1/\rho-2} \cdot \rho \cdot \sqrt{2\pi n}$$

wobei ρ die Wahrscheinlichkeit für das Auftreten von Kanten ist (beim Münzwurf ist $\rho = 1/2$). Alle diese Anzahlen haben eine *Größenordnung* von mehr als $(cn)^{n-2}$ (c eine Konstante). Fig. 23 zeigt auf einer doppelt logarithmischen Skala das enorme Wachstum solcher Funktionen. Schon in unserem kleinen Schnellbahnbeispiel mit nur 336 Knoten und Kantenwahrscheinlichkeit $\rho = 445/\frac{336^2-336}{2} = 0.008$ sind nicht weniger als $E[p_{336,0.008}] = 1.977 \cdot 10^{50}$ Wege zu erwarten! Dieses Wachstumsphänomen ist bekannt als *Kombinatorische Explosion*. Entkommen kann man ihr nicht. Der Ausweg besteht darin, den riesigen *Lösungsraum* zielgerichtet zu durchsuchen. Euler hat das seinerzeit sofort erkannt:

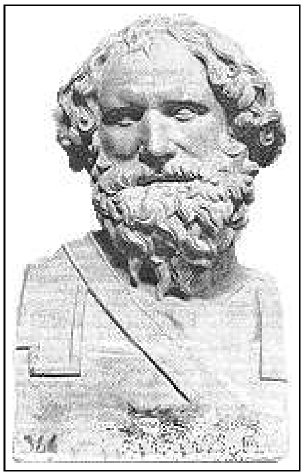
Was das Königsberger Problem von den sieben Brücken betrifft, so könnte man es lösen durch eine genaue Aufzählung aller Gänge, die möglich sind; denn dann wüßte man, ob einer derselben der Bedingung genügt oder keiner. Diese Lösungsart ist aber wegen der großen Zahl von Kombinationen zu mühsam und schwierig, und zudem könnte sie in andern Fragen, wo noch viel mehr Brücken vorhanden sind, gar nicht mehr angewendet werden. Würde die Untersuchung in der eben erwähnten Weise geführt, so würde Vieles gefunden, wonach gar nicht gefragt war; dies ist zweifellos der Grund, warum dieser Weg so beschwerlich wäre. Darum haben ich diese Methode fallengelassen und eine andere gesucht, die nur so weit reicht, daß sie erweist, ob ein solcher Spaziergang gefunden werden kann oder nicht; denn ich vermutete, daß eine solche Methode viel einfacher sein würde.

2.3 Münchhausen versus Archimedes

Eulers Rat, keine überflüssigen Rechnungen durchzuführen, läßt sich auf das Kürzeste-Wege-Problem übertragen. Offensichtlich können viele Wege keine kürzesten sein. Über den dunkelgrauen Artikulationsknoten "Lichtenberg" in Fig. 18 führt der kürzeste Weg vom Alexanderplatz nach Dahlem-Dorf nicht, denn sicher ist der Weg Lichtenberg–Dahlem Dorf allein schon länger als 15 Kanten — oder? Jetzt brauchen wir schon zwei kürzeste Wege und die Fortsetzung dieser Argumentation erinnert sehr an die von Freiherr Karl Friedrich Hieronymus von Münchhausen (1720–1796) erfundene Methode, sich an den eigenen Haaren aus dem Sumpf zu ziehen, siehe Fig. 25.

Figur 24: Münchhausen.

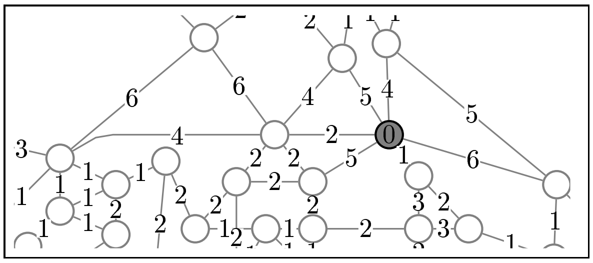
Figur 25: Methode Münchhausen.



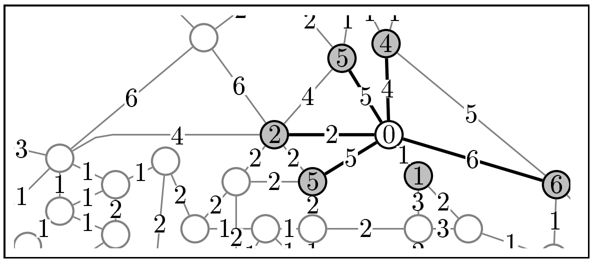
Figur 26: Archimedes.

"Leider" hat der Lügenbaron die Geschichte zur Unterhaltung seiner Gäste nur erfunden. Wir müssen den festen Boden eines wirklichen kürzesten Weges gewinnen, um aus dem Sumpf zu entkommen. Hier ist einer: Der "leere Weg" der Länge 0 vom Alexanderplatz zu sich selbst. Sie finden das ziemlich dürftig? Archimedes von Syrakus (287–212) war anderer Meinung! "Gebt mir einen Platz, wo ich stehen kann, so will ich die Erde bewegen." hat er gesagt. Ob Archimedes Punkt der Methode Münchhausens vorzuziehen ist?

Das "Daumenkino" Fig. 27–32 zeigt, wie man von einem archimedischen Punkt aus den ganzen Graphen mit kürzesten Wegen gangbar macht. Das Verfahren basiert auf dem Konzept einer *Distanzmarke* (distance label), die für jeden (erreichten) Knoten die Länge des kürzesten bisher entdeckten Weges angibt. Die Wege selbst bilden einen *Kürzeste-Wege-Baum*.



Figur 27: Dijkstras Algorithmus: Schritt 0.



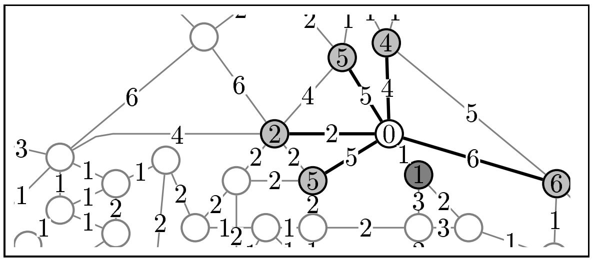
Figur 28: Dijkstras Algorithmus: Schritt 1.

Initialisierung. Der Knoten "Alexanderplatz" wird mit Distanz 0 *temporär markiert.*

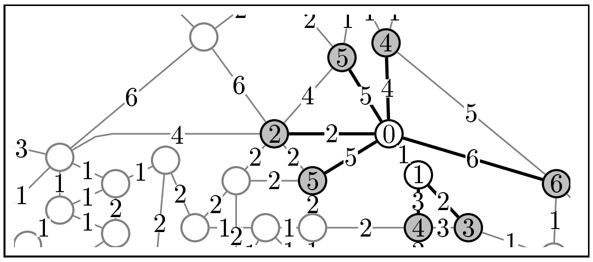
Fig. 27. Der Knoten Alexanderplatz hat die zur Zeit kleinste temporären Distanzmarke und wird *selektiert.* Der Knoten kann nicht auf einem kürzeren Weg erreicht werden. Die temporäre Distanzmarke des Alexanderplatzes wird deshalb *permanent* gemacht.

Fig. 28. Die Nachbarknoten des Alexanderplatzes werden mit den Entfernungen dorthin *markiert*, die Kanten in einen *Kürzeste-Wege-Baum* eingefügt. Markierung und Baum sind *temporär.* (Es kann kürzere Wege geben, die nicht direkt vom Alexanderplatz kommen.)

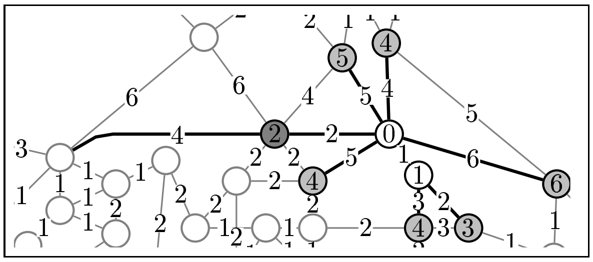
Fig. 29. Der Knoten "Jannowitzbrücke" mit der Distanzmarke 1 kann nicht auf einem kürzeren Weg erreicht werden. Diese entscheidende Beobachtung liefert den zweiten festen archimedischen Punkt. Knoten Jannowitzbrücke wird *selektiert*, Markierung und Kante im Kürzeste-Wege-Baum werden *permanent* gemacht.



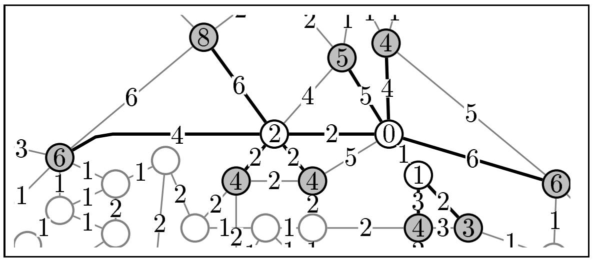
Figur 29: Dijkstras Algorithmus: Schritt 2.



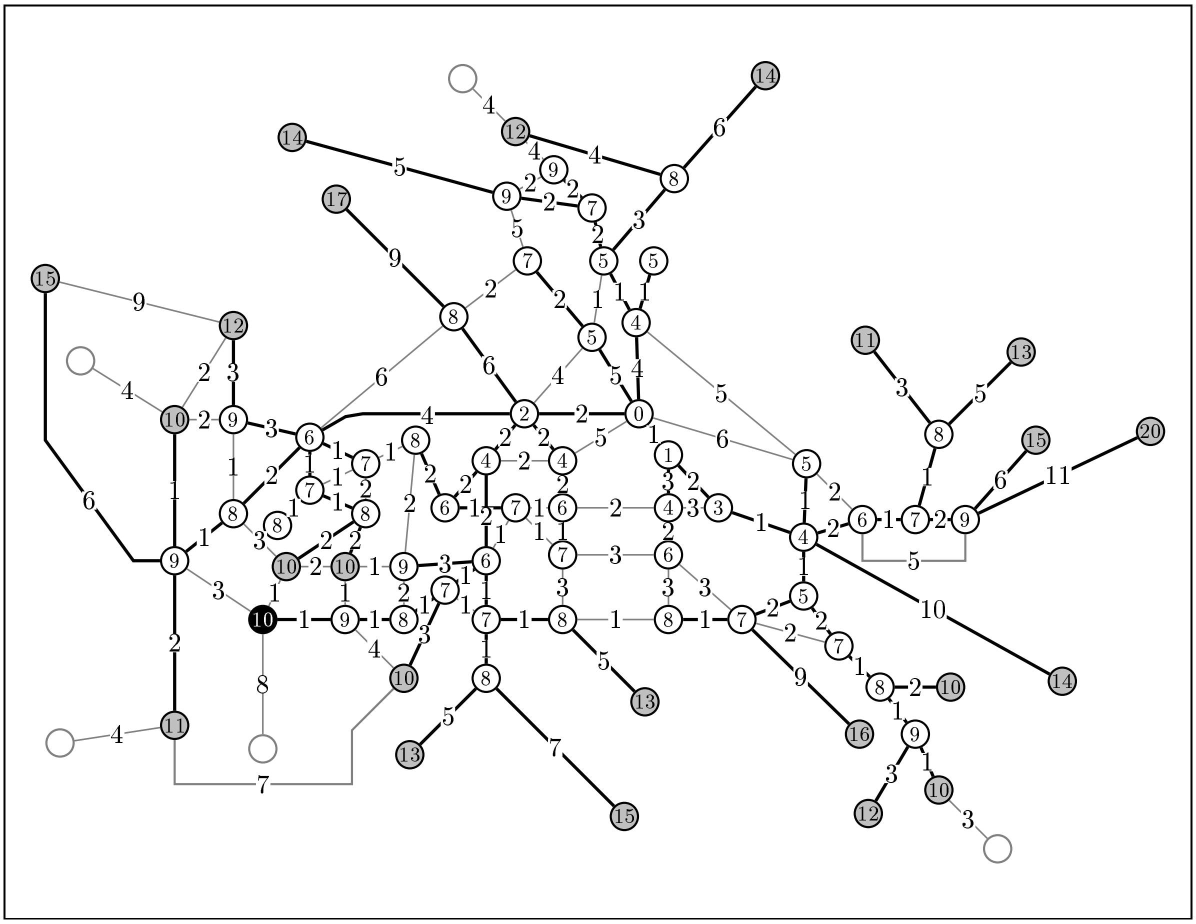
Figur 30: Dijkstras Algorithmus: Schritt 3.



Figur 31: Dijkstras Algorithmus: Schritt 4.



Figur 32: Dijkstras Algorithmus: Schritt 5.



Figur 33: Ein Kürzeste-Wege-Baum.

Fig. 30. Die temporären Distanzmarken der Nachbarn des Knotens Jannowitzbrücke werden *aktualisiert* (label update). Zwei neue Knoten werden entdeckt und *temporär markiert.*

Fig. 31. Analog zu Schritt 0 wird der Knoten "Friedrichstraße" mit dem zur Zeit kleinsten temporären Label 2 *selektiert.* Label und Kürzeste-Wege-Kante werden *permanent* gemacht.

Fig. 32. Beim Distanz-Update der Nachbarknoten werden drei neue Knoten *temporär markiert.* Die Distanz an einem temporär markierten Knoten verringert sich von 5 auf 4. Marke und Kürzeste-Wege-Baum werden entsprechend aktualisiert.

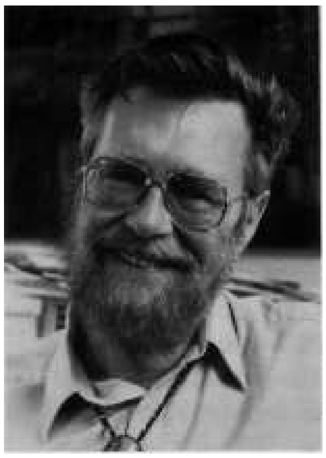
Ende. Die Rechnung endet, wenn der Zielknoten, hier der (schwarze) "Heidelberger Platz", permanent markiert ist (oder wenn alle Knoten permanent markiert sind). Probieren Sie es einmal aus! Sie erhalten Marken und einen Baum wie in Fig. 33. (Der Kürzeste-Wege-Baum kann durch Wahlmöglichkeiten bei gleichen Distanzlabeln ein wenig anders aussehen.) Der Baum enthält eine Überraschung: Statt in 11 Kanten wie in Fig. 17 kann man den Heidelberger Platz auch in 10 erreichen, und damit Dahlem-Dorf in 14 statt 15 Kanten.

*

Eine allgemeine Beschreibung des Verfahrens gibt der *Pseudocode* in Fig. 34. `d(v)` und `pred(v)` sind `arrays` für Distanzmarken und Vorgängerknoten (predecessor) auf dem kürzesten Weg zum Start s. T (tree) ist die Menge der permanent markierten Knoten. $\delta(v)$ ist die Liste der Nachbarn des Knotens v. $w(u,v)$ ist die Länge der Kante von u nach v.

```
algorithm Dijkstra
   forall v ∈ V do d(v) ← ∞;
   d(s) ← 0; pred(s) ← s; T ← {s};
   while (T ≠ ∅) do
      bestimme v ∈ T mit d(v) = min_{u∈T} d(u);
      T ← T ∪ {v};
      forall u ∈ δ(v) do
         if d(u) > d(v) + w(v,u) then
            d(u) ← d(v) + w(v,u); pred(u) ← v;
         endif
      endforall
   endwhile
endalgorithm
```

Figur 34: Dijkstras Algorithmus.



Figur 35: E.W. Dijkstra.

Dijkstras Algorithmus heißt das Verfahren nach seinem Erfinder, dem holländischen Mathematiker Edsger Wybe Dijkstra, der es 1959 zuerst vorgeschlagen hat. Vorausgegangen war der erste Knotenmarkierungsalgorithmus überhaupt des amerikanischen Mathematikers L.R. Ford Jr. aus dem Jahre 1956, von dem sich Dijkstras durch das Konzept der permanenten Markierung von Knoten und die Angabe einer Regel zu deren Auswahl abhebt; Fords Algorithmus markierte in beliebiger Reihenfolge. Ein kleiner aber feiner Unterschied für die Effizienz! Mit den permanenten Marken spart Dijkstras Algorithmus Arbeit gegenüber der Enumeration, indem alle Wege verworfen werden, deren Anfänge nicht in dem wachsenden Kürzeste-Wege-Baum enthalten sind.

Flexibilität ist ein weiteres Plus von Dijkstras Algorithmus. Wir geben drei Beispiele.

Gerichtete Graphen. Das Verfahren funktioniert ohne jede Änderung auch in einem *gerichteten Graphen* (directed graph, kurz: Digraph), dessen *Bögen* wie Einbahnstraßen nur in einer Richtung durchlaufen werden können.

Wabensysteme mit abnehmenden Grenzpreisen (Abschnitt 2.1) behandelt man, indem beim Distanz-Update der korrekte Zusatzpreis addiert wird.

Fahrpläne erfordern Zeitmarken und eine darauf gestützte Suche nach Anschlüssen.

*

Fahrgastinformationssysteme wie die in Tab. 1 genannten basieren alle auf Dijkstras Algorithmus. Das "Mathematics Inside" läßt sich nur vor lauter Oberflächen und Visualisierungen manchmal kaum erkennen. Das gilt natürlich nicht für die Leser dieses Artikels!

Firma	`URL (http://)`
Berliner Verkehrsbetriebe	`www.fahrinfo-berlin.de/fahrinfo/de`
Deutsche Bahn	`bahn.hafas.de`
Norddeutsche Verkehrsbetriebe	`www.efa.de`

Tabelle 1: Fahrgastinformation im Internet.

2.4 Arbeitszeitgesetz für Algorithmen

2.2 Theorem (Rechenaufwand von Dijkstras Algorithmus)
Dijkstras Algorithmus kann auf einer Random Access Maschine (Computertyp mit Adreßarithmetik, kurz: RAM) so implementiert werden, daß der Rechenaufwand für einen Graphen mit n Knoten $O(n^2)$ Operationen beträgt.

Fig. 36 zeigt eine geeignete Implementation. Der Code deutet an, daß man den Rechenaufwand bestimmt, indem man schlicht Zeile für Zeile vorgeht (Wiederholungen in Schleifen werden mehrfach gezählt) und zusammenzählt. Wie diese Rechnung durchgeführt wird und was Theorem 2.2 im einzelnen bedeutet, davon handelt der Rest dieses Abschnittes.

O-Notation ist ein Konzept zum Vergleich des Wachstumsverhaltens von Funktionen.

$$g(n) = O\big(f(n)\big) \iff \text{es existiert eine Konstante } C \text{ mit } g(n) \leq C \cdot f(n) \text{ für alle } n,$$

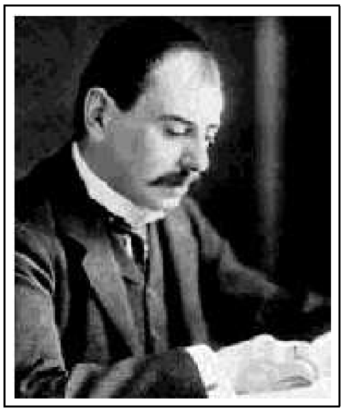
d. h. $g(n) = O\big(f(n)\big)$ (sprich: g ist von der (Größen-)*Ordnung* von f) bedeutet, daß $g(n)$ bis auf Multiplikation mit einer Konstanten stets $\leq f(n)$ ist. Die wichtigsten *Rechenregeln* sind

$$O(1) + O(1) = O(1), \quad O(1) + O(n) = O(n), \quad O(1) + O(n) + O(n^2) = O(n^2)$$

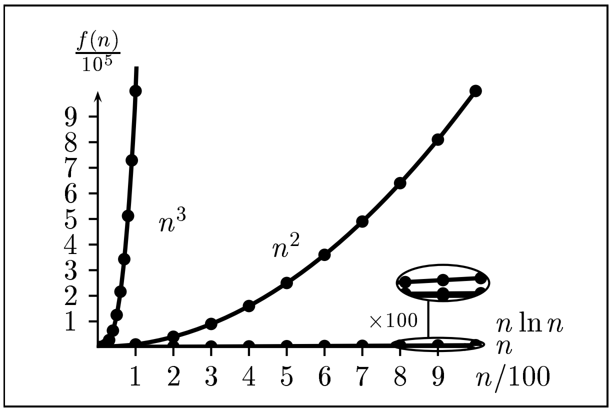
usw.; bei einem Polynom bestimmt der höchste Exponent die Ordnung (daher der Name). Das "Landausche O" wurde von den Zahlentheoretikern Paul Gustav Heinrich Bachmann (1837–1920) und Edmund Georg Hermann Landau (1877–1938) erfunden bzw. popularisiert.

```
algorithm Dijkstra
   forall v ∈ V do d(v) ← ∞; T(v) ← 0;                n× O(1)          = O(n)
   d(s) ← 0; pred(s) ← s;                                               O(1)
   forever do                                        n×
      min_d ← ∞;                                                 O(1)
      forall u ∈ V do                                   n×
         if T(u) = 0 && d(u) < min_d then                  O(1)
            min_d ← d(u); v ← u;                           O(1)
         endif
      endforall                                         n× O(1) = O(n)
      if min_d = ∞ then break;                                   O(1)
      T(v) ← 1;                                                  O(1)
      forall u ∈ δ(v) do                                n×
         if d(u) > d(v) + w(v,u) then                      O(1)
            d(u) ← d(v) + w(v,u); pred(u) ← v;             O(1)
         endif
      endforall                                         n× O(1) = O(n)
   endforever                                        n×          O(n) = O(n²)
endalgorithm                                                            O(n²)
```

Figur 36: Laufzeit von Dijkstras Algorithmus.



Figur 37: E.G.H. Landau.



Figur 38: Polynomiale Funktionen.

Die Größe von Eingabedaten läßt sich mit der O-Notation messen. Wenn man einen Graphen in einer *Adjazenzmatrix* wie in Fig. 22 speichert und dabei Zahl und Adler durch die ganze Zahlen 1 und 0 vom Typ `int` ersetzt, dann ergibt sich ein Speicherplatzbedarf von $\frac{n^2-n}{2} \cdot$ `sizeof(int)` Bits. Der Wert `sizeof(int)` ist eine *hardwareabhängige* Konstante, die Bitlänge einer ganzen Zahl. Datenstrukturen und Hardware führen zu unterschiedlichem Speicherplatzbedarf. Die meisten Abweichungen betreffen aber nicht die *Ordnung* des Platzbedarfes in Abhängigkeit von n, sie verändern nur Konstanten wie `sizeof(int)`. Für einen Graphen mit n Knoten kann man daher von einem Platzbedarf in der Größenordnung von höchstens $O(n^2)$ (Bits) ausgehen.

Worst-Case-Analyse. Der Preis für das Denken in Größenordnungen ist, daß man innerhalb einer Größenklasse immer mit dem ungünstigsten Fall, dem Worst Case, rechnen muß. Dies gilt für Eingabedaten und Rechenzeit.

Die Random Access Maschine (RAM) ist ein Computertyp, der nur vier *elementare Operationen* kennt: $+, -$ (Addition), $*, /$ (Multiplikation), $<, >, =$ (Vergleich) und $\leftarrow$ (Zuweisung). Random Access bedeutet, daß es keine Einschränkungen beim Speicherzugriff gibt. (Das in den Literaturhinweisen genannte Buch von Mehlhorn erklärt die wichtigen Feinheiten dieser Definition.) Das RAM-Modell nivelliert Hardware- und Systemunterschiede in der Mikroprogrammierung: Zählen der elementaren Operationen gibt die Ordnung der dafür benötigten CPU-Taktzyklen auf einem realen Rechner.

Die Laufzeit eines Programms auf einer RAM ist die Anzahl der durchgeführten elementaren Operationen. Man kann zeigen, daß dieses vereinfachende Maß tatsächlich die Ordnung der CPU-Taktzyklen bestimmt. (Das gilt allerdings nur auf den heutigen Rechnern, auf den Parallelrechnern der Zukunft gelten andere Gesetze!) Der größte Vorteil des Begriffs ist, daß die Ordnung der Laufzeit auch unabhängig von Implementationsdetails ist: Ein Befehl hin oder her oder auch zehnmal so viele Befehle machen keinen Unterschied. Diese Invarianz erlaubt es, von *Laufzeiten von Algorithmen* statt von Implementationen zu sprechen.

*

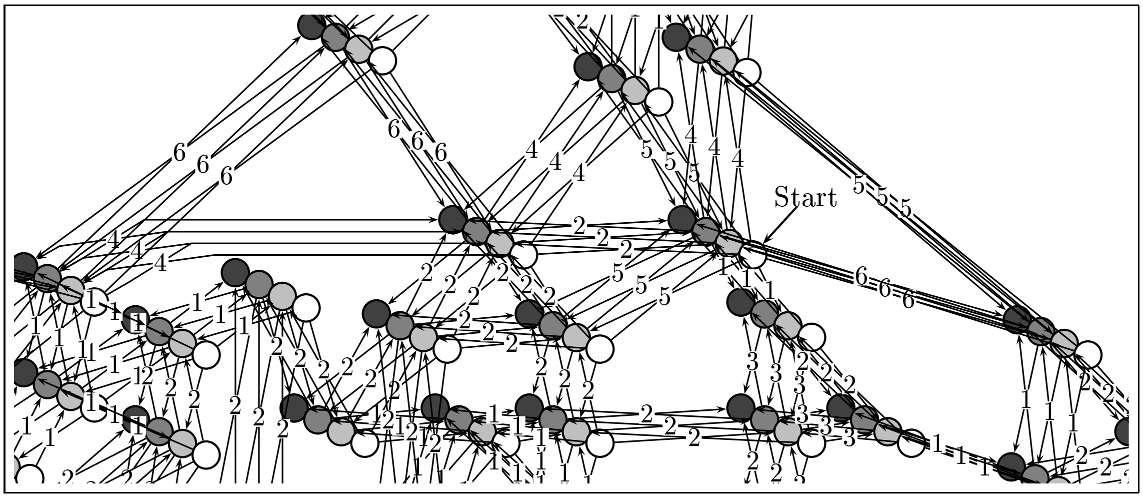
Die Anwendung der Worst-Case-Analyse auf Dijkstras Algorithmus zeigt Fig. 36: Als Gesamtlaufzeit ergibt sich $O(n^2)$. Diese *polynomiale Funktion* zeigt ein anderes Verhalten als die *(super-)exponentiellen Funktionen* in Fig. 23, die die Laufzeit der Enumeration bestimmen. Dieser Qualitätssprung ist die quantitative Begründung für die Überlegenheit von Dijkstras Algorithmus. Daß es auch im polynomialen Bereich deutliche Unterschiede gibt, zeigt die lineare Skala von Fig. 38. Die Quantensprünge im Wachstumsverhalten sind ein großer Ansporn zur Reduktion der Laufzeitordnung. Dijkstras Algorithmus bietet dazu Potenzial. Mit besseren Listen- und Heap-Datenstrukturen kann man die Laufzeit relativ leicht auf $O(m \ln n)$ drücken (falls $m < n^2/\ln n$), wobei m die Anzahl der Kanten ist. Noch ausgefeiltere Datenstrukturen wie eigens zu diesem Zweck entwickelte "Fibonacci Heaps" setzen weiteres Potenzial frei. Schneller als $O(m)$ kann man nicht werden, denn soviel Zeit benötigt man allein zum Einlesen der Daten. Gibt es einen *optimalen SSSP-Algorithmus* mit dieser Laufzeit? Der dänische Mathematiker Mikkel Thorup hat 1996 gezeigt, daß das tatsächlich möglich ist! Es gibt allerdings einen kleinen Haken: Das Verfahren basiert auf einer sog. "Atomic Heap" Datenstruktur der amerikanischen Mathematiker Michael L. Fredman und Daniel E. Willard, deren Verwendung $n > 2^{12^{20}}$ voraussetzt. $2^{12^{20}} = O(1)$, aber *diese* Konstante ist doch so groß, daß der Algorithmus nicht implementierbar ist! Was nun? Thorup schlägt eine "abgespeckte" Variante vor, bei der sich die Laufzeit auf $O(\ln C + m + n \ln\ln\ln n)$ "verschlechtert" (C ist das größte Kantengewicht). Sie sehen schon: Das Thema SSSP wird noch Fortsetzungen haben!

2.5 Beschränkte Ressourcen

Kürzeste-Wege-Probleme treten selten in Reinform auf. Schon bei Tell gab es *Bedingungen* an die Form der Wege. *Ressourcenbeschränkungen* sind dafür ein brauchbarer Rahmen. Solche Modelle sehen neben der Zielfunktion eine Anzahl von *Ressourcen* vor, die entlang der Kanten verbraucht werden. An den Knoten schränken Bedingungen an zulässige *Zustände* des Ressourcenverbrauchs die Form der ankommenden Wege ein. Wir geben zwei Beispiele.

Zeitfenster sind Intervalle $[a_v, b_v]$ an den Knoten $v \in V$, die den Verbrauch einer Zeitressource t_{uv} für das Durchlaufen der Kanten $uv \in E$ einschränken.

Weglängenbeschränkungen werden wie Zeitfenster behandelt. Wenn der Begriff Länge von einer Ressource besetzt ist, spricht man bei der Zielfunktion vom Gewicht.



Figur 39: Ressourcen-Graph.

Kürzeste-Wege-Probleme mit (diskreten) Ressourcenbeschränkungen lassen sich durch eine *Transformation* auf die *gerichtete* Standartform (vgl. Seite 14) zurückführen. Fig. 39 zeigt das Prinzip an einer Variante des Schnellbahnbeispiels mit Längenbeschränkung: Gesucht sind die kürzesten Wege mit genau 3 Kanten, die am Alexanderplatz starten. Entlang jeder Kante wird eine Einheit der Kantenressource verbraucht. An jedem Knoten v im Originalgraphen gibt es vier mögliche zulässige Ressourcenzustände $R_v = \{0, 1, 2, 3\}$ (ein ankommender Weg hat bisher 0, 1, 2 oder 3 Kanten "verbraucht"). Die Transformation stellt von jedem Originalknoten vier Kopien her, eine für jeden Ressourcenzustand. In Fig. 39 sind die Kopien in vier Ebenen 0–3 angeordnet. Gerichtete Bögen verbinden Knotenkopien, deren Originale benachbart sind; die Bögen verlaufen immer "aufwärts" von der niedriger zur höher numerierten Ebene, die Gewichte sind dieselben wie bei den Originalkanten. Der so konstruierte *(Ressourcen-)Zustandsdigraph* hat die Eigenschaft, daß die kürzesten Wege vom Alexanderplatz in Ebene 0 zu den Knoten in Ebene 3 genau den kürzesten Wegen mit 3 Kanten im Originalgraphen entsprechen. Die gerichteten Bögen verhindern die Entstehung von längeren Wege durch "Zurückspringen".

Diese Konstruktion funktioniert allgemein für diskrete Ressourcenzustände R_v. Aus einem Knoten im Originalgraphen entstehen dabei $|R_v|$ Kopien im Zustandsdigraphen, die nach Maßgabe möglicher Ressourcenübergänge durch Bögen verbunden werden. Die Laufzeit von Dijkstras Algorithmus beträgt dort $O\big((\sum_{v\in V} |R_v|)^2\big)$. Diese Größe ist im allgemeinen *nicht* polynomial in der Anzahl n der Knoten des Ausgangsdigraphen. Nur unter besonderen Bedingungen, z. B. wenn $\sum_{v\in V} |R_v|$ polynomial in n ist, ergeben sich polynomiale Laufzeiten. Man nennt ein solches Laufzeitverhalten *pseudopolynomial*.

Die Konstruktion des Zustandsdigraphen bietet Raum für Verbesserungen. Die Idee ist, die Transformation nur *implizit* durchzuführen und im Originaldigraphen mit mehreren Marken pro Knoten zu rechnen. Diese *Multilabel SSSP Algorithmen* benötigen weniger Speicherplatz, manchmal (wie im Fahrplanbeispiel von Seite 14) ist auch die Laufzeit günstiger.

Zuletzt wieder eine Quizfrage: Warum liefert eine n-Ebenen Transformation keine schnellen Algorithmen für das HCP?

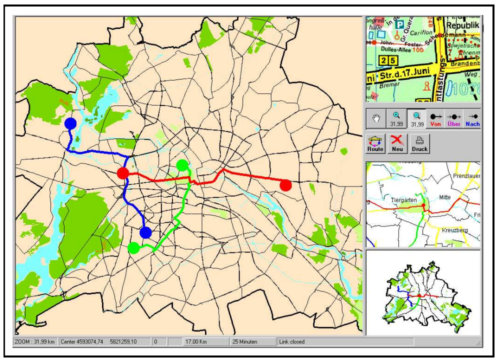
3 Kombinationen von Wegen

Auf die Kürzesten-Wege-Probleme folgt als nächste Stufe die gleichzeitige Planung *mehrerer Wege*. Die typische Schwierigkeit ist, gelegentlich den "richtigen Umweg" zu machen. Wir versuchen nicht, eine Übersicht über die Überdeckungs-, Packungs-, und Partitionierungsprobleme mit Wegen zu geben, um nur die bekanntesten Typen zu nennen und beschränken uns auf ein repräsentatives Beispiel: Branch & Price Verfahren für Set Partitioning Modelle der Fahrzeugeinsatzplanung.

3.1 Telebus Light

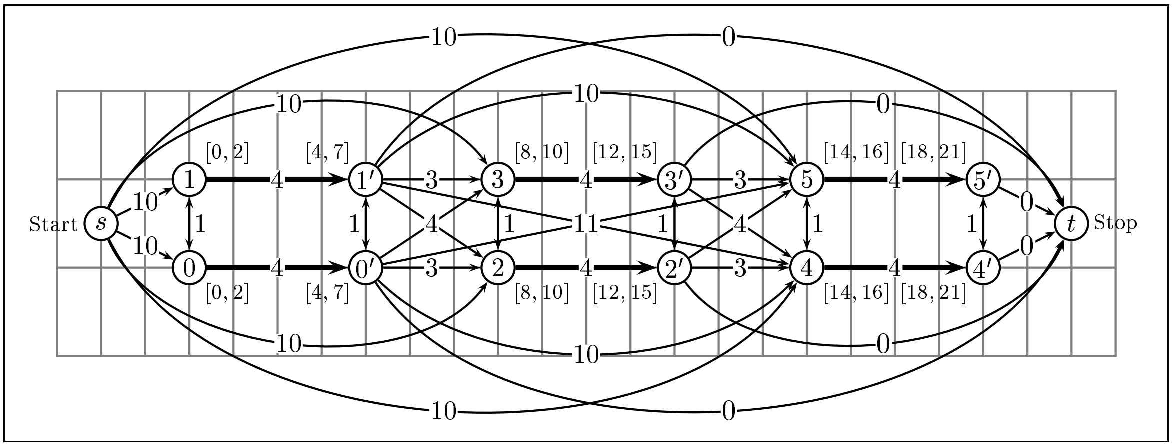
Telebus ist der soziale Behindertenfahrdienst der Stadt Berlin, die mit einer Flotte von Kleinbussen alten und behinderten Bürgern Mobilität verschafft. Jeder Berechtigte (zur Zeit ca. 27.000 Personen) kann pro Monat bis zu 40 Fahrten bei der Telebus-Zentrale buchen. Zum gewünschten Zeitpunkt (mit einer gewissen Toleranz) erscheint der Telebus und bringt den Fahrgast (u.U. gemeinsam mit anderen Personen) zu seinem Ziel, siehe Fig. 40. Bei Bedarf leistet das Personal Hilfe. Auf diese Weise werden jeden Tag 1.500 bis 2.000 Fahrten durchgeführt. Telebus ist ein *Anrufsammeltaxisystem*. Es gibt eine Vorbestellzeit, so daß die (meisten) Bestellungen am Vorabend vorliegen. Die Zentrale disponiert daraus (Bus-) *Touren*. Die Busse werden auf Tagesbasis von karitativen und kommerziellen Anbietern gemietet. Das planerische Kernproblem ist die *Fahrzeugdisposition*. Sie bestimmt Kosten und Servicequalität.

Figur 40: Der Telebus trifft ein.



Figur 41: Disposition von Telebussen.

Fig. 42 zeigt ein "ereignisorientiertes" graphentheoretisches Modell eines vereinfachten *Telebus-Dispositionsproblems*. Es sind 6 Fahrten 0–5 gebucht, die jeweils vom Startknoten i zum Ziel i' führen. Diese Knoten stellen keine Orte dar, sondern die *Ereignisse* des Fahrtbeginns und -endes. Zur leichteren Orientierung haben wir aber doch ein (amerikanisch anmutendes) Straßengitter unterlegt; wir nehmen an, daß der zeitliche Abstand zwischen benachbarten Kreuzungen immer gleich ist (d. h. man fährt senkrecht doppelt so schnell wie waagerecht, eine Skalierung aus zeichnerischen Gründen). Aus der Lage der Ereignisknoten kann man so Fahrzeiten ablesen, die auf den Bögen eingetragen sind. (Zu den Bögen von s und t kommen wir noch.) Zu jedem Startknoten gehört ein Zeitfenster, das von der gewünschten Abholzeit 2 Einheiten nach hinten reicht. Telebus verschafft sich damit dispositive Freiheit. Aus Startfenster plus Fahrzeit ergibt sich ein Zielfenster, das Telebus nochmals um eine Einheit verlängert, um Spiel für das "Einbinden" anderer Fahrten zu gewinnen. Die Busse können zwei Personen gleichzeitig befördern. Der Ein- und Ausstieg geschieht ohne Zeitverlust. Es gilt die Regel, daß mit einem Fahrgast an Bord nicht auf später zusteigende Fahrgäste gewartet werden darf. Die Kosten für die Busse werden nach der Fahrstrecke abgerechnet. Hinzu kommt eine Anfahrtspauschale von 10 Einheiten. s und t sind *Depotknoten*, die Anfang und Ende einer Tour markieren. Die Anfahrtspauschale ist auf die Einsatzbögen gelegt. Das Telebus-Dispositionsproblem ist eine Frage der kostenminimalen *Partitionierung der Ereignisknoten durch Wege* (= Bus-Touren): Jeder Knoten muß auf genau einer Tour liegen.



Figur 42: Vereinfachtes Telebus-Beispiel.

Die Probleme der echten Telebus-Disposition kann man sich vorstellen. Abgesehen von mehr Daten und Regeln tritt dort die Schwierigkeit auf, daß sich Regeln und Ziele (durch Verhandlungen, politische Vorgaben etc.) ändern. Die Gefahr in dieser Situation ist, daß man mit *ad hoc Methoden* ein kompliziertes Modell entwickelt, das sich schwer an Veränderungen anpassen läßt und das Lösungen von zweifelhafter Qualität liefert. Die mathematische Alternative stellen wir im nächsten Abschnitt vor.

3.2 Set Partitioning Modelle

Der Ausgangspunkt für eine bessere Methode ist eine *gedankliche Enumeration* aller Touren. Tab. 2 listet die Ergebnisse in einer komprimierten Form. Jede Spalte steht für eine von 49 möglichen Touren. Oben ist für jede Tour j der Kostenkoeffizient c_j aufgeführt. Es folgt der Tour-*Inzidenzvektor* $a_{\cdot j}$ ($\cdot$ ist ein Platzhalter für einen Index). Er gibt an, welche Fahrten eine Tour durchführt (1) oder nicht durchführt (.). Zeile i gehört dabei zu Fahrt i, $i = 0, \ldots, 5$. Die Übersetzung von Inzidenzvektoren in Bustouren ist nicht immer eindeutig. Tour 6 kann man z. B. "oben herum" (zuerst 0 abholen (Ereignis 0), dann 1 abholen (Ereignis 1), dann 1 abliefern (Ereignis $1'$) und zuletzt 0 abliefern (Ereignis $0'$)) und "unten herum" $(1, 0, 0', 1')$ fahren, und man kann zwischen Abfahrtszeit 0 und 1 wählen. Doch diese Wahlmöglichkeiten sind kostenneutral. Der Inzidenzvektor läßt in diesem Sinne bei der Auswahl eines Tour-*Repräsentanten* etwas Spiel. Ausschließen kann man "dominierte" (d. h. nicht kostenminimale) Touren mit sinnlosen Umwegen.

Diese Vorbereitung macht es leicht, das Telebus-Dispositionsproblem als ein *ganzzahliges Programm* zu formulieren, einem der Modelltypen der diskreten Optimierung. Fig. 43 zeigt das Programm in einem von vielen Codes akzeptierten "LP-Format". Das Programm enthält

c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}	c_{26}	c_{27}	c_{28}	c_{29}	c_{30}	c_{31}	c_{32}	c_{33}	c_{34}	c_{35}	c_{36}	c_{37}	c_{38}	c_{39}	c_{40}	c_{41}	c_{42}	c_{43}	c_{44}	c_{45}	c_{46}	c_{47}	c_{48}
14	14	14	14	14	14	16	21	22	28	29	22	21	29	28	16	21	22	22	21	16	23	23	30	30	23	28	29	30	29	30	23	29	30	29	28	30	23	23	25	30	31	31	30	32	30	31	31	30
$a_{\cdot 0}$	$a_{\cdot 1}$	$a_{\cdot 2}$	$a_{\cdot 3}$	$a_{\cdot 4}$	$a_{\cdot 5}$	$a_{\cdot 6}$	$a_{\cdot 7}$	$a_{\cdot 8}$	$a_{\cdot 9}$	$a_{\cdot 10}$	$a_{\cdot 11}$	$a_{\cdot 12}$	$a_{\cdot 13}$	$a_{\cdot 14}$	$a_{\cdot 15}$	$a_{\cdot 16}$	$a_{\cdot 17}$	$a_{\cdot 18}$	$a_{\cdot 19}$	$a_{\cdot 20}$	$a_{\cdot 21}$	$a_{\cdot 22}$	$a_{\cdot 23}$	$a_{\cdot 24}$	$a_{\cdot 25}$	$a_{\cdot 26}$	$a_{\cdot 27}$	$a_{\cdot 28}$	$a_{\cdot 29}$	$a_{\cdot 30}$	$a_{\cdot 31}$	$a_{\cdot 32}$	$a_{\cdot 33}$	$a_{\cdot 34}$	$a_{\cdot 35}$	$a_{\cdot 36}$	$a_{\cdot 37}$	$a_{\cdot 38}$	$a_{\cdot 39}$	$a_{\cdot 40}$	$a_{\cdot 41}$	$a_{\cdot 42}$	$a_{\cdot 43}$	$a_{\cdot 44}$	$a_{\cdot 45}$	$a_{\cdot 46}$	$a_{\cdot 47}$	$a_{\cdot 48}$
1	.	.	.	.	.	1	1	1	1	1	.	.	.	.	.	.	.	.	.	.	1	1	1	1	1	1	1	1	1	1	.	.	.	.	.	.	.	.	1	1	1	1	1	1	1	1	.	.
.	1	.	.	.	.	1	.	.	.	.	1	1	1	1	.	.	.	.	.	.	1	1	1	1	.	.	.	.	.	.	1	1	1	1	1	1	.	.	1	1	1	1	1	1	.	.	1	1
.	.	1	.	.	.	.	1	.	.	.	1	.	.	.	1	1	1	.	.	.	1	.	.	.	1	1	1	.	.	.	1	1	1	.	.	.	1	.	1	1	1	.	.	.	1	.	1	.
.	.	.	1	.	.	.	.	1	.	.	.	1	.	.	1	.	.	1	1	.	.	1	.	.	1	.	.	1	1	.	1	.	.	1	1	.	.	1	1	.	.	1	1	.	.	1	.	1
.	.	.	.	1	.	.	.	.	1	.	.	.	1	.	.	1	.	1	.	1	.	.	1	.	.	1	.	1	.	1	.	1	.	1	.	1	1	1	.	1	.	1	.	1	1	1	1	1
.	.	.	.	.	1	.	.	.	.	1	.	.	.	1	.	.	1	.	1	1	.	.	.	1	.	.	1	.	1	1	.	.	1	.	1	1	1	1	.	.	1	.	1	1	1	1	1	1
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}	x_{39}	x_{40}	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}

Tabelle 2: Alle Telebus Touren.

für jede Tour eine 0/1 Entscheidungsvariable x_j (`Binaries`, siehe auch Tab. 2 unten) für die Durchführung einer Tour ($x_j = 1$) oder Nichtdurchführung ($x_j = 0$). Die Zielfunktionszeile (`obj`) summiert die Kosten der durchgeführten Touren, die minimiert werden sollen (`Minimize`). Jede der 6 Gleichungen (`C0-C5`) fordert, daß von allen Touren, die eine Fahrt überdecken, genau eine durchgeführt wird (`Ci` gehört zu Fahrt i). Die *Lösungen* dieses Programms entsprechen genau den möglichen Busdispositionen. Weil dabei die Fahrten in Touren partitioniert werden, nennt man diesen Typ eines ganzzahligen Programms ein *Mengenpartitionierungsproblem* (Set Partitioning Problem, kurz: SPP).

```
Minimize
obj: 14 x0 + 14 x1 + 14 x2 + 14 x3 + 14 x4 + 14 x5 + 16 x6 + 21 x7 + 22 x8 + 28 x9
     +29 x10+22 x11+ 21 x12+ 29 x13+ 28 x14+ 16 x15+ 21 x16+ 22 x17+ 22 x18+ 21 x19
     + 16 x20+23 x21+ 23 x22+30 x23+ 30 x24+ 23 x25+ 28 x26+ 29 x27+ 30 x28+ 29 x29
     + 30 x30+23 x31+ 29 x32+30 x33+ 29 x34+ 28 x35+ 30 x36+ 23 x37+ 23 x38+ 25 x39
     + 30 x40+ 31 x41+ 31 x42+ 30 x43+ 32 x44+ 30 x45+ 31 x46+ 31 x47+ 30 x48
Subject To
C0: x0 + x6 + x7 + x8 + x9 + x10 + x21 + x22 + x23 + x24 + x25 + x26 + x27
     + x28 + x29 + x30 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 = 1
C1: x1 + x6 + x11 + x12 + x13 + x14 + x21 + x22 + x23 + x24 + x31 + x32 + x33
     + x34 + x35 + x36 + x39 + x40 + x41 + x42 + x43 + x44 + x47 + x48 = 1
C2: x2 + x7 + x11 + x15 + x16 + x17 + x21 + x25 + x26 + x27 + x31
     + x32 + x33 + x37 + x39 + x40 + x41 + x45 + x47 = 1
C3: x3 + x8 + x12 + x15 + x18 + x19 + x22 + x25 + x28 + x29 + x31
     + x34 + x35 + x38 + x39 + x42 + x43 + x46 + x48 = 1
C4: x4 + x9 + x13 + x16 + x18 + x20 + x23 + x26 + x28 + x30 + x32
     + x34 + x36 + x37 + x38 + x40 + x42 + x44 + x45 + x46 + x47 + x48 = 1
C5: x5 + x10 + x14 + x17 + x19 + x20 + x24 + x27 + x29 + x30 + x33
     + x35 + x36 + x37 + x38 + x41 + x43 + x44 + x45 + x46 + x47 + x48 = 1
Binaries
     x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20 x21 x22 x23 x24 x25
     x26 x27 x28 x29 x30 x31 x32 x33 x34 x35 x36 x37 x38 x39 x40 x41 x42 x43 x44 x45 x46 x47 x48
End
```

Figur 43: Ein ganzzahliges Programm.

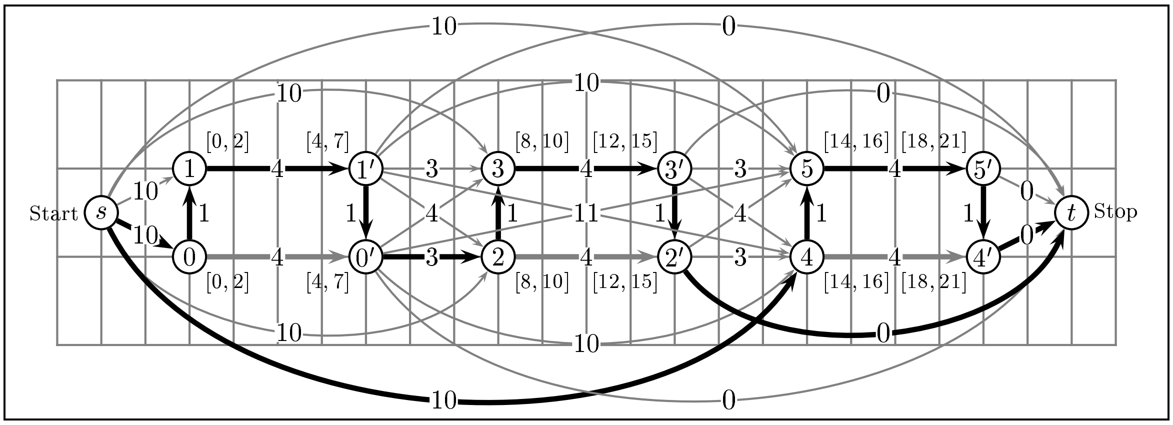
Die mathematische Kurzschreibweise für solche Modelle ist

$$\begin{array}{llllll} \min & \sum_{j=0}^{n-1} c_j x_j & & & \min & c^{\mathrm{T}} x \\ & \sum_{j=0}^{n-1} a_{ij} x_j = 1, & i = 0, \ldots, m-1 & \text{oder} & & Ax = 1 \\ & x_j \in \{0,1\}, & j = 0, \ldots, n-1, & & & x \in \{0,1\}^n, \end{array}$$

wobei $A = (a_{ij})_{\substack{i=0,\dots,m-1\\ j=0,\dots,n-1}}$ die $m \times n$ *Inzidenzmatrix* aus Tab. 2 ist (. wird ersetzt durch 0). Wo ist der Vorteil des Set Partitioning Modells? Wir sehen an zwei Stellen einen Gewinn.

Modelltechnik. Das Modell ist einfach und klar. Die Regeln für die Tourbildung sind in den Daten der Inzidenzmatrix gekapselt. Tourenerzeugung und Optimierung sind getrennt.

Abstraktion erlaubt die Anwendung allgemeiner und hochentwickelter Lösungsmethoden.



Figur 44: Lösung des Telebus-Beispiels.

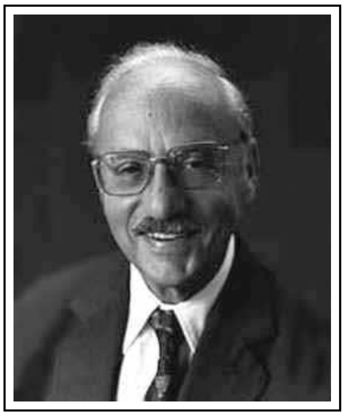
Probleme mit 49 Variablen und 6 Gleichungen löst jeder ernstzunehmende Code wie der Blitz. Unser Solver `SPL` brauchte keine Millisekunde, um die Lösung zu finden *und* ihre Optimalität zu beweisen. Das Ergebnis zeigt Fig. 44: $x_{20} = x_{39} = 1$, die anderen Variablen sind 0.

So leicht macht es uns der echte Telebus nicht. Bei 1.500–2.000 Fahrten gibt es vermutlich mindestens $\binom{2.000}{10} \geq 10^{20}$ Touren, die man nicht mehr enumerieren kann. War alles Set Partitioning umsonst? Das nicht! Wir müssen uns aber von der Vorstellung verabschieden, daß man ein Modell ganz hinschrieben muß, um es zu lösen. Nur einige wenige Touren der Optimallösung sind gefragt (im Beispiel nur 2), die unzähligen anderen (hier 47) sind am Ende nur Ballast. Wie sagte Euler noch: Würde die Untersuchung in der eben erwähnten Weise geführt, so würde Vieles gefunden, wonach gar nicht gefragt war; dies ist zweifellos der Grund, warum dieser Weg so beschwerlich wäre.

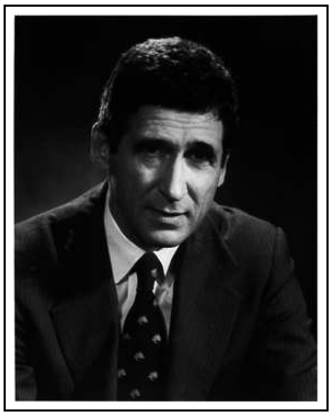
3.3 Pläne = Pfade + Preise + Programme

Vor enormen SPPs der Papierindustrie standen Anfang der sechziger Jahre auch die amerikanischen Mathematiker P.C. Gilmore und Ralph E. Gomory. Enumeration war "noch unmöglicher" als heute. Doch es gab eine neue Wunderwaffe: *Lineare Programmierung*! 25 Jahre zuvor von George B. Dantzig (Logistik der USAF im Pazifikkrieg) und Leonid Vitalyevich Kantorovich (1912-1986) (Produktionsplanung in der UdSSR) erfunden, hatten John von Neumanns (1903–1957) *Spieltheorie*, Wassily Leontiefs (1906–1999) *Input/Output Analyse*, Kantorovichs Produktionsplanung und Tjalling Charles Koopmans (1910-1985) *Transportoptimierung* (Leontief erhielt den Nobelpreis für Ökonomie 1973, Kantorovich und Koopmans 1975) mit dieser Technik die Ökonomie revolutioniert.

Auch ohne Komplexitätstheorie war bekannt, daß die "lineare Programmierung mit ganzzahligen Variablen", die man zur Behandlung *unteilbarer Güter* braucht, schwieriger ist. Doch man konnte Lineare Programmierung benutzen, um solche *Ganzzahligen Programme* zu lösen! Gomory hatte erst 1960 ein bahnbrechendes *Schnittebenenverfahren* vorgestellt, mit dem er *ohne Enumeration* ganzzahlig optimieren konnte (es stellte sich erst später, und zu Gomorys großer Enttäuschung heraus, daß sich seine Methode algorithmisch wie eine Enumeration verhält). Die Idee der Anwendung von Dantzigs *Simplexmethode* auf die Papier-SPPs lag in der Luft. In jeder Simplex-Iteration wird eine einzige Spalte betrachtet ... die man nicht vorher kennen muß, man kann sie bei Bedarf *erzeugen*! Die *Spaltenerzeugung* (column generation), der wichtigste Bestandteil aller *Branch & Price* Algorithmen, war erfunden.



Figur 45: G.B. Dantzig.



Figur 46: R.E. Gomory.

Figur 47: L.V. Kantorovich.

Wie Spaltenerzeugung Set Partitioning Probleme löst, davon wollen wir einen Eindruck geben. Was wir an Linearer Programmierung brauchen, ersetzen wir durch "Marktwirtschaft" ...

Initialisierung. Zum Start wird eine Lösung benötigt. Wir wählen die teure, aber immer mögliche "Taxidisposition" $x_0 = \cdots = x_5 = 1$ mit Kosten $6 \cdot 14 = 84$. Die 6 Inzidenzvektoren $a_{\cdot 0}$–$a_{\cdot 5}$ und die Kosten c_0–c_5 sind alles, was gegenwärtig vom SPP bekannt ist.

Iteration 1: BTRAN. Die *backward transformation* (der Name kommt vom Zurückspulen eines Bandes in den ersten Simplex-Codes) ist eine *Kostenstellenrechnung*: Die Gesamtkosten sollen von den Touren auf die "verursachenden" Fahrten umgerechnet werden. Das Ergebnis sind (Schatten-)*Preise* π_i für die Fahrten. Die Preise ergeben sich als Lösung der Gleichungen $\sum_{i=0}^{5} a_{ij}\pi_i = c_j$ für alle Touren j mit $x_j = 1$. In Beispiel sind das die Gleichungen

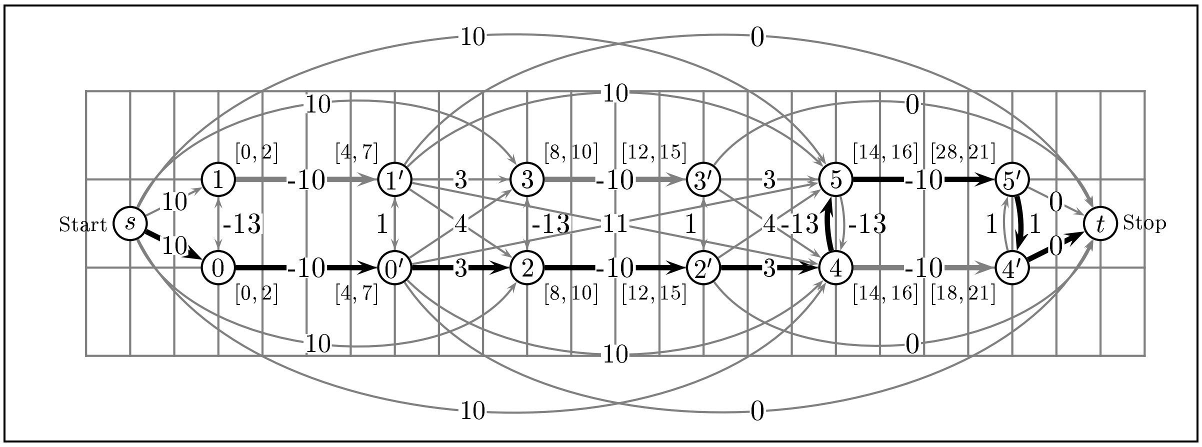
$$
\begin{array}{cccccclr}
\pi_0 & & & & & & = 14 = c_0 & \text{(Preisgleichung Tour 0)} \\
& \pi_1 & & & & & = 14 = c_1 & \text{(Preisgleichung Tour 1)} \\
& & \pi_2 & & & & = 14 = c_2 & \text{(Preisgleichung Tour 2)} \\
& & & \pi_3 & & & = 14 = c_3 & \text{(Preisgleichung Tour 3)} \\
& & & & \pi_4 & & = 14 = c_4 & \text{(Preisgleichung Tour 4)} \\
& & & & & \pi_5 & = 14 = c_5 & \text{(Preisgleichung Tour 5).}
\end{array}
$$

Die Lösung ist $\pi_0 = \cdots = \pi_5 = 14$.

Iteration 1: Pricing ist eine *Investitionsrechnung*: Es wird nach alternativen Touren gesucht, deren Kosten den Schattenpreis unterbieten. Die Differenz, die *reduzierten Kosten*, tut genau dies. Die Formel für die reduzierten Kosten $\overline{c}_j$ von Tour j lautet

$$\overline{c}_j := c_j - \sum_{i=0}^{5} a_{ij}\pi_i. \tag{1}$$

Die Bestimmung einer Tour mit negativen reduzierten Kosten ist — und damit kommen wir auf die Wege zurück — ein Kürzeste-Wege-Problem mit Ressourcenbeschränkungen. Fig. 48 zeigt den Digraphen. Damit die reduzierten Kosten Weglängen entsprechen, haben wir die Preise ähnlich wie in Fig. 20 auf die Bögen umgelegt, die aus den Startknoten der Fahrten herauszeigen. Die Preise gehen in Formel (1) mit negativem Vorzeichen ein und werden deshalb abgezogen. Dabei entstehen einige *negative Bogengewichte*. SSSP Algorithmen haben Schwierigkeiten mit negativen Gewichten, die die permanente Markierung von Knoten verhindern (man kann auf einem Weg Kosten *senken*, und tatsächlich sind SSSPs mit negativen Gewichten sogar $\mathcal{NP}$-schwer). In unserem Fall haben wir Glück: In einem *azyklischen Digraphen* (ohne gerichtete Kreise) funktioniert Dijkstras Algorithmus auch mit negativen Bogengewichten, wenn man die Knoten in der richtigen Reihenfolge (einer *topologischen Ordnung*) "von vorne nach hinten" markiert. Wir überspringen in dieser Iteration die Rechnung und raten Tour 45 (mit Inzidenzvektore $a_{\cdot 45} = (1,0,1,0,1,1)^{\mathrm{T}}$) mit negativen reduzierten Kosten von $\overline{c}_{45} = 10 - 10 + 3 - 10 + 3 - 13 - 10 + 1 + 0 = -26$, siehe Fig. 48.



Figur 48: Iteration 1.

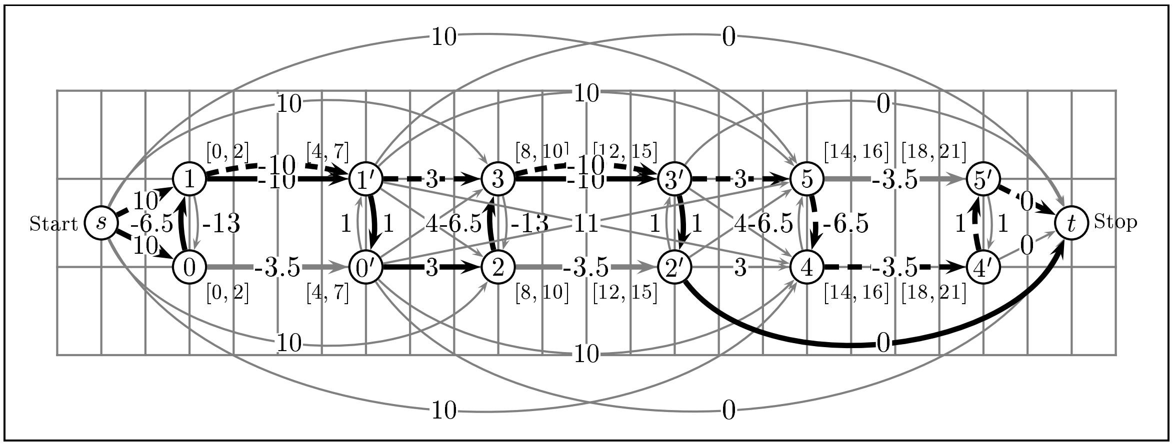
Iteration 1: FTRAN. Die *forward transformation* ist eine *Produktionsrechnung*: Die neue Tour 45 soll einige alte Touren ersetzen. Wie ist klar: Tour 45 ersetzt die Touren 0, 2, 4 und 5, die neue Lösung ist $x_1 = x_3 = x_{45} = 1$. Die Kosten betragen $c_1 + c_3 + c_{45} = 14 + 14 + 30 = 58$, eine Einsparung von $\overline{c}_{45} = -26$.

Iteration 2: BTRAN. Die Preisgleichungen

$$\begin{array}{lllllll} & \pi_1 & & & & = 14 = c_0 & \text{(Preisgleichung Tour 1)} \\ & & & \pi_3 & & = 14 = c_2 & \text{(Preisgleichung Tour 3)} \\ \pi_0 & & +\pi_2 & & +\pi_4 + \pi_5 & = 30 = c_{45} & \text{(Preisgleichung Tour 45)} \end{array}$$

haben mehr als eine Lösung. Wir wählen $\pi_1 = \pi_3 = 14$ und die "demokratische" Variante $\pi_0 = \pi_2 = \pi_4 = \pi_5 = 7.5$.

Iteration 2: Pricing. Die Preisumrechnung ergibt den Digraph in Fig. 49. Wir raten erneut und mit Tour 39 ($a_{\cdot 39} = (1,1,1,1,0,0)^{\mathrm{T}}$) und Tour 48 ($a_{\cdot 48} = (0,1,0,1,1,1)^{\mathrm{T}}$) gleich *zwei* Touren mit negativen reduzierten Kosten von $\overline{c}_{39} = -18$ und $\overline{c}_{48} = -13$. (Die Bestimmung von zwei Touren dient der Demonstration eines Phänomens im folgenden FTRAN-Schritt.)



Figur 49: Iteration 2.

Iteration 2: FTRAN. Die Verwendung der neuen Touren bereitet Schwierigkeiten! Tour 39, 45 und 48 schließen sich jeweils gegenseitig aus. Jede Entscheidung führt zu nichtüberdeckten Fahrten, für die neue Touren gefunden werden müssen, bei denen das Problem erneut auftritt — Münchhausen läßt grüßen. Ein mathematischer Ausweg ist

$$x_{39} = x_{45} = x_{48} = 0.5.$$

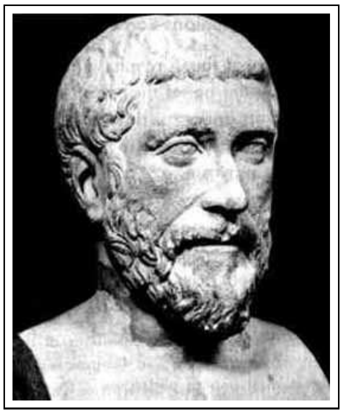
Ganzzahlig ist diese Variablenbelegung nicht. Abgesehen von dieser "Kleinigkeit" sind drei halbe Touren aber sehr vorteilhaft! Die Gleichungen `C0-C5` von Fig. 43 sind erfüllt und die Kosten betragen nur $0.5 \cdot 25 + 0.5 \cdot 30 + 0.5 \cdot 30 = 42.5$.

Was soll diese "Lösung" bedeuten? Die Antwort ist: Überhaupt nichts. *Gebrochene Lösungen* von ganzzahligen Programmen haben keine Bedeutung in der wirklichen Welt. Sie sind nur Zwischenergebnisse in einer Rechnung, an deren Ende eine ganzzahlige Lösung steht. Zwei Parallelen helfen, sich mit gebrochenen Lösungen ganzzahliger Programme anzufreunden.

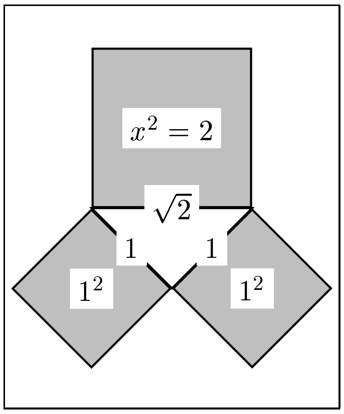
Irrationale Zahlen gab es bei den alten Griechen lange nicht. Pythagoras von Samos (ca. 580–ca. 520 v. Chr.) und seine Schüler hatten alle Zahlen zu Größenverhältnissen (Brüchen) von ganzen Zahlen erklärt. Hippasus von Metapont (ca. 500 v. Chr.), selbst Pythagoräer, war der erste, der Zweifel anmeldete. Pikanterweise lieferte gerade der berühmte *Satz von Pythagoras* für die Länge der Diagonale x im Einheitsquadrat die Beziehung $1^2 + 1^2 = x^2$ und Hippasus hatte entdeckt, daß die Lösung $x = \sqrt{2}$ kein Bruch war. Für diesen Frevel soll er von seinen pythagoräischen Kollegen bei seiner Seefahrt über Bord geworfen worden sein. Heute verstehen wir das "Wesen der Zahlen" nicht besser als die Griechen. Wir haben nur mit der Zahlenmystik aufgehört und uns an die Quadratwurzeln gewöhnt.

Imaginäre Zahlen hatten, wie der Name deutlich zeigt, dasselbe Akzeptanzproblem. Ihr bekanntester Vertreter i sollte die "unmögliche Gleichung" $x^2 = -1$ lösen. Allgemeine Anerkennung und eine Umbenennung in *komplexe Zahlen* erfuhren die imaginären Zahlen erst, als Johann Carl Friedrich Gauss (1777–1855) und andere sie mit einer Interpretation in der *Gaußschen Zahlenebene* geometrisch veranschaulichten.

Heute reicht es uns, mit $\sqrt{2}$ und i zu rechnen und am Ende das gewünschte Ergebnis zu erhalten. Genauso kann man es auch in der ganzzahligen Programmierung halten. Warum wir trotzdem über halbe Touren gesprochen haben? Dieser Versuchung ist schwer zu widerstehen!



Figur 50: Pythagoras.



Figur 51: Satz von Pythagoras.

Figur 52: J.C.F. Gauss.

Iteration 3: BTRAN. Die Preisgleichungen

$$\begin{array}{llllll} \pi_0 + \pi_1 + \pi_2 + \pi_3 & & = 25 = c_{39} & \text{(Preisgleichung Tour 39)} \\ \pi_0 + \pi_2 + \pi_4 + \pi_5 & & = 30 = c_{45} & \text{(Preisgleichung Tour 45)} \\ \pi_1 + \pi_3 + \pi_4 + \pi_5 & & = 30 = c_{48} & \text{(Preisgleichung Tour 48)} \end{array}$$

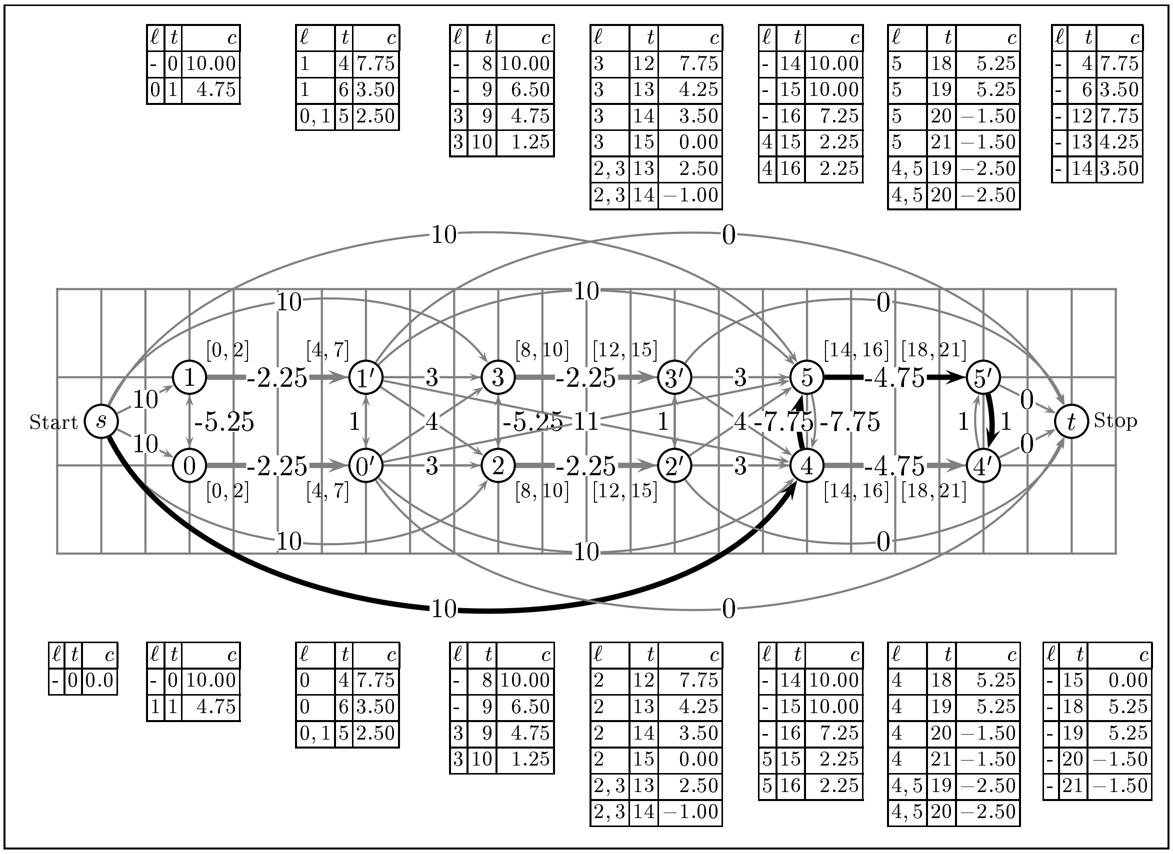
haben (z. B.) die Lösung $\pi_0 = \pi_1 = \pi_2 = \pi_3 = 6.25$ und $\pi_4 = \pi_5 = 8.75$.

Iteration 3: Pricing. Wir wollen uns nicht länger vor der Spaltenerzeugung drücken und berechnen in Fig. 53 die Tour mit der kürzesten Länge (= negativsten reduzierten Kosten) mit einem Multilabel-SSSP-Algorithmus. Neben einer Distanzmarke c benötigen wir an den Knoten Marken für zwei weitere Ressourcen: Eine Liste ℓ der zur Zeit der Ankunft im Fahrzeug befindlichen Personen (die noch ans Ziel gebracht werden müssen) und die Ankunftszeit t (Zeitfenster). ℓ ist eine diskrete Ressource mit endlich vielen (um genau zu sein 2^6) Zuständen. t ist eigentlich keine diskrete Ressource, kann aber wegen der Ganzzahligkeit der Fahrzeiten als $0, 1, 2, \ldots, 23$ ebenfalls diskret angenommen werden. An jedem Knoten ergeben sich auf diese Weise $2^6 \times 24$ mögliche Ressourcenzustände. Sinnvoll sind nur die in Fig. 53 angegebenen (Preprocessing). Die Berechnung der Wege erfolgt in der Reihenfolge wachsender Zeitmarken. Das Ergebnis sind die bereits eingetragenen Distanzmarken (bei t sind Minima durch Warten *nicht* eingetragen). Die Distanzmarke $c = -1.5$ am Depotknoten t zeigt, daß es eine Tour mit negativen reduzierten Kosten gibt. Es stehen vier symmetrische/zeitverschobene Wege zur Auswahl, die alle Repräsentanten von Tour 20 ($a_{\cdot 20} = (0, 0, 0, 0, 1, 1)^{\mathrm{T}}$) sind.

Iteration 3: FTRAN. Für den Austausch der Touren 39, 45 und 48 gegen Tour 20 kann man ein Gleichungssystem ähnlich wie die Preisgleichungen aufstellen. Es lautet

$$\begin{array}{l} a_{20,0} = 0 = y_{39} + y_{45} \\ a_{20,1} = 0 = y_{39} \phantom{+ y_{45}} + y_{48} \\ a_{20,2} = 0 = y_{39} + y_{45} \\ a_{20,3} = 0 = y_{39} \phantom{+ y_{45}} + y_{48} \\ a_{20,4} = 1 = \phantom{y_{39}} + y_{45} + y_{48} \\ a_{20,5} = 1 = \phantom{y_{39}} + y_{45} + y_{48}. \end{array}$$

Die Lösung $y_{39} = -0.5$, $y_{45} = y_{48} = 0.5$ zeigt, daß Tour 20 plus 0.5 Einheiten von Tour 39 je 0.5 Einheiten von Tour 45 und 48 ersetzen. Die neue Lösung $x_{20} = x_{39} = 1$ ist ganzzahlig.



Figur 53: Iteration 3.

Iteration 4: BTRAN. Die Preisgleichungen

$$
\begin{array}{llll}
\pi_0 + \pi_1 + \pi_2 + \pi_3 & & = 25 = c_{39} & \text{(Preisgleichung Tour 39)} \\
& + \pi_4 + \pi_5 & = 16 = c_{20} & \text{(Preisgleichung Tour 20)}
\end{array}
$$

liefern $\pi_0 = \pi_1 + \pi_2 = \pi_3 = 6.25$, $\pi_4 = \pi_5 = 8$.

Iteration 4: Pricing. In dem Digraphen in Fig. 53 gibt es keine Wege mit negativer Länge. In der Theorie der Linearen Optimierung wird gezeigt: Wenn die reduzierten Kosten aller Touren nichtnegativ sind, dann ist die gegenwärtige Lösung optimal unter allen gebrochenen und erst recht unter allen ganzzahligen Lösungen! Das Eintreten dieses *Abbruchkriteriums* ist das Signal zur Einstellung der Rechnung. Die Optimallösung ist gefunden!

*

Nur $6 + 1 + 2 + 1 = 10$ der 49 möglichen Inzidenzvektoren hat die Spaltenerzeugung verwendet, um zur Optimallösung zu kommen (die allerdings gebrochene Werte haben hätte können). Dieses Ergebnis ist kein Zufall. Der zweite Autor hat zusammen mit dem holländischen Mathematiker Alexander Schrijver und dem ungarischen Mathematiker László Lovász eine Theorie der *polynomialen Äquivalenz von Separierung und Optimierung* (hier dual angewandt) entwickelt, die zeigt, daß sich dieses günstige Verhalten immer einstellt. Diese Resultate sind die theoretische Untermauerung für die empirische Effizienz der Spaltengenerierung.

ℓ	t	c
-	0	10.00
0	1	4.75

ℓ	t	c
1	4	7.75
1	6	3.50
0, 1	5	2.50

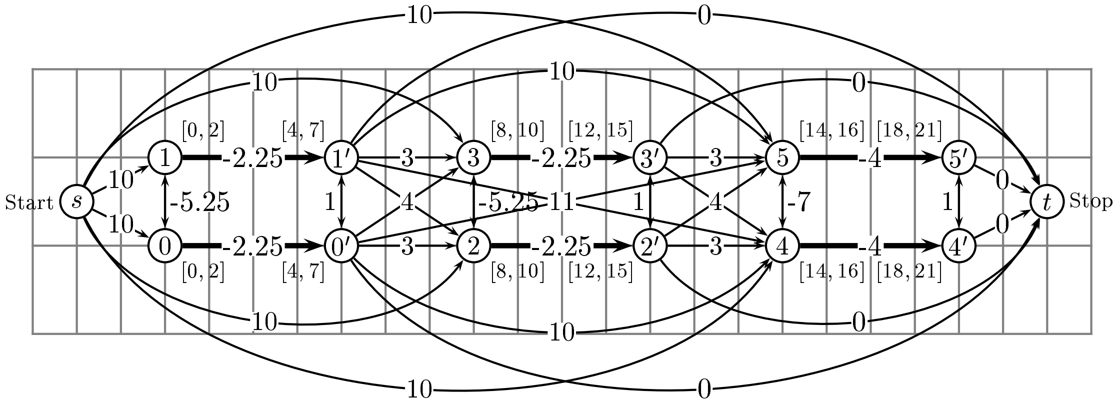
ℓ	t	c
-	8	10.00
-	9	6.50
3	9	4.75
3	10	1.25

ℓ	t	c
3	12	7.75
3	13	4.25
3	14	3.50
3	15	0.00
2, 3	13	2.50
2, 3	14	−1.00

ℓ	t	c
-	14	10.00
-	15	10.00
-	16	7.25
4	15	3.00
4	16	3.00

ℓ	t	c
5	18	6.00
5	19	6.00
5	20	0.00
5	21	0.00
4, 5	19	−1.00
4, 5	20	−1.00

ℓ	t	c
-	4	7.75
-	6	3.50
-	12	7.75
-	13	4.25
-	14	3.50



ℓ	t	c
-	0	0.0

ℓ	t	c
-	0	10.00
1	1	4.75

ℓ	t	c
0	4	7.75
0	6	3.50
0, 1	5	2.50

ℓ	t	c
-	8	10.00
-	9	6.50
3	9	4.75
3	10	1.25

ℓ	t	c
2	12	7.75
2	13	4.25
2	14	3.50
2	15	0.00
2, 3	13	2.50
2, 3	14	−1.00

ℓ	t	c
-	14	10.00
-	15	10.00
-	16	7.25
5	15	3.00
5	16	3.00

ℓ	t	c
4	18	6.00
4	19	6.00
4	20	0.00
4	21	0.00
4, 5	19	−1.00
4, 5	20	−1.00

ℓ	t	c
-	15	0.00
-	18	6.00
-	19	6.00
-	20	0.00
-	21	0.00

Figur 54: Iteration 4.

Leider läßt sich diese Aussage nicht einfach auf Bestimmung ganzzahliger Lösungen durch die *Branch & Price* Verfahren übertragen. Das Problem ist, daß die Spaltenerzeugung (Price) mit einer gebrochenen Optimallösung terminieren kann. Die Ganzzahligkeit kann nur mit einer nachgeschalteten Enumeration (Branch) garantiert werden, die i. a. zu einer exponentiellen Gesamtlaufzeit führt.

Im Beispiel hatten wir Glück und die Spaltenerzeugung lieferte eine ganzzahlig Lösung. Dieses Ergebnis ist nicht die Regel, aber häufig kann man mit einfachen *Heuristiken* aus gebrochenen Lösungen ganzzahlige machen, deren Zielfunktionswerte nahe aneinanderliegen (im Idealfall stimmen sie überein). Dazwischen liegt die *Dualitätslücke* (duality gap), in der sich das unbekannte Optimum befindet. Die Größe der Lücke liefert auch bei *Unkenntnis des genauen Optimums* ein Maß für die *Qualität* einer heuristischen Lösung. Aussagen dieser Art sind wichtig zur Beurteilung von *Einsparpotentialen.*

*

Bei Telebus hat ein Set Partitioning System am 3. Juni 1995 die Disposition übernommen. Hauptproblem war damals ein Anstieg im Fahrtaufkommen von 30% im Nachgang zur Wiedervereinigung. Er konnte kostenneutral aufgefangen werden. Gleichzeitig hat sich die Servicequalität verbessert (Pünktlichkeit, Fahrzeiten). Die Planung ist großteils automatisiert. Mathematik konnte so dazu beitragen, daß der Telebus attraktiv und finanzierbar bleibt.

***

4 Ausblick

Seit sechzig Jahren erlebt die Diskrete Optimierung und mit ihr die Mathematik der Wege einen Aufschwung. Die Entwicklung des Computers und der Linearen Programmierung waren die Triebkräfte und sind es noch immer. Heute stehen gute Modelle, Algorithmen und eine allgemeine Theorie zur Verfügung.

Ihr Einsatz entspricht in der Regel leider nicht dem Stand der Mathematik. Im Vergleich zur Durchdringung der technischen Ingenieursdisziplinen mit Methoden der Differentialrechnung sind diskrete Methoden in Planung, Logistik, Decision Support und Supply Chain Optimization noch Exoten. Es gibt eine Reihe von Gründen: Die Größe diskreter Modelle, mangelnde Kenntnis der hierzu notwendigen Mathematik oder sogar Skepsis gegenüber der Mathematik in manchen Planungskreisen, fehlendes Interesse an Optimierung durch monopolistische Strukturen in wichtigen Feldern wie dem öffentlichen Verkehr.

Diese Lage ändert sich. Die Leistung von Rechnern und Verfahren ist auf einem nutzbaren Stand und steigt. Monopole lösen sich auf. Es besteht die Chance, diskrete Methoden in die Planung und die Logistik zu bringen. Unsere Vision ist, daß *Computer Aided Scheduling* (CAS) denselben Stellenwert in der Logistik bekommt wie CAD und CAM in der Produktion. Die Möglichkeiten liegen auf der Hand: Kosteneinsparung, Qualitätsverbesserung, Planungstempo und -flexibilität, Szenarioanalysen, etc.

Erste Erfolge geben Anlaß zum Optimismus. Beim Telebus haben sich Set Partitioning Modelle bewährt. Im kompetitiven Luftverkehr sind sie bereits Industriestandard. Mit Mehrgüterflußmethoden ist die Umlaufplanung sämtlicher Fahrzeuge von Verkehrsbetrieben wie der BVG (dem viertgrößten der Welt) möglich. Nicht wegzudenken sind Verfahren zur Lösung von Wegproblemen aus dem Netzdesign in der Telekommunikation. Wo noch? Das wird die Zukunft zeigen!

5 Weiterführende Literatur

Sie sind auf den Geschmack gekommen und möchten mehr über die Mathematik der Wege wissen? In diesem Abschnitt finden Sie eine thematisch sortierte Liste mit weiterführender Literatur, die Sie zum Teil aus dem Internet herunterladen können.

Transportprobleme. Unser Artikel *Alcuin's Transportation Problems and Integer Programming* in Band 2 des Buches *Charlemagne and his Heritage: 1200 Years of Civilization and Science in Europe* von Paul Leo Butzer, Hubertus Th. Jongen und Walter Oberschelp, erschienen 1998 bei Brepols, Turnhout, Belgien, gibt eine unterhaltsame Einführung in die mathematische Transportoptimierung am Beispiel des bekannten Wolf-Ziege-Kohl Problems, das sich Karl der Große und sein Chefberater Alkuin von York vor 1200 Jahren ausdachten, um an den fränkischen Schulen den Mathematikunterricht zu verbessern. Der Artikel ist im Internet von `http://www.zib.de/ZIBbib/Publications/` als ZIB Preprint SC 95-27 erhältlich.

Das Königsberger Brückenproblem. Lassen Sie es sich nicht entgehen, Leonhard Eulers gleichnamigen Artikel aus dem Jahre 1736 zu lesen! Sie finden ihn auf den Seiten 290–301 in Dénes Königs Buch *Theorie der Endlichen und Unendlichen Graphen* aus dem Jahre 1936, das 1986 als Band 6 des *Teubner Archivs zur Mathematik* bei B.G. Teubner, Leipzig, wiederaufgelegt wurde.

TSP. Martin Grötschels und Manfred Padbergs Artikel *Die Optimierte Odysse*, erschienen 1999 im *Spektrum der Wissenschaft 4*, Seiten 76–85, und erhältlich im Internet von `http://www.spektrum.de/themen/heft210499.html`, informiert auf unterhaltsame Weise über das berühmteste Problem der kombinatorischen Optimierung.

Telebus. Mehr zur Optimierung des Berliner Telebusses erfahren Sie in den Artikeln *Berliner Telebus bietet Mobilität für Behinderte*, erschienen 1997 in der Zeitschrift *Der Nahverkehr 1–2/97*, und *Optimierung des Berliner Behindertenfahrdienstes* aus den *DMV Mitteilungen 2* des Jahres 1997, beide von Ralf Borndörfer, Martin Grötschel, Fidolin Klostermeier & Christian Küttner, und beide erhältlich von `http://www.zib.de/ZIBbib/Publications/` als ZIB Preprints SC 96-40 bzw. SC 97-10.

Nahverkehr. Die Wagenumlaufplanung im öffentlichen Nahverkehr beschreiben Martin Grötschel, Andreas Löbel & Manfred Völker in dem Artikel *Optimierung des Fahrzeugumlaufs im Öffentlichen Nahverkehr* in dem Buch *Mathematik – Schlüsseltechnologie für die Zukunft*, das von K.-H. Hoffmann, W. Jäger, T. Lohmann und H. Schunck herausgegeben wurde und 1997 im Springer-Verlag erschienen ist.

Die Dienstplanung im öffentlichen Nahverkehr beschreiben Ralf Borndörfer, Andreas Löbel, Uwe Strubbe und Manfred Völker in dem Artikel *Zielorientierte Dienstplanoptimierung*, erschienen 1999 in dem Proceedingsband *Heureka '99: Optimierung in Verkehr und Transport* der Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln.

Die beiden Artikel sind im Internet von `http://www.zib.de/ZIBbib/Publications/` als ZIB Preprints SC 96-08 und SC 98-41 erhältlich.

Algorithmische Graphentheorie. Dieter Jungnickels (1987) Buch *Graphen, Netzwerke und Algorithmen*, erschienen im B.I.–Wissenschaftsverlag, und das schön illustrierte Skript *Graphentheoretische Konzepte und Algorithmen* von Sven Oliver Krumke, Hartmut Noltemeier und Hans-Christoph Wirth, erhältlich von `http://www.zib.de/ZIBbib/Publications/` als ZIB Preprint 00-19, geben fundierte Einführungen.

Lineare Programmierung. Wir empfehlen das spannende Buch *Linear Programming* von Vasěk Chvátal (1980), erschienen bei W.H. Freeman and Company, New York.

Komplexität, Laufzeit, Rechner. Ein Standartwerk ist Kurt Mehlhorns Buch *Datenstrukturen und Effiziente Algorithmen*, Band 1, erschienen 1988 bei B.G. Teubner, Stuttgart.

6 Auflösungen der Fragen

Eulertour. Vier Knoten haben ungeraden Grad. Nach Theorem 1.2 gibt es keine Eulertour.

Ikosaederspiel. Zwei Original-Hamilton-Kreise: B C P N M D F K L T S R Q Z X W V J H G und B C P N M D F G H X W V J K L T S R Q Z.

Bienenwabe. Es gibt keinen Hamiltonkreis. Die Bienenwabe ist *bipartit* (zweifärbbar): Alle Kanten führen von Knoten mit gerade Nummer (g) zu ungeraden (u) Knoten. Ein Hamiltonkreis hätte die Form g u g u ... g u und also eine gerade Anzahl Knoten. Die Bienenwabe hat aber 13 Knoten.

HCP als SSSP, Seite 17. Im Originalgraph kann es zu Knotenwiederholungen kommen.

* Ein herzliches Dankeschön an Regine Kossick für Ihre Hilfe bei den Literaturrecherchen! *