Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARTIN GRÖTSCHEL

Tiefensuche: Bemerkungen zur Algorithmengeschichte

ZIB-Report 08-23 (Juni 2008)

Martin Grötschel

TIEFENSUCHE: BEMERKUNGEN ZUR ALGORITHMENGESCHICHTE

Dieser kurze Aufsatz zur Algorithmengeschichte ist Eberhard Knobloch, meinem Lieblings-Mathematikhistoriker, zum 65. Geburtstag gewidmet. Eberhard Knobloch hat immer, wenn ich ihm eine historische Frage zur Mathematik stellte, eine Antwort gewusst – fast immer auch sofort. Erst als ich mich selbst ein wenig und dazu amateurhaft mit Mathematikgeschichte beschäftigte, wurde mir bewusst, wie schwierig dieses „Geschäft" ist. Man muss nicht nur mehrere (alte) Sprachen beherrschen, sondern auch die wissenschaftliche Bedeutung von Begriffen und Symbolen in früheren Zeiten kennen. Man muss zusätzlich herausfinden, was zur Zeit der Entstehung der Texte „allgemeines Wissen" war, insbesondere, was seinerzeit gültige Beweisideen und -schritte waren, und daher damals keiner präzisen Definition oder Einführung bedurfte. Es gibt aber noch eine Steigerung des historischen Schwierigkeitsgrades: Algorithmengeschichte. Dies möchte ich in diesem Artikel kurz darlegen in der Hoffnung, dass sich Wissenschaftshistoriker dieses Themas noch intensiver annehmen, als sie das bisher tun. Der Grund ist, dass heute Algorithmen viele Bereiche unserer Alltagswelt steuern und unser tägliches Leben oft von funktionierenden Algorithmen abhängt. Daher wäre eine bessere Kenntnis der Algorithmengeschichte von großem Interesse.

1. EINFÜHRUNG

Auch wenn ich gleich zu Beginn des Artikels zusammenfassend feststelle, dass der Stand der Algorithmengeschichte nicht befriedigend ist, ist dieser Artikel keineswegs eine Beschwerde über die Kolleginnen und Kollegen, die sich mit diesem Gebiet beschäftigen. In vielfach mühseliger Arbeit und vermutlich nicht selten begünstigt durch Zufallsfunde sind wunderbare Erkenntnisse darüber gewonnen worden, wann welche Algorithmen entdeckt, wo sie wiedergefunden und wie sie verwendet wurden. Spannend ist hier bereits die Geschichte der algorithmischen Ausführung einfacher arithmetischer Operationen wie Addition, Subtraktion, Multiplikation und Division. Die dafür benutzten Methoden sind u.a. stark abhängig vom benutzten Zahlensystem. Und wie nun die „Rechenkünstler" früherer Zeiten ihre Rechenoperationen genau vorgenommen haben, kann man kaum in „Handbüchern" nachlesen, sondern meistens nur indirekt aus konkreten Zahlenbeispielen erschließen, die sich manchmal nicht wirklich eindeutig interpretieren lassen. Interessant ist die Beobachtung, dass schon in sehr früher Zeit Computer benutzt wurden. Natürlich waren dies keine Geräte in unserem heutigen Sinne, sondern eher Rechenhilfen wie Kieselsteine, Additions- und Multiplikationstabellen, Knotenstricke, Zählbretter, Logarithmentafeln, bis hin zum Abacus oder – mir noch geläufig – zum Rechenschieber. Die Erfindung dieser Rechengeräte erforderte algorithmisches Denken, der Bau Konstruktionsanweisungen, die Benutzung benötigte Regeln und Einweisungen. Hier begegnen wir bereits ersten Schritten hin zur heutigen Programmierung elektronischer Rechner.

Algorithmengeschichte, so wie sie heute geschrieben wird, beschäftigt sich meistens mit explizit in der Literatur formulierten Algorithmen, wie etwa Methoden zum Wurzelziehen, zum Berechnen von Nullstellen von Polynomen, zur Lösung von Gleichungssystemen etc. Hier sind erstaunliche Entdeckungen gemacht worden wie z.B. die Tatsache, dass die Methode zur Lösung von linearen Gleichungssystemen, die wir heute, nach Carl Friedrich Gauß (1777-1855), *Gauß-Algorithmus* nennen, schon 500 Jahre früher in China benutzt wurde.[1] Es ist nicht unwahrscheinlich, dass sie viel älter ist und auch in anderen Kulturen entdeckt wurde, aber es ist unendlich schwierig, so etwas herauszufinden oder gar den ersten Erfinder des Verfahrens zu ermitteln. Im Weiteren werde ich einige überraschende Funde aus neuester Zeit zu erst kürzlich entworfenen Algorithmen erwähnen, bei denen man eigentlich nicht glauben sollte, dass sie heute noch gemacht werden.

Abschnitt 2 dieses Artikels widmet sich dem ungarischen Algorithmus, einem Verfahren zur Lösung des Zuordnungsproblems. Zu seiner Geschichte wurde vor zwei Jahren eine ungewöhnliche Entdeckung gemacht. Diese Beobachtung zeigt mehr als deutlich, wie schwierig es ist, Urheber von Algorithmen zu identifizieren.

Abschnitt 3 vertieft diesen Aspekt bezüglich der Graphentheorie. Ich behaupte hier, dass die Graphentheorie einen riesigen Schatz an unentdeckten Algorithmen birgt. Der Grund liegt darin, dass Graphentheoretiker häufig nicht an Algorithmen interessiert sind. Sie schreiben ihre Konstruktionsvorschriften in Beweisform auf, so dass die algorithmischen Ideen vielfach schwer aufzuspüren sind. Wenn man vom algorithmischen Standpunkt ausgehend (und mit etwas Boshaftigkeit) graphentheoretische Literatur analysiert, so kann man andersherum durchaus mit einigem Recht feststellen, dass Graphentheorie in weiten Teilen nichts anderes als Analyse von heuristischen Algorithmen ist.

Abschnitt 4 widme ich der neueren Geschichte der linearen Programmierung. Hier ist die Frage, wann ist eine Beschreibung einer algorithmischen Idee schon ein Algorithmus, und wem gebührt Priorität bzw. Anerkennung für was? Solche Fragen sind, selbst wenn man (wie ich) einen Teil der Entwicklung selbst wissenschaftlich miterlebt hat, nicht immer einfach zu entscheiden. Wie werden das Historiker in einigen hundert Jahren mit viel geringerer Insiderkenntnis und weniger Quellen tun?

In Abschnitt 5 betrachte ich algorithmische Aspekte von Wahlverfahren, ein eher ungewöhnliches Thema für Algorithmenhistoriker. Auch hier sind viele historische Zusammenhänge unklar, und warum ein und dasselbe Verfahren gleich mehrere Namen tragen kann, erläutere ich an einem aktuellen Ereignis.

Eine kurze Zusammenfassung beendet den Artikel. Ich wage in diesem Artikel zwei Behauptungen: Die Erstentdecker von Algorithmen sind nur in seltenen Fällen auffindbar, und mit großer Wahrscheinlichkeit (zumindest bei der heutigen Form der Erarbeitung der Algorithmengeschichte) wird der größte Teil der jemals entwickelten Algorithmen einfach nicht entdeckt werden. Die Gründe dafür liegen darin, dass Algorithmen in allen Wissenschaften und Anwendungsfeldern entwickelt

1 Siehe hierzu z.B. Chabert 1999, S. 291ff.

wurden. Sie wurden jedoch nicht als Algorithmen gekennzeichnet, sondern waren Teile von Handlungsanweisungen oder Rezepten oder umfangreicheren, meistens mathematischen Argumenten oder Beweisen. Ein paar Anhaltspunkte für meine Behauptungen lege ich hier vor. Für eine wirkliche Quantifizierung der Behauptung, dass die meisten Algorithmen unentdeckt in der Literatur schlummern, habe ich natürlich keinen statistischen Beleg.

2. DER UNGARISCHE ALGORITHMUS

Der ungarische Algorithmus ist eine Methode zur Lösung des *Zuordnungsproblems*. Bei diesem Problem sind zwei endliche Mengen, sagen wir U und V, gleicher Kardinalität gegeben. Dazu gibt es Werte c(u,v) für alle Paare u aus U und v aus V. Gesucht ist eine Zuordnung der Elemente aus U zu den Elementen aus V, so dass die Summe der zugehörigen Werte so klein wie möglich ist. Eine Standardanwendung ist die folgende. U ist die Menge der vorhandenen Arbeiter, V ist eine Menge von Aufgaben, und c(u,v) bezeichnet die Zeit, die Arbeiter u benötigt, um Aufgabe v zu erledigen. Gesucht ist eine Zuordnung der Arbeiter zu den Aufgaben, so dass die Gesamtzeit, die benötigt wird, um alle Aufgaben zu erledigen, so gering wie möglich ist. Das Zuordnungsproblem tritt in der Diskreten Optimierung und im Operations Research an vielen Stellen auf – häufig mit Variationen der Fragestellung oder weiteren Nebenbedingungen, häufig auch als Teilproblem wesentlich komplexerer praktischer Fragestellungen. So werden Algorithmen zur Lösung des Zuordnungsproblems nicht nur bei der Personaleinsatzplanung sondern auch bei einigen Verfahren zur Lösung des Travelling-Salesman-Problems eingesetzt oder bei der Umlaufplanung von Bussen im öffentlichen Nahverkehr.

Noch Mitte des letzten Jahrhunderts haben Mathematiker derartige „kombinatorische Aufgaben“ als trivial erachtet. Diese haben nur endlich viele Lösungen, und eine beste kann daher im Prinzip durch Enumeration aller Lösungen bestimmt werden. Das Zuordnungsproblem hat, wenn n die Anzahl der Elemente von U (und damit auch von V) ist, genau n! Lösungen. Dabei bezeichnet n! das Produkt der ganzen Zahlen 1, 2, 3,…, n-1, n. Der Wert n! wächst ungeheuer schnell, und selbst die größten heutigen Rechner können die Werte von allen n! Zuordnungen für n=25 (das sind 15.511.210.043.330.985.984.000.000) nicht in unserer Lebenszeit bestimmen. Die Behauptung, man könne solche Probleme durch Enumeration lösen, ist theoretisch richtig, aber zeugt von geringem Realitätssinn.

Mathematische Optimierung, Operations Research und die grundsätzliche Methodik zur Lösung von mathematischen Problemen mit Computern nahmen in den 1950er Jahren ihren wirklichen Anfang. Es wurde klar, dass man, um Rechner zu programmieren, Algorithmen klar formulieren und ihre Laufzeit abschätzen muss. Dies war die Zeit, in der viele der heutigen „Standardmethoden“ entstanden, so auch z.B. verschiedene Algorithmen zur Bestimmung kürzester Wege in Graphen.[2]

2 Schrijver 2003, S. 103ff.

Eines der Probleme, die seinerzeit intensiv untersucht wurden, war das Zuordnungsproblem. Wer hat nun, ist unsere Frage, den ersten Algorithmus zur Lösung des Zuordnungsproblems gefunden?

Klar, vollständige Enumeration ist eine Lösungsmethode, aber wir wollen etwas Besseres. Bei Schrijver findet sich hierzu ein umfassender Überblick.[3] Er beginnt im Jahre 1784, in dem Gaspard Monge sich erstmals in einer Publikation mit dem Zuordnungsproblem beschäftigt. Man muss sich allerdings Mühe geben, den Zusammenhang mit dem Zuordnungsproblem zu sehen, da Monge sein Problem in der Sprache der kontinuierlichen Mathematik darstellt. Im Jahre 1928 stellt Appell fest, das das Verfahren von Monge fehlerhaft ist. Der nach Schrijver vermutlich erste veröffentlichte und korrekte Algorithmus für das Zuordnungsproblem stammt von Easterfield aus dem Jahre 1946.[4] Easterfield kannte offenbar die seinerzeit vorhandene Literatur nicht. Man kann das daraus erschließen, dass er in seinem Artikel Resultate bewies, die aus bereits bekannten Sätzen folgen. Sein Algorithmus hat eine Laufzeit von $O(2^n n^2)$, eine Beschleunigung gegenüber O(n!) Rechenschritten, aber immer noch exponentiell.

In seinen Erinnerungen schreibt Harold Kuhn, dass im Jahre 1953, als er begann, sich mit dem Zuordnungsproblem zu beschäftigen, kein Algorithmus und kein Computer in der Lage war, ein 10x10-Zuordnungsproblem zu lösen.[5] Er fand dann ungarische Literatur zum Matchingproblem in bipartiten Graphen von D. König und J. Egerváry, lernte ein wenig ungarisch, um diese lesen zu können, und entdeckte, dass konstruktive Beweisargumente von Egerváry aus dem Jahre 1931 dazu benutzt werden können, um einen kombinatorischen Algorithmus für das Zuordnungsproblem zu formulieren. Da Kuhn die Vorleistungen der ungarischen Kollegen König und insbesondere Egerváry anerkennen wollte, nannte er sein Verfahren *ungarischer Algorithmus*. Dies ist eine schöne Geste, denn Kuhn hätte mühelos verschleiern können, dass er sich auf Ideen aus Artikeln in ungarischer Sprache stützt. Sein Artikel zur ungarischen Methode,[6] wurde im Jahre 2004 von der Zeitschrift *Naval Logistics Quarterly* als wichtigstes Paper gewählt, das seit Bestehen des Journals in diesem erschienen ist. Der Einfluss des Artikels auf die Entwicklung der kombinatorischen Optimierung ist in der Tat enorm. Dies wird u.a. durch Frank gewürdigt.[7] Kuhn hat 1955 die Laufzeit des ungarischen Algorithmus nicht analysiert, er konstatierte lediglich, dass das Verfahren endlich sei. Munkres gab eine erste Laufzeitabschätzung mit $O(n^3)$ an, übersah jedoch die genaue Abschätzung eines einfachen Details, so dass seine Abschätzung tatsächlich $O(n^4)$ ergab.[8] Auch das Munkres-Paper hatte großen Einfluss auf die Untersuchung kombinatorischer Algorithmen, da es nun üblich wurde, die Rechenschritte genau zu zählen und Laufzeitgrößenordnungen zu betrachten. Man kann diesen

3 Ebd., S. 103ff.
4 Ebd.
5 Kuhn 1991.
6 Kuhn 1955.
7 Frank 2005.
8 Munkres 1957.

Artikel zu einem Vorboten der Komplexitätstheorie (polynomiale Laufzeit, $\mathcal{NP}$, $\mathcal{NP}$-Vollständigkeit, etc.) zählen.

Und nun kommt die Überraschung. Am 9. März 2006 erhielt ich die folgende E-Mail von Harold Kuhn, mit dem zusammen ich 2005 in Budapest am 50. Geburtstagsfest der ungarischen Methode teilgenommen hatte:[9]

> Dear Friends:
>
> As participants in the 50th Birthday celebration of the Hungarian Method, you should be among the first to know that `Jacobi` discovered an algorithm that includes both Koenig's Theorem and the Egervary step. I was told about Jacobi's paper by Francois Ollivier who has a website with the original papers and French and English translations. They were published in Latin after his death and so the work was done prior to 1851!!!
> I have discovered the exact relation between Jacobi's algorithm and the Hungarian Method. Ollivier and I plan to submit a note to the Naval Logistics Research journal, he describing the background in systems of differential equations that led `Jacobi` to the Assignment Problem and I describing the algorithmic details.
> I find this development exciting and am not at all unhappy to have found a result that `Jacobi` had found over a hundred years earlier. I am a (mathematical) descendant of `Jacobi`, and Ollivier, Egervary, and I have a common (mathematical) ancestor.
>
> With warm regards to all of you,
> Harold

Carl Gustav Jacob Jacobi (1804-1851) war einer der produktivsten Mathematiker des 19. Jahrhunderts und ist allen heutigen Mathematikern z.B. durch die Jacobi-Matrix bekannt. Jacobi hat sich aber gar nicht mit dem Zuordnungsproblem beschäftigt und schien sein Verfahren nicht für besonders bemerkenswert zu halten. Ich zitiere einige Zeilen aus dem Artikel von Ollivier:[10]

> „Jacobi himself is possibly the first to have forgotten his own work. […] his manuscripts on this subject were written around 1836 and were intended to be a part of a forsaken project of a long paper on differential equations. […] In fact these results are a by-product of his work on the isoperimetric problem[…]."

Dieses Beispiel zeigt die wirkliche Schwierigkeit der Algorithmengeschichte. Zu Jacobis Zeiten gab es die mathematische Disziplin *kombinatorische Optimierung* überhaupt noch nicht. Das Zuordnungsproblem war als „Begriff" genauso wenig bekannt wie die Technik der Algorithmenanalyse. Jacobi wollte lediglich ein gewisses Problem für Matrizen lösen, das im Rahmen seiner Untersuchungen über Differentialgleichungen „nebenbei" auftrat. Seine Lösung des „Matrixproblems" war ein Beweis, der selbst wiederum (aus unserer heutigen Sicht) ein Algorithmus war. Dieses hat Jacobi vermutlich nicht einmal selbst bemerkt oder interessiert. Und welcher Diskrete Mathematiker der heutigen Zeit, der sich für den ungarischen Algorithmus speziell oder Algorithmengeschichte allgemein interessiert, liest Ar-

9 Im Internet findet man weitere Informationen hierzu unter http://www.lix.polytechnique.fr/~ollivier/JACOBI/jacobiEngl.htm? und http://www.sci.ccny.cuny.edu/~ksda/DART-AMS/Ollivier-DART.pdf.

10 Siehe http://www.sci.ccny.cuny.edu/~ksda/DART-AMS/Ollivier-DART.pdf.

tikel über Differentialgleichungen aus dem 19. Jahrhundert in gesammelten Werken im Detail – und dann noch in lateinischer Sprache? Es ist wirklich reiner Zufall, dass der Zusammenhang dieser „Hilfskonstruktion“ mit dem ungarischen Algorithmus entdeckt wurde.

3. GRAPHENALGORITHMEN

Das Thema „Graphenalgorithmen“ wird heute von der Informatik als Teilgebiet dieser Wissenschaftsdisziplin reklamiert, während Graphentheorie als Unterdisziplin der Diskreten Mathematik angesehen wird. Ein Grund dafür ist, dass viele der „klassischen Graphentheoretiker“ sich selbst nach dem Aufblühen des „algorithmischen Standpunktes“ nicht für Algorithmen interessiert haben sondern sich weiterhin (nur) für die Formulierung „schöner Sätze“ begeistern konnten. Dabei haben Graphentheoretiker „unendlich viele“ Algorithmen produziert, sie haben diese jedoch nicht als solche formuliert sondern nur als Beweisargumente benutzt. Zum Beispiel haben fast alle Beweise durch vollständige Induktion algorithmischen Charakter und können auch direkt als Algorithmen formuliert werden – meistens jedoch nicht so kurz und elegant. Gleiches gilt für Beweise, bei denen man minimale oder maximale Gegenbeispiele annimmt und dann durch Konstruktionen zeigt, dass die Existenz dieser Gegenbeispiele im Widerspruch zu den Bedingungen eines Satzes steht. Man kann niemandem vorwerfen, die „impliziten Algorithmen“ nicht explizit darzustellen, aber die Nichterwähnung des algorithmischen Aspekts eines Resultats führt mit Sicherheit dazu, dass einige Juwelen des Algorithmendesigns unbekannt bleiben oder später in anderen Zusammenhängen mehrfach wiederentdeckt werden. Ich möchte hierzu ein Beispiel geben.

Jedes Buch über Graphentheorie enthält einen Abschnitt über hamiltonsche Graphen. Ein *Weg* der Länge k-1 in einem Graphen mit n Knoten ist eine Folge von voneinander verschiedenen Knoten $v_1, v_2, \ldots, v_{k-1}, v_k$, so dass die Knoten v_i und v_{i+1} jeweils durch eine Kante verbunden sind, $1 \leq i \leq k-1$. Gibt es zusätzlich noch eine Kante, welche die Knoten v_1 und v_k verbindet, so liegt ein *Kreis* der Länge k vor. Kreise der Länge n heißen *hamiltonsch* – nach Sir William Rowan Hamilton(1805-1865), der sich 1856 erstmals mit dem Auffinden solcher Kreise beschäftigt hat. Graphen mit $n \geq 3$ Knoten, die einen hamiltonschen Kreis enthalten, heißen *hamiltonsch*. Eines der Ziele der Theorie ist das Auffinden von notwendigen und/oder hinreichenden Bedingungen für die Existenz hamiltonscher Kreise in einem Graphen.

Eine einfache hinreichende Bedingung für die Existenz hamiltonscher Kreise hat G.A. Dirac angegeben.[11] Dirac bezieht sich dabei auf die Grade der Knoten des Graphen. Dies ist ein trivial zu berechnender Parameter, während das Feststellen der Existenz eines hamiltonschen Kreises schwierig ($\mathcal{NP}$-vollständig) ist, was Dirac seinerzeit nicht wusste, weil es die Komplexitätstheorie noch nicht gab. Der *Grad* eines Knoten v ist die Anzahl der Kanten, die v enthalten.

11 Dirac 1952.

Satz (Dirac). Ist G ein Graph ohne parallele Kanten mit $n \geq 3$ Knoten und hat jeder Knoten mindestens den Grad n/2, so ist G hamiltonsch.

Hier ist ein breit ausgewalzter Beweis, den man in ähnlicher Form in vielen Graphentheoriebüchern findet. Man wähle einen beliebigen Knoten v des Graphen. Dann gehe man zu einem Nachbarn w von v (also zu einen Knoten w, der mit v durch eine Kante verbunden ist) und von w zu einem weiteren bisher noch nicht besuchten Nachbarn x von w. Dieses Aufsuchen eines noch nicht besuchten Nachbarn eines gegenwärtigen Knotens führe man so lange fort, bis der gegenwärtig betrachtete Knoten, sagen wir z, keinen Nachbarn mehr hat, der nicht bereits auf dem Weg von v nach z liegt. Dann gehe man zurück zu v und versuche von v aus den Weg auf die gleiche Weise „in die andere Richtung" zu verlängern. Am Ende dieses Suchprozesses hat man einen Weg $W = v_1, v_2, \ldots, v_k$, so dass weder v_1 noch v_k einen Nachbarn haben, der nicht auf diesem Weg liegt. Sind v_1 und v_k durch eine Kante verbunden, so haben wir einen Kreis der Länge k gefunden. Falls diese Kante nicht vorhanden ist, definieren wir die Mengen $N_1 = \{v_i | v_{i+1}$ ist Nachbar von $v_1\}$ und $N_k = \{v_i | v_i$ ist Nachbar von $v_k\}$. Falls im Durchschnitt von N_1 und N_k ein Knoten, sagen wir v_j, liegt, so repräsentiert nach Konstruktion die Knotenfolge $v_1, v_2, \ldots, v_j, v_k, v_{k-1}, \ldots, v_{j+1}$ einen Kreis der Länge k. Nehmen wir an, dass im Durchschnitt von N_1 und N_k kein Knoten liegt, so hat die Vereinigung von N_1 und N_k die Kardinalität n, da sowohl N_1 als auch N_k mindestens die Kardinalität n/2 haben. Aber der Knoten v_k liegt nicht in dieser Vereinigung, also hat die Vereinigung höchstens n-1 Elemente, ein Widerspruch! Folglich haben wir (möglicherweise nach Umnummerierung) einen Kreis $K = v_1, v_2, \ldots, v_k$ der Länge k finden können, und offensichtlich gilt $k \geq n/2+1$.

Gibt es in G einen Knoten w, der nicht in K enthalten ist, so muss w, da der Grad von w mindestens n/2 ist, mindestens einen Nachbarn v_s in K haben. Die Folge $w, v_s, v_{s+1}, \ldots, v_k, v_1, v_2, \ldots, v_{s-1}$ repräsentiert dann einen Weg der Länge k+1. Wir iterieren nun die vorhergehende Argumentation und zeigen, dass dann ein Kreis der Länge k+1 existieren muss. Dies führen wir so lange fort, bis wir einen Kreis der Länge n konstruiert haben. Dies beendet den Beweis.

Wenn man nicht an der expliziten Konstruktion des Kreises der Länge n interessiert ist, kann man die Argumentation erheblich verkürzen. Der entscheidende Punkt ist hier jedoch, dass der Beweis eine Folge von Konstruktionen beschreibt, die z.B. bei Heuristiken für das *Travelling-Salesman-Problem* (TSP) in unterschiedlicher Form verwendet werden. Das TSP ist „das klassische" kombinatorische Optimierungsproblem. Hier wird in einem vollständigen Graphen (jeder Knoten ist mit jedem anderen durch eine Kante verbunden), bei dem für jede Kante eine „Länge" gegeben ist, nach einem hamiltonschen Kreis kürzester Gesamtlänge gesucht.

In der obigen Beweisführung wird zunächst ein langer Weg konstruiert (das ist eine Version der *Nächster-Nachbar-Heuristik* des TSP), dann wird daraus durch Austauschargumente ein Kreis gemacht (TSP-Austauschverfahren wie *Cheapest Insert*), danach wird der Weg und anschließend der Kreis durch Einfügungsoperationen verlängert (TSP-*Insertion-Heuristiken*). Aus algorithmischer

Sicht ist die Basis des Beweises nichts anderes als eine Heuristik für das hamiltonsche Graphenproblem. Der Beweis selbst ist eine Analyse der Heuristik und zeigt, dass die Heuristik, wenn die Gradvoraussetzung des Satzes erfüllt ist, immer einen hamiltonschen Kreis liefert. Heuristik-Analyse gab es also schon lange bevor dieses Wort im Bereich des Operations Research und der Theoretischen Informatik erfunden wurde.

Der Satz von Dirac hat unzählige Verfeinerungen und Verallgemeinerungen erfahren. Aus der hier geschilderten Sicht kann man diese ganz schlicht als Verbesserungen der Algorithmen (neue „Austauschtricks“ wurden z.B. eingeführt und iteriert) und detailliertere Heuristik-Analysen bezeichnen, die dann auch zu allgemeineren Sätzen geführt haben.

Wenn ich behaupte, dass 90% der Ergebnisse der Graphentheorie so wie hier interpretiert werden können (Graphentheorie als Analyse von heuristischen Algorithmen), werden mich Graphentheorie-Puristen vermutlich steinigen. Ich bin jedoch davon überzeugt, dass eine Durchforstung der Graphentheorie-Literatur hervorbringen wird, dass viele der gängigen Graphenalgorithmen bereits implizit in den Beweisen der Graphentheorie vorhanden waren. Was die Sichtweisen jedoch unterscheidet, ist, dass Graphentheoretiker auf die Formulierung eleganter Sätze abzielen, während Informatiker und Operations Researcher mehr Wert auf Datenstrukturen, Laufzeitanalysen und Worst-Case-Abschätzungen legen.

4. LINEARE OPTIMIERUNG

Über die Geschichte der Methoden zur Lösung linearer Gleichungen gibt es sehr viele Untersuchungen.[12] Dagegen scheinen sich Mathematiker viel weniger mit Ungleichungen und insbesondere mit dem Themengebiet, das wir heute *lineare Optimierung* oder *lineare Programmierung* (kurz LP) nennen, beschäftigt zu haben, obwohl LP und Polyedertheorie eng miteinander verknüpft sind und Polyeder in der Geschichte der Mathematik eine wichtige Rolle gespielt haben. Der Artikel von Grattan-Guiness gibt eine Übersicht über die Frühzeit der linearen Optimierung.[13] Die beste Zusammenfassung der Theorie und der Geschichte der linearen Optimierung und der damit verbundenen Gebiete der Polyedertheorie findet man bei Schrijver.[14] Ich möchte hier nur ein paar ergänzende Bemerkungen machen, die einige Aspekte der geschichtlichen Entwicklung erhellen.

Ein lineares Program ist eine Optimierungsaufgabe der Form max $c^T x$ (oder min $c^T x$) unter der Nebenbedingung, dass $Ax \leq b$ gilt. Hierbei sind c und b gegebene reelle Vektoren, A eine gegebene reelle Matrix und x ein Variablenvektor (alle von jeweils passender endlicher Dimension). Gesucht ist also das Maximum oder Minimum einer linearen Funktion über einem Polyeder.

12 Siehe hierzu z.B. Chabert 1999, Kap. 9.
13 Grattan-Guiness 1994.
14 Schrijver 1986, siehe u.a. S. 209ff.

Einer der ersten, der die Bedeutung von Ungleichungen für Anwendungen der Mathematik erkannt hatte, war Jean Baptiste Joseph Fourier (1786-1830).[15] Fourier hat im Jahre 1827 eine Variableneliminationsmethode beschrieben, mit der man prüfen kann, ob ein gegebenes Ungleichungssystem $Ax \leq b$ eine Lösung hat oder nicht. Dieses Verfahren wird heute *Fourier-Motzkin-Methode* genannt. Motzkin hatte die Fourier-Methode 1936 wiederentdeckt und zu verschiedenen Zwecken eingesetzt. Durch Einführung einer Zusatzvariablen x_{n+1} und der beiden Ungleichungen $c^T x \leq x_{n+1} \leq c^T x$ kann man aus jedem LP ein Ungleichungssystem machen. Eliminiert man alle Variablen dieses neuen Systems bis auf die Zusatzvariable x_{n+1}, so liefert die größte untere Schranke für x_{n+1} den Minimalwert und die kleinste obere Schranke für x_{n+1} den Maximalwert des linearen Optimierungsproblems. Fourier hat also im Prinzip bereits einen Algorithmus zur Lösung linearer Programme erfunden. Wie sich später herausstellte, ist Fourier-Motzkin-Elimination ein inhärent exponentielles Verfahren und zur Lösung von LPs nicht praxistauglich.

Fourier beschrieb außerdem in den Jahren 1826 und 1827 eine rudimentäre Version des *Simplex-Algorithmus* für den dreidimensionalen Raum. Er erläuterte, wie man von Ecke zu Ecke eines Polyeders geht, bis man ein Optimum gefunden hat, und stellte fest, dass damit klar sein müsste, wie diese Methode in Räumen beliebiger Dimension funktioniert. In der Tat, dies ist die geometrische Version des Simplex-Algorithmus, wie sie in jeder Vorlesung zur linearen Optimierung zu Beginn der Darstellung dieses Verfahrens beschrieben wird. Ist also Fourier der Erfinder des Simplex-Verfahrens?

Habe ich im vorhergehenden Abschnitt argumentiert, dass viele Algorithmen, die heutzutage entdeckt werden, vermutlich implizit schon bekannt waren und in vielen Fällen die Entdeckung anderen zugeschrieben werden müsste, so will ich nun genau das Umgekehrte tun. Zwar ist korrekt, dass Fourier die Idee der „Wanderung über die Ecken“ hatte. Aber in diesem Falle ist der Weg von der grundsätzlichen Idee zu einem effizienten Algorithmus noch sehr weit. Eine implementierbare und praxistaugliche Version fand George B. Dantzig, der ab 1947 verschiedene Versionen des Simplex-Algorithmus entwickelte. Dantzig übersetzte das geometrische Konzept „Ecke“ in eine algorithmisch handhabbare „Basislösung“ und zeigte, wie man mit der Inversion der zugehörigen „Basis“ so genannte „reduzierte Kosten“ bestimmen kann, um von einer Basislösung zu einer besseren zu gelangen. Auch von diesem Algorithmus kann man bislang nicht zeigen, dass er eine polynomiale Laufzeit hat. Er ist aber empirisch sehr effizient und bis heute das wesentliche Arbeitspferd der linearen Programmierung. Erst diese Transformation von Geometrie in lineare Algebra macht aus der Idee des Simplex-Algorithmus ein brauchbares Rechenverfahren. Deswegen wird George B. Dantzig von allen Optimierern als Vater der linearen Programmierung bezeichnet.

Dies haben nicht alle so gesehen. Als im Jahre 1975 die Ökonomie-Nobelpreise für Beiträge zur linearen Programmierung, genauer für „contributions to the theory of optimum allocation of resources“, vergeben wurden, gingen die Preise

15 Siehe hierzu u.a. Grattan-Guiness 1994.

(nicht unverdient) an L.V. Kantorovich und T.C. Koopmans. Dantzig wurde aber „übersehen“, obwohl sein Simplex-Algorithmus der Ökonomie seit den fünfziger Jahren unschätzbare Dienste geleistet hat und unsere heutige Technologie ohne diese Methodik nicht denkbar wäre.

Die algorithmenhistorisch interessante Frage ist in diesem Fall, wer hat einen wichtigen Beitrag geleistet und wann ist eine algorithmische Idee wirklich schon ein Algorithmus? Das ist schwer zu entscheiden. Im vorliegenden Falle habe ich eine klare persönliche Meinung, aber andere können das durchaus anders sehen, zumal dann, wenn die Quellenlage undurchsichtig ist.

Ich erläutere kurz noch zwei weitere, ähnlich gelagerte Fälle im Bereich der linearen Optimierung: Im Jahre 1979 hat Leonid G. Khachiyan (1952-2005) in einem ganz kurzen Artikel bewiesen, dass lineare Programme in polynomialer Zeit gelöst werden können.[16] Das war ein großer Durchbruch, der weltweit in der öffentlichen Presse gewürdigt wurde. Khachiyan hat dazu die *Ellipsoidmethode* benutzt, die auf Arbeiten von N.Z. Shor (1937-2006) aus dem Jahre 1970 zur nichtlinearen Optimierung zurückgeht und von D.B. Yudin und A.S. Nemirovskii 1976 weiterentwickelt wurde. Khachiyan hat die Ellipsoidmethode nicht verändert, er hat nur einige Parameter „richtig“ eingestellt und mit zahlentheoretischen Überlegungen und Volumenschrumpfungsargumenten zeigen können, dass diese Methode lineare Programme in polynomialer Zeit löst. Das war eine völlig überraschende und neue Vorgehensweise bei der Analyse von LP-Algorithmen. Zweifelsohne stammt die Ellipsoidmethode von Shor, ihre (theoretische) Bedeutung hat sie allerdings erst durch die (geringfügige) Modifikation und die außergewöhnliche Analyse von Khachiyan erhalten. Leider hat sich die Ellipsoidmethode in der Praxis nicht bewährt.

Die Ellipsoidmethode hat sich jedoch als mächtiges Beweiswerkzeug erwiesen, mit dem viele neue polynomiale Algorithmen für geometrische, kombinatorische und andere Probleme entwickelt werden konnten. Sie hat insbesondere zu einem tieferen Verständnis von Schnittebenenverfahren (Schlagwort: polynomiale Äquivalenz von Optimierung und Separierung) geführt.[17]

Ein weiterer, aber anders gelagerter Fall ist der folgende. Im Jahre 1984 hat Narendra Karmarkar spektakuläre Erfolge mit einer *Innere-Punkte-Methode* zur Lösung linearer Programme angekündigt und behauptet, dass dieser Algorithmus in der Praxis schneller sei als die Simplex-Methode.[18] Karmarkar konnte mit eleganter Beweistechnik die Polynomialität seines Verfahrens zeigen, niemand aber konnte mit seiner Methode unabhängig die behaupteten Rechenergebnisse nachvollziehen. Karmarkar trat durch die Nichtnachvollziehbarkeit seiner Rechenergebnisse und die Versuche anderer, die Methodik besser zu verstehen, eine Lawine zur Entwicklung weiterer Innere-Punkte-Methoden los. So wurden dann die verschiedensten Ansätze der nichtlinearen Optimierung (durchaus erfolgreich) eingesetzt, um neue Verfahren zur Lösung linearer Programme zu entwerfen. Ei-

16 Siehe Khachiyan 1979 und zu weiteren Details Grötschel/Lovász/Schrijver 1988.
17 Siehe Grötschel/Lovász/Schrijver 1988.
18 Siehe Karmarkar 1984.

nige davon sind heute der altbewährten Simplex-Methode in verschiedenen Anwendungsbereichen deutlich überlegen. Mehrere Optimierer, darunter meine Kollegen Lustig, Marsten and Shanno, gründeten sogar Firmen, um ihre neuen Innere-Punkte-LP-Codes kommerziell zu vermarkten. Sie waren mehr als erstaunt, als sie von Anwälten von AT&T erfuhren, dass sie damit Patente von AT&T verletzen. Sie wurden aufgefordert, den Verkauf der Programme einzustellen oder AT&T angemessene Lizenzgebühren zu bezahlen. Karmarkar arbeitete seinerzeit bei den AT&T Bell Laboratories, und AT&T hatte sich 1988, von Universitätsmathematikern unbemerkt, weitgehende Patente einräumen lassen, die alle Verfahren abdecken, die lineare Programme dadurch lösen, dass sie durch das Innere der Lösungsmenge vorangehen.[19] Inzwischen hatte sich aber herausgestellt, dass Karmarkars „projektive Innere-Punkte-Methode“ äquivalent zu einem „projected Newton barrier“ Algorithmus ist. Dieser war u.a. in dem Buch von Fiacco und McCormick dargestellt worden.[20] Das Besondere an dem Buch ist, das in ihm die Anwendung dieser Methode auf lineare Programme beschrieben ist. Diesem Buch lag sogar ein Fortran-Code bei, und dessen Parameter-Einstellung war so gewählt, dass er als erster implementierter LP-Algorithmus mit polynomialer Laufzeit anzusehen ist. Lustig, Marsten und Shanno glaubten, damit das AT&T-Patent zu Fall bringen zu können, waren aber finanziell und rechtstechnisch nicht in der Lage, die Patentanfechtungsklage durchzustehen. Sie lösten ihre Firma aufgrund der enormen finanziellen Risiken, die die Drohungen von AT&T nach sich zogen, auf.

Was lernen wir daraus? Algorithmen werden anscheinend auch heute noch in kurzen Abständen wiederentdeckt. Die Literatur ist so umfangreich, dass man einfach nicht alles, selbst in nahen Fachgebieten, kennt. Wem aber gebührt nun der Ruhm für die Entdeckung der Inneren-Punkte-Methoden zur Lösung linearer Programme? Ganz offensichtlich gab es auch vor Karmarkar schon solche Verfahren. Das Problem war, dass niemand sie wirklich ernsthaft für den praktischen Einsatz in Betracht zog oder bei ersten Versuchen Schiffbruch erlitt. Karmarkars Leistung war unzweifelhaft, dass er eine Idee hatte, wie man mit Hilfe von Potentialfunktionen die polynomiale Laufzeit solcher Algorithmen nachweisen konnte. Fiacco und McCormick hatten bereits ein solches Verfahren implementiert, sie wussten aber nicht, dass es eine polynomiale Laufzeit hat und haben wohl selbst nicht recht geglaubt, dass dieser Ansatz sich zu einem Rivalen des Simplex-Algorithmus entwickeln würde.

Was aber ist mit der Priorität? Hier könnten wir den legalen Weg einschlagen und einfach feststellen, dass Karmarkar der erste war, der Innere-Punkte-Methoden für LPs erfunden hat, denn schließlich hat er ja ein Patent erhalten. Das wurde zwar patentrechtlich durchgesetzt, ist aber historisch falsch. Was wird ein Forscher in 500 Jahren davon noch wissen?

19 Z.B. United States Patent 4744028, Karmarkar, May 10, 1988.
20 Fiacco/McCormick 1968.

5. WAHLVERFAHREN

Ich komme nun zu einem Thema, das in der Algorithmengeschichte bisher kaum Berücksichtigung fand. In allen Kulturen und zu allen Zeiten wurde „abgestimmt". Natürlich wurden und werden Führungsrollen nicht selten durch „Gewalt" erobert, aber die Geschichte kennt wichtige Dokumente über die Durchführung von Wahlen (z.B die Goldene Bulle), so dass klar ist, dass Wahlverfahren zu jeder Zeit große Aufmerksamkeit gewidmet wurde. Für unsere Zeit gilt das ganz besonders, denn Demokratien fußen auf Wahlverfahren. Bei der Diskussion von Wahlverfahren steht in der Regel das Thema „Gerechtigkeit eines Verfahrens" im Vordergrund. Hier möchte ich jedoch auf algorithmische Aspekte und einen besonderen geschichtlichen Aspekt eingehen, nämlich die Benamung von Wahlverfahren.

Eine sorgfältige Analyse aller möglichen Wahlsysteme würde den Rahmen dieses Artikels sprengen. Ich will mich daher auf ein „ganz einfaches" Spezialproblem konzentrieren: Sitzzuteilungsverfahren. Und auch hier werde ich nicht auf die enorme Vielfalt von Sonderregeln (Schwellwerte wie die 5%-Hürde, Vermischen von Mehrheits- und Verhältniswahl, Regionallisten und Überhangmandate, etc.) eingehen; sondern lediglich über die „Basisversionen" berichten.

Sitzzuteilungsverfahren sind Methoden, die Wählerstimmen in Abgeordnetenmandate umrechnen. Aus mathematischer Sicht kann man eine Sitzzuteilung als eine Funktion auffassen. Gegeben sind hierbei eine positive ganze Zahl k (P_1, …, P_k sind die zur Wahl zugelassenen k Parteien), eine positive ganze Zahl w (Anzahl aller gültigen Stimmen), und eine positive ganze Zahl s (s ist die Anzahl der zu vergebenden Sitze in einem Parlament, wir gehen hier der einfachen Darstellung halber von einer fest vorgegebenen Sitzzahl aus). Eine *Sitzzuteilung* ist dann eine Funktion, die jedem Vektor (w_1, …, w_k) von k nicht-negativen ganzen Zahlen mit der Eigenschaft $w = w_1+\ldots+w_k$ (w_i ist die Anzahl der für Partei i abgegebenen gültigen Stimmen) einen Vektor (s_1, …, s_k) von nicht-negativen ganzen Zahlen mit der Eigenschaft $s = s_1+\ldots+s_k$ zuordnet (s_i ist die Anzahl der Sitze, die Partei i erhält). Ein *Sitzzuteilungsverfahren* ist ein Algorithmus, der eine Sitzzuteilung aus den gegebenen Daten berechnet.

Hier mache ich einen vielleicht haarspalterisch erscheinenden Unterschied zwischen einer Sitzzuteilung und einem Sitzzuteilungsverfahren, der bei der Diskussion von Wahlverfahren meistens nicht betrachtet wird. Der wesentliche Punkt ist, dass zwei verschiedene Sitzzuteilungsverfahren zur gleichen Sitzzuteilung führen können. Wir bezeichnen zwei Sitzzuteilungsverfahren als *äquivalent*, wenn sie bei allen möglichen Eingangsdaten k, w, s, w_1, …, w_k immer zu derselben Sitzzuteilung s_1, …, s_k führen.

Der Grund dafür, dieses Thema hier anzusprechen ist, dass immer wieder „neue" Sitzzuteilungsverfahren erfunden werden. Es dauert dann meistens einige Zeit, bis man herausfindet, dass sie gar nicht neu sind, sondern dass eine andere Person dasselbe Verfahren (vielleicht mit etwas anderen Worten) schon einmal vorgeschlagen hat. Manchmal sieht das Verfahren neu aus, aber nach genauer mathematischer Analyse des vorgeschlagenen Algorithmus stellt sich heraus, dass dieses neue Verfahren immer dieselbe Sitzverteilung ergibt wie ein bereits bekanntes Ver-

fahren. Das neue Verfahren ist also äquivalent zu einem bekannten Verfahren, nur die Vorschriften zur Berechnung des Ergebnisses sind etwas anders organisiert. Hier sind zwei Beispiele für so etwas. Jeder kennt (vermutlich noch) besondere Alternativrechenregeln aus der Schule wie die Formel zur Berechnung des Quadrats einer ganzen Zahl mit Endziffer 5. Computerarithmetik kann u.a. dadurch beschleunigt werden, dass man die Multiplikation einer binär codierten Zahl mit einer Zweierpotenz durch eine Left-Shift-Operation implementiert.

Natürlich ist es viel wichtiger, die Eigenschaften von Sitzzuteilungsverfahren im Hinblick auf „natürliche Gerechtigkeitskriterien“ zu untersuchen. Das Problem dabei ist, dass man zwar eine wunderbare Liste von erwünschten Eigenschaften aufstellen kann, die ein gerechtes Wahlverfahren besitzen soll, dass man aber (so gut wie immer) zeigen kann, dass es kein Wahlverfahren gibt, das alle diese Eigenschaften hat. Ein sehr schönes Buch, in dem dieser Sachverhalt ausführlich dargelegt wird, ist das von Balinski und Young.[21] Die Diskussion hierzu findet permanent überall auf der Welt statt. Bürger klagen häufig, weil sie Ungerechtigkeiten erkennen; ihnen ist jedoch meistens nicht klar, dass die Beseitigung einer Ungerechtigkeit eine Ungerechtigkeit an anderer Stelle hervorruft. Weil man diese „Paradoxien“ inzwischen besser versteht und Mathematiker Politiker intensiver beraten, werden Verfahren mit größerem Verständnis für die „Nebenwirkungen“ ausgewählt. In Deutschland haben von den vielen existierenden Verfahren drei Eingang in die Wahlgesetzgebung gefunden: das D’Hondt-Verfahren, das Hare/Niemeyer-Verfahren und das Sainte-Laguë/Schepers-Verfahren.[22] Auf der Webseite des Deutschen Bundestags findet man Links zu ausführlichen Erläuterungen dieser Verfahren.[23] Viele Experten sind heutzutage der Ansicht, dass das Sainte-Laguë/Schepers-Verfahren die Idee der Wahlgleichheit bestmöglich erfüllt. Auf Pukelsheims Homepage sind mehrere Dokumente zur Wahlgesetzgebung und zu seiner Beratungstätigkeit hierzu zu finden.[24] Bei den Landtagswahlen des Jahres 2008 in Deutschland fanden folgende Verfahren Anwendung: Bayern und Hessen (Hare/Niemeyer-Verfahren), Hamburg (Sainte-Laguë/Schepers-Verfahren), Niedersachsen (D’Hondt-Verfahren).

Mein Anliegen ist das folgende. Es wäre wunderbar, wenn einmal herausgefunden werden könnte, wer welches Sitzzuteilungsverfahren zuerst entdeckt hat, welche Sitzzuteilungsverfahren im oben definierten mathematischen Sinne äquivalent sind und welche Namen in welchen Ländern für welche Verfahren verwendet werden. Natürlich gibt es dazu schon Vorarbeiten, ich bin jedoch sicher, dass sorgfältige historische Arbeit einige neue Sachverhalte zutage fördern wird. Dazu wäre es schön, die jeweiligen Eigenschaften (Vor- und Nachteile) aufzulisten.[25]

21 Balinski/Young 2001.
22 Hierzu gibt Pukelsheim 2002 einen Überblick.
23 http://www.bundestag.de/ausschuesse/azur/index.html.
24 http://www.math.uni-augsburg.de/stochastik/pukelsheim/.
25 Bei Wikipedia und auf anderen Internetseiten ist eine Fülle von solchen Informationen zu finden.

Ich gebe einige Beispiele zur Begriffsverwirrung im Bereich der Wahlverfahren:

Das *D'Hondt-Verfahren* wird im angelsächsischen Raum *Jefferson-Verfahren* genannt und in der Schweiz *Hagenbach-Bischoff-Verfahren*. Die mathematische Bezeichnung ist *Divisorverfahren mit Abrundung*. Das D'Hondt-Verfahren wurde bis einschließlich 1983 zur Berechnung der Sitzverteilung bei Wahlen zum Deutschen Bundestag verwendet.

Das *Sainte-Laguë/Schepers-Verfahren* nennt man im angelsächsischen Raum *Webster-Verfahren*, mathematisch *Divisorverfahren mit Standardrundung*.

Das *Hare-Niemeyer-Verfahren* heißt im angelsächsischen Raum *Hamilton-Verfahren*; Mathematiker bezeichnen es mit *Quotenverfahren mit Restausgleich nach größten Bruchteilen*. Dieses wurde seit der Wahl im Jahr 1987 zur Berechnung der Sitzverteilung im Deutschen Bundestag angewandt. Am 24. Januar 2008 hat der Bundestag beschlossen, das Hare-Niemeyer-Verfahren ab der folgenden Bundestagswahl durch das Sainte-Laguë/Schepers-Verfahren zu ersetzen.

Weitere Sitzzuteilungsverfahren sind das *Adams-Verfahren*, das *Dean-Verfahren*, das *Hill-Huntington-Verfahren* und das *Lowndes-Verfahren*, und das sind bei weitem noch nicht alle. Eine geschichtliche Übersicht über die Probleme mit diesen Verfahren im Wahlsystem der USA findet sich bei Balinski/Young.[26]

Die bisher betrachteten Wahlverfahren sind Methoden der Verhältniswahl, bei denen es darum geht, die Verhältnisse der abgegebenen Stimmen durch die Sitzverteilung im Parlament möglichst gut widerzuspiegeln. Daneben gibt es vielfältige Formen der Mehrheitswahl. Ich will hier nur die *Condorcet-* und die *Borda-Methode* erwähnen. Die Borda-Methode ist bereits 1433 von Nicolaus Cusanus (1401-1464) in seiner Schrift *De concordantia catholica* vorgeschlagen worden. Auch der mallorquinische Philosoph, Logiker und Theologe Raimundus Lullus (1232-1316) hat schon Studien von Wahlverfahren unternommen. Gerade diese Erkenntnisse legen die Vermutung nahe, dass einige der anderen Verfahren wesentlich älteren Ursprungs sind, denn gewählt wurde auch in Athen und Rom, in Klöstern und anderen Gemeinschaften. Vielleicht hat ja irgendjemand irgendwo die dabei verwendeten Algorithmen skizziert?

Der Grund, dass ein und dasselbe Wahlverfahren unterschiedliche Namen hat, liegt daran, dass die Verfahren zu verschiedenen Zeiten in mehreren Ländern unabhängig voneinander (wieder-)entdeckt wurden und aus Lokalstolz ein einmal gewählter Name beibehalten wird. Heute sollte man das eigentlich besser wissen und machen. Dem ist nicht so, wie die nachfolgende Geschichte zeigt.

Im Züricher Tages-Anzeiger vom 11. Februar 2006 erschien ein Artikel von Edgar Schuler mit der Überschrift *Friedrich Pukelsheim, Statistiker und Vater des neuen Wahlsystems in Zürich. Ein Mathematiker macht Politik"*, aus dem ich einige Sätze zitiere:

> „[…]Der deutsche Mathematikprofessor hat die Methode ausgetüftelt, nach der die Wählerstimmen auf die Parteien und Kandidaten verteilt werden. In Zürich muss sich das System nun zum ersten Mal in der Praxis bewähren. […] Dabei ist es dem 57-jährigen Professor eher peinlich, dass die Methode von Regierungsrat Markus Notter kurzerhand zum ‚doppelten Pu-

26 Balinski/Young 2001.

> kelsheim‘ erklärt wurde. Pukelsheim zieht die wissenschaftliche Bezeichnung ‚doppelt-proportionale Divisormethode mit Standardrundung‘ vor. Vor allem weil das Verfahren nicht von ihm erfunden worden sei, sondern in Paris vom Mathematikerkollegen Michel Balinski. Pukelsheim hat die Methode zusammen mit Christian Schuhmacher von der Justizdirektion für Zürich adaptiert. [….]“

Auch wenn der Kollege Pukelsheim nachdrücklich auf die Urheberschaft anderer verwiesen hat, scheint sich, zumindest in Zürich, ein weiterer Name einzubürgern.

6. ZUSAMMENFASSUNG

Der Artikel beschäftigt sich mit Algorithmengeschichte, aber was ein Algorithmus ist, habe ich noch nicht erklärt. Und das ist auch gar nicht so einfach. Ganz vage wird in der Regel definiert: Ein Algorithmus ist eine Handlungsvorschrift zur Lösung eines Problems. Damit sind auch die Anweisung zum Ausfüllen eines Formulars und ein Kochrezept ein Algorithmus. Man kann die Definition präziser angeben, wie das häufig in Büchern zur Informatik und Logik geschieht. Aber dann muss man einen formalen Apparat aufziehen, der den Rahmen dieses Artikels sprengen würde. Ich habe kein Buch und keinen Artikel zur Algorithmengeschichte gefunden, in denen das so formal gemacht wurde. Ich vermute, dass Menschen, seit sie denken können, Algorithmen entworfen und benutzt haben (Anweisungen zur Jagd, zum Anbau von Pflanzen, zur Behandlung von Ware, etc.). Wirklich in den Fokus wissenschaftlicher Beschäftigung sind Algorithmen jedoch erst im 20. Jahrhundert gekommen. Zunächst war Algorithmentheorie durch die bedeutenden Arbeiten von Gödel, Turing, Church, Kleene und anderer eine Teildisziplin der Logik. Mit dem Aufkommen realer Computer und damit der Informatik traten praktische Fragen der Algorithmenanalyse in den Vordergrund, und die sich entwickelnde Komplexitätstheorie ermöglichte ein präziseres Verständnis von Algorithmen und damit zusammenhängenden Fragen. Donald Knuth, einer der großen Informatik-Pioniere, schrieb im Band 1 seiner Buchserie *The Art of Computer Programming*: „By 1950, the word algorithm was most frequently associated with Euclid’s algorithm.“ Das heißt, wenn man heute Algorithmengeschichte betreibt, kann man nicht nach dem Stichwort Algorithmus suchen und hoffen, etwas vor 1950 zu finden. Man muss *tiefer suchen* nämlich sich mit Verständnis für algorithmische Ideen in die Texte vertiefen und versuchen, algorithmische Vorgehensweisen herauszulesen. Das ist eine wirklich schwere Aufgabe, möglicherweise kommen damit aber sehr interessante wissenschaftshistorische Sachverhalte zu Tage.

Tiefensuche (englisch depth-first search) ist aber auch ein in Informatik und Mathematik sehr beliebtes und erfolgreiches Suchverfahren, das insbesondere in der Graphentheorie vielfältige Anwendungen findet und z.B. sehr gute Algorithmen zum Testen von Zusammenhangseigenschaften liefert. Aber das ist nicht Gegenstand dieses Artikels. Wie es im Internet das *Deep Web* gibt, das von den heutigen Suchmaschinen nicht erfasst wird und dessen Datenmenge nach verschiedenen Untersuchungen einige hundert Mal umfangreicher als das *Visible Web* sein

soll, so vermute ich, dass in den Tiefen der Dokumente vergangener Jahrhunderte viele Juwelen der Algorithmengeschichte schlummern, die mit den gegenwärtigen algorithmenhistorischen Suchverfahren einfach nicht erreicht werden. Es gibt also noch viel zu tun.

LITERATUR

Balinski/Young 2001: Balinski, Michel L. u. H. Peyton Young: Fair Representation: Meeting the Ideal of One Man, One Vote, (second edition), Washington.

Chabert 1999: Chabert, Jean-Luc (Hg.): A History of Algorithms: From the Pebble to the Microchip, Berlin.

Dirac 1952: Dirac, Gabriel Andrew: Some theorems on abstract graphs, in: Proceedings of the London Mathematical Society 2, S. 69-81.

Fiacco/McCormick 1968: Fiacco Anthony V. u. Garth P. McCormick: Nonlinear programming sequential unconstrained minimization techniques, erneut aufgelegt als: Classics in Applied Mathematics, Philadelphia, PA, USA, 1990.

Frank 2005: Frank, Andras: On Kuhn's Hungarian method – A tribute from Hungary, in: Naval Research Logistics 52, S. 2-5.

Grattan-Guiness 1994: Grattan-Guiness, Ivor: „A New Type of Question": On the Prehistory of Linear and Non-Linear Programming, 1770-1940, in: Knobloch, E. und D.E. Rowe (Hg.), The History of Modern Mathematics, Bd. III: Images, ideas and communities, Boston, S. 43-89.

Grötschel/Lovász/Schrijver 1988: Grötschel, Martin, László Lovász u. Alexander Schrijver: Geometric Algorithms and Combinatorial Optimization, Berlin.

Karmarkar 1984: Karmarkar, Narendra: A new polynomial-time algorithm for linear programming, in: Combinatorica 4, S. 373-395.

Khachiyan 1979: Khachiyan, Leonid G.: Polinomialnyi algoritm v lineinom programmirovanii [Russisch], in: Doklady Akademii Nauk SSSR 244, S. 1093-1096 [Englische Übersetzung: A polynomial algorithm in linear programming, in: Soviet Mathematics Doklady 20, 1979, S. 191-194].

Kuhn 1955: Kuhn, Harold W.: The Hungarian method for the assignment problem, in: Naval Logistics Quarterly 2, S. 83-97.

Kuhn 1991: Kuhn, Harold W.: On the origin of the Hungarian method, in Lenstra, J.K., A.H.G. Rinnooy Kan und A. Schrijver (Hg.), History of Mathematical Programming — A Collection of Personal Reminiscences, Amsterdam, S. 77-81.

Munkres 1957: Munkres, James: Algorithms for the assignment and transportation problems, in: Journal of the Society for Industrial and Applied Mathematics 5, S. 32-38.

Pukelsheim 2002: Pukelsheim, Friedrich: Die drei in Deutschland verwendeten Mandatszuteilungsmethoden und ihre Namenspatrone, http://www.math.uni-augsburg.de/stochastik/ pukelsheim/2002g.html zusammengestellt von Friedrich Pukelsheim und von Spektrum der Wissenschaft zur Drucklegung redigiert unter dem Titel „Die Väter der Mandatszuteilungsverfahren", erschienen in: Spektrum der Wissenschaft, September 2002, S. 83.

Schrijver 1986: Schrijver, Alexander: Theory of Linear and Integer Programming, Chichester.

Schrijver 2003: Schrijver, Alexander: Combinatorial Optimization: Polyhedra and Efficiency, Berlin.