Die Kunst, die Gunst des Zufalls zu erzwingen

Raimund Seidel

Einleitung

Der Zufall spielt eine erhebliche Rolle in unserem Leben. Von der Genese unserer eigenen Person (welche Samenzelle befruchtet das Ei) über die Entwicklung unserer Persönlichkeit (welche Menschen treffen wir, welche Bücher und Artikel fallen uns in die Hände) bis hin zum Tode (zur falschen Zeit am falschen Ort), fast immer scheinen Zufälle entscheidend mitzuwirken. Wir leben mit dieser Tatsache, aber es ist uns nicht immer wohl dabei, und wir wehren uns dagegen. Ein Teil des menschlichen Fortschritts kann als erfolgreicher Versuch gesehen werden, den Zufall zu beseitigen oder zumindest seine unpässlichen Konsequenzen zu lindern.

Wie kann man "Zufall beseitigen"? Das geht dann, wenn die Zufälligkeit eines Phänomens nur auf der Unwissenheit des Beobachters beruht oder auf dessen Unvermögen, tiefere Ursachen zu erkennen oder ihre Konsequenzen abzuschätzen. So wird zum Beispiel einem Binnenländer, der ahnungslos zum Meer kommt, die Gezeitenstärke vorerst zufällig und unvorhersagbar erscheinen. Sobald ihm aber der Zusammenhang zu Mond und Sonne hergestellt wird, ist die scheinbare Zufälligkeit dieses Phänomens vollkommen verschwunden.

Als zweites Beispiel nehmen wir eine Person, die sich auf eine Waldlichtung stellt und den Vogelflug beobachtet. Anfangs werden die beobachteten Flüge wohl zufällig erscheinen. Aber mit längerer Beobachtung werden sich Regelmäßigkeiten herauskristallisieren. Wenn die Person nun auch immer mehr über die Lage von Nestern und Futterquellen herausfindet und auch die grundsätzlichen Lebensgewohnheiten der einzelnen Vogelarten kennenlernt, dann wird ihr es immer leichter fallen, richtige Voraussagen über anstehende Vogelflüge zu machen.

Wie soll es aber möglich sein, "den Zufall zu nutzen"? Im vorangegangenen Beispiel könnte ein Langzeitbeobachter der Waldlichtung durch seine guten Vorhersagen einen neu angekommenen Beobachter, dem die Vogelflüge noch zufällig erscheinen, verblüffen und der Langzeitbeobachter

könnte sich so vielleicht auch Vorteile erzielen. Den Auguren der alten Römer wird das wohl nicht ganz unbekannt gewesen sein. Aber das kann man kaum "Nutzung des Zufalls" nennen, sondern eher Übertölpelung.

Tatsächliche Nutzung, so wie wir sie meinen, zeigt sich eher im folgenden Beispiel: Ein junger Mann muß zwischen zwei Freundinnen wählen, aber er kann sich nicht entscheiden. Er fragt seine Mutter um Rat. Sie aber bevorzugt insgeheim eine der beiden Freundinnen und sagt zum Sohn: Geh nachmittags zur U-Bahnstation. Deine beiden Freundinnen wohnen doch in verschiedenen Richtungen. Nimm den ersten Zug, der kommt. Die Freundin, zu der er dich bringt, nimm zur Frau. Dem Sohn scheint das einleuchtend. Nachmittags verkehren die Züge in beide Richtungen im 20 Minuten Takt. Wenn er also zu einem zufälligen Zeitpunkt zur Station käme, wäre das Zuerstkommen eines Zuges für beide Richtungen gleichwahrscheinlich, und damit käme es so zu einer fairen zufälligen Wahl zwischen den beiden Alternativen. Dem ist aber nicht so. Denn die Züge in Richtung A kommen 13, 33 und 53 Minuten nach der vollen Stunde in der Station an, und die Züge in Richtung B um 15, 35 und 55 Minuten nach der vollen Stunde. Damit gibt es in der Stunde nur sechs Minuten (13–15, 33–35 und 53–55), in denen zuerst ein Zug in Richtung B zur Station kommt, die übrigen 54 Minuten kommt zuerst ein Zug in Richtung A. Die Freundin in Richtung A hat also neunmal bessere Chancen gewählt zu werden. Die Mutter wusste genau, warum sie zu dieser Art des zufälligen Wählens riet.

Die Mutter nutzt hier den Zufall aus. Aber ist das nicht wieder einfach Übertölpelung so wie vorher beim Vogelflugbeispiel? Bei genauerem Hinsehen merkt man aber, daß die Sache hier ganz anders ist. Es dreht sich überhaupt nicht darum, ein zufälliges Ereignis, nämlich den Ankunftszeitpunkt des Sohnes bei der U-Bahnstation, vorauszusagen. Das Wesentliche ist, daß die Entscheidungsmethode so entworfen wurde, daß bei einer zufälligen (gleichverteilten) Ankunftszeit des Sohnes eine bestimmte der beiden Alternativen wesentlich wahrscheinlicher zum Ergebnis der Entscheidung wird.

Dies ist genau die Art und Weise, wie in der Informatik "die Gunst des Zufalls genutzt" wird. Innerhalb von Methoden und Algorithmen lässt man es zu Zufallsentscheidungen kommen, aber so, daß es mit hoher Wahrscheinlichkeit zu einem positiven Gesamtergebnis kommt. Man nennt diesen kontrollierten Einsatz von Zufall das *Randomisieren*. Wie das in etwa wirklich passiert, soll in den folgenden Kapiteln exemplarisch gezeigt werden.

Dieser Aufsatz richtet sich an ein breites Publikum und ist daher zum

größten Teil bewusst einfach gehalten. Für eine technisch anspruchsvollere, sowie auch breitere Behandlung des Einsatzes des Zufalls in der Informatik sei der interessierte Leser auf das Buch von Motwani und Raghavan [Motwani und Raghavan, 1995] oder auf den Übersichtsartikel von Karp [Karp, 1991] verwiesen.

Den Zufall kontrollieren: ein Brückenproblem

In diesem Beispiel soll klar werden, was mit "*kontrolliertem* Einsatz von Zufall" gemeint ist. Wir wollen uns zwar auf den Zufall verlassen, seine Quelle darf aber nicht ganz beliebig sein, sie muß unserer Kontrolle unterliegen.

Als sehr idealisiertes Beispiel, oder vielleicht Gedankenexperiment, stelle man sich eine große Stadt vor, an der ein Fluss vorbeiführt. Über diesen Fluss gebe es zwei Brücken, nennen wir sie B_0 und B_1, und auf der anderen Seite des Flusses liege ein großes Stadion. Es sei Samstagnachmittag, eine Veranstaltung finde im Stadion statt, 20 000 Autos wollen aus der Stadt zum Stadion.

Problem: Wie kann man erreichen, daß beide Brücken gleich ausgelastet werden, also jeweils von ungefähr gleich vielen Autos benutzt werden?

Leser, denen dieses Problem zu wirklichkeitsfremd erscheint, mögen Stadt und Stadion jeweils durch ein Datennetz ersetzen, die beiden Brücken durch zwei die beiden Datennetze verbindende Kabel, und die Autos durch Nachrichten von einem Netz ins andere, die sich jetzt jeweils für eines der beiden Verbindungskabel entscheiden müssen.

Lösung 1 (laissez faire): Man macht nichts und überlässt es einfach jedem Autofahrer, nach seiner Wahl Brücke B_0 oder Brücke B_1 zu befahren.

Diese Lösung verlässt sich auf den Zufall und funktioniert wohl auch erfahrungsgemäß recht gut. Aber kann man irgendeine Art von Garantie über die gleiche Auslastung der Brücken abgeben? Nur schwer, oder nur, wenn man statistische Annahmen über das Verhalten der Autofahrer macht. Solche Annahmen mögen wohl fundiert sein, sie können aber wegen Baustellen, Gerüchten und aus anderen Gründen leicht verletzt werden. Bei der Datennetzversion unseres Beispiels kann es sogar schwierig sein, vernünftig fundierte statistische Annahmen zu machen. Fazit ist, daß man sich hier auf den Zufall verlässt, seine Quelle aber nicht unter Kontrolle hat und damit nur sehr beschränkt Garantien über die Gleichauslastung der Brücken abgeben kann.

Lösung 2 (deterministisch–diktatorisch): Vor seiner Abfahrt muß jeder Autofahrer eine zentrale Leitstelle anrufen, die ihm mitteilt, ob er Brücke B_0 oder B_1 verwenden soll. An diese Anweisung muß sich der Autofahrer auch halten.

Mit dieser Lösung kann sicher eine gleiche Auslastung der beiden Brücken erreicht werden. Aber diese Lösung ist offensichtlich nicht ideal, um nicht zu sagen, sehr teuer: jeder muß telefonieren, und es muß diese zentrale Leitstelle geben.

Lösung 3 (randomisiert): Vor seiner Abfahrt muß jeder Autofahrer eine Münze werfen. Kommt Zahl, muß er die Brücke B_0 verwenden, kommt Wappen, die Brücke B_1.

Diese Lösung verursacht offensichtlich kaum Kosten, wie es die deterministische Lösung 2 tut. Man muß auch kein Wahrscheinlichkeitstheoretiker sein, um zu sehen, daß, wenn sich jeder an diese Münzwurfregel hält, es mit hoher Wahrscheinlichkeit zu einer fast gleichen Brückenauslastung kommt. Und mit ein klein wenig Mühe kann man auch quantifizierte Aussagen machen, wie z.B. die Wahrscheinlichkeit ist kleiner als 0.024 (oder 0.0000092), daß eine der Brücken von mehr als 10 300 (oder 10 500) der 20 000 Autos verwendet wird. Diese Wahrscheinlichkeitsgarantien beruhen hier auf keinerlei statistischen Annahmen über Autofahrer (außer, daß sie sich an die vorgegebene Regel halten), sondern nur auf der Annahme des fairen Münzwurfes, daß also Zahl und Wappen jeweils mit Wahrscheinlichkeit $1/2$ auftreten.

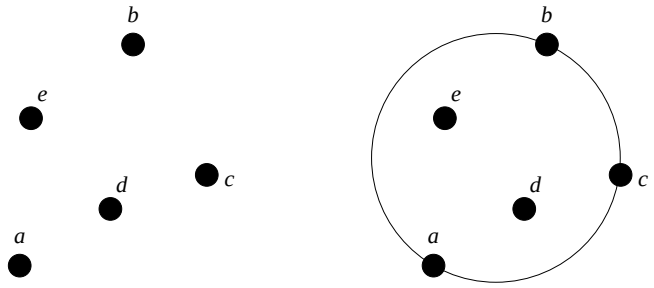
Diese randomisierte Lösung verlässt sich auf den Zufall, aber die Quelle des Zufalls ist genau bekannt und unter Kontrolle.

Zufällige Ordnung und ein geometrisches Optimierungsproblem

Das Brückenbeispiel des vorigen Kapitels hat vielleicht für manche wenig mit Berechnen zu tun, sondern "nur" mit der Koordination von unabhängigen Agenten. In diesem Kapitel wollen wir ein Beispiel geben, bei dem es sich ganz klar ums Berechnen handelt, und wo der Zufall verwendet wird, um die Berechnung zu beschleunigen.

Problem: Finde Für eine Menge S von n Punkten in der Ebene den kleinsten *umschriebenen Kreis*, also den kleinsten Kreis, der alle Punkte in S enthält.

Für manchen mag jetzt gar nicht klar sein, wo denn das Problem überhaupt steckt. Betrachtet man zum Beispiel die Punktmenge in Abbildung 2.1(a), dann *"sieht man doch ganz klar"*, daß der kleinste umschriebene



(a) (b)

Abbildung 2.1.

Kreis der durch die Punkte a, b, und c definierte ist, so wie es in Abbildung 2.1(b) gezeigt ist.

Aber so einfach ist es nun doch nicht. Denn typischerweise bekommt man die Punktmenge S nicht gezeichnet, sondern man bekommt die einzelnen Punkte durch ihre Koordinaten angegeben, also ein Zahlenpaar pro Punkt. Hier wäre ein Beispielmenge von 20 Punkten:

```
(34.20, 9.90) (41.85,12.60) (45.45,15.30) (47.70,19.35)
(48.60,23.85) (48.60,32.85) (46.80,36.90) (46.35,40.50)
(40.50,42.75) (34.65,44.10) (29.25,44.10) (24.75,42.75)
(18.45,38.70) (14.85,31.95) (13.95,28.35) (14.40,22.50)
(15.75,19.35) (18.00,14.40) (23.85,11.25) (26.55, 9.90)
```

Wenn man sich die Mühe macht, diese Punkte aufzuzeichnen, wie in Abbildung 2.2 getan, kommt man drauf, daß die Punkte alle schon fast auf einem Kreis liegen, aber eben nicht genau auf einem Kreis. Welches jetzt *genau* der gesuchte kleinste umschriebene Kreis ist, das sieht man gar nicht mehr ganz klar.

Mathematisch geneigte Leser mögen jetzt vielleicht sagen, so schwer sei das Problem aber trotzdem nicht: Drei nicht auf einer Geraden liegende Punkte a,b,c, definieren genau einen Kreis $K_{a,b,c}$, auf dem sie alle drei liegen (den Umkreis des Dreicks a, b, c). Zwei Punkte a,b definieren genau einen "Diametralkreis" $D_{a,b}$, auf dem sie beide liegen und dessen Durchmesser sie aufspannen (siehe Abbildung 2.3). Der kleinste umschriebene Kreis der Ausgangspunktemenge S kann nur ein Kreis der Form $K_{a,b,c}$ sein (wie in Abbildung 2.1) oder der Form $D_{a,b}$ (wie in Abbildung 2.3), mit a, b, c Punkte

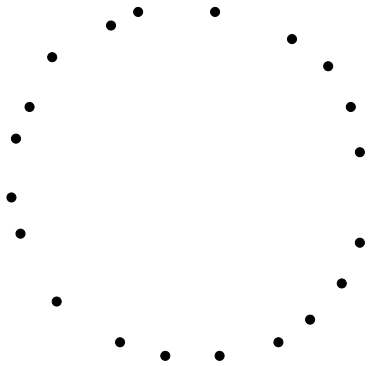


Abbildung 2.2.

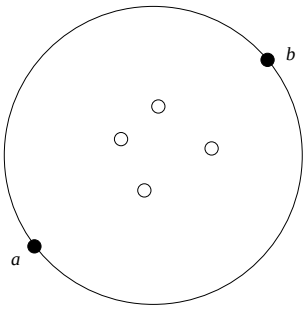


Abbildung 2.3.

in S. Nennen wir diese zwei oder drei Punkte, die den kleinsten umschriebenen Kreis von S bestimmen, die *Umpunkte* von S.

Um den kleinsten umschriebenen Kreis (bzw. die Umpunkte) von S zu finden, genügt es also, für jedes mögliche Paar a, b und jedes mögliche Tripel a, b, c von Punkten in S zu testen, ob der Kreis $D_{a,b}$ bzw. $K_{a,b,c}$ die Menge S umschreibt (alle Punkte von S enthält), und von den so gefundenen umschriebenen Kreisen den kleinsten auszuwählen.

Der mathematisch geneigte Leser hat damit recht, daß sich der kleinste umschriebene Kreis so bestimmen lässt. Aber geht das wirklich "leicht"? Bei 20 Punkten gibt es schon 190 Paare a, b und 1140 Tripel zu betrachten. Das ist für die Geduld eines Menschen zu viel. Aber für einen modernen Computer sollte das doch kein Problem sein. Das stimmt. Aber wenn man das Problem für 20 000 Punkte lösen möchte, dann gibt es rund 200 Millionen Paare zu betrachten und mehr als eine Billion Tripel! Nehmen wir

an, ein Computer könnte pro Sekunde ein Million Tripel abarbeiten — derzeit eine zweifelhafte Annahme, da ja für jedes Tripel a, b, c getestet werden muß, ob die anderen 19 997 Punkte alle innerhalb des Kreises $K_{a,b,c}$ liegen — so bräuchte der Computer allein für das Abarbeiten aller Tripel schon mehr als eine Million Sekunden. Das sind mehr als elf Tage! So lange möchte man wohl nicht warten, um den kleinsten umschriebenen Kreis von 20 000 Punkten zu finden.

Wie kann nun der Zufall helfen, dieses Problem schneller zu lösen? Die Idee ist folgende: Machen wir einmal das 20 000 Punkte Problem ein bisschen einfacher, indem wir einen Punkt, nennen wir ihn p, wegnehmen. Nehmen wir an, Kreis K' wäre die Lösung für das etwas einfachere 19 999 Punkte Problem und wir wüssten K'. Wie können wir nun den Lösungskreis K für das 20 000 Punkte Problem erhalten? Da gibt es nun zwei Fälle: Der *gute* Fall wäre, daß der weggenommene Punkt p im Kreis K' liegt. Dann ist nämlich K' nicht nur der kleinste umschriebene Kreis der 19 999 Punkte, sondern auch der kleinste umschriebene Kreis aller 20 000 Punkte. Also haben wir mit $K = K'$ die Gesamtlösung gefunden. Der *schlechte* Fall wäre, daß der Punkt p nicht im Kreis K' liegt. Dann haben wir zwar nicht den Gesamtlösungskreis K gefunden, aber wir haben wichtige Informationen über ihn gewonnen, denn p muß jetzt auf seinem Rand liegen. Das heisst, p ist ein Umpunkt der 20 000 Punkte und K kann jetzt nur noch von der Form $D_{a,p}$ oder $K_{a,b,p}$ sein.

Jetzt kommt der Zufall ins Spiel. Wenn p zufällig aus den 20 000 Punkten ausgewählt wird, dann ist es recht unwahrscheinlich, daß der *schlechte Fall* eintritt. Denn eine "*schlechte*" Wahl für p wären nur die Umpunkte, und davon gibt es unter den 20 000 entweder zwei oder drei, also höchstens[1] drei. Das heisst, der schlechte Fall tritt mit einer Wahrscheinlichkeit von höchstens 3/20 000 ein.

Diese Überlegungen führen zu folgender *randomisierter* Methode zur Bestimmung des kleinsten umschriebenen Kreises K einer Menge S von n Punkten.

1. (Randomisiere) Bringe die Punkte von S in eine zufällige Reihenfolge $p_1, p_2, \ldots, p_n$.

2. (Initialisiere) Sei K der Diametralkreis von p_1 und p_2.

3. (Inkrementiere) Für jedes i von 3 bis n mache folgendes:

> Falls p_i nicht im Kreis K liegt, dann löse das Unterproblem, den kleinsten umschriebenen Kreis von $\{p_1, \ldots, p_{i-1}\}$ zu finden, der p_i auf dem Rand hat. Von nun an sei K der Lösungskreis dieses Unterproblems.

Wie löst man das Unterproblem? Ganz analog zum ursprünglichen Problem, nur mit zwei kleinen Änderungen: erstens die Initialisierung, da K jetzt den Punkt p_i auf dem Rand haben muß, und zweitens die Inkrementierung, bei der sich das Unter-Unterproblem ergibt, den kleinsten umschriebenen Kreis zu finden, der zwei vorgegeben Punkte auf dem Rand hat. Man kann sich leicht davon überzeugen, daß dieses Unter-Unterproblem leicht zu lösen ist.

Die interessante Frage ist aber jetzt, wie schnell die vorgeschlagene Methode ist. Das hängt natürlich ganz stark davon ab, wieviel Zeit man für die Lösung eines Unterproblems braucht, und natürlich auch davon, wie oft der schlechte Fall eintritt, daß ein Unterproblem gelöst werden muß.

Nehmen wir an, daß ein Computer zur Unterproblemlösung für i Punkte $3i$ Mikrosekunden braucht. Wie oft Unterprobleme gelöst werden müssen, hängt natürlich von der im Schritt 1 gewählten Reihenfolge der Punkte ab. Ist sie schlecht, dann könnte es eintreten, daß für jedes i ein Unterproblem gelöst werden muß. Ist sie gut, muß vielleicht überhaupt kein Unterproblem gelöst werden. Das Interessante ist nun, daß man von einer zufällig gewählten Reihenfolge erwarten kann, daß sie ziemlich gut ist.

Hier ist die Argumentation: Wir fragen uns, was ist der *erwartete* Zeitaufwand für den i-ten Durchgang der Iteration? Wenn die Reihenfolge zufällig gewählt wurde, dann ist es für jeden der ersten i Punkte in der Reihenfolge gleichwahrscheinlich an i-ter Stelle zu stehen. Das heisst, mit Wahrscheinlichkeit höchstens $3/i$ steht einer der höchstens drei Umpunkte von $\{p_1, \ldots, p_i\}$ an letzter Stelle, und damit tritt der schlechte Fall, daß ein Unterproblem gelöst werden muß, höchstens mit Wahrscheinlichkeit $3/i$ ein. Der Zeitaufwand für die Lösung dieses Unterproblems ist dann, wie oben angenommen, $3i$ Mikrosekunden. Der *erwartete* Aufwand ist damit $(3i) \cdot (3/i)$, also 9 Mikrosekunden. Zählen wir jetzt noch (konservativ) eine Mikrosekunde für den Test, ob p_i im Kreis K liegt, dazu, ergibt sich als *erwarteter* Zeitaufwand für den i-ten Durchgang der Iteration 10 Mikrosekunden. Alle $n - 2$ Durchgänge zusammen brauchen daher *im Erwartungswert* weniger als $10n$ Mikrosekunden. Nehmen wir noch n Mikrosekunden für das Herstellen der zufälligen Reihenfolge in Schritt 1 dazu, dann ist die *erwartete*

Gesamtzeit dieser Methode für das Bestimmen des kleinsten umschriebenen Kreises von n Punkten weniger als $11n$ Mikrosekunden.

Für 20 000 Punkte braucht diese randomisierte Methode "im Durchschnitt" also nur 220 000 Mikrosekunden, also etwas weniger als eine Viertelsekunde, und nicht über 11 Tage.

Der aufmerksame Leser wird jetzt fragen, wie realistisch die Annahme sei, daß ein Unterproblem für i Punkte in $3i$ Mikrosekunden gelöst werden könne. Das ist ziemlich realistisch, wenn man mit "Lösungszeit" die *erwartete* Lösungszeit meint. Denn für die vorgeschlagene Lösung des Unterproblems kann man die gleiche Analyse anwenden wie eben, nur daß jetzt die Wahrscheinlichkeit höchstens $2/i$ ist, daß der schlechte Fall eintritt und ein Unter-Unterproblem gelöst werden muß. Dieses Unter-Unterproblems kann man für i Punkte realistisch sehr leicht in i Mikrosekunden lösen.

Genauere Einzelheiten findet man in der Arbeit von Welzl [Welzl, 1991], in der diese Methode vorgestellt wurde. Dort wird auch gezeigt, daß sich diese Methode für das analoge Problem im 3-dimensionalen Raum, finde die kleinste umschriebene Kugel von n Punkten, bzw. auch im d-dimensionalen Raum verwenden lässt. Allerdings ist der erwartete Zeitaufwand dann proportional zu $d\,!\,n$, was nur für recht kleine d erträglich ist. Verfeinerte und allgemeinere Methoden findet man in Arbeiten von Matoušek, Sharir und Welzl [Matoušek et al., 1996] und von Gärtner [Gärtner, 1995]. Der Letztere stellt auch unter [Gärtner, 1999] Software zur Verfügung.

Zufällige Zeugen und Primzahlbestimmung

Eine natürliche Zahl $N > 1$ heißt *Primzahl*, wenn sie nur durch 1 und durch sich selbst teilbar ist. Eine Zahl, die keine Primzahl ist, heißt *zusammengesetzt*, denn sie ist darstellbar als $N = A \cdot B$ mit $1 < A, B < N$.

In diesem Abschnitt geht es um das Problem, für eine vorgegebene Zahl N festzustellen, ob sie eine Primzahl ist oder nicht.

Die Definition von Primzahl gibt eigentlich schon eine Möglichkeit an, wie man das Problem lösen kann: man probiert einfach für jede Zahl $A = 2, 3, 4, \ldots, N-1$, ob A die Zahl N teilt. Man kann das auch noch etwas beschleunigen. Erstens braucht man nach $A = 2$ nur die ungeraden Zahlen zu probieren. Zweitens reicht es, nur Zahlen bis höchstens $\sqrt{N}$ zu probieren. Denn wenn N zusammengesetzt ist, also $N = A \cdot B$, dann ist sowohl A als auch B Teiler von N und es können nicht beide größer als $\sqrt{N}$

sein, denn dann gälte $A \cdot B > N$.

Selbst diese beschleunigte Methode ist leider nicht sehr schnell, außer N ist relativ klein. Wenn man sie auf die 30-stellige Zahl $N = 10^{30} - 11$ anwendet, dann müssen ungefähr $0.5 \cdot 10^{15}$ Teilbarkeitstests durchgeführt werden, um festzustellen, daß N prim ist. Nehmen wir an, ein Computer könne ein Million "primitive" Operationen pro Sekunde durchführen, wobei primitive Operationen so etwas sind wie Multiplikation zweier Zahlen oder deren Division mit Restbildung (de facto der Teilbarkeitstest) und wobei die Operanden, sagen wir, höchstens 100-stellige Zahlen sind. Auf so einem Computer bräuchte dann diese "beschleunigte" Methode zum Testen von $N = 10^{30} - 11$ ungefähr $0.5 \cdot 10^9$ Sekunden. Das sind mehr als 15 Millionen Jahre! Da würde es auch nicht viel nützen, wenn der Computer tausendmal schneller wäre. In der modernen Kryptographie verwendet man aber routinemäßig Primzahlen mit mehreren hundert Stellen. Wie kann man so große Zahlen auf Primalität testen?

Zuerst einmal ein klein wenig Zahlentheorie: Ein Satz von Fermat besagt, wenn N eine Primzahl ist, dann gilt für jede ganze Zahl X mit $1 \leq X < N$

$$X^{N-1} \bmod N = 1\,.$$

Dabei bedeutet $U \bmod V$ der Rest bei der ganzzahligen Division von U durch V. Zur Illustration das Beispiel $N = 5$ (eine Primzahl)

X	1	2	3	4
X^4	1	16	81	256
$X^4 \bmod 5$	1	1	1	1

und zum Gegensatz $N = 8$ (eine zusammengesetzte Zahl)

X	1	2	3	4	5	6	7
X^7	1	128	2187	16384	78125	279936	823543
$X^7 \bmod 8$	1	0	3	0	5	0	7

Hat man also ein X mit $X^{N-1} \bmod N \neq 1$, dann kann N keine Primzahl sein. Nennen wir so ein X einen *Zeugen* für die Zusammengesetztheit von N.

Nehmen wir an, daß für jede zusammengesetzte Zahl N mindestens die Hälfte der Kandidatenzahlen in $Z_N = \{1, 2, 3, 4, \ldots, N-1\}$ Zeugen für die Zusammengesetztheit von N sind.

Betrachten wir nun folgende randomisierte Methode zum Testen, ob eine Zahl N zusammengesetzt oder prim ist:

> Versuche 60 mal durch *zufällige* Wahl einer Zahl aus Z_N einen Zeugen für die Zusammengesetztheit von N zu finden.
> Falls einer der Versuche erfolgreich ist, erkläre N für zusammengesetzt.
> Wenn alle 60 Versuche fehlschlagen, erkläre N für prim.

Ist diese Methode korrekt? Wenn sie zum Schluss kommt, daß N zusammengesetzt ist, dann stimmt das auch, denn es wurde ja ein Zeuge dafür gefunden. Wenn sie aber zum Schluss kommt, daß N prim ist, dann könnte sich die Methode auch geirrt haben. Denn N könnte zusammengesetzt sein, aber bei jedem der 60 Versuche wurde aus Z_N zufällig eine Zahl gewählt, die kein Zeuge für die Zusammengesetztheit von N ist.

Die Wahrscheinlichkeit, daß es zu diesem Irrtum kommt, ist allerdings sehr klein. Nach unserer Annahme sind bei zusammengesetztem N mindestens die Hälfte der Zahlen in Z_N Zeugen. Die Wahrscheinlichkeit, bei zufälliger Auswahl aus Z_N einen Nichtzeugen zu ziehen, ist daher kleiner als $1/2$, und die Wahrscheinlichkeit, dies 60 mal hintereinander zu tun, ist kleiner als $1/2^{60}$ (das ist kleiner als 10^{-18}).

Stellt also unsere randomisierte Methode fest, eine Zahl ist zusammengesetzt, dann stimmt das. Stellt sie fest, eine Zahl ist prim, dann kann sie sich irren, und zwar mit einer Wahrscheinlichkeit kleiner als 10^{-18}.
Nun wird sicher für manchen eine Feststellung der Art "Diese Zahl ist *höchstwahrscheinlich* prim" nicht zufriedenstellend sein. Dazu sind drei Dinge zu sagen: Erstens kann man durch eine größere Anzahl von Wiederholungen die Fehlerwahrscheinlichkeit beliebig klein machen. Zweitens ist die Wahrscheinlichkeit dieses Fehlers vergleichbar mit der Wahrscheinlichkeit anderer Fehler, die beim Betreiben und Benutzen von Rechnern auftreten, oder sogar kleiner. Zum Beispiel braucht ein 10^9-Operationen-pro-Sekunden-Rechner mehr als 31 Jahre für 10^{18} Operationen; die Wahrscheinlichkeit ist erheblich, daß es in diesem Zeitraum zu einem Hardwarefehler kommt. Drittens sind zur Zeit leider keine *effizienten* fehlerfreien Methoden zur Primzahlbestimmung bekannt, mit Ausnahme der Methode von Miller [Miller, 1976], deren Korrektheit allerdings wiederum auf einer *un-*

bewiesenen Verallgemeinerung der *unbewiesenen* Riemannschen Vermutung beruht.
Zwei Dinge bedürfen nun aber noch der Klärung:

1. Ist die Annahme berechtigt, bei zusammengetztem N seien mindestens die Hälfte der Kandidaten in Z_N Zeugen?

2. Wie stellt man wirklich fest, ob eine Zahl $X \in Z_N$ ein Zeuge für die Zusammengesetztheit von N ist?

Die Antwort zu Frage 2) erscheint zuerst klar: nach Definition von Zeugen berechnet man einfach $X^{N-1} \bmod N$ und testet, ob das Ergebnis 1 ist. Wie macht man aber dies auf effiziente Art und Weise? Die Zahl X einfach $N-1$ mal mit sich selbst zu multiplizieren braucht sicher zu lange, wenn N groß ist (z.B. eine 30-stellige Zahl). Durch den Trick des sogenannten "wiederholten Quadrierens" kommt man aber um dieses Problem herum: zum Beispiel lässt sich X^{16} durch nur 4 Multiplikationen berechnen, nämlich

$$\left(\left((X^2)^2\right)^2\right)^2 ,$$

oder, als etwas komplizierteres Beispiel, lässt sich X^{26} durch

$$\left(\left(\left(\left((X^2)\cdot X\right)^2\right)^2\right)\cdot X\right)^2$$

mit nur 6 Multiplikationen berechnen. Im Allgemeinen lässt sich mit Hilfe von wiederholtem Quadrieren ein Ausdruck der Form X^M bei einer n-stelligen Zahl M durch höchstens ungefähr $7n$ Multiplikationen berechnen. Bei unserer 30-stelligen Beispielzahl $10^{30}-11$ bräuchte man also weniger als 210 Multiplikationen für einen Zeugentest und nicht 10^{30}.

Selbst mit wiederholtem Quadrieren werden allerdings die Zwischenergebnisse so groß, daß die Annahme, man könne eine Multiplikation in einer konstanten Zeiteinheit (in unserem Beispiel 10^{-6} Sekunden bei 100-stelligen Operanden) durchführen, nicht mehr gerechtfertigt ist[2]. In unserem Fall, wo $X^{N-1} \bmod N$ gefragt ist, kann man aber die Zwischenergebnisse klein halten, indem jedes Zwischenergebnis Y durch den Rest $Y \bmod N$ ersetzt wird und damit weitergerechnet wird. Dies ist wegen der Rechenregel

$$(A\cdot B\cdot C) \bmod N = (((A\cdot B) \bmod N)\cdot C) \bmod N$$

zulässig.

Bei unserer Beispielzahl $10^{30} - 11$ braucht dann die randomisierte Methode nur höchstens $60 \cdot 2 \cdot 210 = 25200$ primitive Operationen (die Berechnung des "modularen Produkts" $A \cdot B \bmod N$ braucht 2 Operationen). Auf unserem Beispielcomputer bräuchte dies etwas mehr als eine vierzigstel Sekunde, und nicht 15 Millionen Jahre.

Schließlich müssen wir noch die Frage 1) klären. Die Antwort dazu ist zwar für fast alle zusammengesetzten Zahlen N positiv, im Allgemeinen aber negativ. Die Antwort wird aber für *alle* N positiv, wenn man die Definition von "X ist Zeuge für die Zusammengesetztheit von N" etwas erweitert, und zwar gilt X auch als Zeuge, wenn während der Berechnung von $X^{N-1} \bmod N$ durch wiederholtes Quadrieren eine nichttriviale Wurzel von 1 entdeckt wird, d.h. eine Berechnung $Y^2 \bmod N$ ergibt 1, wobei $Y \neq 1$ und $Y \neq N - 1$. Für eine Primzahl N kann so etwas nämlich nicht auftreten. Einen Beweis, daß bei zusammengesetztem N immer mindestens die Hälfte der Kandidaten Zeugen in diesem erweiteten Sinne sind, findet man in [Rabin, 1980].

Man beachte die unterschiedlichen Nutzungen von Zufall in diesem und im vorherigen Kapitel. Die randomisierte Optimierungsmethode in Kapitel 3 liefert immer eine korrekte Antwort, nur die Zeit, die gebraucht wird, um zu dieser Antwort zu kommen, hängt vom Zufall ab und ist im Erwartungswert klein. Diese Art von randomisierter Methode wird *Las Vegas Methode* genannt. Dies steht im Gegensatz zur *Monte Carlo Methode* zum Primzahltesten in diesem Abschnitt: sie hat eine vom Zufall im Wesentlichen unabhängige Laufzeit, dafür liefert sie aber nicht immer die korrekte Antwort. Die Korrektheit der Antwort hängt also vom Zufall ab, aber auf eine so günstige Art und Weise, daß die Wahrscheinlichkeit eines Fehlers in der Antwort verschwindend klein ist.

Schnelle Monte Carlo Methoden zum Primzahltesten wurden von Solovay und Strassen [Solovay und Strassen, 1977] und von Rabin [Rabin, 1976] vorgestellt. Die hier umrissene Methode folgt Ansätzen von Rabin [Rabin, 1980] und Miller [Miller, 1976]. Es ist hochinteressant, daß alle diese Methoden eine Zahl N als zusammengesetzt erkennen, ohne einen Teiler von N zu liefern. Es wird allgemein angenommen (und es ist ein Grundpfeiler der modernen Kryptographie), daß das Faktorisieren zusammengesetzter Zahlen, also das Bestimmen der Teiler, *wesentlich* schwieriger und zeitaufwendiger ist als das reine Erkennen solcher Zahlen. Diese Annahme ist allerdings unbewiesen und interessanterweise für das (jetzt noch) futuristische Modell des Quantenrechners sogar falsch [Shor, 1997].

Daß beim ungefähren Zählen, also beim Schätzen, der Zufall von Nutzen sein kann, entspricht unserer Erfahrung. Wollten wir wissen, wieviele Worte in einem 500 Seiten Buch vorkommen, dann könnten wir ein paar, sagen wir fünf, zufällige Seiten aussuchen und die darauf vorkommenden Worte zählen. Das Ergebnis mal hundert wäre dann ein guter Schätzwert für die Zahl aller Worte in dem Buch. Wollten wir wissen, wie oft das Wort "Zufall" in diesem Buch vorkommt, könnten wir genau so vorgehen. Intuitiv wissen wir aber, daß im zweiten Fall der so erhaltene Schätzwert nicht so vertrauenswürdig ist wie im ersten, denn es wird eine größere Streuung geben, das heisst, das Wort "Zufall" wird kaum über alle Seiten des Buches annähernd gleich verteilt sein. So kommt es zum Beispiel auf den Seiten dieses Aufsatzes wohl um einiges öfter vor als auf den Seiten anderer Aufsätze dieses Bandes. Aber um unseren Schätzwert zu verbessern, könnten wir eine größere Anzahl von Seiten betrachten. Die Wissenschaft der Statistik gibt uns quantitativ an, wie viele Seiten wir bei welcher Streuung betrachten müssen, um eine bestimmte Güte des Schätzwerts zu erreichen.

Im Falle der Zahl der Worte in einem Buch wäre das Schätzen eigentlich nicht notwendig. Man könnte die Worte ja einfach abzählen. Für einen Menschen wäre das etwas mühsam, für einen Computer wäre es aber nur eine Sache weniger Sekunden, sollte das Buch in elektronischer Form vorhanden sein. In diesem Abschnitt wollen wir Zählprobleme behandeln, die, obwohl einfach formulierbar, so kompliziert sind, daß selbst mit den besten Computern, auch wenn man sie noch millionenfach beschleunigte, keine genauen Lösungen in vernünftiger Zeit errechnet werden können. Ja, selbst die Frage, ob man in vernünftiger Zeit halbwegs gute Schätzwerte errechnen kann, ist erst teilweise gelöst und ist gegenwärtig ein interessanter Forschungsgegenstand.

Wir wollen uns mit dem *Paarungsanzahl*-Problem beschäftigen, bei dem die Anzahl der *vollständigen Paarungen* in einem *bipartiten Graphen* gezählt werden soll. Was ist ein bipartiter Graph? Man malt sich n schwarze und n weiße Knoten auf und Kanten, die manche schwarze mit manchen weißen Knoten verbinden. Ein Beispiel mit $n = 4$ ist in Abbildung 2.4 zu sehen.

Was ist eine vollständige Paarung? Das sind n Kanten, die jeden Knoten mit genau einem Knoten der anderen Farbe verbinden, anders gesagt, n schwarz-weiße Knotenpaare, in denen alle Knoten vorkommen. Im Graphen von Abbildung 2.4 gibt es genau drei solche vollständige Paarungen.

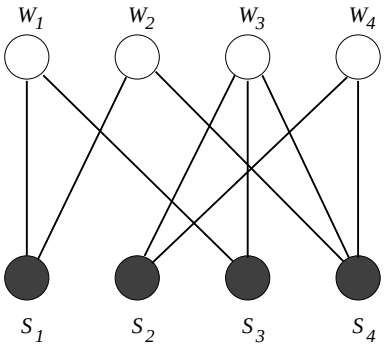


Abbildung 2.4.

Sie werden in Abbildung 2.5 gezeigt.

Der Leser möge sich davon überzeugen, daß es in dem Beispielgraphen keine weiteren vollständigen Paarungen gibt. Das wird wahrscheinlich nicht ganz leicht fallen und zeigt, daß selbst schon bei solch kleinen Beispielen das Paarungsanzahlproblem nicht einfach ist. Bei größeren Graphen wird das noch durch die riesige mögliche Anzahl der vollständigen Paarungen erschwert. In einem Graphen, in dem jeder der n schwarzen Knoten mit jedem der n weissen Knoten verbunden ist, gibt es n Partnerkandidaten für W_1, für jede Wahl bleiben für W_2 dann $n-1$ Partnerkandidaten übrig, $n-2$ Kandidaten für W_3, und so fort. Insgesamt gibt es dann $n \cdot (n-1) \cdot (n-1) \cdots 2 \cdot 1 = n!$ viele vollständige Paarungen. Für $n = 10$ sind das etwas mehr 3,6 Millionen, für $n = 20$ schon über 2,4 Trillionen ($2,4 \cdot 10^{18}$).

Für das Paarungsanzahlproblem ist derzeit keine vernünftige, exakte und deterministische Lösungsmethode bekannt. Es ist nicht zu erwarten, daß je eine gefunden wird, da das Problem #P-vollständig ist. Das bedeutet, fände man eine vernünftig rasche Lösungsmethode für dieses Problem, würde man damit gleichzeitig Lösungsmethoden für eine ganze Horde anderer Zählprobleme finden, die allesamt als sehr schwierig gelten. Ein tatsächlicher Beweis, daß es so eine "vernünftige" Lösungsmethode nicht geben kann, steht allerdings noch aus.

Man versucht daher, zumindest approximative Lösungsmethoden für das Paarungsanzahlproblem zu finden. Es ist noch nicht klar, ob das immer mit vernünftigem Aufwand und auch vernünftiger Approximationsgüte möglich ist. Es sind aber schon gewisse Teilerfolge erzielt worden. Zwei

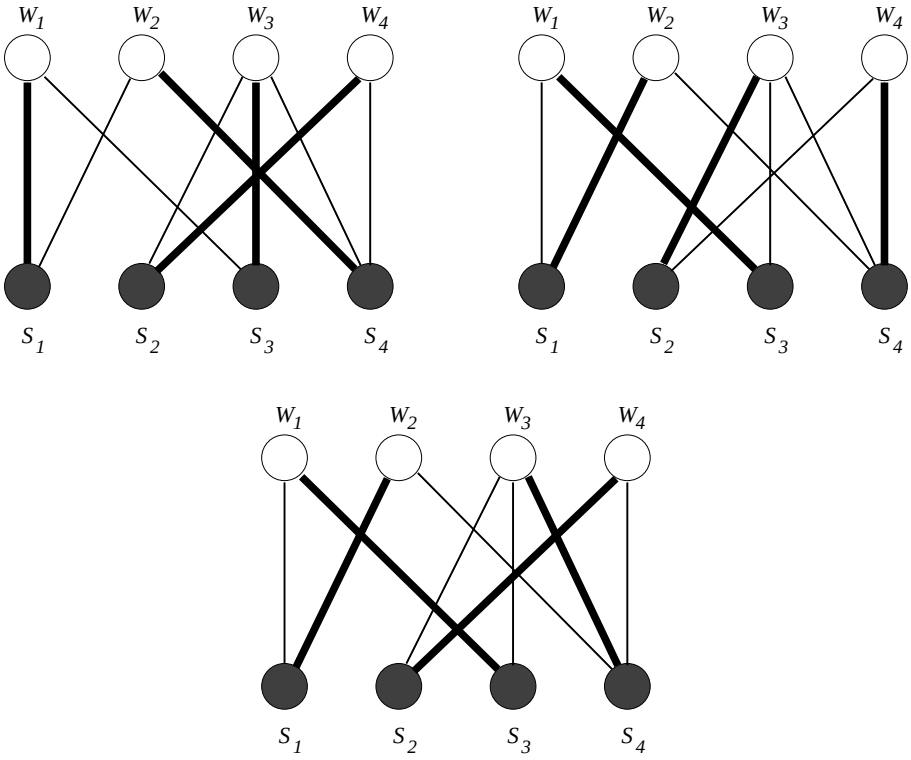


Abbildung 2.5.

Ansätze sollen hier kurz vorgestellt werden. Beide verwenden das Randomisieren, also den kontrollierten Einsatz von Zufall, auf sehr raffinierte Art und Weise. Die erste Methode beruht auf der Algebra, die zweite auf sogenannten zufälligen Wanderungen oder Irrfahrten.

Zuvor aber noch zur Beruhigung des Lesers: dies ist nicht alles *l'art pour l'art*. In der statistischen Physik möchte man genau solche Paarungsanzahlprobleme lösen.

Eine algebraische Methode

Man kann einen bipartiten Graphen mit n weissen Knoten $W_1, \ldots, W_n$ und n schwarzen Knoten $S_1, \ldots, S_n$ auf natürliche Weise durch seine *Adjazenzmatrix* darstellen. Das ist eine $n \times n$ Matrix A, in der $A_{i,j}$, der Eintrag in der i-ten Zeile und j-ten Spalte, zu 1 gemacht wird, wenn die Knoten W_i und S_j

durch eine Kante verbunden sind, und zu 0 sonst. Die Adjazenzmatrix des Graphen von Abbildung 2.4 sieht dann folgendermaßen aus:

$$\begin{array}{c} \\ W_1 \\ W_2 \\ W_3 \\ W_4 \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

Eine vollständige Paarung entspricht dann einer Auswahl von n Einsen in dieser Matrix, sodaß in jeder Zeile und in jeder Spalte genau eine Eins gewählt wurde. Die drei möglichen vollständigen Paarungen unseres Beispiels sehen dann so aus:

$$\begin{pmatrix} \boxed{1} & 0 & 1 & 0 \\ 1 & 0 & 0 & \boxed{1} \\ 0 & 1 & \boxed{1} & 1 \\ 0 & \boxed{1} & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & \boxed{1} & 0 \\ \boxed{1} & 0 & 0 & 1 \\ 0 & \boxed{1} & 1 & 1 \\ 0 & 1 & 0 & \boxed{1} \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & \boxed{1} & 0 \\ \boxed{1} & 0 & 0 & 1 \\ 0 & 1 & 1 & \boxed{1} \\ 0 & \boxed{1} & 0 & 1 \end{pmatrix}$$

Um dies nun algebraisch zu fassen, brauchen wir den Begriff der Permutation. Eine Permutation der Größe n ist eine Auflistung der Zahlen von 1 bis n, sodaß jede Zahl in dieser Auflistung genau einmal vorkommt. Zum Beispiel sind $\rho = (1, 4, 3, 2)$ und $\sigma = (3, 1, 2, 4)$ Permutationen der Größe 4, aber $\tau = (2, 4, 1, 2)$ ist keine. Wenn π eine Permutation ist, dann bezeichnet $\pi(i)$ einfach die Zahl, die an i-ter Stelle der Auflistung π steht; z.B. $\sigma(3)$ ist 2. Wenn π eine Permutation der Größe n ist und A eine $n \times n$ Matrix, dann ist

$$A_{1,\pi(1)}, A_{2,\pi(2)}, A_{3,\pi(3)}, \ldots, A_{n,\pi(n)}$$

eine Auswahl von n Elementen der Matrix, sodaß aus jeder Zeile und aus jeder Spalte genau ein Element gewählt wurde. Jede Auswahl mit dieser Eigenschaft lässt sich durch genau eine Permutation identifizieren. So entsprechen die oben angegebenen Permutationen ρ und σ genau zwei der in unserem Beispiel getroffenen Auswahlen (welchen, das möge der Leser sich überlegen). Die Permutation $\pi = (1, 2, 3, 4)$ entspricht der Auswahl $A_{1,1}, A_{2,2}, A_{3,3}, A_{4,4}$, die aber in unserem Beispiel wiederum keiner vollständigen Paarung entspricht, da $A_{2,2}$ gleich 0 ist.

Wir sehen nun, daß vollständige Paarungen genau solchen Permutationen π entsprechen, für die gilt $A_{i,\pi(i)} = 1$ für alle i, oder anders ausgedrückt,

für das Produkt gilt

$$A_{1,\pi(1)} \cdot A_{2,\pi(2)} \cdot A_{3,\pi(3)} \cdots A_{n,\pi(n)} = 1 \,.$$

Für Permutationen, die keinen vollständigen Paarungen entsprechen, ist dieses Produkt null. Damit kann man jetzt die Anzahl der vollständigen Paarungen in einem bipartiten Graphen schön algebraisch ausdrücken als Summe über alle möglichen permutationsentsprechenden Produkten:

$$\sum_{\substack{\pi \text{ ist Permutation} \\ \text{der Größe } n}} A_{1,\pi(1)} \cdot A_{2,\pi(2)} \cdot A_{3,\pi(3)} \cdots A_{n,\pi(n)} \,.$$

Dies mag zwar hübsch aussehen, ist aber vorerst nicht besonders hilfreich, denn es gibt $n!$ viele Permutationen der Größe n. Es gibt also viel zu viele Summanden, als daß diese Summe in vernünftiger Zeit ausgewertet werden könnte.

Aber diese Summe ähnelt stark der Definition der *Determinante* $\det(A)$ einer Matrix A:

$$\det(A) = \sum_{\substack{\pi \text{ ist Permutation} \\ \text{der Größe } n}} s(\pi) \cdot A_{1,\pi(1)} \cdot A_{2,\pi(2)} \cdot A_{3,\pi(3)} \cdots A_{n,\pi(n)} \,,$$

wobei $s(\pi)$ ein von der Permutation abhängiges Vorzeichen ist, also $+1$ oder -1. Obwohl die Determinantenfunktion durch diese Vorzeichen in ihrer Summendefinition komplizierter aussieht, lässt sie sich mit Hilfe alternativer Definitionen relativ leicht berechnen. Diese Tatsache wird nun unter Zuhilfenahme von Zufall ausgenutzt.

Godsil und Gutman [Godsil und Gutman, 1981] schlugen 1981 folgende Methode vor: Es sei A die Adjazenzmatrix eines bipartiten Graphens G. Ändere jede 1 in A zufällig, mit Wahrscheinlichkeit $1/2$ (und unabhängig von den anderen Einsen) zu -1. Berechne die Determinante der so erhaltenen zufälligen Matrix und nimm ihr Quadrat als Schätzwert für die Anzahl der vollständigen Paarungen im Graphen G.

Warum soll das ein guter Schätzwert sein? Es stellt sich heraus, daß, wenn man alle möglichen Muster von $+1/-1$ der Matrix A betrachtete und man für jedes Muster den Schätzwert wie angegeben berechnete, dann wäre der Durchschnitt der so erhaltenen Schätzwerte genau die Anzahl der vollständigen Paarungen. Mathematisch ausgedrückt: der Erwartungswert des Schätzwertes ist die Paarungsanzahl.

Das ist nicht sehr schwer zu beweisen. Das Quadrat der Determinante von A lässt sich schreiben als

$$\left(\sum_{\substack{\pi \text{ ist Permutation} \\ \text{der Größe } n}} s(\pi) \cdot A_{1,\pi(1)} \cdots A_{n,\pi(n)} \right)^2 ,$$

was mit Hilfe des Distributivgesetztes zu

$$\sum_{\substack{\pi,\, \rho \text{ Permutationen} \\ \text{der Größe } n}} \left(s(\pi) \cdot A_{1,\pi(1)} \cdots A_{n,\pi(n)}\right) \left(s(\rho) \cdot A_{1,\rho(1)} \cdots A_{n,\rho(n)}\right)$$

wird. Wir wollen den Erwartungswert dieser Summe wissen. Der ist die Summe der Erwartungswerte der einzelnen Summanden. Betrachten wir also einen Summanden

$$\left(s(\pi) \cdot A_{1,\pi(1)} \cdots A_{n,\pi(n)}\right) \left(s(\rho) \cdot A_{1,\rho(1)} \cdots A_{n,\rho(n)}\right) .$$

Sollte π oder ρ keiner vollständigen Paarung entsprechen, d.h. $A_{i,\pi(i)} = 0$ oder $A_{i,\rho(i)} = 0$ für irgendein i, dann ist dieser Ausdruck (und damit auch sein Erwartungswert) 0.

Im Fall $\rho \neq \pi$ muß es ein i geben mit $\pi(i) \neq \rho(i)$. Der Term $A_{i,\pi(i)}$ kommt dann in dem Produkt

$$\left(s(\pi) \cdot A_{1,\pi(1)} \cdots A_{n,\pi(n)}\right) \left(s(\rho) \cdot A_{1,\rho(1)} \cdots A_{n,\rho(n)}\right)$$

genau einmal vor. Aufgrund seiner gleichwahrscheinlichen $-1/+1$ Belegung ist der Erwartungswert von $A_{i,\pi(i)}$ gleich 0, und damit (und aufgrund der Unabhängigkeit der zufälligen $-1/+1$-Zuweisungen) ist auch der Erwartungswert des gesamten Produkts gleich 0.

Schließlich im Fall, daß $\rho = \pi$, ergibt dieser Ausdruck

$$\left(s(\pi) \cdot A_{1,\pi(1)} \cdots A_{n,\pi(n)}\right)^2 ,$$

und das ist immer 1, ganz gleich wie das Vorzeichen $s(\pi)$ aussieht und welche $+1/-1$ Werte den Matrixelementen zugewiesen wurden. Damit bekommen wir genau für jede Permutation, die einer Paarung entspricht, einen Beitrag von 1 zum Erwartungswert der gesamten Summe, und damit ist der Erwartungswert dieser Summe gleich der gesuchten Paarungsanzahl.

Dieser Schätzwert von Godsil und Gutman hat nun zwar den richtigen Erwartungswert, nur ist leider seine Streuung so groß, daß man etwas mehr

als $3^{n/2}$ viele unabhängige Schätzwerte ermitteln muß, damit ihr Durchschnitt mit vernünftiger Wahrscheinlichkeit eine vernünftige Approximation der wahren Paarungsanzahl ist.

Dies wurde inzwischen etwas verbessert. 1993 zeigten Karmakar et al. [Karmarkar et al., 1993], daß folgende leichte Veränderung des Verfahrens von Godsil und Gutman zu einem guten Schätzverfahren mit geringerer Streuung führt und somit nur etwas mehr als $2^{n/2}$ Schätzwerte ermittelt werden müssen.

Ändere jede 1 in A zufällig, mit $1/4$ Wahrscheinlichkeit (und unabhängig von den anderen Einsen) zu $-1, +1, I, -I$, wobei $I = \sqrt{-1}$. Berechne die Determinante der so erhaltenen zufälligen Matrix und nimm das Quadrat ihres Absolutbetrags als Schätzwert für die Anzahl der vollständigen Paarungen im Graphen G.

Schließlich zeigte Rasmussen [Rasmussen, 1998] in 1998, aufbauend auf Arbeiten von Barvinok [Barvinok, 1999], daß man über die komplexen Zahlen hinweg noch einen Schritt weiter machen kann, um die Streuung und damit die Anzahl der nötigen Schätzwerte zu verringern. Er verwendet zufällige Ersetzungen der Einsen durch die acht Quaternionen $\pm 1, \pm I, \pm J, \pm K$ und reduziert damit die Anzahl der zu ermittelnden Schätzwerte auf etwas über $(3/2)^{n/2}$. Dies ist zwar noch immer exponentiell, also viel zu groß und führt nicht zu einer Schätzmethode mit vernünftiger Laufzeit, aber es legt nahe, daß durch die Verwendung von höher-dimensionalen Algebren, sogenannten Clifford Algebren, die Streuung auf ein vernünftiges Maß reduziert werden könnte. Ob dies wirklich möglich ist, ist zur Zeit nicht bekannt und Gegenstand der Forschung.

Eine Irrfahrtmethode

Hier ist eine vollkommen andere randomisierte Methode, um die Paarungsanzahl in einem bipartiten Graphen zu schätzen. Diese Methode ist viel weniger direkt als die gerade gezeigte.

Definieren wir eine *k-Paarung* in einem bipartiten Graphen als eine Menge P von k Kanten, sodaß jeder Knoten im Graphen an höchstens einer der Kanten in P anliegt. Im folgenden sei nun G ein bipartiter Graph mit $2n$ Knoten. Es seien M_k die Menge aller k-Paarungen in G und m_k die Anzahl der verschiedenen k-Paarungen in G. Eine vollständige Paarung in G ist jetzt nichts anderes als eine n-Paarung, also ein Element von M_n, und die Paarungsanzahl in G zu bestimmen oder zu schätzen ist nichts anderes als das Bestimmen oder Schätzen von m_n.

Für $i = 2, \ldots, n$ sei $r_i = m_i/m_{i-1}$. Dann gilt

$$m_1 \cdot r_2 \cdot r_3 \cdots r_{n-1} \cdot r_n = m_1 \cdot \frac{m_2}{m_1} \cdot \frac{m_3}{m_2} \cdots \frac{m_{n-1}}{m_{n-2}} \cdot \frac{m_n}{m_{n-1}} = m_n \,.$$

Die Idee ist nun diese: m_1 ist bekannt, es ist die Anzahl der Kanten von G; wenn man jedes der r_i's schätzt, dann ergibt das Produkt von m_i und von allen diesen Schätzungen nach der angegebenen Gleichung eine Schätzung der gesuchten Zahl m_n.

Wie kann man aber ein r_i schätzen, und zwar mit hinreichender Genauigkeit? Die Grundidee hier ist wieder einfach: Angenommen, man hätte einen "Zufallspaarungsgenerator", der auf Anfrage eine zufällige Paarung in $M_{i-1} \cup M_i$ erzeugt, jede mit gleicher, oder zumindest fast gleicher Wahrscheinlichkeit. Dann könnte man eine ganze Reihe von Anfragen an diesen Generator richten und zählen, wie oft er eine $(i-1)$-Paarung und wie oft er eine i Paarung liefert, sagen wir jeweils c_{i-1} und c_i mal. Wenn r_i nicht zu groß oder zu klein ist (was derzeit nur garantiert werden kann, wenn jeder Knoten in G an hinreichend viele, sagen wir mindestens $n/2$, Kanten anliegt) und wenn hinreichend viele Anfragen gestellt wurden, dann liefert das Verhältnis c_{i-1}/c_i mit hoher Wahrscheinlichkeit eine gute Schätzung für das gesuchte r_i.

Jetzt brauchen wir also nur mehr so einen "Zufallspaarungsgenerator". Dieser funktioniert folgendermaßen: Man beginnt mit irgendeiner Paarung P_0 in $M_{i-1} \cup M_i$. Diese verändert man auf zufällige Art und Weise durch eine kleine "Mutation" zu einer Paarung P_1 in $M_{i-1} \cup M_i$. Durch eine weitere kleine zufällige Änderung bekommt man davon Paarung P_2, und so fort. Nach t solchen "zufälligen, kleinen" Veränderungen erreicht man eine Paarung P_t in $M_{i-1} \cup M_i$, und es ist plausibel, daß bei hinreichend großem t die so in vielen zufälligen kleinen Schritten erreichte Paarung P_t mit fast gleicher Wahrscheinlichkeit jede der Paarungen in $M_{i-1} \cup M_i$ sein kann.

Dies ist bei geeigneter Spezifikation von "kleiner zufälligen Veränderung" (im Prinzip ändert sich die Paarung in höchstens einer zufälligen Kante) tatsächlich der Fall. Der Beweis dafür ist aber äußerst kompliziert und weit außerhalb des Rahmens dieses Aufsatzes.

Der gerade skizzierte Zufallspaarungsgenerator basiert auf der Methode der "Irrfahrt" oder "zufälligen Wanderung". Diese Namen erklären sich dadurch, daß man sich vorstellen kann, daß man auf zufällige Art und Weise entlang der Kanten eines Graphens U wandert, also eine Irrfahrt macht, wobei in unserem Fall die Paarungen in $M_{i-1} \cup M_i$ die Knoten von U sind, und

zwei Knoten (also Paarungen) durch eine Kante in U verbunden sind, wenn sie durch "kleine Änderungen" auseinander hervorgehen. Diese Sichtweise und die damit verbundene Theorie der Markoff-Ketten hat sich bei einigen Zähl- und Schätzproblemen als sehr fruchtbar erwiesen. Wir verweisen den Leser auf das Buch [Motwani und Raghavan, 1995] von Motwani und Raghavan und auf die Monographie [Sinclair, 1992] von Sinclair.

Zusammenfassend können wir sagen, daß die hier gerade skizzierte randomisierte Methode für das Schätzen der Paarungsanzahl eines bipartiten Graphen G im Fall, daß jeder Knoten von G an hinreichend vielen Kanten anliegt, nachweislich mit "vernünftig" viel Rechenaufwand mit hoher Wahrscheinlichkeit zu einer "vernünftig" genauen Lösung führt. Was "vernünftig" hier genau bedeuten soll, lassen wir offen. Der technisch versierte Leser möge das Wort "polynomiell" dafür einsetzen. Man soll aber keinesfalls glauben, daß "vernünftig viel Rechenaufwand" hier mit praktischer Effizienz gleichzusetzen sei.

Schlußbemerkungen

Rechnen, Berechnen und Problemlösen tragen üblicherweise die Attribute "exakt", "vorhersagbar" und "deterministisch", also eigentlich gerade das Gegenteil von "zufällig". Die Beispiele in diesem Aufsatz haben den Leser hoffentlich davon überzeugt, daß der Zufall sehr wohl fürs Rechnen und Problemlösen nützlich sein kann, wenn er nur richtig eingesetzt wird. Die dargestellten Beispiele reichten vom Koordinieren von unabhängigen Agenten über rasches Lösen von Optimierungsproblemen bis hin zum Testen von speziellen Eigenschaften von Zahlen und zum algorithmischen Schätzen komplizierter Größen. Dies ist aber wahrlich nur ein sehr kleiner Ausschnitt aus dem Gebiet der randomisierten Algorithmen. Beim Verwalten von Information findet Randomisierung ebenso Anwendung wie beim Lösen geometrischer oder zahlentheoretischer Probleme, oder bei Algorithmen der Textverarbeitung. Der interessierte Leser sei an das Buch von Motwani und Raghavan [Motwani und Raghavan, 1995] verwiesen.

Randomisierte Algorithmen zeichnen sich oft durch ihre Einfachheit aus. Wenn überhaupt etwas dabei kompliziert wird, dann sind es typischerweise die Beweise ihrer Korrektheit- oder Laufzeiteigenschaften. Dies macht sie für den praktischen Einsatz oft besonders attraktiv.

In randomisierten Methoden muß oft "eine Münze geworfen" werden, oder es soll eine zufällige ganze Zahl zwischen 0 und n generiert werden.

Mancher Leser wird fragen, durch welchen Mechanismus dies im Ablauf von Computerprogrammen tatsächlich realisiert wird. Dieses Problem ist schon in sich selbst einen Aufsatz wert. Wir verweisen dafür auf das Buch von Luby [Luby, 1996] für einige theoretische Grundlagen der sogenannten Pseudozufallszahlenerzeugung und auf den zweiten Band des großen Werkes von Knuth [Knuth, 1997].

Danksagung

Die Abschnitte *Den Zufall kontrollieren* und *Zufällige Zeugen und Primzahlbestimmung* dieser Arbeit erschienen schon in einem früheren Aufsatz des Autors mit dem Titel "Der Zufall in der Informatik". Der Abdruck erfolgt mit freundlicher Genehmigung der Deutschen Akademie der Naturforscher Leopoldina aus: Nova Acta Leopoldina NF Bd. 79, Nr. 308 "Der Zufall", S. 127–139 (1999), Herausgeber: K. Krickeberg, H. Bauer, H. Föllmer, J. Moser und V. Strassen.

Anmerkungen

[1] Wir nehmen hier an, daß die Punktemenge S in "allgemeiner Lage" ist, in dem Sinne, daß keine vier Punkte in S auf einem gemeinsamen Kreis liegen. Aber selbst wenn dies nicht der Fall ist, stimmt die Wahrscheinlichkeitsaussage noch immer, ja, die Wahrscheinlichkeit für den schlechten Fall wird sogar kleiner.

[2] Mathematische Leser mögen bitte verzeihen, daß darauf verzichtet wurde, hier Restklassenringe einzuführen.

Literatur

[Barvinok, 1999] Barvinok, A. I. (1999). Polynomial time algorithms to approximate permanents and mixed discriminants within a simply exponential factor. *Random Structures & Algorithms* 14, 29–61.

[Birkhoff, 1946] Birkhoff, G., (1946). Tres observaciones sobre el algebra lineal. *Rev. univ. nac. Tucum n (A)* 5, 147–151.

[Brogan, 1989] Brogan, W. L., (1989). Algorithm for ranked assignments with application to multiobject tracking. *Journal of Guidance* 12, 357–364.

[Gärtner, 1995] Gärtner, B. (1995). A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing* 24, 1018–1035.

[Gärtner, 1999] Gärtner, B. (1999). Smallest enclosing balls – fast and robust in C++.
`http://www.inf.ethz.ch/personal/gaertner/miniball.html`

[Godsil und Gutman, 1981] Godsil, C. und Gutman, I. (1981). On the matching polynomial of a graph. *Algebraic Methods in Graph Theory I.* Lovász, L. und Sós, V., Hrsg., Colloq. Math. Soc. János Bolyai, 25, North-Holland, Amsterdam, 241–249.

[Karp, 1991] Karp, R. (1991). An introduction to randomized algorithms. *Discrete Applied Mathematics,* 34, 165–201.

[Knuth, 1997] Knuth, D. (1997). *Seminumerical Algorithms*, Band 2 von *The Art of Computer Programming.* Addison-Wesley, Dritte Ausgabe.

[Luby, 1996] Luby, M. (1996). *Pseudorandomness and Cryptographic Applications.* Princeton University Press.

[Matoušek et al., 1996] Matoušek, J., Sharir, M. und Welzl, E. (1996). A subexponential bound for linear programming. *Algorithmica,* 16, 498-516.

[Miller, 1976] Miller, G. (1976). Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences,* 13, 300–317.

[Motwani und Raghavan, 1995] Motwani, R. und Raghavan, P. (1995). *Randomized Algorithms.* Cambridge University Press.

[Rabin, 1976] Rabin, R. (1976). Probabilistic algorithms. In *Algorithms and Complexity, Recent Results and New Directions*, Traub, J., Hrsg., Academic Press, 21–39.

[Rabin, 1980] Rabin, M. (1980). Probabilistic algorithm for testing primality. *Journal of Number Theory,* 12, 128–138.

[Rasmussen, 1998] Rasmussen, L. E. (1998) *On Approximating the Permanent and other #P-complete Problems.* Ph.D. Thesis, Computer Science Division, Univ. of California, Berkeley.

[Shor, 1997] Shor, P. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26, 1484–1509.

[Sinclair, 1992] Sinclair, A. (1992). *Algorithms for Random Generation and Counting: A Markov Chain Approach.* Progress in Theoretical Computer Science. Birkhäuser, Boston.

[Solovay und Strassen, 1977] Solovay, R. und Strassen, V. (1977, 1978). A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6(1), 84–85, 1977; und auch *SIAM Journal on Computing*, 7(1), 118, 1978.

[Welzl, 1991] Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science*, Maurer, H., Hrsg., Springer Lecture Notes in Computer Science, 555, 359–370.