Georg-August-Universität Göttingen
Institut für Theoretische Physik

Vortrag zum Seminar

Statistische Mechanik ungeordneter Systeme

bei PD Dr. Alexander Karl Hartmann
und
Dr. Timo Aspelmeier

Das Zahlenaufteilungsproblem (Number Partitioning Problem)

von Silke Dreißigacker
und Alexander Mann

April 2005

Inhaltsverzeichnis

Die Autoren sind per E-Mail erreichbar unter
silk30a@web.de (Silke Dreißigacker) und fuenfundachtzig@gmx.de (Alexander Mann).

1. Einleitung

In der Thermodynamik haben wir Phasenübergänge kennengelernt und beschrieben, wie wir sie in der physikalischen Realität beobachten, wenn Wasser zu Eis gefriert oder verdampft. Ein Ordnungsgrad ändert sich dabei sprunghaft, beim Verdampfen von Wasser zum Beispiel die Dichte. Auch aus anderen physikalischen System wie den Ferromagneten sind uns Phasenübergänge bekannt. Phasenübergänge können jedoch auch bei nicht physikalischen Systemen auftreten, beispielsweise bei Untersuchungsobjekten der Mathematik, den Zahlenmengen.

In diesem Vortrag werden wir sehen, dass die bekannten Werkzeuge der Thermodynamik auf ein typisches Problem aus der Numerik, das Zahlenaufteilungsproblem, anwendbar sind. Wie in [7] dargestellt werden wir Bedingungen für den Phasenübergang zwischen einfach und schwer zu lösenden Situationen analytisch herleiten und die Phasenübergänge in Computerexperimenten nachweisen. Hierzu benötigen wir effiziente Algorithmen. Leider gehört das Zahlenaufteilungsproblem, wie wir sehen werden, eigentlich zu der Gruppe der „harten“ Computerprobleme. Wir werden darauf eingehen, warum es trotzdem möglich ist, einen relativ schnellen Algorithmus zu finden und was die Bereiche auszeichnet, die der Phasenübergang trennt.

Anwendungen für das Zahlenaufteilungsproblem finden sich bei Optimierungsaufgaben wie der Aufgabenverteilung auf Mehrprozessorcomputern oder in der Industrie bei der Entwicklung von hoch integrierten Schaltungen (VLSI = Very Large Scale Integration), um nur zwei Beispiele zu nennen.

1.1. Definition des Number Patitioning Problems ($\mathcal{NPP}$)

Bei dem Zahlenaufteilungsproblem handelt es sich um eine recht einfach zu formulierende Optimierungsaufgabe:
Es sei eine Menge $\mathcal{A}$ von N natürlichen Zahlen: $\mathcal{A} = \{(a_i),\ a_i \in \mathbb{N}\backslash\{0\},\ i = 1, \ldots, N\}$ gegeben. Die Aufgabe besteht nun darin, die Zahlen so in zwei Teilmengen zu unterteilen, dass die Summen der Zahlenwerte in beiden Teilmengen möglichst wenig voneinander abweichen. Mathematisch formuliert man dies so, dass eine Teilmenge $\mathcal{B} \subset \mathcal{A}$ zu finden ist, so dass sich die Summe der Elemente der Menge $\mathcal{B}$ und die Summe der Elemente ihres Komplements in $\mathcal{A}$, $\overline{\mathcal{A}} \doteq \mathcal{A}\backslash\mathcal{B}$, sich um möglichst wenig unterscheiden.
Die Differenz der beiden Summen liefert die Kostenfunktion

$$E \doteq \left| \sum_{\substack{i=1 \\ a_i \in \mathcal{B}}}^{N} a_i - \sum_{\substack{i=1 \\ a_i \in \mathcal{A}\backslash\mathcal{B}}}^{N} a_i \right|. \tag{1}$$

Hieraus leiten wir nun einige Definitionen ab:

Definition Eine *perfekte Aufteilung* bezeichnet die grundsätzlich „bestmögliche“ Unterteilung der Menge $\mathcal{A}$ bei gegebener Summe aller Zahlen aus $\mathcal{A}$.

Wir müssen hier zwei Fälle unterscheiden. Ist die Summe aller Zahlen in $\mathcal{A}$ gerade, so ist die minimale Kostenfunktion $E = 0$. Ist die Summe ungerade, so ist eine Unterteilung in zwei Hälften mit betragsgleichen Summen von vornherein nicht möglich. Ist die Differenz jedoch minimal, d. h. ist $E = 1$, so spricht man auch hier von einer *perfekten Aufteilung*.
In unserem Vortrag wird in diesem Zusammenhang ein weiterer Terminus eine Rolle spielen, und zwar der der *ausgewogenen Aufteilung*. So werden Partitionen bezeichnet, in denen die Anzahl der

Zahlen in beiden Teilmengen übereinstimmt, bzw. sich im Falle ungerader Gesamtanzahl um maximal 1 unterscheidet. Mathematisch erhält man daraus folgende

Definition Falls $|\mathcal{B}| - |\mathcal{A}\backslash\mathcal{B}| = n \mod 2$, so heißt eine Aufteilung ausgewogen.

Das $\mathcal{NPP}$ kann beliebig auf nicht ganzzahlige Verteilungen oder die Verteilung der Zahlen auf $q > 2$ Untergruppen ausgeweitet werden. Wir wollen uns in unserer Arbeit an die obige (eingeschränkte) Definition halten.

2. Komplexitätstheorie (Computational Complexity)

Damit wir uns vertieft mit der Materie des $\mathcal{NPP}$ beschäftigen können, müssen wir uns zunächst mit einigen Klassifikationen der Informatik beschäftigen.

Die *Zeitkomplexität* soll eine quantitative Aussage über die Zeit liefern, die ein Algorithmus zur Lösung eines konkreten Problems benötigt. Es liegt nahe, bei der Definition der Zeitkomplexität eines Problems, das auf einem Computer gelöst werden soll, die benötigte Rechenzeit als Maß heranzuziehen. Man kann die Rechenzeit dabei jedoch nicht einfach mit einer Stoppuhr messen, da man sonst für denselben Algorithmus auf jedem Computer aufgrund der verschiedenen Rechenleistungen eine unterschiedliche Zeitkomplexität erhielte. Auch kann es für ein und dieselbe Aufgabe wie zum Beispiel das Sortieren von Zahlen unterschiedlich schnelle Algorithmen geben. Schlichte (dafür meist sehr intuitive) Algorithmen, wie z.B. Bubblesort oder Selectsort, haben eine Laufzeit von $\mathcal{O}(n^2)$, d.h. sie benötigen eine Rechenzeit, die quadratisch mit der Anzahl n der zu sortierenden Zahlen wächst. Zu den schnellsten Algorithmen zum Sortieren von Zahlen gehören Mergesort und Heapsort mit einer Laufzeit von $\mathcal{O}(n \log n)$.

Was genau eine Laufzeit von $\mathcal{O}(x)$ heißen soll, darüber werden wir uns im nächsten Abschnitt Gedanken machen. Klar ist, dass wir eine Möglichkeit finden müssen, eine von der *Implementation des Algorithmus und Rechenleistung unabhängige Klassifikation* der Komplexität eines Problems zu erreichen. Eine sehr gut lesbare Einführung hierzu bietet [5].

2.1. Leicht und schwer lösbare Probleme

Allgemein hängt die Laufzeit eines Algorithmus von der Größe der Eingabe ab, also von der Größe des Problems an sich und der Größe der Darstellung der Daten. Man nennt dies eine *Instanz* des Problems. Um die Abhängigkeit der Komplexität von der konkreten Instanz unabhängig zu machen, wählt man als Maß für die Zeitkomplexität ein „worst-case Szenario“. Wir definieren:

$$T(n) = \max_{|x|=n} t(x) \tag{2}$$

Hierin ist t(x) die Laufzeit des Algorithmus für eine Instanz x mit der Länge n. Man erhält also mit $T(n)$ ein Maß, das nur noch von der Länge der Eingabe abhängt, und betrachtet nun den Grenzfall $n \to \infty$. In der Komplexitätstheorie hat man sich auf eine auf den ersten Blick sehr grobe Einteilung geeinigt: Probleme mit polynomialer (worst case) Laufzeit, für die $T(n) = \mathcal{O}(n^c)$ für irgendein c ist, heißen *leicht*, solche mit einer Laufzeit ohne polynomiale obere Schranke, also z.B. $T(n) = \mathcal{O}(2^n)$ oder $T(n) = \mathcal{O}(n!)$ heißen *schwer* lösbar. Die Wahl der Bezeichnung wird klar, wenn man bedenkt, was passiert, wenn n nur um eins größer wird: Für einen schwer lösbaren Algorithmus der Ordnung 2^n verdoppelt sich die Laufzeit bei Erhöhung der Systemgröße um eins schlagartig, wodurch man man schnell an die Grenzen der zur Verfügung stehenden Rechenleistung stößt.

2.2. Die $\mathcal{P}$- und die $\mathcal{NP}$-Klasse

Eine Fragestellung, die sich eindeutig mit „ja“ oder „nein“ beantworten lässt, bezeichnet man als *Entscheidungsproblem*. Ausgehend von der oben getroffenen Unterscheidung der Probleme in leicht lösbar und schwer lösbar leitet man nun eine Unterteilung in zwei Klassen, $\mathcal{P}$ und $\mathcal{NP}$, ab:

Definition Ein Entscheidungsproblem $\mathcal{P}$ gehört zur *Klasse* $\mathcal{P}$ genau dann, wenn es einen Algorithmus gibt, welcher es in polynomialer Laufzeit löst.

Wir werden in Abschnitt (2.3) einige Beispiele kennenlernen. Auch das Sortieren von Zahlen zählt zur Klasse $\mathcal{P}$, auch wenn es sich hierbei nicht um ein Entscheidungsproblem handelt. Alle leichten Probleme gehören nach der obigen Definition zur Klasse $\mathcal{P}$, schwer sind alle diejenigen, die nicht in $\mathcal{P}$ liegen. Es wäre nun einfach zu sagen, zu $\mathcal{NP}$ gehört alles übrige. Die tatsächliche Definition ist jedoch anders. Wir benötigen für sie den Konzept des *nicht-deterministischen Algorithmus*, welcher den üblichen Befehlssatz um eine Anweisung erweitert, die allerdings nur theoretisch vorstellbar ist:

goto both label1, label2

Dies ist sozusagen eine „gleichzeitige Verzweigung“ (siehe auch Abb. (1)). Der Algorithmus soll an dieser Stelle gleichzeitig an zwei verschiedene Stellen verzweigen. Die CPUs unserer Rechner arbeiten Befehle linear ab, d.h. einen nach dem anderen. Um die „goto both“-Anweisung ausführen zu können, müsste sich also die Anzahl der CPUs schlagartig verdoppeln, damit der Prozess gleichzeitig an beiden Stellen, an die verzweigt wurde, fortgesetzt werden kann. Durch diese Anweisung wäre es möglich, beliebig viele Verzweigungen in einem Suchbaum gleichzeitig zu durchsuchen, praktisch umsetzbar ist sie zum heutigen Zeitpunkt nicht, da sich die Rechenleistung nicht auf Knopfdruck verdoppeln lässt. Sie führt uns aber zu folgender Definition:

Definition Ein Entscheidungsproblem P gehört zur Klasse $\mathcal{NP}$ genau dann, wenn es einen nicht-deterministischen Algorithmus gibt, der es in polynomialer Laufzeit löst.

Es lässt sich zeigen, dass diese Definition zu folgender äquivalent ist (siehe [13]):

Definition Ein Entscheidungsproblem P gehört zur Klasse $\mathcal{NP}$ genau dann, wenn ein Lösungskandidat für eine „ja“-Antwort in polynomialer Zeit mit einem deterministischen Algorithmus überprüft und bestätigt werden kann.

Ein Problem liegt also in der Klasse $\mathcal{NP}$, wenn man ein Problem dadurch in polynomialer Zeit lösen kann, dass man die Zweige eines Suchbaumes (vgl. Abbildung (1)) gleichzeitig abarbeitet. Hat man eine Lösung gefunden, so muss die Überprüfung durch einen deterministischen Algorithmus in polynomialer Zeit möglich sein. Sie kann dadurch erfolgen, dass der Algorithmus an jeder Verzweigung diejenige wählt, die ihn zur „ja“-Antwort führt. Die Ausführungszeit bleibt dabei linear in der Baumtiefe, auch wenn der Suchbaum beliebig in die Breite geht und dadurch eine ausführliche Suche eine exponentielle Laufzeit hat. Es ist offensichtlich $\mathcal{P} \subseteq \mathcal{NP}$, denn falls ein Problems x in polynomialer Zeit gelöst werden kann ($x \in \mathcal{P}$), so folgt, dass die Lösung auch in polynomialer Zeit überprüft werden kann ($\Rightarrow x \in \mathcal{NP}$).

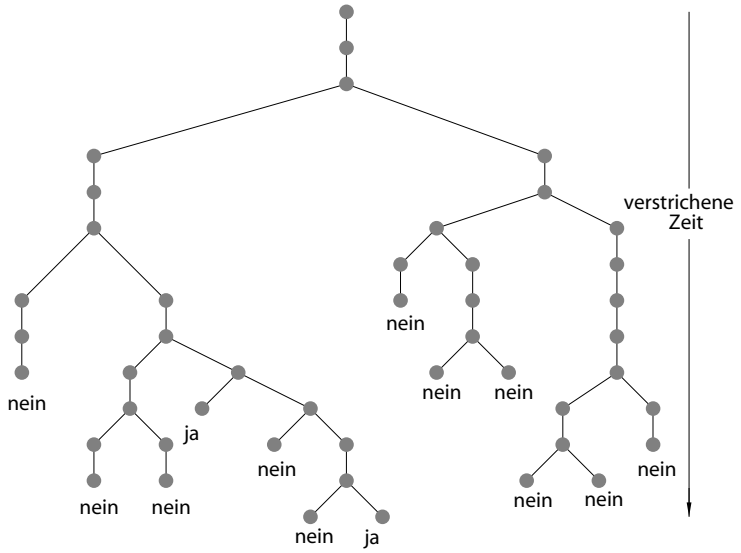

Abbildung 1: Beispiel für die Ausführungsweise eines nicht-deterministischen Algorithmus. (aus [5])

2.2.1. Erlauchte Gesellschaft: $\mathcal{NP}$-vollständige Probleme

Innerhalb der $\mathcal{NP}$-Probleme gibt es eine Klasse, die sich durch eine besondere Eigenschaft auszeichnet: Würde es gelingen, ein Problem aus dieser Klasse der *vollständigen $\mathcal{NP}$-Probleme* zu lösen, so könnte man den entsprechenden Algorithmus auf alle anderen Probleme der $\mathcal{NP}$-Klasse übertragen und somit wäre bewiesen, dass $\mathcal{P} = \mathcal{NP}$. Diese Möglichkeit des Übertragens sei wie folgt definiert:

Definition Ein Problem P_1 heißt *polynomial rückführbar* auf ein anderes Problem P_2, wenn eine vollständige mit polynomialer Zeitkomplexität berechenbare Funktion $\varphi(x)$ existiert, so dass $x =$ „ja-Instanz für Problem 1“ $\Leftrightarrow \varphi(x) =$ „ja-Instanz für Problem 2“. Wir schreiben dafür $P_1 \leq_p P_2$.

In anderen Worten: $P_1 \leq_p P_2$ soll heißen, dass das Problem P_1 nicht schwieriger zu lösen sein kann als das Problem P_2. Damit können wir die Klasse der $\mathcal{NP}$-vollständigen Probleme wie folgt definieren:

Definition Ein Problem P heißt *$\mathcal{NP}$-vollständig*, wenn $P \in \mathcal{NP}$ und $Q \leq P \; \forall Q \in \mathcal{NP}$.

Die $\mathcal{NP}$-vollständigen Probleme sind die härtesten aller $\mathcal{NP}$-Probleme. Jedes von ihnen könnte der Schlüssel zu der ganzen Klasse sein. Auch das Zahlenaufteilungsproblem gehört in die besondere Klasse der $\mathcal{NP}$-vollständigen Probleme. Zum Abschluss noch eine letzte Definition:

Definition Ein Problem P heißt $\mathcal{NP}$-hart, wenn seine Entscheidungsvariante $\mathcal{NP}$-vollständig ist.

Als Entscheidungsvariante eines Problems bezeichnet man die Abwandlung der entsprechenden Fragestellung auf eine Entscheidungsfrage, d.h. eine Frage, die sich mit ja oder nein beantworten lässt. Geht es zum Beispiel beim $\mathcal{NPP}$ darum, eine Aufteilung einer Zahlenmenge zu finden bzw.

die Differenz der bestmöglichen Aufteilung, so könnte eine Entscheidungsvariante des $\mathcal{NPP}$ so aussehen: „Gibt es eine perfekte Aufteilung?“ oder „Lässt sich eine ausgewogene Aufteilung finden, die eine Differenz von weniger als 10 aufweist?“

Wichtig wird für uns sein, dass auch das $\mathcal{NPP}$ ein Vertreter der Klasse der $\mathcal{NP}$-vollständigen Probleme ist. Der Beweis hierfür findet sich in [10] und erfolgt durch Rückführung auf das 3-SAT-Problem, was einfacher als der „direkte“ Beweis ist.

2.2.2. Die sechs grundlegenden NP-complete Probleme

Garey und Johnson listen in ihrem Buch [10] sechs grundlegende Probleme aus der Klasse der $\mathcal{NP}$-vollständigen Probleme auf, mit denen wir den Überblick über die Thematik der Komplexitätstheorie abrunden wollen. Alle sechs sind Entscheidungsprobleme.

3-dimensional matching Gegeben sei eine Menge $M \subseteq W \times X \times Y$, wobei X, Y und W disjunkte Mengen mit $|X| = |Y| = |W| = q$ Elementen sind. Die Frage ist: Gibt es eine Teilmenge $M' \subseteq M$, so dass $|M'| = q$ und keine zwei Elemente von M' in irgendeiner Koordinate übereinstimmen?

Vertex Cover Gegeben sei ein Graph $G(V, E)$[1] und eine Zahl $K \in \mathbb{Z}^+, K \leq |V|$. Die Frage ist: Gibt es einen Teilmenge (genannt vertex cover) $V' \subseteq V$ mit $|V'| \leq K$, so dass von jeder Kante $\{u, v\} \in E$ mindestens einer der Endpunkte u oder v zu V' gehört?

Clique Gegeben sei ein Graph $G(V, E)$ und eine Zahl $K \in \mathbb{Z}^+, K \leq |V|$. Die Frage ist: Gibt es eine Teilmenge (genannt clique) $V' \subseteq V$ mit $|V'| \geq K$, so dass jedes Paar von Knoten in V' durch eine Kante in E verbunden ist?

Hamiltonian Circuit Gegeben sei ein Graph $G(V, E)$. Die Frage ist: Gibt es eine Anordnung $< \nu_1, \ldots, \nu_n >$ aller Knoten in V, so dass die Kanten $\{\nu_n, \nu_1\} \in E$ und $\{\nu_i, \nu_{i+1}\} \in E,\ 1 \leq i \leq n-1$ enthalten sind?

Die anderen beiden sind das Zahlenaufteilungsproblem und das wichtige „3-SAT“-Problem, mit dem wir uns in Abschnitt 2.3 ausführlich beschäftigen wollen.

2.2.3. $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$

Es stellt sich natürlich die Frage, ob man überhaupt zwischen $\mathcal{P}$ und $\mathcal{NP}$ unterscheiden muss. Könnte es nicht sein, dass sich jedes $\mathcal{NP}$-Problem in polynomialer Zeit lösen lässt und bloß noch niemand den richtigen Algorithmus gefunden hat? Würde man einen Algorithmus finden, der irgendein Problem aus der Klasse der vollständigen $\mathcal{NP}$-Probleme in polynomialer Zeit lösen könnte, so könnte man ihn auf alle anderen Probleme dieser Klasse übertragen und es wäre bewiesen, dass $\mathcal{P} = \mathcal{NP}$ (siehe Abschnitt (2.2.1)). Obwohl diese Herausforderung schon einige Jahrzehnte besteht, ist es bisher noch niemandem gelungen, einen solchen Algorithmus zu finden. Andererseits war es bisher auch nicht möglich, das Gegenteil zu beweisen, dass also $\mathcal{P} \neq \mathcal{NP}$. Aufgrund der gesammelten Erfahrungen und der unzähligen vergeblichen Versuche, einen polynomialen Algorithmus für $\mathcal{NP}$-Probleme zu finden, liegt es nahe anzunehmen, dass die Vermutung $\mathcal{P} = \mathcal{NP}$ falsch ist. Letztendlich muss man aber eingestehen, dass diese Frage noch nicht endgültig geklärt ist.

[1] Bei Graphen soll bei uns V immer die Menge der Knoten (vertices) und E die Menge der Kanten (edges) bezeichnen.

Das Clay Mathematics Institute [12] hat die Klärung der Frage, ob $\mathcal{P} = \mathcal{NP}$ ist, zu einem ihrer sieben „Millenium Problems“ ernannt und auf ihre Lösung 1000000 Dollar ausgelobt.

2.3. SAT als einfaches Beispiel für ein $\mathcal{NP}$-Problem

Nachdem wir nun die Theorie kennengelernt haben, wollen wir unser Wissen auf zwei Beispiele anwenden. Wir beginnen mit dem Satisfiability-Problem (kurz SAT), das sich wie folgt darstellt:

Sei $F(x_1, \ldots, x_n) : \{0,1\}^n \mapsto \{0,1\}$ eine gegebene Boolsche Funktion in den n Boolschen Variablen[2] x_i. Sie stellt eine logische Verknüpfung dieser Variablen dar, d.h. einen Ausdruck, in dem die x_i durch die binären Operatoren UND und ODER und die Negation $\neg$ verknüpft sind. Die Variablen x_i und ihre Verneinungen $\neg\, x_i \equiv \overline{x_i}$ heißen Literale, ihre Verknüpfungen nennt man Klauseln. Die logischen Operatoren sind gegeben durch:

- UND $\wedge$: Der Ausdruck $q_1 \wedge q_2$ ist WAHR genau dann, wenn q_1 WAHR ist *und* q_2 WAHR. ist
- ODER $\vee$: Der Ausdruck $q_1 \vee q_2$ ist WAHR genau dann, wenn q_1 WAHR ist *oder* q_2 WAHR ist oder wenn beide WAHR sind.
- Negation $\bar{\cdot}$: Der Ausdruck $\overline{q_1}$ ist WAHR genau dann, wenn q_1 FALSCH ist.

Jede Boolsche Funktion lässt sich in der *konjunktiven Normalform* darstellen (engl. conjunctive normal form, CNF), die aus ein oder mehreren Klauseln besteht, die ausschließlich durch UND-Operatoren verknüpft sind, während in jeder der Klauseln die Literale ausschließlich durch ODER-Operatoren verknüpft sind. Ein Beispiel für eine CNF wäre also

$$F = (q_1 \vee q_2) \wedge (q_2 \vee \overline{q_3})$$

(Für dieses Beispiel ist es natürlich einfach, eine Lösung „durch Hinsehen“ zu finden.) Die SAT ist ein wichtiges Problem in der Informatik. Ein Schaltkreis aus logischen Gattern kann z.B. als SAT-Problem aufgefasst werden.

Wir wollen nun als *n-SAT-Problem* ein SAT-Problem bezeichnen, dessen Klauseln immer genau n Literale enthalten. Die 1-SAT ist langweilig und lässt sich in linearer Zeit lösen, wie man selbst schnell rausfindet. Auch für die 2-SAT lässt sich ein Algorithmus mit linearer Zeitkomplexität finden (siehe [3]). Damit hört es aber auch schon auf. Bisher ist für die 3-SAT kein Algorithmus bekannt, der wesentlich effizienter[3] ist als das sture Durchprobieren aller 2^n möglichen Kombinationen der Wahrheitswerte in den Variablen. In der Tat gehört die 3-SAT zu den $\mathcal{NP}$-vollständigen Problemen.

2.4. Zweites Beispiel: Das MST-Problem

Nachdem wir eben ein Problem gesehen haben, dass trotz seiner Schlichtheit schon zu den $\mathcal{NP}$-vollständigen Problemen gehört, werden wir nun noch einen Vertreter aus der Klasse der $\mathcal{P}$-Probleme kennenlernen:

Angenommen wir wären Manager eines großen Energiekonzerns und wollten unsere Kraftwerke vernetzen, damit bei einem Ausfall nicht gleich eine ganze Region und Tausende unserer teuren Kunden ohne Strom sind. Die dafür nötigen Überlandleitungen sind sehr teuer und deshalb begnügen wir uns damit, dass jedes Kraftwerk mit irgendeinem anderen verbunden wird. Zusätzlich hängen die Kosten für eine Verbindung davon ab, wie weit die Kraftwerke voneinander entfernt sind und ob

[2] Boolsche Variablen können nur zwei Werte annehmen: 0 (= falsch) und 1 (= wahr).

[3] Wesentlich effizienter im Sinne einer polynomialen statt einer exponentiellen Zeitkomplexität.

ein Gebirge im Weg ist und so weiter. Unser Problem ist jetzt, die billigste Vernetzung zu finden. Dazu zeichnen wir uns die möglichen Verbindungen und ihre Kosten in einem Graph auf, wie er in Abbildung (2) zu sehen ist.

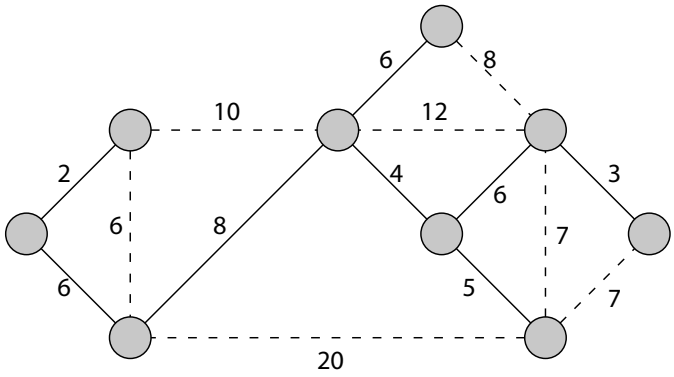

Abbildung 2: Ein gewichteter Graph als Beispiel für das MST-Problem

Die allgemeine mathematische Formulierung dieses Problems ist die: Gegeben seien die Knoten V und die Kanten E in einem gewichteten Graph $G(V, E)$. Gesucht ist derjenige Teilgraph $T \subseteq G$, der alle Knoten verbindet und das geringste Gesamtgewicht hat. Das Problem wird als *MST (minimum spanning tree)* bezeichnet.

Wie beim SAT würde auch hier das Durchprobieren aller möglichen Graphen exponentiell viel Zeit verschlingen, denn es gibt n^{n-2} verschiedene Möglichkeiten, n Knoten zu verbinden. Anders als beim SAT lässt sich aber ein einfacher Algorithmus finden, der in Abbildung (3) dargestellt ist. Er wird mit einem gewichteten Graphen G aufgerufen und lässt in T nach und nach den Teilgraphen wachsen, bis daraus der MST entstanden ist. In Abbildung (2) sind die Linien, die zum MST gehören, durchgehend gezeichnet. Die Zeitkomplexität variiert je nachdem, wie man die Datenstruktur gestaltet, die den Graphen repräsentiert. Eine obere Grenze ist aber $\mathcal{O}(n^2 \log n)$ (siehe [5]). Damit gehört das MST-Problem in die $\mathcal{P}$-Klasse der leichten Probleme.

```
1: procedure PRIM'S ALGORITHMUS(G)
2:     T ← beliebiger Knoten v aus G;          ▷ Initialisierung des Teilgraphen T
3:     while T verbindet weniger als n Knoten do
4:         Finde die Kante mit den geringsten Kosten, die T und T − G verbindet;
5:         Füge diese Kante zu T hinzu;
6:     end while
7: end procedure
```

Abbildung 3: Prim's Algorithmus zur Lösung des MST-Problems

3. Analytischer Teil

3.1. Das $\mathcal{NPP}$ als „easy“ und „hard“ Problem

Obwohl das $\mathcal{NPP}$ Problem zu der Klasse der $\mathcal{NP}$-vollständigen Probleme gehört, gibt es eine Reihe von Beispielen, in denen man mit einfachen Algorithmen Lösungen finden kann (vgl. Abschnitt (4). Vor dem Hintergrund dieser Erfahrungen hat sich vor einigen Jahren eine Forschungskampagne, zusammengesetzt aus Physikern, Mathematikern und Informatikern, mit dem Zahlenaufteilungsproblem beschäftigt. Man fand heraus, dass es beim Zahlenaufteilungsproblem sowohl einfach als auch schwer zu lösende Fälle gibt. Bei der Standardklassifikation in $\mathcal{P}$ und $\mathcal{NP}$ nimmt man an, dass die Schwierigkeit, ein Problem zu lösen als Funktion der Größe des Problems anwächst.
Beim $\mathcal{NPP}$ bestimmen zwei Faktoren die Größe eines speziellen Problems, die Anzahl und die Größe der aufzuteilenden Zahlen. Die Größe eines bestimmten Problems ist definiert durch die Anzahl von Bits, die zu dessen Darstellung nötig sind. Diese hängt wiederum von der Anzahl der (Integer-) Zahlen N und der Anzahl von Bits zur Repräsentation der jeweiligen Zahlen M ab.
Für eine Menge aus 100 Integerzahlen mit einer durchschnittlichen Größe von 2^{100}, ist beispielsweise $N = M = 100$ und die Größe des Problems ergibt sich damit zu $10^4\ bits$. Nun könnte man annehmen, dass die Lösbarkeit des Problems schwieriger wird, wenn das Produkt aus N und M anwächst. Dies ist zwar nicht ganz falsch, da Algorithmen aufgrund der Einlesezeit generell mehr Zeit für größere Probleme benötigen, überraschenderweise gibt jedoch nicht das Produkt $M \cdot N$, sondern das Verhältnis M/N die beste Auskunft über die Lösbarkeit des Problems.
Dies sieht man wie folgt ein: Hat man viele kleine Zahlen gegeben, so besteht eine sehr große Flexibilität bei der Suche nach perfekten Partitionen, denn es kann viele verschiedene Verteilungen geben, welche die Kostenfunktion E minimieren. Mit größer werdenden Zahlen nimmt diese Flexibilität ab. Sind wenige große Zahlen gegeben, so reduziert sich die Anzahl der möglichen perfekten Konfigurationen immer weiter, unter Umständen gibt es gar keine.
Argumentativ kann man schon im Vorhinein eine recht gute Abschätzung für den Ort des Phasenübergangs finden. Gehen wir davon aus, dass der Phasenübergang genau dann stattfindet, wenn die Anzahl möglicher Kostenfunktionen (Diskrepanzen) gerade gleich der Anzahl möglicher Verteilungen ist. Die Anzahl der Konfigurationen beträgt bei N Gewichten 2^N. Man erhält nun eine Bedingung für die Existenz einer perfekten Partition: Die Anzahl der potentiellen Diskrepanzen muss kleiner sein als die Anzahl der möglichen Partitionen (vergleiche [9]).
Wir betrachten eine aufzuteilende Menge $\mathcal{A}$, bestehend aus natürlichen, gleichverteilten Zufallszahlen a_j kleiner oder gleich $a_{max} = 2^M$:

$$\begin{aligned} \sum_{i=0}^{\sum_{j=1}^{N} a_j} 1 &< 2^N \\ \log_2\left(1 + \sum_{j=1}^{N} a_j\right) &< N. \end{aligned} \tag{3}$$

Mit $a_j \approx \frac{1}{2} a_{max} = \frac{1}{2} 2^M$ folgt

$$\begin{aligned} \log_2\left(\frac{N}{2} 2^M\right) &< N \\ M &< N - \log_2\left(\frac{N}{2}\right). \end{aligned} \tag{4}$$

Falls die Anzahl der Zahlen N gegen unendlich geht, findet man perfekte Partitionen, solange die Zahl der Bits, die zur Kodierung der Zahlen notwendig sind, kleiner als N ist. Für endliche N erhält man logarithmische Korrekturen.
Können wir jedoch für eine gegebene Anzahl N und feste „Bitgöße“ der Zahlen keine perfekte Lösung finden, so gibt es immer genau eine Lösung, welche die minimale Abweichung von einer gleichmäßigen Aufteilung in zwei Hälften angibt.
Das Verhältnis M/N teilt also den Raum des $\mathcal{NPP}$ in zwei Gebiete ein, wobei in dem einen eine Vielfalt an perfekten Lösungsmöglichkeiten, in dem anderen gar keine perfekte Lösung existiert. Es muss also ein Übergangsgebiet existieren, in dem ein *Phasenübergang* stattfindet.

3.2. Phasenübergänge

Das Phänomen der Phasenübergänge ist uns aus dem Alltag gut bekannt. Ändert sich beispielsweise die Temperatur von Wasser, so ändert sich bei bestimmten Temperaturen sprunghaft die Dichte. Allgemein ist ein Phasenübergang erster Ordnung durch die sprunghafte Änderung eines sogenannten Ordnungsparameters (in unserem Beispiel die Dichte, allgemein ein Ordnungsgrad) charakterisiert, wenn ein *Kontrollparameter* κ (z. B. die Temperatur) einen kritischen Wert κ_c überschreitet.
Es existieren viele $\mathcal{NP}$-vollständige Probleme, für die man einen Phasenübergang nachweisen kann. Ob der in Abschnitt (3.1) proklamierte Phasenübergang beim $\mathcal{NPP}$ tatsächlich stattfindet, war lange Zeit umstritten. Yaotian Fu untersuchte (in [1]) die thermodynamischen Eigenschaften des Zahlenaufteilungsproblems und erklärte, dass es sich hier um ein $\mathcal{NP}$-vollständiges Problem ohne Phasenübergang handele. Mertens [7] fand jedoch heraus, dass man auch bei dem Zahlenaufteilungsproblem einen Phasenübergang feststellen kann. In der einen Phase existieren exponentiell viele, in der anderen Phase keine perfekten Partitionen, entsprechend einer einfach zu lösenden und einer sehr schwer lösbaren Phase. Im Folgenden wollen wir diese Phasen mit „easy“ und „hard“ bezeichnen.
Der Ordnungsparameter ist beim Zahlenaufteilungsproblem gegeben durch die Wahrscheinlichkeit, dass eine perfekte Partition existiert. Der Kontrollparameter muss dabei, wie in Abschnitt (3.1) beschrieben, eine Funktion sein, die von der Größe und der Anzahl der Zahlen abhängt.

3.3. „Antimagnetische“ Zahlen

Um zu verstehen, was ein Phasenübergang bei einem Problem ohne direkten physikalischen Bezug bedeutet, schlagen wir einen Bogen und ziehen einen Vergleich zum Modell eines physikalischen Systems. Hierzu ändern wir unsere Vorgehensweise bei der Verteilung der Zahlen (mathematisch) etwas ab. Anstatt eine Menge von Zahlen in zwei separate Mengen zu unterteilen, bleiben wir bei der einen ursprünglichen Menge und multiplizieren eine Teilmenge davon mit -1. Die Aufgabe ist nun, gerade die zu negierende Teilmenge zu finden, so dass sich die Gesamtsumme gerade möglichst wenig von Null unterscheidet. Der Bezug zur Physik ist nun dadurch gegeben, dass sich die Ansammlung von positiven und negativen Zahlen vergleichen lässt mit der Anordnung von Atomen in magnetischer Materie, wobei „plus“ und „minus“ gerade „spin up“, bzw. „spin down“ entsprechen. Konkret spiegelt unser System einen unendlich dimensionalen Antiferromagneten wieder, in dem jedes Atom den Einfluss des anderen Atoms spürt und in welchem die angestrebte Anordnung gerade derart ist, dass alle Spins in entgegengesetzte Richtungen orientiert sind.
Mathematisch können wir diese Situation mit folgender Hamiltonfunktion ausdrücken:

$$H = E^2 = \sum_{i,j=1}^{N} s_i a_i a_j s_j \tag{5}$$

Mit Hilfe der statistischen Mechanik können nun anhand des Wechselspiels zwischen Entropie und Energie Aussagen über den Gleichgewichtszustand des Spinsystems getroffen werden. Die Energie stammt von dem Zusammenspiel der Atomspins, bzw. in unserem Fall zwischen den Zahlen unterschiedlichen Vorzeichens. Sie wird minimiert, wenn die verschiedenen Spins gerade entgegengesetzt ausgerichtet sind, bzw. die Summe der Teilmengen Null ergibt. Die Entropie ist ein Maß für die Anzahl der Realisierungsmöglichkeiten, welche zu dieser Grundzustandsenergie gehören. Für ein System mit einem eindeutigen Grundzustand (oder einer einzigen perfekten Unterteilung) ist die Entropie Null, während bei einer großen Lösungsvielfalt die Entropie groß wird.
Wie in Abschnitt (3.2) bereits erläutert, muss der Kontrollparameter eine Funktion von Anzahl und Größe der Zahlen sein, wobei insbesonders gilt

$$f = M/N \tag{6}$$

(M = Anzahl der Bits, N = Anzahl der Integers). In Anlehnung an Mertens [7] zeigen wir in den folgenden Abschnitten, dass der Phasenübergang beim $\mathcal{NPP}$ im Limes großer N bei dem Verhältnis $M/N = 1$ liegt. Der etwas geringere Wert von $0{,}96$ in vorangegangen Arbeiten von Gent und Walsh [2] entstand dadurch, dass der Limes für $N \to \infty$ nicht streng genug überprüft wurde.

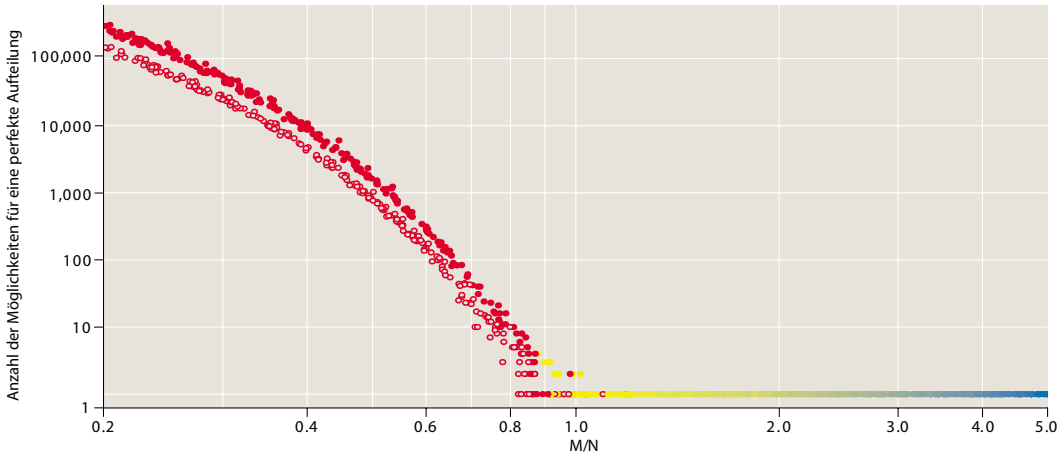

Abbildung 4: Anzahl der möglichen perfekten Partitionen in Abhängigkeit von M/N (entnommen aus [4])
Hierbei ist $N = 25$ und $M \in [5, \ldots, 125]$, wobei jeder Punkt der Mittelung über 1000 Instanzen entspricht. Ausgefüllte Punkte stehen für eine Summendifferenz von 1, leere Punkte für eine Summendifferenz von 0.

3.4. Die Bedeutung der Zustandssumme für Optimierungsprobleme

Für Optimierungsprobleme, wie das Zahlenaufteilungsproblem, ist vor allem der Grundzustand des Systems interessant. Dieses System habe diskrete Energieeigenzustände $E_0 < E_1 < E_2 < ...$ mit den Multiplizitäten $n_0, n_1, n_2, ...$
Die freie Energie enthält die Information über sämtliche mögliche Mikrozustände. Bezeichnen x die

Nebenbedingungen und s die Anzahl möglicher Mikrozustände, so ergibt sie sich zu

$$\begin{aligned} F(T) &= -T \ln Z \\ &= -T \ln \sum_{s \in S} \exp\left(-\frac{H(s,x)}{T}\right) \\ &= -T \ln \left(n_0 \exp\left(-\frac{E_0}{T}\right) \left(1 + \frac{n_1}{n_0} \exp\left(-\frac{E_1 - E_0}{T}\right) + ...\right)\right) \\ &= E_0 - T \ln n_0 - T \ln \left(1 + \frac{n_1}{n_0} \exp\left(-\frac{E_1 - E_0}{T}\right) + ...\right). \end{aligned} \tag{7}$$

Somit erhalten wir für die Grundzustandsenergie E_0 und deren Multiplizität n_0:

$$E_0 = \lim_{T \to 0} F(T) \tag{8}$$

und

$$\ln n_0 = \lim_{T \to 0} -\frac{\delta}{\delta T} F(T) = \lim_{T \to 0} S(T), \tag{9}$$

wobei S ist die (kanonische) Entropie ist.

3.5. Kanonische Behandlung des Zahlenaufteilungsproblems

Um auf das Zahlenaufteilungsproblem den Formalismus der statistischen Mechanik anwenden zu können, müssen wir zunächst eine Hamilton-Funktion angeben:

$$H = E = \left| \sum_{j=1}^{N} s_j a_j \right| \tag{10}$$

Die Wahl dieser Funktion ist dabei nicht eindeutig. Man beschränkt sich bei ihrer Wahl jeweils auf ein bestimmtes Phasenraumvolumen.
Wir wollen die thermodynamischen Eigenschaften dieses Problems, in Analogie zu den von Mertens in [7] veröffentlichten Ergebnissen, im kanonischen Formalismus betrachten. Hierzu muss künstlich eine Temperatur eingeführt werden. Das Tieftemperaturverhalten beschreibt dann die optimalen Partitionen des $\mathcal{NPP}$.
Aus der Hamilton-Funktion (s. Gl. (10)) berechnen wir zunächst die Zustandssumme[4]

$$Z = \sum_{\{s_i\}} \exp(-\beta \cdot E) = \sum_{\{s_i\}} \exp(-\frac{E}{T}) \tag{11}$$

$$= \sum_{\{s_i\}} \int_{-\infty}^{\infty} dx \exp(-|x|) \delta \left(x - \frac{1}{T} \sum_{j=1}^{N} a_j s_j \right), \tag{12}$$

wobei wir die $\delta-$Funktion über ihr Fourierintegral definieren:

$$\delta(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\hat{x} \exp\left(ix\hat{x}\right). \tag{13}$$

[4] vgl. Anhang, Abschnitt (A.2)

Damit erhalten wir

$$\begin{aligned} Z &= \sum_{\{s_i\}} \int_{-\infty}^{\infty} \exp(-|x|) \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp\left(i \left(x - \frac{1}{T} \sum_{j=1}^{N} a_j s_j \right) k \right) dk dx \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \underbrace{\sum_{s_i} \exp\left(-\frac{ik}{T} \sum_{j=1}^{N} a_j s_j \right)}_{:=a} \underbrace{\int_{-\infty}^{\infty} \exp\left(-|x| + ikx\right) dx}_{:=b} dk \end{aligned} \tag{14}$$

Wir berechnen die Terme a und b getrennt. Umformen ergibt für a:

$$\begin{aligned} a &= \sum_{s_i} \exp\left(-\frac{ik}{T} \sum_{j=1}^{N} a_j s_j \right) = \sum_{s_i} \prod_{j=1}^{N} \exp\left(-\frac{ik}{T} a_j s_j \right) \\ &= \prod_{j=1}^{N} \sum_{s_i} \exp\left(-\frac{ik}{T} a_j s_j \right) = 2^N \prod_{j=1}^{N} \sum_{s_i} \frac{1}{2} \exp\left(-\frac{ik}{T} a_j s_j \right) \\ &= 2^N \prod_{j=1}^{N} \cos\left(\frac{k}{T} a_j \right). \end{aligned} \tag{15}$$

Zur Auswertung des Integrals b

$$b = \int_{-\infty}^{\infty} \exp\left(-|x| + ikx\right) dx \tag{16}$$

schreiben wir

$$\begin{aligned} b &= \int_{-\infty}^{0} dx \exp(x + ikx) + \int_{\infty}^{0} dx \exp(x + ikx) \\ &= \frac{1}{1 + ik} + \frac{1}{1 - ik} \\ &= \frac{2}{1 + k^2} \end{aligned} \tag{17}$$

Substituieren wir nun

$$k = \tan y \qquad \Rightarrow \qquad dk = dy \left(1 + \tan^2 y\right), \tag{18}$$

so ergibt sich für die Zustandssumme

$$\begin{aligned} Z &= 2^N \int_{-\pi/2}^{\pi/2} \frac{1 + \tan^2 y}{2\pi} \prod_{j=1}^{N} \cos\left(\frac{a_j}{T} \tan y \right) \int_{-\infty}^{\infty} \exp\left(-|x| + ikx\right) dx dy \\ &= 2^N \int_{-\pi/2}^{\pi/2} \frac{1}{\pi} \prod_{j=1}^{N} \cos\left(\frac{a_j}{T} \tan y \right). \end{aligned} \tag{19}$$

Wir schreiben Z in der Form

$$Z = 2^N \int_{-\pi/2}^{\pi/2} \frac{1}{\pi} \exp\left(NG(y)\right) dy \tag{20}$$

mit

$$G = \frac{1}{N} \sum_j \ln \cos \left(\frac{a_j}{T} \tan y \right). \tag{21}$$

Das Integral in Gleichung (19) lässt sich mit Hilfe der Sattelpunktmethode auswerten.[5]
Um die Sattelpunkte zu finden, nehmen wir vereinfachend an, dass a nur Werte annehmen kann, die sich als Vielfaches einer bestimmten reellen Zahl Δa darstellen lassen. Δa kann als ein Maß für die Auflösung angesehen werden. Für ganzzahlige Verteilungen ist $\Delta a = 1$, für Fließkommazahlverteilungen ist Δa die kleinste Zahl, die durch die zur Verfügung stehende Anzahl an Bits ausgedrückt werden kann.
Die Sattelpunkte liegen dann bei

$$y_k = \arctan \left(\frac{\pi T}{\Delta a} k \right), \qquad k = 0, \pm 1, \pm 2, \dots . \tag{22}$$

Für die Ableitungen von G nach y erhalten wir

$$G' = \sum_j -\frac{1}{N} \tan \left(\frac{a_j}{T} \tan y \right) \cdot \frac{a_j}{T} \left(1 + \tan^2 y \right) \tag{23}$$

$$\begin{aligned} G'' &= \sum_j -\frac{1}{N} \left(1 + \tan^2 \left(\frac{a_j}{T} \tan y \right) \right) \left(\frac{a_j}{T} \left(1 + \tan^2 y \right) \right)^2 \\ &\quad -\frac{1}{N} \cdot \frac{a_j}{T} \tan \left(\frac{a_j}{T} \tan y \right) \left(2 \tan y \left(1 + \tan^2 y \right) \right) \end{aligned} \tag{24}$$

und damit

$$G''(y_k) = 2^N \sum_j -\frac{1}{N} \left(1 + \left(\frac{\pi T}{\Delta a} k \right)^2 \frac{a_j}{T} \right)^2 . \tag{25}$$

Die Zustandssumme ergibt sich zu einer Reihe von Gaußintegralen:

$$\begin{aligned} Z &= \frac{2^N}{\pi} \sum_{k=0,\pm 1,\pm 2,\dots} \int_{\mathbb{R}} \exp \left(\frac{N}{2} G''(y_k) \left(y - y_k \right)^2 \right) dy \\ &\overset{\text{Gauß}}{=} \frac{2^N}{\pi} \sum_{k=0,\pm 1,\pm 2,\dots} \sqrt{-\frac{2\pi}{N G''}} \\ &= 2^N \sum_{k=0,\pm 1,\pm 2,\dots} \sqrt{\frac{2T^2}{\pi \sum a_j^2}} \cdot \frac{1}{1 + \left(\frac{\pi T}{\Delta a} k \right)^2}. \end{aligned} \tag{26}$$

Mit Hilfe der Identität

$$\sum_{k=0,\pm 1,\pm 2,\dots} \frac{1}{1 + (xk)^2} = \frac{\pi}{x} \coth \frac{\pi}{x} \tag{27}$$

erhalten wir

$$Z = 2^N \sqrt{\frac{2 \cdot \Delta a^2}{\sum a_j^2 \cdot \pi}} \cdot \coth \frac{\Delta a}{T}. \tag{28}$$

[5] vgl. Anhang, Abschnitt (A.3)

Daraus ergibt sich die mittlere Energie[6] mit

$$\frac{\delta}{\delta\beta}\coth(\Delta a\beta) = \left(1-\coth^2(\Delta a\beta)\right)\Delta a \tag{29}$$

zu

$$\begin{aligned}\frac{E}{T} &= -\frac{1}{T}\cdot\frac{\delta}{\delta\beta}\ln Z \\ &= -\frac{1}{T}\cdot\frac{\delta}{\delta\beta}\ln\left(\coth(\Delta a\beta)\cdot 2^N\sqrt{\frac{2\cdot\Delta a^2}{\sum a_j^2\cdot\pi}}\right) \\ &\overset{\beta=1/T}{=} \frac{\Delta a}{T}\frac{\coth^2\frac{\Delta a}{T}-1}{\coth\frac{\Delta a}{T}}.\end{aligned} \tag{30}$$

Für die Entropie gilt

$$S = \frac{E-F}{T}, \tag{31}$$

wobei die freie Energie F gegeben ist durch[6]

$$F = -\frac{1}{\beta}\ln Z.$$

Einsetzen von Gleichung (28) ergibt

$$F = -T\left(N\ln 2 + \frac{1}{2}\ln\left(\frac{2\Delta a^2}{\pi\sum_j a_j^2}\right) + \ln\coth\frac{\Delta}{T}\right)$$

und damit folgt für die Entropie aus Beziehung (31)

$$S = N\ln 2 - \frac{1}{2}\ln\left(\frac{\pi\sum_j a_j^2}{2\Delta a^2}\right) + \widetilde{S}\left(\frac{\Delta a}{2T}\right), \tag{32}$$

mit

$$\widetilde{S}\left(\frac{\Delta a}{2T}\right) = \ln\coth\frac{\Delta}{T} + \frac{\Delta a}{T}\frac{\coth^2\frac{\Delta a}{T}-1}{\coth\frac{\Delta a}{T}}. \tag{33}$$

Im Limes kleiner Temperaturen, $T \to 0$ bei endlichen Werten von Δa geht $\widetilde{S}$ gegen Null. Um dies einzusehen, drücken wir den $\coth$ mit Hilfe der Exponentialfunktion aus:

$$\begin{aligned}\lim_{T\to 0}\widetilde{S} &= \lim_{x\to\infty}\ln\left(\frac{\exp(2x)+1}{\exp(2x)-1}\right) + x\left(\frac{\exp(2x)+1}{\exp(2x)-1} - \frac{\exp(2x)-1}{\exp(2x)+1}\right) \\ &= 0\end{aligned} \tag{34}$$

[6] vgl. Anhang, Abschnitt (A.2)

Im Folgenden soll die Entropie näher diskutiert werden.

$$\begin{aligned} S &= N\ln 2 - \frac{1}{2}\ln\left(\frac{\pi\sum_j a_j^2}{2\Delta a^2}\right) + \widetilde{S} \\ &= N\ln 2 - \frac{1}{2}\ln\left(\frac{\pi N}{6}\cdot 3\frac{\sum_j a_j^2}{\Delta a^2 N}\right) + \widetilde{S} \\ &= N\ln 2\left(\underbrace{1 - \frac{\ln\left(\frac{\pi}{6}N\right)}{2N\ln 2}}_{\kappa_c(N)} - \underbrace{\frac{\ln\left(\frac{3}{\Delta a^2 N}\sum_j a_j^2\right)}{2N\ln 2}}_{\kappa(\Delta a,\,N)}\right) + \widetilde{S} \\ &= N\ln 2\cdot(\kappa_c - \kappa) + \widetilde{S} \end{aligned} \tag{35}$$

mit

$$\kappa_c(N) = 1 - \frac{\ln\left(\frac{\pi}{6}N\right)}{2N\ln 2} \tag{36}$$

und

$$\kappa(\Delta a,\,N) = \frac{\ln\left(\frac{3}{\Delta a^2 N}\sum_j a_j^2\right)}{2N\ln 2}. \tag{37}$$

Dass diese Umformung sinnvoll ist, sieht man folgendermaßen ein. Betrachten wir zunächst den Fall $\kappa < \kappa_c$. Für $T = 0$ ist die Entropie hier eindeutig extensiv. Die Energie ist nach Gleichung (29) Null. Nach Gleichung (9) berechnet sich die Anzahl perfekter Verteilungen in Übereinstimmung mit numerischen Experimentenzu $2^{N(\kappa_c-\kappa)}$.
Im Bereich $\kappa > \kappa_c$ ergäbe sich bei $T = 0$ nach Gleichung (35) jedoch eine negative Entropie, was der minimal möglichen Entropie (im Limes $T \to 0$), in unserem konkreten Beispiel $S_{min} = \ln 2$, widerspricht.
Nach Gleichung (36) und (37) ist $\kappa > \kappa_c$ gleichbedeutend mit

$$\begin{aligned} \ln\left(\frac{3}{\Delta a^2 N}\sum_j a_j^2\right) &> 2N\ln 2 - \ln\left(\frac{\pi}{6}N\right) \\ \ln 2^{-N} &> \frac{1}{2}\ln\frac{2\Delta a^2}{\pi\sum_j a_j^2} \\ 2^{-N} &> \Delta a\sqrt{\frac{2}{\pi\cdot\sum_j a_j^2}}. \end{aligned} \tag{38}$$

Somit ist $\Delta a = \mathcal{O}\left(2^{-N}\right)$, das bedeutet $\Delta a \to 0$ für $N \to \infty$, und für endliche T ist $\widetilde{S} = \mathcal{O}(N)$.

$$\begin{aligned} \widetilde{S}\left(\frac{\Delta a}{2T}\right) &= \ln\coth\frac{\Delta a}{T} + \frac{\Delta a}{T}\frac{\coth^2 - 1}{\coth\frac{\Delta a}{T}} \\ &= \ln\frac{\cosh\frac{\Delta a}{T}}{\sinh\frac{\Delta a}{T}} + \frac{\Delta a}{T}\left(\frac{\cosh\frac{\Delta a}{T}}{\sinh\frac{\Delta a}{T}} - \frac{\sinh\frac{\Delta a}{T}}{\cosh\frac{\Delta a}{T}}\right) \\ &\approx \ln\frac{T}{\Delta a} + \frac{\Delta a}{T}\left(\frac{T}{\Delta a} - \frac{\Delta a}{T}\right) \\ &= \ln\frac{T}{\Delta a} + 1 + \mathcal{O}\left(\left(\frac{\Delta a}{T}\right)^2\right). \end{aligned} \tag{39}$$

Hierbei wurde die Taylorentwicklung des $\sinh$, bzw. $\cosh$ in erster Ordnung verwendet.
Wir sehen also, dass $\widetilde{S}$ für endliche Temperaturen nicht mehr vernachlässigt werden kann. Man berücksichtigt diesen Term durch Einführen einer *effektiven* Nulltemperatur T_0.
Aus

$$S_{min} = \ln 2 = N \ln 2(\kappa_c - \kappa) + \widetilde{S} \overset{(39)}{=} N(\kappa_c - \kappa) \ln 2 + \ln\left(\frac{T_0}{\Delta a}\right) \tag{40}$$

folgt

$$\begin{aligned} \ln 2\,(1 + N(\kappa - \kappa_c)) &= \ln \frac{T_0}{\Delta a} \\ \ln 2 + \ln 2^{N(\kappa-\kappa_c)} &= \ln \frac{T_0}{\Delta a} \end{aligned}$$

und somit ergibt sich die effektive Nulltemperatur durch Einsetzen von κ und κ_c zu

$$T_0 = 2\Delta a 2^{N(\kappa-\kappa_c)} = \sqrt{2\pi \sum_j a_j^2} 2^{-N}. \tag{41}$$

Daraus folgt unmittelbar die Grundzustandsenergie für $\kappa > \kappa_c$

$$\begin{aligned} E_0 &= T_0 = \sqrt{2\pi \sum_j a_j^2} 2^{-N} \\ &= \sqrt{2\pi \left\langle a_j^2 \right\rangle \cdot N} 2^{-N}. \end{aligned} \tag{42}$$

Durch das Einführen der effektiven Nulltemperatur erhalten wir eine positive Entropie. Dies zeigt, dass unser Ansatz in sich stimmig ist und der scheinbare Widerspruch sich als nichtig erweist.

4. Numerische Darstellung

Da $\mathcal{NPP}$ ein $\mathcal{NP}$-Problem ist, können wir nicht erwarten, einen Algorithmus zu finden, der das Entscheidungsproblem des $\mathcal{NPP}$, die Frage ob eine perfekte Aufteilung überhaupt existiert, in polynomialer Zeit löst. Wie wir schon in Abschnitt (3.1) angedeutet haben, gibt es aber durchaus Algorithmen, die gute Näherungslösungen des $\mathcal{NPP}$ in polynomialer Zeit finden. Diese sind für konkrete Anwendungen ausreichend, bei denen es weniger darauf ankommt, irgendwann die perfekte Lösung zu finden, sondern bei denen schnell eine gute Lösung gefunden werden soll. Eine solche Vorgehensweise nennt man Heuristik. Wichtig ist, jeweils zu unterscheiden, ob die von einem Algorithmus gefundenen Aufteilungen ausgewogen sind oder nicht. Im Folgenden wollen wir einige dieser Algorithmen vorstellen.

4.1. Algorithmen

Greedy-Algorithmus Der einfachste Algorithmus, mit dem sich eine vorgegebene Menge von Zahlen aufteilen lässt, ist der „greedy-Algorithmus“. Er erinnert an das Verfahren, wie die Mannschaftsaufteilung in der Schule zustandezukommen pflegte: Es werden zwei Mannschaftskapitäne ernannt, die solange abwechselnd einen Spieler in ihre Mannschaft wählen dürfen, bis jeder einen Platz gefunden hat. Dabei werden die Mannschaftskapitäne – abgesehen von ihren persönlichen Sympathien und Antipathien – wahrscheinlich immer den stärksten der verbliebenen Spieler auswählen. Genauso arbeitet der greedy-Algorithmus, mit einer kleinen Ausnahme. Während die Mannschaften meist auch von der Anzahl der Spieler her gleich stark sein sollen, kümmert sich der greedy-Algorithmus nicht um eine ausgewogene Aufteilung.

```
1:  procedure GREEDY(x[1..k])
2:      SORT(x[1..k]);                          ▷ Aufsteigend sortieren
3:      for i ← k, ..., 1 do
4:          if ∑ Haufen 1 < ∑ Haufen 2 then
5:              Lege x_i auf Haufen 1
6:          else
7:              Lege x_i auf Haufen 2
8:          end if
9:      end for
10: end procedure
```

Abbildung 5: Der greedy-Algorithmus

Ein Blick auf Abbildung (6) zeigt, dass der Greedy-Algorithmus noch weit vom Optimum entfernt ist. Die wichtigste Grundidee bei der Suche nach besseren Algorithmen besteht in der sogenannten Differenzoperation, bei der zwei der zu verteilenden Zahlen durch ihre Differenz ersetzt werden, da wir uns nur für die Summendifferenz der beiden Hälften interessieren. Dies entspricht der Entscheidung, diese beiden Zahlen auf unterschiedliche Haufen zu verteilen. Wir betrachten im folgenden zwei Algorithmen, die auf diesem Prinzip beruhen.

Paired Differencing Method Bei der *„Paired Differencing Method“* (PDM) müssen die n aufzuteilenden Zahlen sortiert vorliegen. Man führt nun $\lfloor n/2 \rfloor$ Differenzoperationen durch, und zwar auf die beiden größten Zahlen, auf die dritt- und viertgrößte Zahl, etc. Es wird immer die kleinere von der

größeren abgezogen. Dadurch erhält man $\lceil n/2 \rceil$ neue Elemente ($x_1 - x_2$, $x_3 - x_4$, etc.), die sortiert werden, und mit denen man diese Prozedur wiederholt, bis man nur noch ein Element übrig behält. Dessen Wert entspricht dann der Differenz der gefundenen Aufteilung. Welche Zahlen auf welchem Haufen gelandet sind, kann man danach aus den durchgeführten Operationen schlussfolgern. In Tabelle (1) ist ein Beispiel dargestellt, welches das Vorgehen der PDM veranschaulicht. Tabelle (2) zeigt die Rekonstruktion der Aufteilung der Zahlen auf die beiden Teilmengen, von unten beginnend.

aufzuteilende Zahlenmenge	Operation
(2) (3) (4) (7) (15) (21) (24)	24-21=3, 15-7=8, 4-3=1. Sortieren.
(1) (2) (3) (8)	8-3=5, 2-1=1. Sortieren.
(1) (5)	5-1=4
(4)	Dies ist die gefundene Aufteilungsdifferenz.

Tabelle 1: Beispiel für den Ablauf der PDM für $n = 7$ Zahlen.
Jeweils zwei Zahlen werden durch ihre Differenz ersetzt. Da wir zunächst eine ungerade Anzahl von Zahlen haben, wird die überschüssige 2 unverändert in den 2. Schritt übernommen. Dort haben wir dann $\lceil n/2 \rceil = 4$ Zahlen.

Menge $\mathcal{B}_1$	Menge $\mathcal{B}_2$	Ersetzungsschritt
(4)		Die (4) entstand aus (5) und (1).
(5)	(1)	(5) aus (8) und (3), ...
(8)	(3) (1)	... (1) aus (2) und (1).
(8) (1)	(3) (2)	Und so weiter ...
(21) (15) (4)	(24) (7) (3) (2)	Differenz der Teilmengen: $40 - 36 = 4$ ✓

Tabelle 2: Die Aufteilung der Zahlen auf die beiden Teilmengen ergibt sich aus Tabelle (1), indem von unten beginnend jede Zahl wieder durch die beiden Zahlen ersetzt wird, aus deren Differenz sie gebildet wurde. Die größere der beiden kommt in die Menge, in der die Differenz steht, die kleine in die andere.

Largest Differencing Method Die *„Largest Differencing Methode“* (LDM) basiert ebenso wie die PDM auf der Differenzoperation. Auch hier werden die Zahlen der Größe nach behandelt: Die beiden größten Zahlen werden durch ihre Differenz ersetzt und dann wird - im Unterschied zur PDM - sofort neu einsortiert. Man fährt solange fort, bis nur noch ein Element übrig ist, das wieder der Teilungsdifferenz entspricht. Die eigentliche Verteilung der Zahlen erhält man wie bei der PDM rückwärts. Der Vorteil der LDM gegenüber der PDM sind kleinere Summendifferenzen der entstehenden Teilmengen. Nachteil der LDM ist, dass die entstehenden Teilmengen in Bezug auf die Anzahl der Elemente nicht ausgewogen sind, während die PDM ausschließlich ausgewogene Teilungen liefert. Bei der PDM werden in jedem Schritt in beide Teilmengen Elemente hinzugefügt, anders als bei der LDM, bei der er sein kann, dass immer wieder nur auf eine Seite neue Elemente „abgespalten“ wer-

den. Man betrachte hierfür zum Beispiel die Zahlenmenge $(2,\ 3,\ 4,\ 8,\ 23)$. Die PDM ergibt hierfür eine Summendifferenz E von 12, bei einer Anzahldifferenz m von 1. (Da $n = 5$ ungerade ist, ist dies „ausgewogen“.) Die LDM ergibt $E = 6$ mit $m = 3$, die bestmögliche unausgewogene Aufteilung in $(23), (2,\ 3,\ 4,\ 8)$. Die bestmögliche ausgewogene Aufteilung wäre $(3,\ 4,\ 8), (2,\ 23)$ mit $E = 10$.

Balanced Largest Differencing Method Die *„Balanced Largest Differencing Method“* (BLDM) wurde von Yakir (siehe [14]) entwickelt und ist eine Verbesserung der LDM, die darin besteht – wie der Name schon vermuten lässt – dass sie ausgewogene Teilungen ergibt. Der Trick besteht darin, dass zunächst ein Durchgang der PDM durchgeführt wird. Die erhaltenen $\lceil n/2 \rceil$ Zahlen verarbeitet man dann mit der LDM weiter, bis man wieder nur noch ein Element übrig behält. Die wieder rückwärts zu erhaltende Aufteilung ist dadurch ausgewogen, aber man erhält trotzdem eine Summendifferenz, wie sie bei der LDM zu erwarten ist.

Angewendet auf unser obiges Beispiel $(2\ ,3\ ,4\ ,8\ ,23)$ wird die optimale Aufteilung allerdings auch von der BLDM nicht gefunden, da durch den ersten PDM-Schritt bereits die 3 und die 4 in unterschiedliche Mengen verteilt werden. Wir sehen also, dass wir auch von der BLDM wie eingangs erwähnt nur Näherungslösungen erwarten können, sonst würden wir $\mathcal{NPP}$ nicht der Klasse $\mathcal{NP}$ zuordnen. Die Zeitkomplexität aller drei Methoden, der PDM, der LDM wie auch der BLDM, ist wegen der nötigen Suchvorgänge in der Ordnung $\mathcal{O}(n \log n)$. Im folgenden Abschnitt werden wir einen Algorithmus kennenlernen, der garantiert die beste Lösung findet - lässt man ihm genügend Zeit.

4.2. Vergleich der bisherigen Algorithmen

Mertens (in [6]) nennt unter Verweis auf andere Quellen zu erwartende Summendifferenzen von $\Theta(n^{-1})$ für die PDM und $n^{-\Theta(\log n)}$ für die LDM[7]. Anhand des vergleichenden Graphen in Abbildung (6) lässt sich dies qualitativ bestätigen. Wir sehen, dass der einfachste Algorithmus, der Greedy Algorithmus, (natürlich) am schlechtesten abschneidet. Wenig besser ist die PDM, der allerdings zugute gehalten werden muss, dass sie im Gegensatz zum Greedy-Algorithmus auf ausgewogene Aufteilungen eingeschränkt ist. Die LDM ist dies nicht und dadurch deutlich im Vorteil, aber auch die BLDM, die wie die PDM ausgewogene Aufteilungen erzeugt, ist durch die oben beschriebene geschickte Kombination von PDM und LDM nur wenig schlechter als die LDM. Die cBLDM, die wir als Höhepunkt der hier behandelten Algorithmen kennenlernen werden, mit in diese Grafik aufzunehmen, ist eigentlich unfair, denn hierbei handelt es sich nicht wie bei den anderen vier Algorithmen um einen heuristischen Algorithmus, sondern um einen, der den gesamten Suchraum absucht. Sie benötigt also im schlimmsten Fall wesentlich mehr Rechenzeit. Dafür findet die cBLDM immer die bestmöglichen Aufteilungen. Wir sehen, dass ab einem bestimmten kritischen Wert von N die minimale Aufteilungsdifferenz auf $0{,}5$ absinkt. Dieser Wert ergibt sich aus der Mittelung über Zahlenmengen mit ungerader ($\Rightarrow\ \Delta_{min} = 1$) und gerader Summe ($\Rightarrow\ \Delta_{min} = 0$). Für kleinere N fällt die Kurve exponentiell ab. Die Treppenstufen, die sich dabei bilden, begründen sich in der Tatsache, dass die cBLDM nur ausgewogene Aufteilungen zulässt, und wir für eine ungerade Anzahl von aufzuteilenden Zahlen doppelt so viele Aufteilungen als ausgewogen definiert haben, nämlich solche mit Anzahldifferenzen von $\Delta m \stackrel{!}{=} \pm 1$, als für gerade Anzahlen mit $\Delta m \stackrel{!}{=} 0$. Daraus ergibt sich statistisch eine um einen Faktor von etwa 2 bessere minimale Aufteilungsdifferenz.

[7] Er definiert $f(n) = \Theta(g(n))$ als $\exists c_1, c_2, n_0$ s.d. $c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0$.

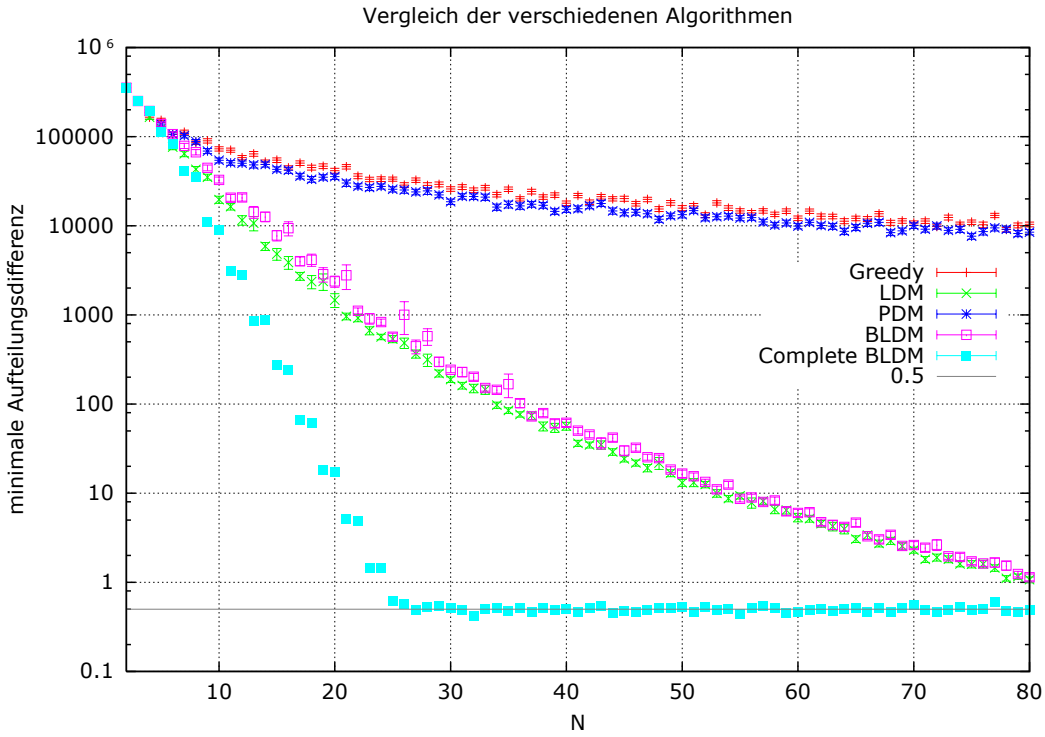

Abbildung 6: Vergleich der vorgestellten Algorithmen
Für $N \in [2{,}80]$ zu verteilende Zahlen $a_i \in [0{,}10^6]$ sind die über 300 Durchläufe gemittelten Summendifferenzen der von den Algorithmen gefundenen Aufteilungen aufgetragen. Der Complete BLDM-Algorithmus findet immer die bestmögliche Aufteilung.

4.3. Vollständige Algorithmen für das $\mathcal{NPP}$

Wer auf der Suche nach der bestmöglichen Aufteilung seiner Zahlenmenge ist, darf sich nicht auf die bisher beschriebenen heuristischen Algorithmen verlassen, wie dieses einfache Beispiel zeigt:

$$A = \{8,\ 7,\ 5,\ 5,\ 5\}$$

Alle heuristischen Algorithmen beißen sich hieran die Zähne aus, denn sie ergeben eine Aufteilungsdifferenz von $\Delta = 4$, da sie die 8 und die 7 in unterschiedliche Zahlenmengen ordnen, dabei ist die Lösung doch so einfach, ausgewogen und perfekt:

$$A_1 = 8 + 7 = 15$$
$$A_2 = 5 + 5 + 5 = 15$$

Abhilfe schaffen die nun beschriebenen, „vollständigen“ Algorithmen. Wir kommen nicht umhin, den gesamten Suchraum wieder in Betracht zu ziehen. Wir werden aber sehen, wir uns davor bewahren können, alle 2^{N-1} möglichen Aufteilungen durchprobieren zu müssen.

Korfs Complete-Verbesserung der LDM Richard Korf veröffentlichte 1998 einen Algorithmus, mit dem die LDM zu einem sogenannten anytime-Algorithmus erweitert werden kann. Unter einem *anytime-Algorithmus* versteht man einen Algorithmus, der zunächst eine Näherungslösung ergibt, d.h. eine gute, aber nicht unbedingt die beste Lösung, und danach solange weiterarbeitet, wie man ihm Zeit gibt. Er sucht nach besseren Lösungen, bis er entweder angehalten wird, oder er den gesamten Suchraum abgegrast hat. Bei speziellen Problemen, zu denen auch unser $\mathcal{NPP}$ gehört, lässt sich eine einfache Bedingung für die bestmöglich zu erreichende Lösung definieren, hier eine Differenz der beiden Teilmengen von 0 bzw. 1. In diesen Fällen kann (und sollte) der anytime-Algorithmus die Suche beenden, sobald er eine solche beste Lösung gefunden hat.

Korfs anytime-Algorithmus zur Lösung des $\mathcal{NPP}$ basiert auf der LDM: An jedem Knoten werden die beiden größten Zahlen durch ihre Differenz ersetzt, was effektiv einer Einsortierung der beiden Zahlen in verschiedene Teilmengen gleichkommt. Die einzige andere Möglichkeit wäre, die beiden Zahlen derselben Teilmenge zuzuordnen und sie durch ihre Summe zu ersetzen. Dadurch erhält man einen binären Suchbaum, dessen Basis ein Knoten mit einer Liste der zu verteilenden Zahlen ist. An jedem Knoten spaltet sich der Baum in zwei Äste, wobei in der linken Verzweigung die beiden größten verbliebenen Zahlen durch ihre Differenz ersetzt werden, in der rechten durch ihre Summe:

$$x_1, x_2, x_3, \ldots \mapsto \begin{cases} |x_1 - x_2|, x_3, \ldots & \text{linker Ast} \\ x_1 + x_2, x_3, \ldots & \text{rechter Ast} \end{cases} \tag{43}$$

Korfs Algorithmus gibt zunächst die LDM-Lösung zurück, die dem Endknoten ganz links entspricht, und beginnt von dort seine Suche. Der Suchbaum hat demnach für n Zahlen eine Tiefe von $n-1$ und 2^{n-1} Endknoten mit nur noch einem Element, welches als mögliche Lösung (Summendifferenz einer Aufteilung) in Frage kommt. Den gesamten Baum abzusuchen würde im schlimmsten Fall eine in n exponentielle Zeit erfordern, es sei denn, der Suchvorgang trifft auf eine perfekte Aufteilung, also auf einen Endknoten mit einer Differenz gleich 0 bzw. 1. Es gibt aber noch eine zweite Möglichkeit, um Teile des Suchbaums auszuschließen, in denen keine bessere Lösung zu erwarten ist. Trifft der Algorithmus bei seiner Suche auf einen Knoten, bei dem die Differenz zwischen dem größten Elements und der Summe aller anderen Elemente größer ist als die beste (= kleinste) bisher gefundene Aufteilungsdifferenz Δ, so wird es in dem Teil des Baums, der von diesem Knoten ausgeht, keine bessere Lösungen geben:

$$2 \max_i x_i - \sum_i x_i \geq \Delta \Rightarrow \text{in diesem Ast keine bessere Lösung zu erwarten} \tag{44}$$

Lässt man ihm genügend Zeit, so wird dieser Algorithmus also immer bessere Lösungen bis hin zur bestmöglichen Lösung finden. Einen Algorithmus, der wie Korfs Algorithmus alle möglichen Aufteilungen (oder allgemeiner alle möglichen Lösungen) berücksichtigt, bezeichnet man als einen vollständigen Algorithmus (engl. *complete algorithm*). Allerdings eignet sich Korfs Algorithmus nicht für das ausgewogene $\mathcal{NPP}$, denn auf die Anzahldifferenzen der hervorgebrachten Teilmengen wird keine Rücksicht genommen. Die Lösung hierfür bringt der folgende, letzte hier dargestellte Algorithmus.

4.4. Complete Anytime Algorithm for Balanced Number Partitioning

Stephan Mertens hat Korfs Algorithmus auf die BLDM ausgeweitet und 1999 einen anytime-Algorithmus für das ausgewogene $\mathcal{NPP}$ vorgestellt ([6]). Er beruht auf dem Gedanken, während des

Durchforstens des Suchbaums über die Anzahldifferenzen Buch zu führen. Dazu führt er eine „effektive Kardinalität“ ein, die wir im folgenden mit m_i bezeichnen und die Auskunft darüber gibt, um wieviele Zahlen eine Teilmenge wachsen würde, wenn ihr das Element x_i zugeordnet würde. Zu Beginn ist für jede Zahl $m_i = 1$. Die Differenzoperationen der beiden Verzweigungen an jedem Knoten müssen nun auf die m_i erweitert werden:

$$(x_1, m_1), (x_2, m_2), (x_3, m_3), \ldots \mapsto \begin{cases} (|x_1 - x_2|, m_1 - m_2), (x_3, m_3), \ldots & \text{linker Ast} \\ (x_1 + x_2, m_1 + m_2), (x_3, m_3), \ldots & \text{rechter Ast} \end{cases} \tag{45}$$

Für die in den Endknoten einzeln stehenden Elemente gibt die effektive Kardinalität die Anzahldifferenz der entsprechenden Aufteilung an. Als in Frage kommende Lösungen betrachten wir nur noch Endknoten, in denen m_i gleich der gewünschten Anzahldifferenz ist. Für das ausgewogene $\mathcal{NPP}$ sind dies alle Endknoten mit $m_i = n \mod 2$, mit diesen Algorithmus lässt sich aber jede beliebige Kardinalitätsdifferenz als Bedingung vorgeben.

Mithilfe der effektiven Kardinalitäten fallen erneut Teile des Suchbaums weg, weil in ihnen keine Endknoten mit der gewünschten Anzahldifferenz zu erwarten sind, wodurch sich die Suche erneut beschleunigen lässt. Dazu sei $m_{max} \doteq \max_i\{|m_i|\}$ die größte effektive Kardinalität und $M \doteq \sum_{i=1}^{k} |m_i|$ die Summe der effektiven Kardinalitäten der k Elemente irgendeines Knotens. Die effektiven Kardinalitäten aller aus diesem Knoten hervorgehenden Knoten ist beschränkt durch

$$2m_{max} - M \leq |m| \leq M. \tag{46}$$

Liegt die gesuchten Anzahldifferenz außerhalb dieses Bereiches, so braucht der Suchbaum unterhalb dieses Knotens nicht durchsucht zu werden.

Während bei Korfs Algorithmus die erste gültige Aufteilung nach n Knoten gefunden wird, so ist dies bei dem hier vorgestellten Algorithmus nicht garantiert, da die Kardinalitätsbedingung nicht erfüllt sein muss. Deshalb wollen wir uns jetzt auf die Suche nach ausgewogenen Aufteilungen einschränken, für die wir die BLDM-Strategie anwenden können. Am Ende der PDM-Phase der BLDM erhalten wir Knoten mit $\lceil n/2 \rceil$ Elementen, die alle effektive Kardinalitäten von $m_i = 0$ haben (bis auf ein einziges $m_j = 1$, falls n ungerade ist). Verfahren wir nun nach dem Korf-Algorithmus mit obigen Erweiterungen, so erhalten wir zuerst die BLDM-Lösung und danach kann die Suche nach besseren Lösungen beginnen.

Abbildung (8) zeigt den cBLDM-Algorithmus als Pseudocode, anhand dessen das oben gesagte bestimmt klarer wird. Die Trennung in PDM- und LDM-Phase wird dadurch erreicht, dass erst ab einer Tiefe von $\lceil n/2 \rceil$ die Zahlen erneut sortiert werden (Zeile 16-18) und zuvor während der PDM-Phase neue Elemente immer vorne eingefügt werden, während nur die hinteren beiden für Differenzoperationen verwendet werden. Dadurch entsprechen die ersten $\lfloor n/2 \rfloor$ Schritte dem Vorgehen bei der PDM.

4.5. Ein $\mathcal{NPP}$-Algorithmus mit $\mathcal{O}(n^x)$?

Mertens erwähnt in [7], dass das $\mathcal{NPP}$ zu den pseudo-polynomialen Problemen zählt. Wir werden hier sehen, was es mit einem *pseudo-polynomialen Algorithmus* auf sich hat. Dazu wollen wir folgenden Vorschlag für einen neuen Algorithmus zur Lösung des Entscheidungs-$\mathcal{NPP}$ betrachten:

Gegeben sei eine Instanz des $\mathcal{NPP}$ bestehend aus einer Menge n Zahlen $A = \{a_1, \ldots, a_n\}$. Unser Algorithmus soll die Frage beantworten, ob eine perfekte Aufteilung möglich ist. Der Einfachheit halber sollen hier nur Aufteilungen ein „ja“ ergeben, deren Summendifferenz wirklich 0 ist. Falls also

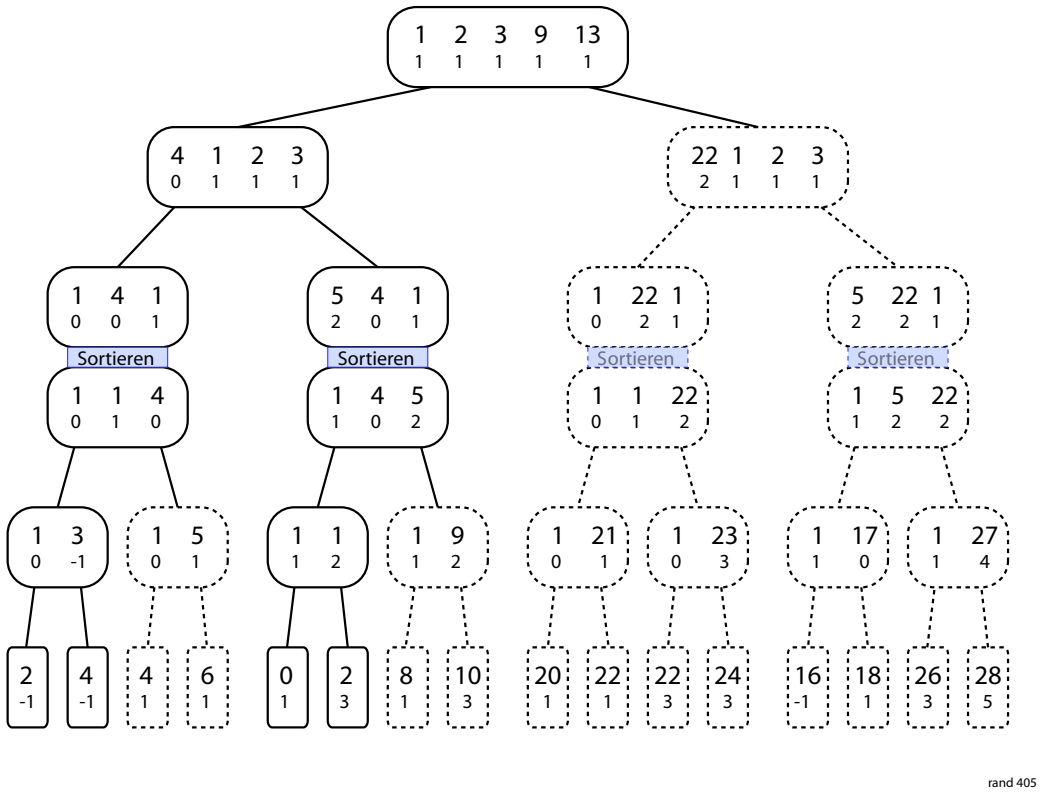

Abbildung 7: Beispiel eines Suchbaums des cBLDM
An den gepunktet gezeichneten Knoten braucht wegen der im Text erläuterten Regeln die Suche nicht in die Tiefe fortgesetzt zu werden. - Statt in der LDM-Phase jedes Mal neu zu sortieren, werden die neuen Elemente gleich am richtigen Platz eingefügt.

$\sum_i a_i \mod 2 \neq 0$, so kann der Algorithmus gleich mit einer negativen Antwort beendet werden. Sei andernfalls $B \doteq \sum_i a_i$. Wir legen eine Tabelle mit n Zeilen und $B/2+1$ Spalten an, deren Einträge $t(i,j),\ 1 \leq i \leq n,\ 0 \leq j \leq B/2$ die Antwort auf die Frage „Gibt es eine Teilmenge von $\{a_1, a_2, \ldots, a_i\}$ deren Elementsumme genau j ist?“ enthalten. Ein Beispiel für die Menge $A = \{1,9,5,3,8\}$ gibt Tabelle (3), in der J für Ja und N für Nein steht.

Das Besondere ist nun, dass sich diese Tabelle mit einfachen Regeln füllen lässt. Die erste Zeile ist am einfachsten: Es ist $t(1,j) = ja$ genau dann, wenn $j = 0$ oder $j = a_1$. Jede folgende Zeile lässt sich mit Hilfe der vorangegangenen ausfüllen:

$$t(i,j) = ja \iff t(i-1,j) = ja \vee j \geq a_i \wedge t(i-1, j-a_i) = ja \tag{47}$$

(Man verdeutliche sich die Regeln anhand der Beispieltabelle (3).) Ist die gesamte Tabelle ausgefüllt, so haben wir bereits die Lösung der gegebenen Instanz, denn die Menge A hat genau dann eine perfekte Teilung, wenn $t(n, B/2) = ja$.

Das Ausfüllen der Tabelle kann von einem Algorithmus durchgeführt werden, der polynomial in der Zahl der Tabelleneinträge (also in nB) ist, womit scheinbar ein Algorithmus gefunden wurde, der die Entscheidungsvariante des $\mathcal{NPP}$ in polynomialer Zeit bewältigen kann. Wie kann dieser Widerspruch aufgelöst werden?

```
1:  procedure COMPLETE-BLDM(x[1..k], m[1..k])
2:      if k = 1 then                                                    ▷ Endknoten
3:          if |m[1]| = (n mod 2) and x[1] < Δ then
4:              Δ ← x[1];                                                ▷ neue kleinste Differenz
5:          end if
6:      else
7:          if Δ = (Σ_{i=1}^{N} a_i) then
8:              return                                                   ▷ Stop: Perfekte Lösung bereits gefunden.
9:          end if
10:         if 2 · max_i{x[i]} − Σ_i x[i] ≥ Δ then
11:             return                                                   ▷ Stop: nur noch schlechtere Δ-Werte zu erwarten
12:         end if
13:         if 2 · max_i{|m[i]|} − Σ_i |m[i]| > |m| or Σ_i |m[i]| < m then
14:             return                                                   ▷ Stop: keine balanced-Partition mehr möglich
15:         end if
16:         if k ≤ ⌈n/2⌉ then
17:             SORT(x[1..k], m[1..k]);                                  ▷ LDM-Abschnitt → Aufsteigend sortieren
18:         end if
                                                                         ▷ Verzweigen „−“ :
19:         COMPLETE-BLDM(x[k] − x[k − 1], x[1..k − 2], m[k] − m[k − 1], m[1..k − 2]);
                                                                         ▷ Verzweigen „+“ :
20:         COMPLETE-BLDM(x[k] + x[k − 1], x[1..k − 2], m[k] + m[k − 1], m[1..k − 2]);
21:     end if
22: end procedure
```

Abbildung 8: Complete balanced largest differencing method algorithm
Aufruf mit einer aufsteigend sortierten Liste von n Zahlen $x_1, \ldots, x_n$ mit $m_i = 1,\ 1 \leq i \leq n$ und $\Delta = \infty$. In Δ wird die bisher gefundene minimale Aufteilungsdifferenz gespeichert. Wird eine perfekte Teilung gefunden, so wird der Algorithmus beendet.

Jede Zahl würde in der Eingabe durch eine Bitfolge mit einer Länge von $\mathcal{O}(\log a_i)$ beschrieben. Dadurch wäre die ganze Probleminstanz von der Größe $\mathcal{O}(n \log B)$. Es gibt aber kein Polynom $p(x)$, so dass für fast alle $x = n \log B$ gelten würde $nB \leq p(n \log B)$. nB ist also durch keine polynomiale Funktion von $n \log B$ beschränkt, weshalb der beschriebene Algorithmus kein polynomialer Algorithmus für das (unbeschränkte) Zahlenaufteilungsproblem ist.

Dieses Beispiel macht deutlich, dass die NP-Vollständigkeit des $\mathcal{NPP}$ stark darauf beruht, dass extrem große Eingabezahlen erlaubt sind. Gäbe es von vornherein eine obere Schranke für die Zahlen (selbst wenn diese polynomial in der Länge der Eingabe wäre), wäre der beschriebene Algorithmus ein polynomialer Algorithmus zur Lösung des eingeschränkten Problems. Einen solchen Algorithmus nennt man *pseudo-polynomial*.

4.6. Vergleich der analytischen und numerischen Ergebnisse

4.6.1. Phasenübergang in der numerischen Simulation

Die numerischen Berechnungen sind Ergebnis einer Implementation des cBLDM-Algorithmus wie in Abbildung (8) dargestellt mit der Programmiersprache C. Aufgrund des verwendeten Typs zur Dar-

	j													
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	J	J	N	N	N	N	N	N	N	N	N	N	N	N
2	J	J	N	N	N	N	N	N	N	J	J	N	N	N
3	J	J	N	N	N	J	J	N	N	J	J	N	N	N
4	J	J	N	J	J	J	J	N	J	J	J	N	J	J
5	J	J	N	J	J	J	J	N	J	J	J	J	J	**J**

Tabelle 3: Beispiel für den pseudo-polynomialen Algorithmus zur Lösung des $\mathcal{NPP}$
Der Eintrag „ja“ in Zelle $t(5,13)$ sagt uns, dass das gegebene Problem $(1,9,5,3,8)$ eine perfekte Partition hat.

stellung der Zahlen durfte in unserer Umsetzung die Summe der zu verteilenden Zahlen 31 Bit nicht übersteigen. Abbildung (9) zeigt eine graphische Darstellung der mittleren Anzahl der erzeugten Knoten im Suchbaum in Abhängigkeit der Anzahl N der zu verteilenden Zahlen sowie die Wahrscheinlichkeit der Existenz einer perfekten Aufteilung $p_{\text{perf}}(N)$. Gemittelt wurde hierbei über 300 Läufe. Die Anzahl der Knoten liefert dabei ein Maß für die Komplexität der jeweiligen Instanz des $\mathcal{NPP}$. Sie wächst exponentiell und erreicht ein Maximum am Phasenübergang, fällt danach wieder exponentiell ab, bis sie ein Minimum erreicht, wonach sie linear mit N anwächst. In diesem Bereich (der „weichen“ Phase) sind perfekte Aufteilungen so zahlreich, dass so gut wie immer die vom cBLDM zuerst gefundene BLDM-Lösung schon perfekt ist, so dass nur $2N$ Knoten erzeugt werden müssen. Es sind dies N Knoten, bis die Lösung gefunden wird, die weiteren N erzeugten Knoten werden sofort abgebrochen, da bereits eine bestmögliche Lösung gefunden wurde.

Handelt es sich wie bei unseren Simulationen um eine ausgewogene Aufteilung, so ergibt sich eine ähnliche Zustandssumme bzw. Entropie wie im unausgewogenen Fall, der in Abschnitt (3) analytisch behandelt wurde. Der einzige Unterschied liegt in den Ausdrücken für die Kontrollparameter κ und κ_c. Nach [8] gilt für ausgewogene Aufteilungen:

$$\kappa_c = 1 - \frac{1}{N}\log_2\left(\frac{\pi}{\sqrt{12}}N\right), \tag{48}$$

bzw.

$$\kappa = \frac{1}{N}\log_2\left(\frac{\sqrt{12}}{\Delta a}\sqrt{\langle a^2\rangle - \langle a\rangle^2}\right). \tag{49}$$

Um die Wahrscheinlichkeit für die Existenz einer perfekten Aufteilung zu berechnen, werden wir die entsprechende Formel aus [9] verwenden:

$$p_{\text{perf}}(N) \approx 1 - \exp\left(-\frac{2^{N(\kappa_{c,0}-\kappa)}}{2!}\right) \tag{50}$$

Die aufzuteilenden Zahlen a_i waren gleichmäßig verteilte ganze Zufallszahlen aus dem Bereich

$$a_i \in [0,\ldots,2^{23}-1 = 8\ 388\ 607],\ 1 \le i \le N \tag{51}$$

Um den Phasenübergang bei einer ganzzahligen Verteilung ($\Delta a = 1$) zu finden, setzen wir die ent-

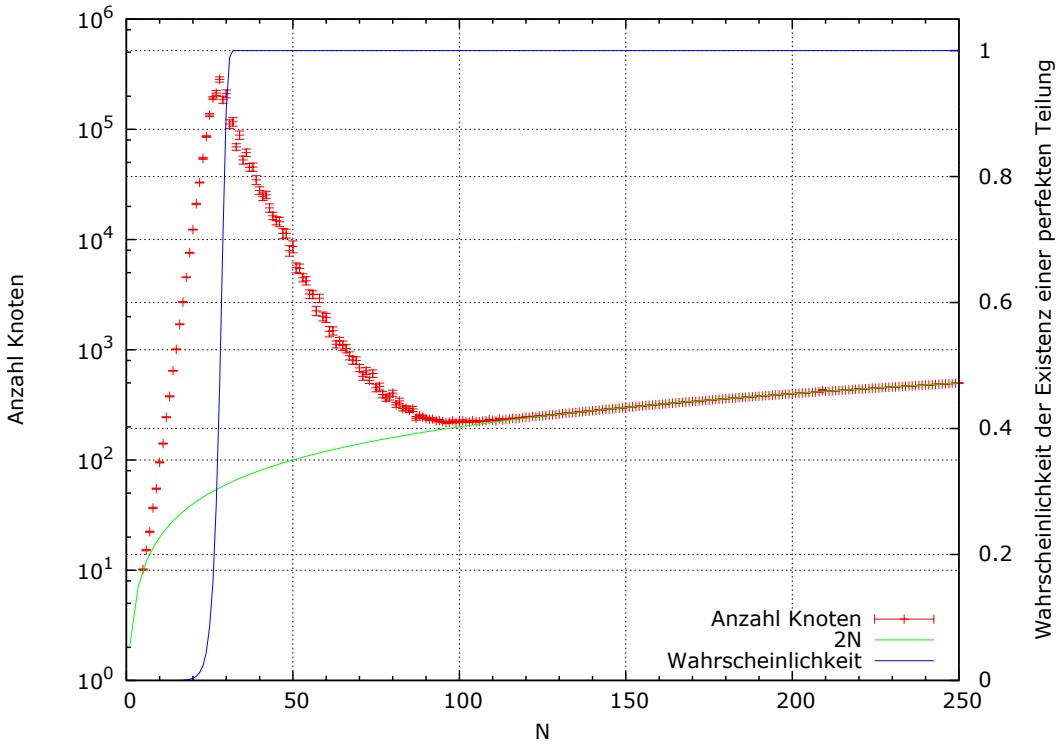

Abbildung 9: Phasenübergang beim $\mathcal{NPP}$. Jeweils 300 Durchläufe mit gleichverteilten zufälligen Ganzzahlen im Bereich von $[0{,}2^{23}-1]$.

sprechenden Ausrücke für κ und κ_c gleich:

$$\begin{aligned} \kappa &\stackrel{!}{=} \kappa_c \\ \Rightarrow \frac{1}{N}\log_2\left(\frac{\sqrt{12}}{\Delta a}\sqrt{\langle a^2\rangle_i - \langle a\rangle_i^2}\right) &= 1 - \frac{1}{N}\log_2\left(\frac{\pi}{\sqrt{12}}N\right) \\ \Rightarrow \log_2\left(\sqrt{12}\sqrt{\langle a^2\rangle_i - \langle a\rangle_i^2}\right) &= N - \log_2\left(\frac{\pi}{\sqrt{12}}N\right) \\ \Rightarrow N_c - \log_2 N_c &= \log_2\left(\pi\sqrt{\langle a^2\rangle_i - \langle a\rangle_i^2}\right). \end{aligned} \tag{52}$$

Den Phasenübergang erwarten wir also nach Formel (52) bei $N_c = 27{,}6$. Dies stimmt in zufriedenstellendem Maße mit dem numerisch gefundenen Maximum überein (vgl. Abb. (9)). Auch in Abbildung (6) lässt sich ein Phasenübergang beobachten. Für die Größe der dort verwendeten Zufallszahlen ergibt sich $N_c = 24{,}4$. Allerdings ist der Phasenübergang aus der minimalen Aufteilungsdifferenz in diesem Graphen nicht so schön deutlich ersichtlich wie aus der Anzahl der Knoten in Abbildung (9).

Abbildung (10) gibt einen überzeugenden Eindruck, wie sich die vom cBLDM gefundene Lösung Δ gegenüber der BLDM-Lösung Δ_{BLDM} im Laufe der Zeit verbessert. Die Grafik wurde [6] entnom-

men. Dargestellt sind 100 Instanzen des $\mathcal{NPP}$ mit je 100 gleichverteilten 150 Bit langen ganzen Zufallszahlen. Durch die große Größe der Zufallszahlen soll sichergestellt werden, dass der Algorithmus nicht vorzeitig die bestmögliche Aufteilung findet, sondern sich über den gesamten Zeitraum Verbesserungen aufzeichnen lassen. Man sieht, dass sich Verbesserungen von mehreren Größenordnungen ergeben.

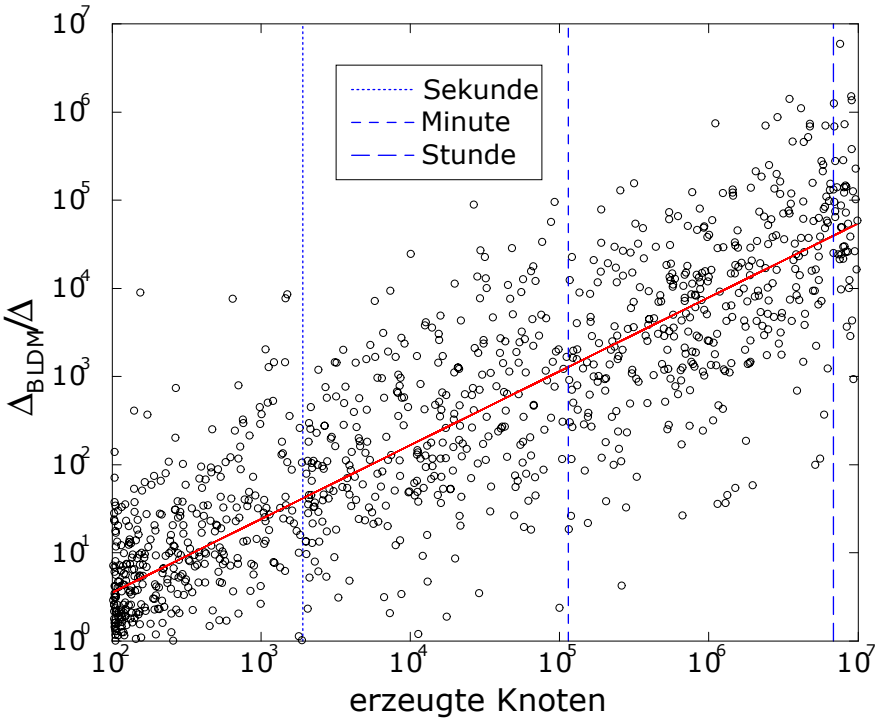

Abbildung 10: Verbesserung der Lösungsqualität des cBLDM über die Zeit (aus [6])

4.6.2. Überprüfung der Grundzustandsenergie

Die analytisch gewonnenen Ergebnisse können nun numerisch überprüft werden. Als Beispiel wählen wir reelle Zahlen aus dem Intervall $[0,1)$. Δa strebt also gegen Null und wir können den Erwartungswert $< a_j^2 >= \sum_j a_j^2/N$ mit Hilfe eines Integrals nähern. Damit erhalten wir

$$\int_0^1 a_j^2 = \frac{1}{3}$$

$$\sum_j a_j^2 = \frac{N}{3}. \tag{53}$$

Die Grundzustandsenergie ergibt sich somit nach Gleichung (42) zu

$$E_0 = \sqrt{\frac{2}{3}\pi N 2^{-N}}. \tag{54}$$

In Abbildung (11) werden die analytischen Ergebnisse aus Gleichung (54) mit den numerisch bestimmten Werten für diese Grundzustandsenergie verglichen. Die geringe Abweichung ist recht überzeugend.

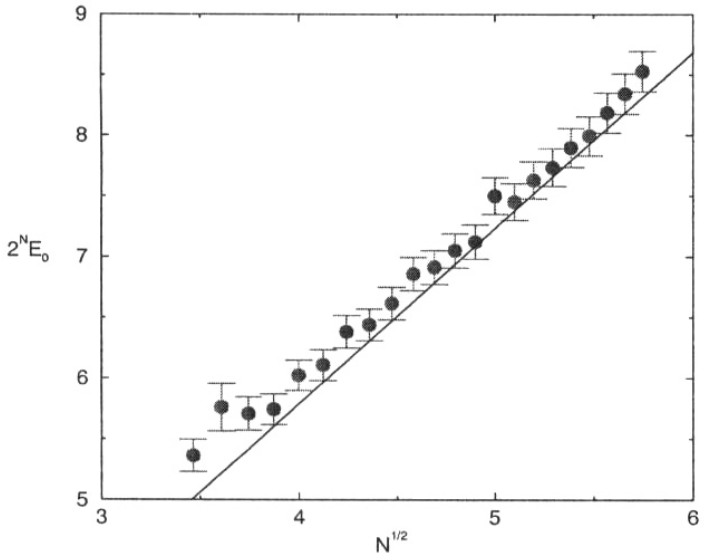

Abbildung 11: Durchschnittliche Grundzustandsenergie des $\mathcal{NPP}$ mit reellen Random-Zahlen $0 \leq a_i < 1$ im Vergleich mit dem analytischen Ergebnis aus Gl. (54) (durchgezogene Linie) (aus [7])

4.6.3. Überprüfung des Kontrollparameters

Um zu überprüfen, ob $\kappa(N)$ tatsächlich ein Kontrollparameter ist und bei dem kritischen Wert $\kappa_c(N)$ ein Phasenübergang stattfindet, simulierte Mertens numerisch die Fälle, bei denen bei festem N und κ mindestens eine perfekte Unterteilung gefunden werden kann ([7], vgl. Abbildung 12). Für $N \to \infty$ strebt $\kappa_c(N)$ gegen 1.
Die beiden Phasen unterscheiden sich sehr in der Komplexität der Lösbarkeit. Im Bereich $\kappa < \kappa_c$ kann immer innerhalb kurzer Zeit eine perfekte Unterteilung gefunden werden. Im Bereich $\kappa > \kappa_c$ hingegen wird für die eindeutige beste Lösung exponentiell viel Zeit benötigt.
Bei kleinen Werten von N und kleinen Δa sind wir in der *hard phase*. Es existiert keine perkfekte Lösung und ein Suchalgorithmus braucht viel Zeit. Vergrößerung von N lässt den Suchraum exponentiell ansteigen, ebenso wie die Laufzeit. Andererseits gelangt man mit wachsendem N auch näher an die Phasengrenze. Hinter dieser Grenze wächst die Anzahl perfekter Unterteilungen exponentiell mit der Anzahl N und ein Suchalgorithmus wird schnell auf eine perfekte Unterteilung stoßen. Im Vergleich mit numerischen Experimenten stellen wir fest, dass die Laufzeit nun mit *wachsendem N* sogar abnimmt!

5. Zusammenfassung

In diesem Vortrag haben wir gezeigt, dass die Methoden der Statistischen Mechanik auch auf nicht physikalische Probleme - in unserem Beispiel auf das Zahlenaufteilungsproblem der theoretischen Informatik - übertragen werden können. Selbst unter dem Gesichtspunkt, dass man bei weiterführendem Studium des $\mathcal{NPP}$ auch auf Grenzen der Methoden der Statistischen Physik stößt, wird dieser allein durch die Übertragbarkeit auf nicht physikalische Systeme ein ganz neues Gewicht verliehen. Auch wird der Begriff des Phasenüberganges, der uns bislang nur in Bezug auf physikalische Phänomene bekannt ist, wesentlich erweitert. Schließlich wurden die analytischen Ergebnisse durch numerische

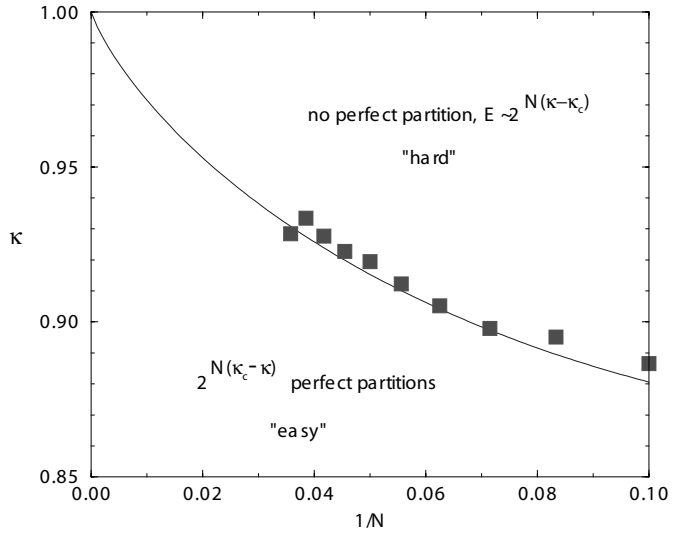

Abbildung 12: Phasendiagramm des $\mathcal{NPP}$(entnommen aus [7]).
Die Quadrate sind numerisch gefundene Werte, die durchgezogene Linie ist durch κ_c nach Gl. (36) gegeben. Für $\kappa < \kappa_c$ ist die Nulltemperaturentropie extensiv, ein Suchalgorithmus findet typischerweise schnell eine der $\mathcal{O}(2^N)$ perfekte Unterteilungen. Für $\kappa > \kappa_c$ existieren keine perfekten Partitionen und das Optimierungsproblem hat eine schwer zu findende, eindeutige Lösung.

Simulationen mit effizienten Algorithmen in überzeugendem Maße bestätigt.

A. Anhang

A.1. Grundlegende Formeln

A.2. Statistische Mechanik - Grundlegende Begriffe und Formeln

Die Statistische Mechanik versucht die makroskopischen Eigenschaften von Vielteilchensystemen aus mikroskopischen Wechselwirkungen abzuleiten. Die Subjektive Unkenntnis über den Mikrozustand erfordert die Wahrscheinlichkeitstheorie. Ein System ist vollständig charakterisiert durch die Hamiltonfunktion

$$H = H(q_1, ..., q_{NF}, p_1, ..., p_{Nf}), \tag{55}$$

wobei die q_i für die Orts- und die p_i für die Impulskoordinaten stehen.N ist die Anzahl der Phasenraumzellen, f die Anzahl der Freiheitsgrade.
Dem Gibbschen Zugang zur Statistischen Mechanik liegt die Idee zu Grunde, dass man den Mikrozustand nicht genau kennt, aber eine Wahrscheinlichkeitsverteilung der möglichen Mikrozustände bei den gegebenen Bedingungen angegeben werden kann.
Ein *Thermodynamisches Ensemble* ist eine Schar von gedachten Systemen, die alle Abbild des realen systems sind. Für Z mögliche Mikrozustände gibt es somit Z (im Rahmen der Ergodentheorie) gleichberechtigte Mitglieder. Statt *scharfe Mikrozustände* betrachtet man die Gesamtheiten von Mikrozuständen, die sämtlich mit den Nebenbedingungen verträglich sind, mi Wahrscheinlichkeitsinterpretation.
In der klassischen Mechanik beispielsweise, entspricht ein Ensemble allen Punkten im Phasenraum, die mit den Zwangsbedingungen verträglich sind.
Die wichtigsten Ensembles sind in der folgenden Tabelle zusammengestellt:

konstante Parameter	Ensemble	physikalische Bedingungen
N, V, E	mikrokanonisch	abgeschlossenes System
N, V, T	kanonisch	Kontakt mit Wärmebad
μ,V,T	großkanonisch	Kontakt mit Wärmebad und Teilchenreservoir

Tabelle 4: Die wichtigsten Ensembles

Befindet sich ein System im thermischen Gleichgewicht mit der Umgebung, so heißt die dazugehörige Verteilung *kanonische Verteilung*, die entsprechende Gesamtheit *kanonische Gesamtheit*. Die konstanten Parameter sind die Temperatur T, das Volumen V und die Teilchenzahl N.
Die *Zustandssumme Z* ist definiert über die Beziehung

$$Z(T, V, N, ...) := \sum_l \exp\left(-\beta E_l\right) \tag{56}$$

Der Faktor β ist dabei gegeben durch

$$\beta = \frac{1}{k_B T}. \tag{57}$$

mit der Boltzmann-Konstante k_B. Die *Freie Energie F* berechnet sich zu

$$F(T, V, N, ...) := -k_B T \ln Z = -\frac{1}{\beta} \ln Z \tag{58}$$

und mit der mittleren Energie

$$\overline{E} = -\frac{\partial}{\partial\beta}\ln Z \tag{59}$$

erhält man daraus die *Entropie S*

$$\begin{aligned} S &= -\frac{\delta F}{\delta T} = \frac{E-F}{T} \\ &= k\beta\overline{E} + k\ln Z - k\ln C \end{aligned} \tag{60}$$

folgt.

A.3. Die Sattelpunkts-Methode

Die Laplace- oder Sattelpunkts-Methode ist eine Methode zur asymptotischen Nä herung von Integralen der Form

$$\int f(k)\exp\left(NG(k)\right)dk. \tag{61}$$

Wir beschränken uns hier zweckmäßig auf eindimensionale Integration. Für große N wird das Integral von Beiträgen in der Nähe der absoluten Maxima k_n der Funktion $G(k)$ dominiert. Entwicklung der Funktion $G(k)$ um k_n in zweiter polynomialer Ordnung, liefert die Näherung

$$\begin{aligned} &\int f(k)\exp\left(NG(k)\right)dk \\ &\qquad\approx \sum f\left(k_n\right)\exp\left(NG(k_n)\right)\int_{\mathbb{R}}\exp\left(NG''(k_n)\frac{(k-k_n)^2}{2}\right)dk \\ &\qquad= \sum f\left(k_n\right)\exp\left(NG(k_n)\right)\sqrt{\frac{2\pi}{-NG''(k_n)}} \end{aligned} \tag{62}$$

Im letzen Schritt wurde hier das Gaußintegral

$$\int_0^\infty \exp\left(-a^2x^2\right)dx = \frac{\sqrt{\pi}}{2a} \tag{63}$$

verwendet.

Literatur

[1] Fu, Y., in *Lectures in the Sciences of Complexity*, edited by. D. L. Stein, Addison-Wesley Publiching Company, Reading, Massachusetts, 1989

[2] Gent, I. P. and Walsh, T., in Proceedings of ECAI-96, edeted by W. Wahlster, John Wiley & Sons, New York (1996)

[3] Hayes, Brian: Can't get no satisfaction, American Scientist **85**, 108 (1997)

[4] Hayes, Brian: The Easiest Hard Problem, Magazine of Sigma Xi, the Scientific Research Society, Vol. 90, No. 2 (2002)

[5] Mertens, Stephan: Computational Complexity for Physicists, Computing in Science & Engineering **4**, 31 (2002)

[6] Mertens, Stephan: A complete anytime algorithm for balanced number partitioning, preprint cs. DS/9903011 (1998)

[7] Mertens, Stephan: Phase Transition in the Number Partitioning Problem, Physical Review Letters **81**, 4281 (1998)

[8] Mertens, Stephan: A physicist's approach to number partitioning, arXiv:cond-mat/00092320, 2000

[9] Bauke, Heiko: Statistische Mechanik des Zahlenaufteilungsproblem, Diplomarbeit an der Universität Magdeburg 2002

[10] Michael R. Garey und David S. Johnson: Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman, New York, 1997

[11] http://www.sortieralgorithmen.de/

[12] http://www.claymath.org/millennium/

[13] D.S. Johnson und C.H. Papadimitriou, „Computational Complexity - The Traveling Salesman Problem“, Lawier et al., eds., 1985, pp. 37-85

[14] Benjamin Yakir, The differencing algorithm LDM for partitioning: a proof of a conjecture of Karmarkar and Karp, Math. Oper. Res., 21(1):85-99, 1996

Die Quellen [13] und [14] lagen nicht vor, sondern wurden aus Verweisen in [6] übernommen.