Algorithmische Graphentheorie

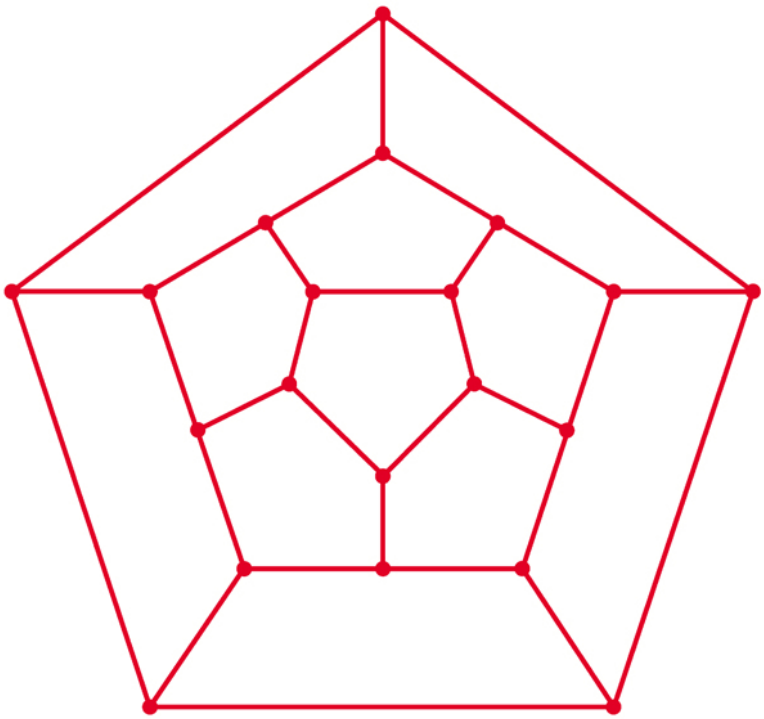
Bearbeitet von
Volker Turau

überarbeitet 1996. Taschenbuch. XIII, 458 S. Paperback
ISBN 978 3 486 59057 9
Format (B x L): 17 x 24 cm
Gewicht: 842 g

Volker Turau

Algorithmische Graphentheorie

3. Auflage



Oldenbourg

Algorithmische Graphentheorie

von
Prof. Dr. Volker Turau

3., überarbeitete Auflage

Oldenbourg Verlag München

Prof. Dr. Volker Turau hat seit 2002 eine Professur für Verteilte Systeme an der Technischen Universität Hamburg-Harburg und leitet seit 2008 das Institut für Telematik. Seine aktuellen Forschungsgebiete sind verteilte und selbststabilisierende Algorithmen, drahtlose Sensornetze und energiebewusste Systeme.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
oldenbourg.de

Lektorat: Dr. Margit Roth
Herstellung: Anna Grosser
Coverentwurf: Kochan & Partner, München
Gedruckt auf säure- und chlorfreiem Papier
Gesamtherstellung: Grafik + Druck, München

ISBN 978-3-486-59057-9

Vorwort

Für die dritte Auflage des Buches wurde eine kompaktere Darstellung der Algorithmen gewählt. Hinzugefügt wurden unter anderem ein Abschnitt zur Implementierung von Graphalgorithmen in objektorientierten Programmiersprachen und Algorithmen zur Analyse sozialer Netzwerke. Wie schon bei der vorangegangenen Auflage möchte ich den Lesern für Ihre Anregungen danken. Mein besonderer Dank gilt den Leserinnen Prof. Dr. Sigrid Knust und Jutta Müller.

Vorwort zur 2. Auflage

Die erfreuliche Nachfrage nach dem Buch war Anlaß, eine in vielen Teilen erweiterte und überarbeitete Neuauflage zu verfassen. In fast allen Kapiteln kamen neue Beispiele hinzu oder es wurden vorhandene Beispiele erweitert bzw. verbessert. Viele dieser Änderungen gehen auf Anregungen von Lesern zurück, denen an dieser Stelle herzlich gedankt sei. Auf vielfachen Wunsch sind im Anhang für alle Aufgaben Lösungen angegeben. Mein besonderer Dank gilt folgenden Lesern: Elke Hocke, Thomas Emden-Weinert, Holger Dörnemann, Philipp Flach, Bernhard Schiefer, Ralf Kastner, Andreas Görz und Felix Yu. Dem Oldenbourg Verlag möchte ich für die angenehme Zusammenarbeit danken.

Hamburg

Volker Turau

Vorwort zur 1. Auflage

Graphen sind die in der Informatik am häufigsten verwendete Abstraktion. Jedes System, welches aus diskreten Zuständen oder Objekten und Beziehungen zwischen diesen besteht, kann als Graph modelliert werden. Viele Anwendungen erfordern effiziente Algorithmen zur Verarbeitung von Graphen. Dieses Lehrbuch ist eine Einführung in die algorithmische Graphentheorie. Die Algorithmen sind in kompakter Form in einer programmiersprachennahen Notation dargestellt. Eine Übertragung in eine konkrete Programmiersprache wie C++ oder Pascal ist ohne Probleme durchzuführen. Die mei-

sten der behandelten Algorithmen sind in der dargestellten Form im Rahmen meiner Lehrveranstaltungen implementiert und getestet worden. Die praktische Relevanz der vorgestellten Algorithmen wird in vielen Anwendungen aus Gebieten wie Compilerbau, Betriebssysteme, künstliche Intelligenz, Computernetzwerke und Operations Research demonstriert.

Dieses Buch ist an allejene gerichtet, die sich mit Problemen der algorithmischen Graphentheorie beschäftigen. Es richtet sich insbesondere an Studenten der Informatik und Mathematik im Grund- als auch im Hauptstudium.

Die neun Kapitel decken die wichtigsten Teilgebiete der algorithmischen Graphentheorie ab, ohne einen Anspruch auf Vollständigkeit zu erheben. Die Auswahl der Algorithmen erfolgte nach den folgenden beiden Gesichtspunkten: Zum einen sind nur solche Algorithmen berücksichtigt, die sich einfach und klar darstellen lassen und ohne großen Aufwand zu implementieren sind. Der zweite Aspekt betrifft die Bedeutung für die algorithmische Graphentheorie an sich. Bevorzugt wurden solche Algorithmen, welche entweder Grundlagen für viele andere Verfahren sind oder zentrale Probleme der Graphentheorie lösen.

Unter den Algorithmen, welche diese Kriterien erfüllten, wurden die effizientesten hinsichtlich Speicherplatz und Laufzeit dargestellt. Letztlich war die Auswahl natürlich oft eine persönliche Entscheidung. Aus den genannten Gründen wurde auf die Darstellung von Algorithmen mit kompliziertem Aufbau oder auf solche, die sich auf komplexe Datenstrukturen stützen, verzichtet. Es werden nur sequentielle Algorithmen behandelt. Eine Berücksichtigung von parallelen oder verteilten Graphalgorithmen würde den Umfang dieses Lehrbuchs sprengen.

Das erste Kapitel gibt anhand von mehreren praktischen Anwendungen eine Motivation für die Notwendigkeit von effizienten Graphalgorithmen. Das zweite Kapitel führt in die Grundbegriffe der Graphentheorie ein. Als Einstieg in die algorithmische Graphentheorie wird der Algorithmus zur Bestimmung des transitiven Abschlusses eines Graphen diskutiert und analysiert. Das zweite Kapitel stellt außerdem Mittel zur Verfügung, um Zeit- und Platzbedarf von Algorithmen abzuschätzen und zu vergleichen.

Kapitel 3 beschreibt Anwendungen von Bäumen und ihre effiziente Darstellung. Dabei werden mehrere auf Bäumen basierende Algorithmen präsentiert. Kapitel 4 behandelt Suchstrategien für Graphen. Ausführlich werden Tiefen- und Breitensuche, sowie verschiedene Realisierungen dieser Techniken diskutiert. Zahlreiche Anwendungen auf gerichtete und ungerichtete Graphen zeigen die Bedeutung dieser beiden Verfahren.

Kapitel 5 diskutiert Algorithmen zur Bestimmung von minimalen Färbungen. Für allgemeine Graphen wird mit dem Backtracking-Algorithmus ein Verfahren vorgestellt, welches sich auch auf viele andere Probleme anwenden läßt. Für planare und transitiv orientierbare Graphen werden effiziente Algorithmen zur Bestimmung von minimalen Färbungen dargestellt.

Die beiden Kapitel 6 und 7 behandeln Flüsse in Netzwerken. Zunächst werden zwei Algorithmen zur Bestimmung von maximalen Flüssen vorgestellt. Der erste basiert auf Erweiterungswegen minimaler Länge, der zweite verwendet die Technik der blockierenden Flüsse. Im Mittelpunkt von Kapitel 7 stehen Anwendungen dieser Verfahren: Bestimmung von maximalen Zuordnungen in bipartiten Graphen, Bestimmung der Kanten-

und Eckenzusammenhangszahl eines ungerichteten Graphen und Bestimmung von minimalen Schnitten.

Kapitel 8 betrachtet verschiedene Varianten des Problems der kürzesten Wege in kantenbewerteten Graphen. Es werden auch Algorithmen zur Bestimmung von kürzesten Wegen diskutiert, wie sie in der künstlichen Intelligenz Anwendung finden.

Kapitel 9 gibt eine Einführung in approximative Algorithmen. Unter der Voraussetzung $\mathcal{P} \neq \mathcal{NP}$ wird gezeigt, daß die meisten $\mathcal{NP}$-vollständigen Probleme keine approximativen Algorithmen mit beschränktem absoluten Fehler besitzen und daß sich die Probleme aus $\mathcal{NPC}$ bezüglich der Approximierbarkeit mit beschränktem relativen Fehler sehr unterschiedlich verhalten. Breiten Raum nehmen approximative Algorithmen für das Färbungsproblem und das Traveling-Salesman Problem ein. Schließlich werden Abschätzungen für den Wirkungsgrad dieser Algorithmen untersucht.

Ein unerfahrener Leser sollte zumindest die ersten vier Kapitel sequentiell lesen. Die restlichen Kapitel sind relativ unabhängig voneinander (mit Ausnahme von Kapitel 7, welches auf Kapitel 6 aufbaut). Das Kapitel über approximative Algorithmen enthält viele neuere Forschungsergebnisse und ist aus diesem Grund das umfangreichste. Damit wird auch der Hauptrichtung der aktuellen Forschung der algorithmischen Graphentheorie Rechnung getragen.

Jedes Kapitel hat am Ende einen Abschnitt mit Übungsaufgaben; insgesamt sind es etwa 250. Der Schwierigkeitsgrad ist dabei sehr unterschiedlich. Einige Aufgaben dienen nur zur Überprüfung des Verständnisses der Verfahren. Mit $*$ bzw. $**$ gekennzeichnete Aufgaben erfordern eine intensivere Beschäftigung mit der Aufgabenstellung und sind für fortgeschrittene Studenten geeignet.

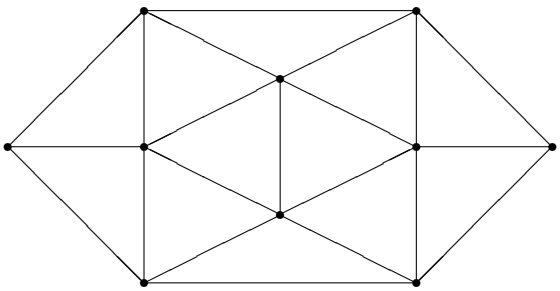
Mein Dank gilt allen, die mich bei der Erstellung dieses Buches unterstützt haben. Großen Anteil an der Umsetzung des Manuskriptes in LaTeX hatten Thomas Erik Schmidt und Tim Simon. Einen wertvollen Beitrag leisteten auch die Studentinnen und Studenten mit ihren kritischen Anmerkungen zu dem Stoff in meinen Vorlesungen und Seminaren. Ein besonderer Dank gilt meiner Schwester Christa Teusch für die kritische Durchsicht des Manuskriptes. Den beiden Referenten Professor Dr. A. Beutelspacher und Professor Dr. P. Widmayer danke ich für ihre wertvollen Verbesserungsvorschläge zu diesem Buch.

Wiesbaden, im Februar 1996
Volker Turau

Inhaltsverzeichnis

Kapitel 1

Einleitung



In vielen praktischen und theoretischen Anwendungen treten Situationen auf, die durch ein System von Objekten und Beziehungen zwischen diesen Objekten charakterisiert werden können. Die Graphentheorie stellt zur Beschreibung von solchen Systemen ein Modell zur Verfügung: den Graphen. Die problemunabhängige Beschreibung mittels eines Graphen läßt die Gemeinsamkeit von Problemen aus den verschiedensten Anwendungsgebieten erkennen. Die Graphentheorie ermöglicht somit die Lösung vieler Aufgaben, welche aus dem Blickwinkel der Anwendung keine Gemeinsamkeiten haben. Die algorithmische Graphentheorie stellt zu diesem Zweck Verfahren zur Verfügung, die problemunabhängig formuliert werden können. Ferner erlauben Graphen eine anschauliche Darstellung, welche die Lösung von Problemen häufig erleichtert.

Im folgenden werden sechs verschiedene Anwendungen diskutiert. Eine genaue Betrachtung der Aufgabenstellungen führt ganz natürlich auf eine graphische Beschreibung und damit auch auf den Begriff des Graphen; eine Definition wird bewußt erst im nächsten Kapitel vorgenommen. Die Beispiele dienen als Motivation für die Definition eines Graphen; sie sollen ferner einen Eindruck von der Vielfalt der zu lösenden Aufgaben geben und die Notwendigkeit von effizienten Algorithmen vor Augen führen.

1.1 Verletzlichkeit von Kommunikationsnetzen

Ein Kommunikationsnetz ist ein durch Datenübertragungswege realisierter Verband mehrerer Rechner. Es unterstützt den Informationsaustausch zwischen Benutzern an verschiedenen Orten. Die *Verletzlichkeit* eines Kommunikationsnetzes ist durch die Anzahl von Leitungen oder Rechnern gekennzeichnet, die ausfallen müssen, damit die Verbindung zwischen zwei beliebigen Benutzern nicht mehr möglich ist. Häufig ist eine Verbindung zwischen zwei Benutzern über mehrere Wege möglich. Somit ist beim Ausfall einer Leitung oder einer Station die Verbindung nicht notwendigerweise unterbrochen. Ein Netzwerk, bei dem schon der Ausfall einer einzigen Leitung oder Station gewisse Verbindungen unmöglich macht ist verletzlicher, als ein solches, wo dies nur beim Ausfall von mehreren Leitungen oder Stationen möglich ist.

Die minimale Anzahl von Leitungen und Stationen, deren Ausfall die Funktion des Netzwerkes beeinträchtigt, hängt sehr stark von der Beschaffenheit des Netzwerkes ab. Netzwerke lassen sich graphisch durch Knoten und Verbindungslinien zwischen den Knoten darstellen: Die Knoten entsprechen den Stationen, die Verbindungslinien den Datenübertragungswegen. In Abbildung 1.1 sind vier Grundformen gebräuchlicher Netzwerke dargestellt.

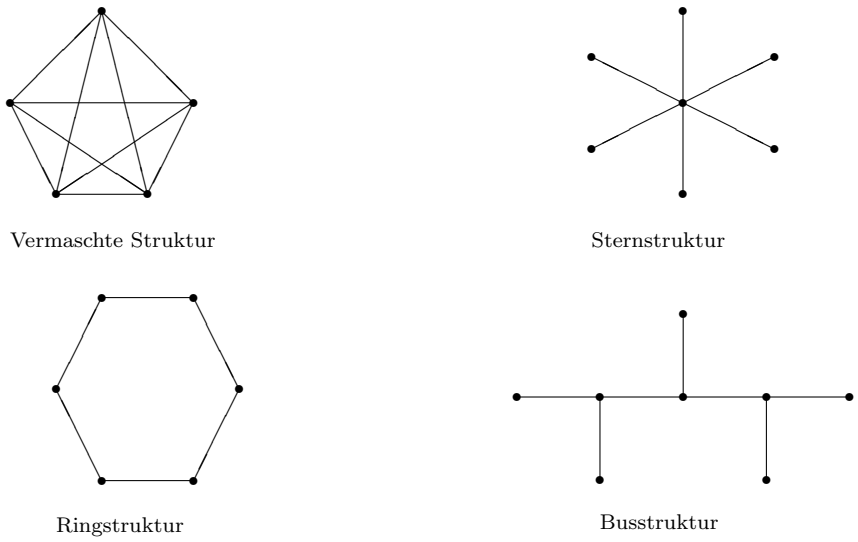


Abbildung 1.1: *Netzwerktopologien*

Bei der vermaschten Struktur ist beim Ausfall einer Station die Kommunikation zwischen den restlichen Stationen weiter möglich. Sogar der Ausfall von bis zu sechs Datenübertragungswegen muß noch nicht zur Unterbrechung führen. Um die Kommunikation mit einem Benutzer zu unterbrechen, müssen mindestens vier Datenübertragungswege ausfallen. Bei der Sternstruktur ist nach dem Ausfall der zentralen Station keine Kommunikation mehr möglich. Hingegen führt in diesem Fall der Ausfall einer Leitung nur zur Abkopplung eines Benutzers.

Eine Menge von Datenübertragungswegen in einem Kommunikationsnetzwerk heißt *Schnitt*, falls ihr Ausfall die Kommunikation zwischen irgendwelchen Stationen unterbricht. Ein Schnitt mit der Eigenschaft, daß keine echte Teilmenge ebenfalls ein Schnitt ist, nennt man *minimaler Schnitt*. Die Anzahl der Verbindungslinien in dem minimalen Schnitt mit den wenigsten Verbindungslinien nennt man *Verbindungszusammenhang* oder auch die *Kohäsion* des Netzwerkes. Sie charakterisiert die Verletzlichkeit eines Netzwerkes. Die Kohäsion von Bus- und Sternstruktur ist gleich 1, die der Ringstruktur gleich 2, und die vermaschte Struktur hat die Kohäsion 4.

Analog kann man auch den Begriff *Knotenzusammenhang* eines Netzwerkes definieren. Er gibt die minimale Anzahl von Stationen an, deren Ausfall die Kommunikation der restlichen Stationen untereinander unterbrechen würde. Bei der Bus- und Sternstruktur ist diese Zahl gleich 1 und bei der Ringstruktur gleich 2. Fällt bei der vermaschten Struktur eine Station aus, so bilden die verbleibenden Stationen immer noch eine vermaschte Struktur. Somit ist bei dieser Struktur beim Ausfall von beliebig vielen Stationen die Kommunikation der restlichen Stationen gesichert.

Verfahren zur Bestimmung von Verbindungszusammenhang und Knotenzusammenhang eines Netzwerkes werden in Kapitel 7 behandelt.

1.2 Wegplanung für Roboter

Ein grundlegendes Problem auf dem Gebiet der Robotik ist die Planung von kollisionsfreien Wegen für Roboter in ihrem Einsatzgebiet. Von besonderem Interesse sind dabei die kürzesten Wege, auf denen der Roboter mit keinem Hindernis in Kontakt kommt. Zum Auffinden dieser Wege muß eine Beschreibung der Geometrie des Roboters und des Einsatzgebietes vorliegen. Ohne Einschränkungen der Freiheitsgrade des Roboters und der Komplexität des Einsatzgebietes ist dieses Problem praktisch nicht lösbar.

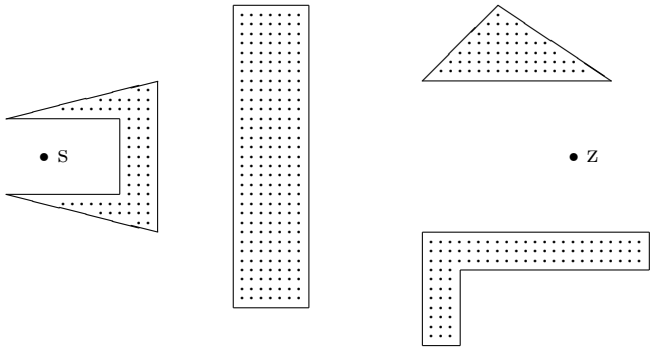


Abbildung 1.2: *Hindernisse für einen Roboter*

Eine stark idealisierte Version dieses komplizierten Problems erhält man, indem man die Bewegung auf eine Ebene beschränkt und den Roboter auf einen Punkt reduziert. Diese Ausgangslage kann auch in vielen Anwendungen durch eine geeignete Transformation

geschaffen werden. Das Problem stellt sich nun folgendermaßen dar: Für eine Menge von Polygonen in der Ebene, einen Startpunkt s und einen Zielpunkt z ist der kürzeste Weg von s nach z gesucht, welcher die Polygone nicht schneidet; Abbildung 1.2 zeigt eine solche Situation.

Zunächst wird der kürzeste Weg analysiert, um dann später ein Verfahren zu seiner Bestimmung zu entwickeln. Dazu stellt man sich den kürzesten Weg durch ein straff gespanntes Seil vor. Es ergibt sich sofort, daß der Weg eine Folge von Geradensegmenten der folgenden Art ist:

a) Geradensegmente von s oder z zu einer konvexen Ecke eines Hindernisses, welche keine anderen Hindernisse schneiden

b) Kanten von Hindernissen zwischen konvexen Ecken

c) Geradensegmente zwischen konvexen Ecken von Hindernissen, welche keine anderen Hindernisse schneiden

d) das Geradensegment von s nach z, sofern kein Hindernis davon geschnitten wird

Für jede Wahl von s und z ist der kürzeste Weg eine Kombination von Geradensegmenten der oben beschriebenen vier Typen. Der letzte Typ kommt nur dann vor, wenn die direkte Verbindung von s und z kein Hindernis schneidet; in diesem Fall ist dies auch der kürzeste Weg. Abbildung 1.3 zeigt alle möglichen Geradensegmente für die Situation aus Abbildung 1.2.

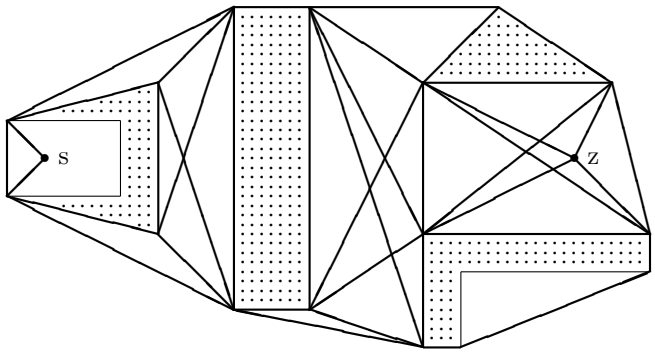


Abbildung 1.3: *Geradensegmente für den kürzesten Weg*

Um einen kürzesten Weg von s nach z zu finden, müssen im Prinzip alle Wege von s nach z, welche nur die angegebenen Segmente verwenden, betrachtet werden. Für jeden Weg ist dann die Länge zu bestimmen, und unter allen Wegen wird der kürzeste ausgewählt. Das Problem läßt sich also auf ein System von Geradensegmenten mit entsprechenden Längen reduzieren. In diesem ist dann ein kürzester Weg von s nach z zu finden.

Die Geradensegmente, die zu diesem System gehören, sind in Abbildung 1.3 fett gezeichnet. Abbildung 1.4 zeigt den kürzesten Weg von s nach z. Verfahren zur Bestimmung von kürzesten Wegen in Graphen werden in Kapitel 8 behandelt.

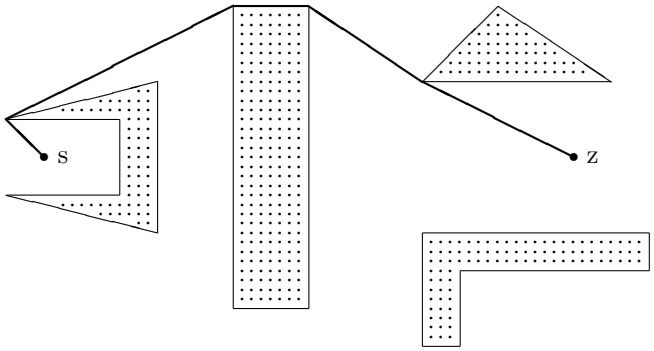


Abbildung 1.4: *Der kürzeste Weg von s nach z*

1.3 Optimale Umrüstzeiten für Fertigungszellen

Die Maschinen einer Fertigungszelle in einer Produktionsanlage müssen für unterschiedliche Produktionsaufträge verschieden ausgerüstet und eingestellt werden. Während dieser Umrüstzeit kann die Fertigungszelle nicht produktiv genutzt werden. Aus diesem Grund wird eine möglichst geringe Umrüstzeit angestrebt. Die Umrüstzeiten sind abhängig von dem Ist- und dem Sollzustand der Fertigungszelle. Liegen mehrere Produktionsaufträge vor, deren Reihenfolge beliebig ist, und sind die entsprechenden Umrüstzeiten bekannt, so kann nach einer Reihenfolge der Aufträge gesucht werden, deren Gesamtumrüstzeit minimal ist. Hierbei werden Aufträge, welche keine Umrüstzeiten erfordern, zu einem Auftrag zusammengefaßt. Das Problem läßt sich also folgendermaßen beschreiben: Gegeben sind n Produktionsaufträge $P_1, \ldots, P_n$. Die Umrüstzeit der Fertigungszelle nach Produktionsablauf P_i für den Produktionsablauf P_j wird mit T_{ij} bezeichnet. Im allgemeinen ist $T_{ij} \neq T_{ji}$, d.h. die Umrüstzeiten sind nicht symmetrisch. Eine optimale Reihenfolge hat nun die Eigenschaft, daß unter allen Reihenfolgen $P_{i_1}, \ldots, P_{i_n}$ die Summe

$$\sum_{j=1}^{n-1} T_{i_j i_{j+1}}$$

minimal ist. Diese Summe stellt nämlich die Gesamtumrüstzeit dar.

Das Problem läßt sich auch graphisch interpretieren: Jeder Produktionsablauf wird durch einen Punkt in der Ebene und jeder mögliche Umrüstvorgang durch einen Pfeil vom Anfangs- zum Zielzustand dargestellt. Die Pfeile werden mit den entsprechenden Umrüstzeiten markiert. Abbildung 1.5 zeigt ein Beispiel für vier Produktionsaufträge.

Eine Reihenfolge der Umrüstvorgänge entspricht einer Folge von Pfeilen in den vorgegebenen Richtungen. Hierbei kommt man an jedem Punkt genau einmal vorbei. Die Gesamtumrüstzeit entspricht der Summe der Markierungen dieser Pfeile; umgekehrt entspricht jeder Weg mit der Eigenschaft, daß er an jedem Punkt genau einmal vorbeikommt, einer möglichen Reihenfolge. Bezieht man noch den Ausgangszustand in diese

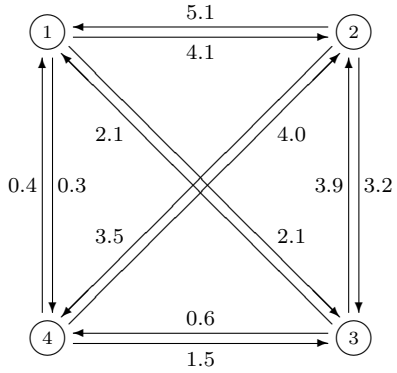


Abbildung 1.5: *Umrüstzeiten für vier Produktionsaufträge*

Darstellung ein und gibt ihm die Nummer 1, so müssen alle Wege bei Punkt 1 beginnen. Für n Produktionsaufträge gibt es somit $n+1$ Punkte. Es gibt dann insgesamt $n!$ verschiedene Wege. Eine Möglichkeit, den optimalen Weg zu finden, besteht darin, die Gesamtumrüstzeiten für alle $n!$ Wege zu berechnen und dann den Weg mit der minimalen Zeit zu bestimmen. Interpretiert man die Darstellung aus Abbildung 1.5 in dieser Art (d.h. drei Produktionsaufträge und der Ausgangszustand), so müssen insgesamt sechs verschiedene Wege betrachtet werden. Man sieht leicht, daß der Weg 1–4–3–2 mit einer Umrüstzeit von 5.7 am günstigsten ist. Die Anzahl der zu untersuchenden Wege steigt jedoch sehr schnell an. Bei zehn Produktionsaufträgen müssen bereits 3628800 Wege betrachtet werden. Bei größeren n stößt man bald an Zeitgrenzen. Für $n = 15$ müßten mehr als $1.3 \cdot 10^{12}$ Wege betrachtet werden. Bei einer Rechenzeit von einer Sekunde für 1 Million Wege würden mehr als 15 Tage benötigt. Um zu vertretbaren Rechenzeiten zu kommen, müssen andere Verfahren angewendet werden.

In Kapitel 9 werden Verfahren vorgestellt, mit denen man in einer annehmbaren Zeit zu einem Resultat gelangt, welches relativ nahe an die optimale Lösung herankommt.

1.4 Objektorientierte Programmiersprachen

Eine der wichtigsten Entwicklungen der letzten Jahre auf dem Gebiet der Programmiersprachen sind die objektorientierten Sprachen. Smalltalk und C++ sind Beispiele für solche Sprachen. In traditionellen Programmiersprachen sind Daten und Prozeduren separate Konzepte. Der Programmierer ist dafür verantwortlich, beim Aufruf einer Prozedur diese mit aktuellen Parametern vom vereinbarten Typ zu versorgen. In objektorientierten Programmiersprachen steht das Konzept der Objekte im Mittelpunkt. Objekte haben einen internen Zustand, welcher nur durch entsprechende Methoden verändert werden kann. Methoden entsprechen Prozeduren in traditionellen Programmiersprachen, und das Konzept einer Klasse ist die Verallgemeinerung des Typkonzeptes. Jedes Objekt ist Instanz einer Klasse; diese definiert die Struktur und die Methoden zum Verändern des Zustandes ihrer Instanzen.

Ein Hauptziel der objektorientierten Programmierung ist es, eine gute Strukturierung und eine hohe Wiederverwendbarkeit von Software zu erzielen. Dazu werden Klassen in Hierarchien angeordnet; man spricht dann von Ober- und Unterklassen. Jede Methode einer Klasse ist auch auf die Instanzen aller Unterklassen anwendbar. Man sagt, eine Klasse vererbt ihre Methoden rekursiv an ihre Unterklassen. Der Programmierer hat aber auch die Möglichkeit, die Implementierung einer geerbten Methode zu überschreiben. Die Methode behält dabei ihren Namen; es wird nur die Implementierung geändert. Enthält ein Programm den Aufruf einer Methode für ein Objekt, so kann in vielen Fällen während der Übersetzungszeit nicht entschieden werden, zu welcher Klasse dieses Objekt gehört. Somit kann auch erst zur Laufzeit des Programms die korrekte Implementierung einer Methode ausgewählt werden (*late binding*). Der Grund hierfür liegt hauptsächlich darin, daß der Wert einer Variablen der Klasse C auch Instanz einer Unterklasse von C sein kann.

Die Auswahl der Implementierung einer Methode zur Laufzeit nennt man *Dispatching*. Konzeptionell geht man dabei wie folgt vor:

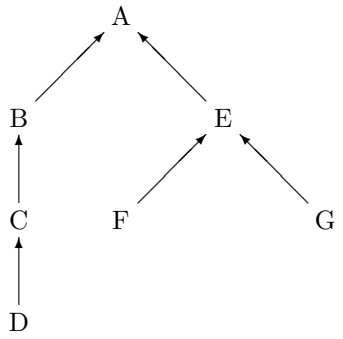
a) Bestimmung der Klasse, zu der das Objekt gehört.

b) Falls diese Klasse eine Implementierung für diese Methode zur Verfügung stellt, wird diese ausgeführt.

c) Andernfalls werden in aufsteigender Reihenfolge die Oberklassen durchsucht und wie oben verfahren.

d) Wird keine Implementierung gefunden, so liegt ein Fehler vor.

Im folgenden wird nur der Fall betrachtet, daß jede Klasse maximal eine Oberklasse hat (*single inheritance*). Hat eine Klasse mehrere Oberklassen (*multiple inheritance*), so muß die Reihenfolge, in der in Schritt c) die Oberklassen durchsucht werden, festgelegt werden.



$A : m_1$
$B : m_1, m_2$
$C : m_3$
$D : m_3$

$E : m_1, m_5$
$F : m_1, m_4$
$G : m_4$

Abbildung 1.6: *Eine Klassenhierarchie*

Abbildung 1.6 zeigt eine Klassenhierarchie bestehend aus sieben Klassen $A, B, \ldots, G$. Die Unterklassenrelation ist durch einen Pfeil von einer Klasse zu ihrer Oberklasse

gekennzeichnet. Ferner ist angegeben, welche Klassen welche Methoden implementieren. Beispielsweise kann die Methode m_1 auf jedes Objekt angewendet werden, aber es gibt vier verschiedene Implementierungen von m_1.

Welche Objekte verwenden welche Implementierung von m_1? Die folgende Tabelle gibt dazu einen Überblick; hierbei wird die Adresse einer Methode m, welche durch die Klasse K implementiert wird, durch $m \mid K$ dargestellt. Wird z.B. die Methode m_1 für eine Instanz der Klasse G aufgerufen, so ergibt sich direkt, daß $m_1 \mid E$ die zugehörige Implementierung ist.

Klasse	A	B	C	D	E	F	G
Implementierung	$m_1 \mid A$	$m_1 \mid B$	$m_1 \mid B$	$m_1 \mid B$	$m_1 \mid E$	$m_1 \mid F$	$m_1 \mid E$

Ein Programm einer objektorientierten Sprache besteht im wesentlichen aus Methodenaufrufen. Untersuchungen haben gezeigt, daß in manchen objektorientierten Sprachen bis zu 20% der Laufzeit für Dispatching verwendet wird. Aus diesem Grund besteht ein hohes Interesse an schnellen Dispatchverfahren. Die effizienteste Möglichkeit, Dispatching durchzuführen, besteht darin, zur Übersetzungszeit eine globale *Dispatchtabelle* zu erzeugen. Diese Tabelle enthält für jede Klasse eine Spalte und für jede Methode eine Zeile. Ist m der Name einer Methode und C eine Klasse, so enthält der entsprechende Eintrag der Dispatchtabelle die Adresse der Implementierung der Methode m, welche für Instanzen der Klasse C aufgerufen wird. Ist eine Methode auf die Instanzen einer Klasse nicht anwendbar, so bleibt der entsprechende Eintrag in der Dispatchtabelle leer. Diese Organisation garantiert, daß die Implementierung einer Methode in konstanter Zeit gefunden wird; es ist dabei genau ein Zugriff auf die Dispatchtabelle notwendig. Abbildung 1.7 zeigt die Dispatchtabelle für das oben angegebene Beispiel.

	A	B	C	D	E	F	G
m_1	$m_1 \mid A$	$m_1 \mid B$	$m_1 \mid B$	$m_1 \mid B$	$m_1 \mid E$	$m_1 \mid F$	$m_1 \mid E$
m_2		$m_2 \mid B$	$m_2 \mid B$	$m_2 \mid B$			
m_3			$m_3 \mid C$	$m_3 \mid D$			
m_4						$m_4 \mid F$	$m_4 \mid G$
m_5					$m_5 \mid E$	$m_5 \mid E$	$m_5 \mid E$

Abbildung 1.7: *Eine Dispatchtabelle*

Der große Nachteil einer solchen Dispatchtabelle ist der Platzverbrauch. Für die komplette Klassenhierarchie eines Smalltalk Systems mit 766 Klassen und 4500 verschiedenen Methoden ergibt sich ein Platzverbrauch von über 13 MBytes. Hierbei benötigt eine Adresse 4 Bytes. Aus diesem Grund sind Dispatchtabellen in dieser Form praktisch nicht verwendbar.

Um den Speicheraufwand zu senken, macht man sich zunutze, daß die meisten Einträge der Dispatchtabelle nicht besetzt sind. Der Speicheraufwand kann reduziert werden,

indem man für zwei Methoden, welche sich nicht beeinflussen, eine einzige Zeile verwendet. Zwei Methoden beeinflussen sich nicht, wenn es kein Objekt gibt, auf welches beide Methoden anwendbar sind. Im obigen Beispiel beeinflussen sich die Methoden m_3 und m_5 nicht; somit können beide Methoden in der Dispatchtabelle eine gemeinsame Zeile verwenden. Um eine optimale Kompression zu finden, wird zunächst untersucht, welche Methoden sich nicht beeinflussen. Dazu wird ein sogenannter *Konfliktgraph* gebildet. In diesem werden Methoden, welche sich beeinflussen, durch eine Kante verbunden. Abbildung 1.8 zeigt den Konfliktgraphen für das obige Beispiel.

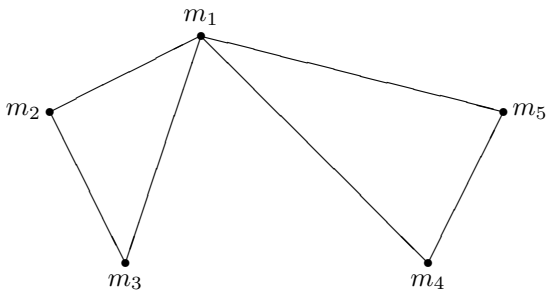


Abbildung 1.8: *Ein Konfliktgraph*

Um die minimale Anzahl von Zeilen zu finden, wird eine Aufteilung der Menge der Methoden in eine minimale Anzahl von Teilmengen vorgenommen, so daß die Methoden in jeder Teilmenge nicht durch Kanten verbunden sind. Jede Teilmenge entspricht dann einer Zeile. Für das obige Beispiel ist $\{m_1\}$, $\{m_2, m_5\}$ und $\{m_3, m_4\}$ eine solche minimale Aufteilung. Die Dispatchtabelle wird nun mittels zwei Tabellen dargestellt. Die erste Tabelle gibt für jede Methode die entsprechende Zeile in der zweiten Tabelle an. Abbildung 1.9 zeigt diese Tabellen für das obige Beispiel.

Methode	m_1	m_2	m_3	m_4	m_5
Zeile	1	2	3	3	2

	A	B	C	D	E	F	G
1	$m_1 \mid A$	$m_1 \mid B$	$m_1 \mid B$	$m_1 \mid B$	$m_1 \mid E$	$m_1 \mid F$	$m_1 \mid E$
2		$m_2 \mid B$	$m_2 \mid B$	$m_2 \mid B$	$m_5 \mid E$	$m_5 \mid E$	$m_5 \mid E$
3			$m_3 \mid C$	$m_3 \mid D$		$m_4 \mid F$	$m_4 \mid G$

Abbildung 1.9: *Komprimierte Dispatchtabelle*

Die Auswahl einer entsprechenden Methode erfordert nun zwei Tabellenzugriffe. Die Bestimmung der zugehörigen Zeilen kann dabei schon zur Übersetzungszeit erfolgen;

somit ist zur Laufzeit weiterhin nur ein Tabellenzugriff erforderlich. Der Platzbedarf wird durch dieses Verfahren häufig erheblich reduziert. Wendet man dieses Verfahren auf das oben angegebene Smalltalk System an, so reduziert sich der Speicheraufwand auf etwas mehr als 1 MByte. Dies entspricht einer Reduktion von etwa 90%.

Das Problem bei dieser Vorgehensweise ist die Bestimmung einer Aufteilung der Methoden in eine minimale Anzahl von Teilmengen von sich nicht beeinflussenden Methoden. Dieses Problem wird ausführlich in den Kapiteln 5 und 9 behandelt.

1.5 Suchmaschinen

Das World Wide Web ist inzwischen zu einer bedeutenden Informationsquelle von enormer Größe geworden. Die Suche nach Informationen zu einem bestimmten Thema in diesem weltumspannenden Netzwerk ist schwierig. Allein durch manuelles Durchsuchen von Web-Seiten mit einem Browser ist eine umfassende Informationssuche nicht möglich. Man hat eine bessere Chance die gewünschte Information zu finden, wenn man die Dienste von Suchmaschinen in Anspruch nimmt. Diese verwalten in der Regel einen Index, der Stichwörter mit Web-Seiten verbindet. Eine Anfragesprache ermöglicht eine gezielte Suche in diesem Index. Die Indizes der Suchmaschinen werden automatisch durch so genannte Web-Roboter aufgebaut.

Das Konzept der Suchmaschinen basiert auf einer graphentheoretischen Modellierung des Inhaltes des WWW. Die Grundlage für das Auffinden neuer Dokumente bildet die Hypertextstruktur des Web. Diese macht das Web zu einem gerichteten Graphen, wobei die Dokumente die Ecken bilden. Enthält eine Web-Seite einen Verweis (*Link*) auf eine andere Seite, so zeigt eine Kante von der ersten Seite zu der zweiten Seite. Abbildung 1.10 zeigt ein Beispiel mit vier Web-Seiten und fünf Verweisen. Die wahre Größe des WWW ist nicht bekannt, über die Anzahl der verschiedenen Web-Seiten gibt es nur Schätzungen. Die Betreiber von Suchmaschinen veröffentlichen gelegentlich Angaben über die Anzahl der durch sie indizierten Web-Seiten. Der Marktführer Google indiziert zurzeit mehr als eine Billion Web-Seiten.

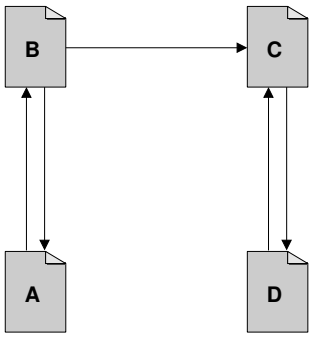


Abbildung 1.10: *Vier Web-Seiten mit ihren Verweisen*

Um möglichst viele Seiten aufzufinden, ohne dabei Seiten mehrfach zu analysieren, müssen Web-Roboter bei der Suche nach neuen Dokumenten systematisch vorgehen. Häufig werden von einer Web-Seite aus startend rekursiv alle Verweise verfolgt. Ausgehend von einer oder mehreren Startadressen durchsuchen Web-Roboter das Web und extrahieren nach verschiedenen Verfahren Wörter und fügen diese in die Indizes ein. Zur Verwaltung der Indizes werden spezielle Datenbanksysteme verwendet. Dabei werden sowohl statistische Methoden wie Worthäufigkeiten und inverse Dokumentenhäufigkeiten als auch probabilistische Methoden eingesetzt. Viele Roboter untersuchen die Dokumente vollständig, andere nur Titel und Überschriften. Zur Unterstützung komplexer Anfragen werden neben Indizes auch die Anfänge der Dokumente abgespeichert. Bei der Analyse von Web-Seiten kommen HTML-Parser zum Einsatz, die je nach Aufgabenstellung den Quelltext mehr oder weniger detailliert analysieren.

Anwender nutzen den Dienst einer Suchmaschine, indem sie ihre Suchbegriffe in eine entsprechende Suchmaske eingeben. Die Suchmaschine durchsucht daraufhin ihren Index nach dem Vorkommen der Suchbegriffe und bestimmt die Treffermenge. Die Größe des Web bedingt, daß die Anzahl der gefundenen Dokumente sehr groß sein kann. Darüber hinaus haben viele der gefundenen Dokumente oft nur eine geringe oder sogar keine Relevanz für die Anfrage. Im Laufe der Zeit wurden deshalb verschiedene Verfahren zur Bewertung von Webseiten mit dem Ziel der Relevanzbeurteilung durch Suchmaschinen entwickelt. Die gefundenen Seiten werden dann nach ihrer vermeintlichen Relevanz sortiert. Dies erleichtert Anwendern die Sichtung des Suchergebnisses.

Ein aus unmittelbar nachvollziehbaren Gründen auch heute immer noch von praktisch allen Suchmaschinen genutzter Maßstab ist das Vorkommen eines Suchbegriffs im Text einer Webseite. Dieses Vorkommen wird nach verschiedensten Kriterien wie etwa der relativen Häufigkeit und der Position des Vorkommens oder auch der strukturellen Platzierung des Suchbegriffs im Dokument gewichtet.

Im Zuge der wachsenden wirtschaftlichen Bedeutung von Suchmaschinen versuchten Autoren ihre Web-Seiten so zu gestalten, daß sie bei der Relevanzbeurteilung durch Suchmaschinen gut abschnitten. Beispielsweise wurden zugkräftige Wörter mehrfach an wichtigen Stellen in den Dokumenten plaziert (sogar in Kommentaren). Um einem solchen Mißbrauch entgegenzuwirken, entwickelten Suchmaschinenbetreiber weitere Relevanzkriterien. Ein einfaches Verfahren besteht darin, die Anzahl der Verweise auf ein Dokument als ein grundsätzliches Kriterium für die Bedeutung einzubeziehen. Ein Dokument ist dabei um so wichtiger, je mehr Dokumente auf es verweisen. Es zeigte sich aber bald, daß auch dieses Konzept schnell von Webmastern unterlaufen werden konnte, indem sie automatisch Web-Seiten mit einer Vielzahl von Verweisen auf ihre eigentlichen Seiten generierten.

Ein vom Suchdienst Google zum ersten Mal eingesetztes Konzept heißt *PageRank*. Es basiert auf der Idee, daß ein Dokument zwar bedeutsam ist, wenn andere Dokumente Verweise auf es enthalten, nicht jedes verweisende Dokument ist jedoch gleichwertig. Vielmehr wird einem Dokument unabhängig von seinem Inhalten ein hoher Rang zugewiesen, wenn anderen bedeutende Dokumente auf es verweisen. Die Wichtigkeit eines Dokuments bestimmt sich beim PageRank-Konzept aus der Bedeutsamkeit der darauf verweisenden Dokumente, d.h., die Bedeutung eines Dokuments definiert sich rekursiv aus der Bedeutung anderer Dokumente. Somit hat im Prinzip der Rang jedes Doku-

ments eine Auswirkung auf den Rang aller anderen Dokumente, die Verweisstruktur des gesamten Web wird also in die Relevanzbewertung eines Dokumentes einbezogen.

Der von Lawrence Page und Sergey Brin entwickelte PageRank-Algorithmus ist sehr einfach nachvollziehbar. Für eine Web-Seite W bezeichnet $LC(W)$ die Anzahl der Verweise, welche von Seite W ausgehen, d.h., Verweise die im Text von W enthalten sind, und $PL(W)$ bezeichnet die Menge der Web-Seiten, welche einen Verweis auf Seite W enthalten. Der PageRank $PR(W)$ einer Seite W berechnet sich wie folgt:

$$PR(W) = (1-d) + d \sum_{D \in PL(W)} \frac{PR(D)}{LC(D)}$$

Hierbei ist $d \in [0,1]$ ein einstellbarer Dämpfungsfaktor. Der PageRank einer Seite W bestimmt sich dabei rekursiv aus dem PageRank derjenigen Seiten, die einen Verweis auf Seite W enthalten. Der PageRank der Seiten $D \in PL(W)$ fließt nicht gleichmäßig in den PageRank von W ein. Der PageRank einer Seite wird stets anhand der Anzahl der von der Seite ausgehenden Links gewichtet. Das bedeutet, daß je mehr ausgehende Links eine Seite D hat, um so weniger PageRank gibt sie an W weiter. Der gewichtete PageRank der Seiten $D \in PL(W)$ wird addiert. Dies hat zur Folge, daß jeder zusätzliche eingehende Link auf W stets den PageRank dieser Seite erhöht. Schließlich wird die Summe der gewichteten PageRanks der Seiten $D \in PL(W)$ mit dem Dämpfungsfaktor d multipliziert. Hierdurch wird das Ausmaß der Weitergabe des PageRanks von einer Seite auf einer andere verringert.

Die Eigenschaften des PageRank werden jetzt anhand des in Abbildung 1.10 dargestellten Beispieles veranschaulicht. Der Dämfungsfaktor d wird Angaben von Page und Brin zufolge für tatsächliche Berechnungen oft auf 0.85 gesetzt. Der Einfachheit halber wird d an dieser Stelle ein Wert von 0.5 zugewiesen, wobei der Wert von d zwar Auswirkungen auf den PageRank hat, das Prinzip jedoch nicht beeinflußt. Für obiges Beispiel ergeben sich folgende Gleichungen:

$$\begin{aligned}
PR(A) &= 0.5 + 0.5\, PR(B)/2 \\
PR(B) &= 0.5 + 0.5\, PR(A) \\
PR(C) &= 0.5 + 0.5\, (PR(B)/2 + PR(D)) \\
PR(D) &= 0.5 + 0.5\, PR(C)
\end{aligned}$$

Dieses Gleichungssystem läßt sich sehr einfach lösen. Es ergibt sich folgende Lösung:

$$PR(A) = 0.71428 \;\; PR(B) = 0.85714 \;\; PR(C) = 1.2857 \;\; PR(D) = 1.1428$$

Somit hat Seite C den höchsten und A den niedrigsten PageRank.

Es stellt sich die Frage, wie der PageRank von Web-Seiten berechnet werden kann. Eine ganzheitliche Betrachtung des WWW ist sicherlich ausgeschlossen, deshalb erfolgt in der Praxis eine näherungsweise, iterative Berechnung des PageRank. Dazu wird zunächst jeder Seite ein PageRank mit Wert 1 zugewiesen, und anschließend wird der PageRank aller Seiten in mehreren Iterationen ermittelt. Abbildung 1.11 zeigt das Ergebnis der ersten fünf Iterationen für das betrachtete Beispiel. Bereits nach sehr wenigen Iterationen wird eine sehr gute Näherung an die tatsächlichen Werte erreicht. Für die Berechnung

Iteration	$PR(A)$	$PR(B)$	$PR(C)$	$PR(D)$
0	1	1	1	1
1	0.75	0.875	1.21875	1.109375
2	0.71875	0.859375	1.2695312	1.1347656
3	0.71484375	0.8574219	1.2817383	1.1408691
4	0.71435547	0.85717773	1.284729	1.1423645
5	0.71429443	0.8571472	1.285469	1.1427345

Abbildung 1.11: *Die iterative Berechnung des PageRank*

des PageRanks für das komplette Web werden von Page und Brin etwa 100 Iterationen als hinreichend genannt.

Die bei der systematischen Suche nach neuen Web-Seiten von Web-Robotern eingesetzten Verfahren wie Tiefen- und Breitensuche werden in Kapitel 4 ausführlich vorgestellt.

1.6 Analyse sozialer Netze

Der Begriff *Soziales Netzwerk* bezeichnet eine soziale Struktur, die zwischen menschlichen Akteuren mittels ihrer Interaktion entsteht. Der Begriff ist recht weit gefasst. Es ist nicht festgelegt, um welche Art von Interaktion es sich handelt und ob die Akteure Individuen, Gruppen oder Organisationen sind. Soziale Netzwerke lassen sich einfach als Graphen modellieren. Die Akteure bilden die Ecken und Interaktionen zwischen ihnen werden als Kanten dargestellt. Der US-amerikanische Psychologe S. Milgram untersuchte bereits in den 60-er Jahren soziale Netze. Er vertrat die Hypothese, daß jeder Mensch auf der Welt mit jedem anderen über eine überraschend kurze Kette von Bekanntschaftsbeziehungen verbunden ist. Im zugehörenden sozialen Netzwerk sind die Menschen die Ecken und die Bekanntschaft zweier Menschen wird durch eine Kante repräsentiert. Diese Art der Darstellung von sozialen Beziehungen wird auch *Soziogramm* genannt. Da es unmöglich ist diesen Graph explizit anzugeben, beschränkte sich Milgram auf empirische Untersuchungen (*Small world experiment*). Er interpretierte die Ergebnisse in der Art, daß Bürger der USA im Durchschnitt durch eine Kette von 6 Personen verbunden sind. Die Schlußfolgerungen aus seinen Untersuchungen sind wegen der geringen Datenlage allerdings umstritten.

Im Zeitalter des Internet erreichte die Erforschung sozialer Netze eine neue Dimension. Das World Wide Web bietet zahllose Plattformen, mit denen Benutzer ein persönliches Profil verwalten können. Des Weiteren können Beziehungen zu anderen Personen abgebildet werden. Allein in Deutschland nutzten im Jahr 2009 mehr als 10 Millionen Menschen solche Websites. Jede dieser Plattformen definiert ein soziales Netzwerk, wobei die Nutzer die Akteure sind. Die Kommunikationsmuster in Diskussionsforen oder Email-Listen können ebenfalls durch soziale Netzwerke repräsentiert werden. Abbildung 1.12 zeigt ein soziales Netzwerk mit 10 Akteuren, diese sind mit den Buchstaben A bis J gekennzeichnet. Hierbei sind zwei Personen durch eine Kante verbunden, falls

sie regelmäßig Emails austauschen. Dieses Beispiel wird wegen seiner Form auch Kite-Netzwerk genannt und wurde erstmals von D. Krackhard verwendet.

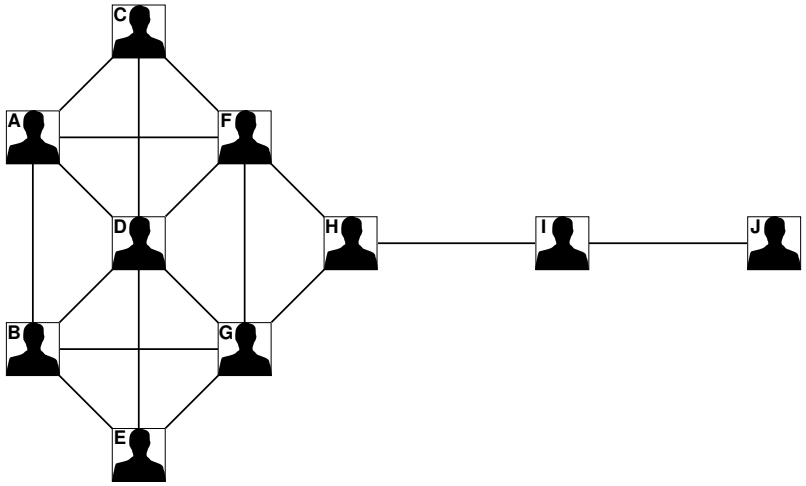


Abbildung 1.12: *Das Kite-Netzwerk, ein Beispiel für ein soziales Netzwerk*

Der Zusammenhang zwischen der Struktur eines sozialen Netzwerkes auf der einen und dem Verhalten der darin eingebetteten Akteure und Gruppen auf der anderen Seite ist seit vielen Jahrzehnten Gegenstand sozialwissenschaftlicher Untersuchungen. Die hierbei verwendeten Methoden werden unter dem Begriff *Analytik sozialer Netzwerke* zusammengefasst. Im Folgenden werden wichtige Konzepte dieser Analytik und ihre graphentheoretische Interpretation vorgestellt.

Ein wichtiges Merkmal eines sozialen Netzwerkes ist seine *Dichte*. Sie bezeichnet das Verhältnis der Anzahl der Kanten in einem Soziogramm zur Zahl der möglichen Kanten. Diese Maßzahl verdeutlicht die Verbundenheit des sozialen Netzwerkes. Die Dichte des in Abbildung 1.12 dargestellten Netzwerks beträgt $18/45 = 0.4$. Die Dichte liegt immer im Intervall $[0, 1]$. Je näher die Dichte an 1 liegt, desto schneller kann sich beispielsweise eine Informationen im Netzwerk ausbreiten. Im Folgenden bezeichne n stets die Anzahl der Akteure und m die Anzahl der Kanten in einem sozialen Netzwerk. Die Dichte ist durch folgenden Ausdruck gegeben:

$$\frac{2m}{n(n-1)}$$

Ein weiteres Mittel zur Untersuchung von Gesamtnetzwerken ist die *Cliquenanalyse*. Unter einer *Clique* versteht man eine Teilmenge der Ecken, die alle untereinander direkt verbunden sind. Eine Clique kennzeichnet eine Gruppe von Akteuren mit hoher Kohäsion. Die Cliquenanalyse unterteilt ein Gesamtnetzwerk in disjunkte Cliquen. Diese Aufteilung bringt Erkenntnisse über die Struktur und den inneren Zusammenhang

eines sozialen Netzwerkes. Das in Abbildung 1.12 dargestellte Netzwerk kann beispielsweise in die disjunkten Cliquen $\{A, C, D, F\}$, $\{B, E, G\}$ und $\{H, I\}$ unterteilt werden. Es gibt keine Clique, die mehr als vier Akteure enthält.

Die Untersuchung eines sozialen Netzwerkes kann auch den Fokus auf einzelne Akteure richten. Dabei geht es um die Frage, wie zentral der Akteur ist bzw. welches Prestige er besitzt. Zur Messung von Zentralität (*Centrality*) wurden verschiedene Verfahren entwickelt, einige davon werden im folgenden vorgestellt. Ein Akteur ist zentral im Sinne der *Degree-Centrality*, wenn er direkte Beziehungen zu möglichst vielen anderen Akteuren hat. Für einen Akteur, der mit $d(e)$ anderen Akteuren durch eine Kante verbunden ist, ist die Degree-Centrality $C_D(e)$ durch $d(e)/(n-1)$ gegeben. Diese Maßzahl kennzeichnet die Eingebundenheit eines Akteurs in einem Netzwerk. Je näher $C_D(e)$ an 1 liegt, desto höher ist das Prestige des Akteurs. In Abbildung 1.13 sind die Degree-Centrality Werte der 10 Akteure aus Abbildung 1.12 zusammengefasst. Akteur D hat mit $2/3 \approx 0.67$ den höchsten Wert.

	A	B	C	D	E	F	G	H	I	J
Degree	0.44	0.44	0.33	0.67	0.33	0.56	0.56	0.33	0.22	0.11
Betweenness	0.02	0.02	0	0.10	0	0.23	0.23	0.39	0.22	0
Closeness	1.89	1.89	2	1.67	2	1.56	1.56	1.67	2.33	3.22

Abbildung 1.13: *Die Werte der Zentralitätsmaße für die Akteure aus Abbildung 1.12*

Die *Betweenness-Centrality* ist ein Maß für die Bedeutung eines Akteurs für den Informationsaustausch. Ein Akteur hat signifikanten Einfluß auf den Informationsaustausch zwischen zwei anderen Akteuren, wenn er auf einem kürzesten Pfad zwischen diesen beiden Akteuren liegt. Ein Akteur gilt dann als zentral, wenn er auf möglichst vielen kürzesten Pfaden liegt. In Abbildung 1.12 haben die Akteure F, G und H zentrale Positionen. Die Betweenness-Centrality $C_B(e)$ eines Akteurs e kann wie folgt bestimmt werden:

$$C_B(e) = \frac{2}{(n-1)(n-2)} \sum_{s \neq e \neq t, s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

Hierbei bezeichnet σ_{st} die Anzahl der kürzesten Pfade von s nach t und $\sigma_{st}(e)$ die Anzahl dieser Pfade, die e enthalten. Summiert wird über alle ungeordneten Paare s, t von Akteuren mit $s \neq e \neq t$ und $s \neq t$. Der Faktor $2/(n-1)(n-2)$ dient der Normierung des Wertes auf das Intervall $[0, 1]$. Je näher $C_B(e)$ an 1 ist, desto weitreichender sind die Kontrollmöglichkeiten, die dem Akteur e aufgrund seiner strategischen Position im Netzwerk zufallen. Dieses Maß geht implizit davon aus, daß Kommunikation immer entlang kürzester Wege erfolgt.

Zur Bestimmung von $C_B(H)$ bezüglich des in Abbildung 1.12 dargestellten Netzwerks beachte man, daß $\sigma_{IJ}(H) = 0$ und $\sigma_{st}(H) = 0$ für $\{s, t\} \subset \{\text{A,B,C,D,E,F,G}\}$ gilt. Da H auf allen kürzesten Wegen zwischen Ecken in den Mengen {I,J} und {A,B,C,D,E,F,G} liegt, gilt $\sigma_{sI}(H) = \sigma_{sI}$ und $\sigma_{sJ}(H) = \sigma_{sJ}$ für $s \in \{\text{A,B,C,D,E,F,G}\}$. Somit ergibt sich

$C_B(H) = 2/72 \cdot 14 \approx 0.39$, für alle anderen Akteure ist C_B kleiner. Die restlichen Werte sind in Abbildung 1.13 zusammengefasst.

Bei der *Closeness-Centrality* C_C werden neben den direkten auch die indirekten Beziehungen betrachtet. Der Wert dieser Maßzahl für einen Akteur e entspricht den aufsummierten Längen der kürzesten Pfade, über die der betrachtete Akteur zu allen anderen in Beziehung steht, dividiert durch $n - 1$:

$$C_C(e) = \frac{\sum_{t \neq e} d(e,t)}{n-1}$$

Summiert wird über alle Akteure $t \neq e$, welche von e aus erreichbar sind. Hierbei bezeichnet $d(e,t)$ die Anzahl der Kanten auf einem kürzesten Weg von e nach t. $C_C(e)$ ist ein Maß für die Nähe eines Akteurs e zu allen anderen. Je kleiner $C_C(e)$ ist, desto schneller kann der Akteur mit den restlichen Akteuren interagieren. Diese Maßzahl ist vor allem in der Bewertung der Kommunikation zwischen Akteuren von Bedeutung. In Abbildung 1.12 haben F und G mit jeweils 1.56 die kleinste Closeness-Centrality. Von diesen Akteuren verbreiten sich Informationen am schnellsten im gesamten Netzwerk aus. Die restlichen Werte sind in Abbildung 1.13 zusammengefasst.

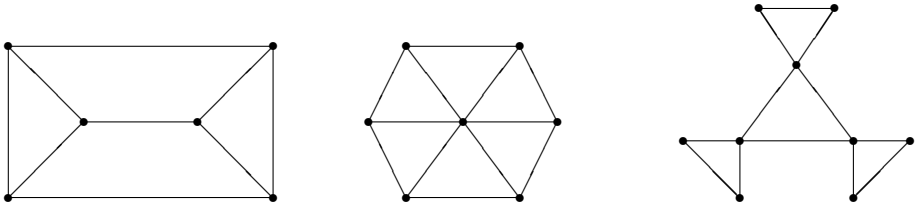
Cliquen werden in Kapitel 5 behandelt. Die Aufgaben 37 in Kapitel 9 und 29 in Kapitel 2 beschreiben Verfahren zur Bestimmung großer Cliquen. Effiziente Algorithmen zur Bestimmung der Betweenness- und der Closeness-Centrality werden in Abschnitt 8.5 und Aufgabe 37 in Kapitel 8 vorgestellt.

1.7 Literatur

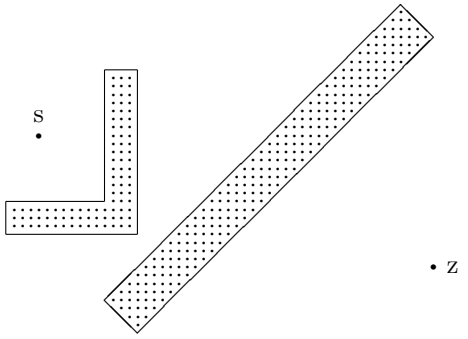
Verletzlichkeit von Kommunikationsnetzen ist ein Teilgebiet der *Zuverlässigkeitstheorie*. Eine ausführliche Darstellung der graphentheoretischen Aspekte findet man in [82]. Eine gute Übersicht über *Wegeplanungsverfahren* in der Robotik findet man in [20]. Das Problem der optimalen Umrüstzeiten ist eng verwandt mit dem *Traveling-Salesman Problem*. Eine ausführliche Darstellung findet man in [89]. Verfahren zur Optimierung von Methodendispatching in objektorientierten Programmiersprachen sind in [6] beschrieben. Weitere Information zur Bestimmung des PageRank einer Web-Seite findet man in [18, 104]. Weitere praktische Anwendungen der Graphentheorie sind in [64, 65] beschrieben. Die Grundzüge der Analyse sozialer Netzwerke sind in [17] beschrieben.

1.8 Aufgaben

1. Bestimmen Sie Verbindungs- und Knotenzusammenhang der folgenden drei Netzwerke.



2. Bestimmen Sie für die im folgenden dargestellte Menge von Hindernissen das System aller Geradensegmente analog zu Abbildung 1.3. Bestimmen Sie den kürzesten Weg von s nach z.

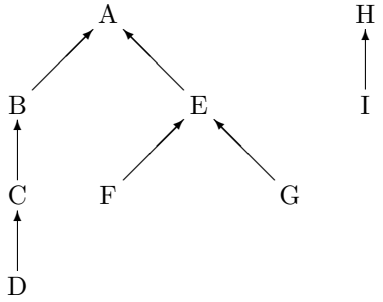


3. Im folgenden sind die Umrüstzeiten T_{ij} für vier Produktionsaufträge für eine Fertigungszelle in Form einer Matrix gegeben.

$$\begin{pmatrix} 0 & 1.2 & 2 & 2.9 \\ 0.8 & 0 & 1.0 & 2.3 \\ 2 & 1.2 & 0 & 3.4 \\ 5.1 & 1.9 & 2.1 & 0 \end{pmatrix}$$

Bestimmen Sie die optimale Reihenfolge der Produktionsaufträge, um die geringste Gesamtumrüstzeit zu erzielen. Die Fertigungszelle befindet sich anfangs im Zustand 1.

4. Bestimmen Sie für die folgende Klassenhierarchie und die angegebenen Methoden den Konfliktgraphen und geben Sie eine optimal komprimierte Dispatchtabelle an.



$A : m_1$
$B : m_1, m_4$
$C : m_1, m_3$
$D : m_1$
$E : m_2, m_3$
$F : m_1$
$G : m_1$
$H : m_4$
$I : m_5$

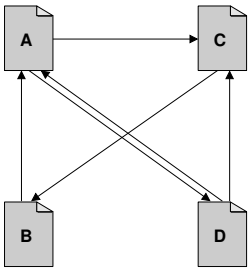
5. Es sei M eine Menge von Web-Seiten mit folgenden beiden Eigenschaften:

   a) Jede Seite in M enthält mindestens einen Verweis und

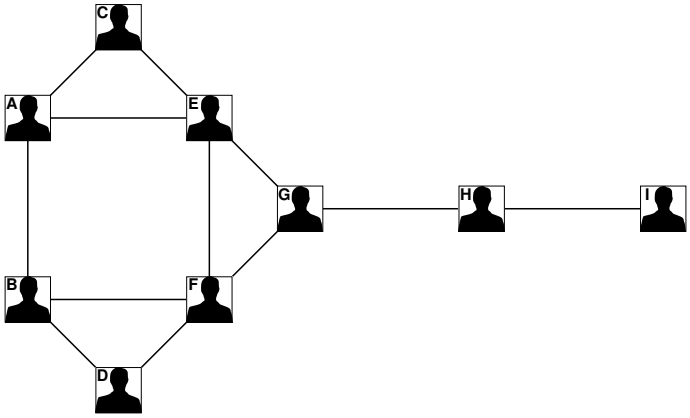
   b) außerhalb von M gibt es keine Seiten, welche auf Seiten in M verweisen.

   Beweisen Sie, daß die Summe der Werte des PageRank aller Seiten aus M gleich der Anzahl der Seiten in M ist.

6. Bestimmen Sie den PageRank aller Seiten der folgenden Hypertextstruktur ($d = 0.5$).



7. Bestimmen Sie die Dichte des folgendes sozialen Netzwerks. Berechnen Sie Degree-Centrality, Betweenness-Centrality und Closeness-Centrality für jeden Akteur.



8. Berechnen Sie Degree-Centrality, Betweenness-Centrality und Closeness-Centrality für die Ecken eines Graphen G, bei dem jede Ecke mit jeder anderen Ecken verbunden ist.

9. Geben Sie ein Beispiel für ein soziales Netzwerk an, in dem es eine Ecke e mit $C_B(e) = 1$ gibt.

Index