Grundzustandsstruktur ungeordneter Systeme und Dynamik von Optimierungsalgorithmen

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von
Wolfgang Barthel
aus Halle (Saale)

Göttingen 2005

D7
Referent: PD Dr. Alexander K. Hartmann
Korreferentin: Prof. Dr. Annette Zippelius
Tag der mündlichen Prüfung: 8. November 2005

Inhaltsverzeichnis

1 Komplexe ungeordnete Systeme – Schnittstelle zwischen Statistischer Physik und Theoretischer Informatik

Die Statistische Physik beschreibt makroskopische Eigenschaften von Systemen aus einer sehr großen Anzahl mikroskopisch miteinander wechselwirkender Teilchen.

Ein klassisches Beispiel hierfür sind Ising-Spins S_i, welche die beiden Werte $S_i = \pm 1$ annehmen können. Die Wechselwirkungen zwischen den Spins werden durch Kopplungskonstanten J_{ij} beschrieben. Die Energie des Systems ist durch die Hamiltonfunktion

$$H_{\{J_{ij}\}}(\{S_i\}) = -\sum J_{ij} S_i S_j, \tag{1.1}$$

gegeben, wobei die Summe über nächste Nachbarn in einem Gitter, ausgewählte Paare von Spins oder alle Paare von Spins (*mean-field*) läuft.

In *Spingläsern* [1, 2] sind die Wechselwirkungen J_{ij} zufällig ferro- oder antiferromagnetisch. Im Labor findet man dieses Verhalten u. a. in bestimmten Metalllegierungen mit magnetischen Verunreinigungen wie Mangan in Kupfer oder Eisen in Gold. Dies führt dazu, dass sich nicht alle Paare von Spins entsprechend ihrer bevorzugten Ausrichtung einstellen können, das System ist *frustriert* (Abbildung 1.1).

Diese Materialien haben einige ungewöhnliche Eigenschaften. So verschwindet zwar die Gesamtmagnetisierung $m = \sum_i S_i$, man findet aber (in Dimensionen $d > 2$ unterhalb einer kritischen Temperatur T_c) eine Art Ordnung für die relative Lage benachbarter Spins, die im Zeitmittel zu einem nicht verschwindenden Wert von $\sum_i \langle S_i \rangle^2$ führt. Die Beschreibung der Grundzustandslandschaft endlich-dimensionaler Spingläser ist trotz jahrzehntelanger Bemühungen immer noch eine ungeklärte Frage.

In der Theoretischen Informatik gibt es Optimierungsprobleme, die eine ganz ähnliche Struktur wie Spingläser aufweisen. Ein solches Problem ist das *Vertex-Cover-Problem* (siehe Abb. 1.2): Gegeben ist ein Graph G mit N Knoten und M Kanten. Eine *Überdeckung* ist eine Teilmenge der Knoten, die von jeder Kante mindestens einen Endknoten enthält. Wie viele Knoten muss für gegebenes G eine Überdeckung mindestens enthalten?

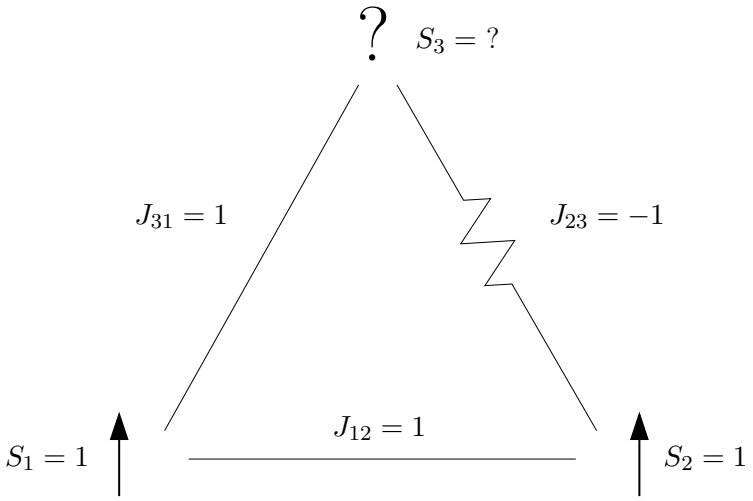


Abbildung 1.1: Frustriertes Ising-Spin-System

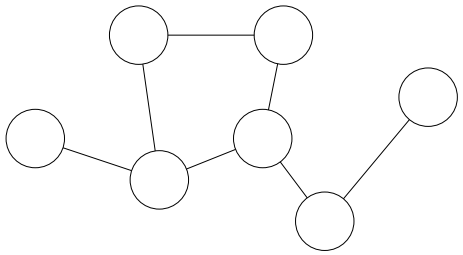


Abbildung 1.2: Vertex-Cover-Problem: Wie viele Knoten muss man mindestens auswählen, damit von jeder Kante mindestens ein Endpunkt enthalten ist?

Sowohl Spinkonfigurationen mit minimaler Energie als auch Überdeckungen mit minimaler Größe werden als *Grundzustände* bezeichnet. Beiden Problemen ist gemeinsam, dass auf Grund der mikroskopischen Struktur die Bestimmung von Grundzuständen extrem schwierig ist: Welche der Wechselwirkungen in einer optimalen Spinglas-Konfiguration verletzt werden müsssen, ist lokal nicht erkennbar. Ebenso lässt sich nicht lokal bestimmen, ob für eine Kante beide Endknoten zu einer optimalen Überdeckung gehören oder nur einer von beiden.

Es wird vermutet, dass diese algorithmischen Schwierigkeiten mit der Struktur der Grundzustandslandschaft in Verbindung stehen [3]. Sind alle Grundzustände im Phasenraum verbunden und ist die Struktur trivial? Oder gibt es mehrere voneinander getrennte Bereiche, so genannte *Cluster*?

Die Grundzustandslandschaft von Optimierungsproblemen aus der Physik und der Theoretischen Informatik und Algorithmen zur Bestimmung dieser Grundzustände sind Themen dieser Arbeit. Dabei werden wir uns mit den folgenden drei Problem-

stellungen befassen:

- Durch Verwendung eines verbesserten Algorithmus zur Bestimmung von Grundzuständen dreidimensionaler Ising-Spingläser soll die Energiedifferenz zwischen Grundzustand und erstem angeregten Zustand untersucht werden, um Aussagen zur Komplexität der Grundzustandslandschaft treffen zu können.
- Die Grundzustandsstruktur von Vertex-Cover für ein bestimmtes Ensemble von Zufallsgraphen wird mit ursprünglich u. a. für Spingläser entwickelten *Clusterungsmethoden* betrachtet. Es ist bereits bekannt, dass man in bestimmten Bereichen des Ensembles eine einfache Grundzustandsstruktur findet, gleichzeitig kennt man in diesem Bereich Algorithmen, die eine optimale Überdeckung in kurzer Zeit finden. Wir wollen uns vor allem auf den Bereich konzentrieren, in dem die Grundzustandslandschaft noch unbekannt ist und in dem bislang auch kein effizienter Lösungsalgorithmus existiert.
- Für ein anderes Optimierungsproblem, das sogenannte *Erfüllbarkeitsproblem*, werden wir die Dynamik eines Algorithmus zur Bestimmung von Grundzuständen analytisch beschreiben. Im dabei betrachteten Ensemble werden wir zwei Phasen identifizieren: In der einen findet der Algorithmus Lösungen schnell, d. h. in einer Zeit, die linear mit der Systemgröße steigt. In der anderen Phase steigt die Lösungszeit dagegen exponentiell.

Die Arbeit beginnt mit einer Einführung in die Komplexitätstheorie und der Vorstellung verwendeter generischer Optimierungsalgorithmen in Kapitel 2. Anschließend werden in den Kapiteln 3 bis 5 die drei genannten Problemstellungen untersucht und abschließend in Kapitel 6 die wichtigsten Ergebnisse zusammengefasst.

2 Algorithmische Grundlagen

2.1 Grundlagen der Komplexitätstheorie

Um algorithmisch behandelbare Probleme nach ihrer Schwierigkeit einzuteilen, unterscheidet man in der Theoretischen Informatik verschiedene (Zeit-)Komplexitätsklassen. Diese sollen hier nur einführend beschrieben werden, eine genaue Definition findet man z. B. in [4, 5].

Zu dieser Klassifizierung ist es hilfreich, nicht das Optimierungsproblem selbst zu betrachten, sondern ein zugehöriges Entscheidungsproblem, das für die Minimierung der Spinglas-Hamiltonfunktion (1.1) folgendermaßen lauten kann:

> Gegeben eine Schranke C, gibt es einen Zustand $\{S_i\}$ mit $H_{\{J_{ij}\}}(\{S_i\}) < C$?

Durch konkrete Angabe von $\{J_{ij}\}$ und C wird eine *Instanz* des Entscheidungsproblems definiert. Ihre *Länge* ist die Anzahl der zur Kodierung eines Zustands notwendigen Bits, hier also die Anzahl der Spins bzw. die Anzahl der Knoten des Graphen. Alle Zustände $\{S_i\}$, für die für eine gegebene Instanz $H_{\{J_{ij}\}}(\{S_i\}) < C$ gilt, bezeichnen wir als *Lösungen* dieser Instanz. Eine Instanz heißt *Ja-Instanz*, wenn zu ihr mindestens eine Lösung existiert, andernfalls *Nein-Instanz*.

Zur Lösung eines solchen Entscheidungsproblems benötigen wir einen Algorithmus, der als Eingabe eine beliebige Instanz akzeptiert und die (richtige) Antwort „Ja“ oder „Nein“ ausgibt. Die benötigte Rechenzeit wird im Allgemeinen von der konkreten Instanz abhängen. In der Komplexitätstheorie betrachtet man die Instanz vorgegebener Länge mit der längsten Laufzeit, den *Worst-Case*.

Existiert nun zu einem Entscheidungsproblem ein polynomialer Lösungsalgorithmus, d. h. ein Algorithmus, dessen Worst-Case-Laufzeit asymptotisch höchstens polynomial mit der Länge der Eingabe wächst, so ist das Problem in der *Klasse P* (polynomial).

Solche Probleme sind algorithmisch gut handhabbar: Verdoppelt man die Rechenleistung, so steigt auch die maximal erreichbare Systemgröße um einen festen Faktor, der vom Grad des Polynoms abhängt, durch das die Laufzeit beschränkt ist.

Es gibt allerdings viele Entscheidungsprobleme, für die zwar bis heute kein polynomialer Algorithmus bekannt ist, für die sich aber Lösungen leicht überprüfen lassen.

Die *Klasse NP* (nondeterministic-polynomial) enthält alle Entscheidungsprobleme, für die ein polynomialer Algorithmus mit folgenden Eigenschaften existiert:

- Eingabe sind eine beliebige Instanz und ein *Zertifikat* (z. B. beim Spinglas-Grundzustandsproblem ein Zustand $\{S_i\}$ mit hinreichend kleiner Energie).
- Bei Nein-Instanzen wird für alle Zertifikate „Nein“ ausgegeben.
- Für Ja-Instanzen existiert mindestens ein Zertifikat, für das der Algorithmus „Ja“ zurückgibt.

Man beachte, dass der Algorithmus selbst keine Lösung findet, sondern nur (für Ja-Instanzen) eine gegebene Lösung überprüft.

Es gilt $P \subseteq NP$, da sich jeder polynomiale Lösungsalgorithmus zu einem polynomialen Zertifikat-Überprüfungsalgorithmus erweitern lässt, bei dem das Zertifikat einfach ignoriert wird. Umgekehrt konnte aber bis heute nicht geklärt werden, ob auch $NP \subseteq P$ gilt oder ob es Probleme in NP gibt, für die man zeigen kann, dass kein polynomialer Lösungsalgorithmus existieren kann. Diese Frage $P \stackrel{?}{\neq} NP$ ist die fundamentale Frage der Komplexitätstheorie [6].

Die Klasse der NP-vollständigen Probleme (*NP-complete*) ist eine wichtige Teilmenge von NP. Ein Problem A gehört zu dieser Klasse, wenn sich jedes andere Problem B aus NP durch eine Transformation in polynomialer Zeit in A überführen lässt. Würde man für ein NP-vollständiges Problem einen polynomialen Algorithmus finden, so ließe sich somit (durch vorherige Anwendung der jeweiligen Transformation) jedes andere Problem aus NP ebenfalls in polynomialer Zeit lösen. Die NP-vollständigen Probleme sind also die härtesten Probleme in NP.

Die Nichtexistenz eines polynomialen Algorithmus begrenzt die erreichbaren Systemgrößen für Probleme aus $NP \setminus P$ drastisch: Wächst die Rechenzeit beispielsweise exponentiell, so steigert die Verwendung eines doppelt so schnellen Computers die erreichbare Systemgröße nur um eine feste Konstante. Man bezeichnet diese Probleme deshalb auch als *nicht handhabbar*.

Im Rahmen dieser Arbeit werden drei Probleme aus der Statistischen Physik und der Theoretischen Informatik betrachtet, die alle zur Klasse der NP-vollständigen Probleme gehören:

1. Vertex-Cover-Problem,
2. K-Erfüllbarkeitsproblem,
3. Grundzustandsbestimmung von Ising-Spingläsern in drei Dimensionen.

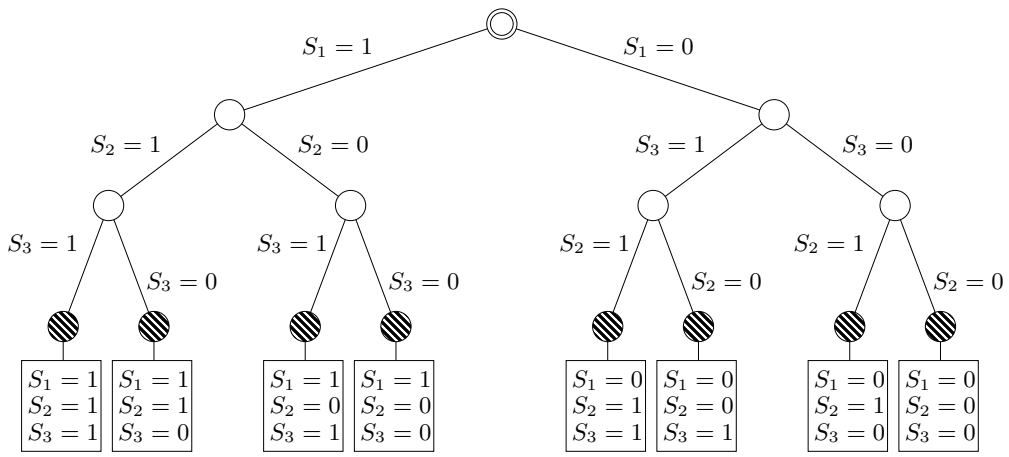


Abbildung 2.1: Mögliche Darstellung der 2^3 möglichen Konfigurationen von $N = 3$ binären Variablen in einem binären *Entscheidungsbaum*. Jedes *Blatt* (schraffiert) entspricht einem Zustand.

2.2 Generische Lösungsalgorithmen

2.2.1 Vollständige Enumeration

Die grundlegenden Objekte der hier betrachteten Optimierungs- bzw. Entscheidungsprobleme sind N binäre Variablen, d. h. solche, die nur zwei bestimmte Werte annehmen können. Je nach Problem sind das beispielsweise die zwei möglichen Orientierungen von Ising-Spins oder die beiden Werte „wahr“ und „falsch“ einer booleschen Variablen.

Der Phasenraum eines solchen Systems besteht demnach aus 2^N verschiedenen Konfigurationen. Um ihn systematisch zu durchsuchen, stellt man diese in einem binären Baum dar (siehe Abb. 2.1). Der Baum besteht aus Knoten (durch Kreise symbolisiert), die hierarchisch in Ebenen angeordnet sind. Der Knoten in der höchsten Ebene heißt *Wurzel* (Doppelkreissymbol), die Knoten in der untersten Ebene heißen *Blätter* (schraffierte Kreise). Die Höhe des Baums ist die um eins verminderte Anzahl seiner Ebenen, der Suchbaum in Abb. 2.1 hat also Höhe 3. Jeder Knoten außer der Wurzel ist mit genau einem Knoten aus der Ebene darüber verbunden, jeder Knoten außer den Blättern mit zwei Knoten (seinen beiden *Nachfolgern*) aus der Ebene darunter. Als *Nachfahren eines Knotens* bezeichnen wir die Menge seiner Nachfolger, der Nachfolger seiner Nachfolger usw. Jeder Knoten ist gleichzeitig Wurzel des *Unterbaums*, der ihn selbst und alle seine Nachfahren enthält.

Beginnend in der *Wurzel* wird in jeder Ebene die Belegung einer Variable festgesetzt, wobei in verschiedenen Unterbäumen in einer Ebene nicht unbedingt dieselbe Variable festgesetzt werden muss. Jede der beiden von einem Knoten nach unten

ausgehenden Verzweigungen entspricht einem der möglichen Werte, jedem der 2^N Blätter entspricht eine mögliche Konfiguration.

Die Antwort für eine gegebene Instanz des Entscheidungsproblems ist genau dann „Ja“, wenn mindestens eines der Blätter eine Lösung der Instanz ist.

Die naivste Lösungsmethode besteht deshalb darin, einfach den gesamten Baum zu durchsuchen. Dies kann beispielsweise durch *Backtracking* geschehen (siehe Algorithmus 2.1). Dieser rekursive Algorithmus steigt im Suchbaum zunächst von der Wurzel nach unten bis zu einem Blatt und wählt dabei in jedem Knoten die Verzweigung nach links. Solange noch keine Lösung gefunden wurde, steigt er jeweils wieder soweit nach oben, bis er zu einem Knoten gelangt, bei dem noch nicht beide Verzweigungen untersucht wurden, um in der dort noch nicht besuchten Verzweigung wieder bis zu einem Blatt hinabzusteigen.

Das Verfahren ist deterministisch: Entweder eines der Blätter ist tatsächlich eine Lösung, dann bricht der Algorithmus ab, sobald er diese gefunden hat. Oder es existiert keine Lösung, dann bricht der Algorithmus nach der Untersuchung aller 2^N Blätter (bzw. $2^{N+1} - 1$ Knoten) ebenfalls ab.

Der Algorithmus 2.1 lässt sich auf *Optimierungsprobleme* übertragen. Anstatt *schranke* wird das bisher gefundene Minimum $minimum_{\text{akt}}$ übergeben. Jeweils beim Erreichen eines Blattes wird überprüft, ob ein neues Minimum vorliegt und in diesem Fall $minimum_{\text{akt}}$ aktualisiert und nicht abgebrochen. Der Algorithmus stoppt erst, wenn der gesamte Baum durchsucht wurde und gibt das gefundene Optimum zurück.

Für praktische Belange ist dieser Algorithmus weitgehend unbrauchbar: Selbst wenn pro Sekunde 10^6 Blätter besucht werden könnten, bräuchte man für ein System mit $N = 50$ über 35 Jahre. Und selbst wenn man einen zehnmal so schnellen Rechner verwenden würde, könnte man in der gleich langen Zeit auch nur ein System mit $N = 53$ lösen.

Entscheidend ist also, den Algorithmus an sich zu verbessern. Zwei generische Methoden, die bei den Problemen dieser Arbeit verwendet wurden, sollen hier allgemein vorgestellt werden:

- *Bounding*: Unterbäume des Suchbaums werden ausgelassen, wenn bereits beim Besuch dessen Wurzel klar ist, dass sie keine Lösung bzw. kein neues Minimum enthalten.

  Wenn beispielsweise im Erfüllbarkeitsproblem, bei dem eine Anzahl von Bedingungen *sämtlich* erfüllt sein müssen, durch die Belegung mehrerer Variablen *eine einzige* Bedingung verletzt wird, dann kann die Suche in diesem Teil des Suchbaums abgebrochen werden. Denn unabhängig von der Belegung der weiteren Variablen kann kein Blatt dieses Unterbaums eine Lösung sein.

```
algorithm backtrack_naive_rekursion(V, S⃗, schranke)
begin
   if(V = ∅)    /* Blatt erreicht */
   then
      if(H(S⃗) ≤ schranke)
      then
         print „S⃗ ist Lösung!“
         stop
   else
      wähle i ∈ V
      V := V\i
      S_i := 1
      backtrack_naive_rekursion(V, S⃗, schranke)
      S_i := 0
      backtrack_naive_rekursion(V, S⃗, schranke)
end

algorithm backtrack_naiv(N, schranke)
begin
   V = {1, 2, ..., N}
   backtrack_naive_rekursion(V, S⃗, schranke)
   print „Keine Lösung gefunden!“
end
```

Algorithmus 2.1: Naiver Backtracking-Algorithmus für das Entscheidungsproblem „Gibt es eine Konfiguration $\vec{S}$ mit $H(\vec{S}) \leq schranke$?“. Der Baum aus Abb. 2.1 wird rekursiv durchsucht, ohne einen Knoten auszulassen. Der rekursiv aufgerufenen Funktion **backtrack_naive_rekursion** werden als Parameter die Menge der noch nicht belegten Variablen $\mathcal{V}$, die aktuellen Werte der belegten Variablen $\vec{S}$ und die Schranke übergeben. Der Algorithmus stoppt, wenn entweder eine Lösung gefunden wurde ($H(\vec{x}) \leq schranke$) oder alle Knoten besucht wurden.

Ebenso können bei einem Optimierungsproblem Unterbäume ausgelassen werden, wenn man durch Berechnung einer unteren Schranke zeigen kann, dass auch bei günstigster Belegung weiterer Variablen keines der Blätter besser als das bisher gefundene Minimum sein kann, welches eine obere Schranke für das tatsächliche Minimum darstellt.

Dieses sehr mächtige Verfahren zur Durchsuchung des Konfigurationsraums wird als *Branch and Bound* bezeichnet. [7]

- *Constraint propagation*: Durch das Setzen einer Variablen auf einen bestimmten Wert werden z. B. durch lokale Betrachtung des Systems weitere Variablen festgelegt.

  Wenn beispielsweise ein Spin S_i ferromagnetisch mit allen seinen Nachbarn wechselwirkt und diese alle in dieselbe Richtung zeigen, dann muss in einer Lösung minimaler Energie auch S_i in diese Richtung zeigen.

  Diese Technik ist deshalb besonders effizient, weil sie die Variablen in einer Reihenfolge festlegt, bei der möglichst weit oben im Suchbaum Unterbäume wegfallen.

Durch diese Modifikationen verringert sich einerseits die Anzahl der zu durchsuchenden Knoten, im Extremfall fällt die Suche sogar ganz weg. Andererseits steigt jedoch der Aufwand zum Bearbeiten eines Knotens mit der Komplexität der Berechnung der verwendeten Schranken bzw. der lokalen Untersuchung, u. U. ist er größer als für das Durchsuchen des dadurch ausgelassenen Unterbaums nötig ist.

2.2.2 Stochastische lokale Suchalgorithmen

Die im vorigen Abschnitt beschriebenen Algorithmen überprüfen für jedes Blatt des Suchbaums direkt oder durch Betrachten ganzer Teilbäume, ob es zu einer Lösung gehört. Diese Verfahren werden deshalb auch als *vollständige Algorithmen* bezeichnet. Für die in dieser Arbeit betrachteten Probleme reichen die damit erreichbaren Systemgrößen aber oft nicht aus, um verlässliche Aussagen beispielsweise zur Struktur der Grundzustandslandschaft zu geben. Wir werden deshalb auch auf *stochastische Suchalgorithmen* zurückgreifen, welche zur Klasse der *unvollständigen Suchalgorithmen* gehören.

Im Allgemeinen starten diese Algorithmen in einem zufällig gewählten Zustand und durchlaufen anschließend zufällig den Phasenraum, wobei Schritte in Richtung einer Lösung bevorzugt werden. Ihr Nachteil ist, bei Entscheidungsproblemen überhaupt nur dann erfolgreich sein zu können, wenn die Instanz eine Ja-Instanz ist. Findet der Algorithmus keine Lösung, so ist nicht klar, ob es sich um eine Nein-Instanz handelt oder ob eine an sich existierende Lösung nicht gefunden wurde. Dies liegt daran, dass im Gegensatz zu vollständigen Suchverfahren nicht garantiert alle Zustände

betrachtet werden. Aus dem gleichen Grund können unvollständige Algorithmen bei einem Optimierungsproblem nicht beweisen, dass das von ihnen gefundene Minimum auch das tatsächliche ist.

Bei Ja-Instanzen sind unvollständige Algorithmen den vollständigen dagegen meist überlegen. Zudem sind die im Folgenden beschriebene *Monte-Carlo-Suche* (MC) und das darauf aufbauende *Parallel Tempering* (PT) auch für die Untersuchung physikalischer Systeme sehr interessant, da sich mit ihnen Zustände entsprechend ihres thermodynamischen Gewichtes auswählen und damit beispielsweise Erwartungswerte numerisch bestimmen lassen. Hier werden nur einführend die für das Verständnis der Arbeit notwendigen Grundlagen erläutert, für eine genauere Beschreibung siehe [8, 9].

Monte-Carlo-Suche und Parallel Tempering gehören zu einer speziellen Klasse von stochastischen Algorithmen, bei denen die Wahrscheinlichkeit $w(S \to S')$, dass der Algorithmus aus dem Zustand S in einen Zustand S' übergeht, nicht vom vorherigen Verlauf des Algorithmus abhängt. Die Menge der Zustände $\{S\}$ zusammen mit den gegebenen Übergangswahrscheinlichkeiten $w(S \to S')$ wird als *Markov-Prozess* bezeichnet. Im einfachsten (praktisch aber meist unbrauchbaren) Fall sind dabei die Übergangswahrscheinlichkeiten $w(S \to S') = 1/|\{S\}|$, d. h. jede Konfiguration wird gleichwahrscheinlich erreicht, der Algorithmus durchläuft den Phasenraum völlig zufällig. Gebräuchlicher sind Übergangswahrscheinlichkeiten, die die Änderung der Anzahl der verletzten Wechselwirkungen im Spinglas-Problem oder der Anzahl der Knoten in der Überdeckung beim Vertex-Cover-Problem berücksichtigen.

Sei $p(S,t)$ die Wahrscheinlichkeit, dass der Algorithmus zum Zeitpunkt t im Zustand S ist. Die zeitliche Entwicklung von $p(S,t)$ ist dann durch die *Mastergleichung*

$$p(S,t+1) - p(S,t) = \sum_{S'} w(S' \to S)p(S',t) - w(S \to S')p(S,t) \qquad \forall S \tag{2.1}$$

gegeben. Die zugehörige Differentialgleichung hat unter bestimmten Bedingungen eine stationäre, d. h. zeitunabhängige Lösung, deren Dichten $p^*(S,t) \equiv p^*(S)$ damit

$$0 = p^*(S,t+1) - p^*(S,t) \qquad \forall S \tag{2.2}$$

erfüllen. Dafür ist hinreichend, dass sich schon die Beiträge jedes einzelnen Paars (S,S') von Zuständen wegheben, d. h. dass

$$w(S' \to S)p^*(S') = w(S \to S')p^*(S) \qquad \forall (S,S') \tag{2.3}$$

gilt (genannt *detailliertes Gleichgewicht*).

In dieser Arbeit werden wir insbesondere das großkanonische Ensemble bei Temperatur 0 und chemischem Potential μ für ein System verwenden, dessen Zustände $\vec{x}$ durch N Besetzungszahlen $x_1, x_2, \ldots, x_N$ mit $x_i \in \{0,1\}$ charakterisiert werden. Das System ist über das chemische Potential μ an ein Teilchenreservoir gekoppelt, die Summe $n := \sum_{i=1}^{N} x_i$ gibt die Teilchenanzahl im Zustand $\vec{x}$ an.

Das Gewicht $p(\vec{x})$ eines Zustandes $\vec{x}$ in diesem Ensemble ist dann gegeben durch

$$p(\vec{x}) = \frac{1}{\Xi} \exp\left(\mu \sum_{i=1}^{N} x_i\right) = \frac{1}{\Xi} \exp\left(\mu n\right), \tag{2.4}$$

wobei die großkanonische Zustandssumme Ξ die Wahrscheinlichkeiten normiert, so dass die Summe über alle Zustände 1 ist.

Das chemische Potential μ hat die Rolle eines Gewichtungsfaktors und bevorzugt für große Werte Zustände, in denen die Summe der Besetzungszahlen, d. h. die Anzahl n der Teilchen, möglichst groß ist.

Ziel ist es, den Mittelwert einer Observablen $O(\vec{x})$ in diesem Ensemble zu bestimmen, d. h.

$$\langle O \rangle = \sum_{\vec{x}} O(\vec{x}) p(\vec{x}), \tag{2.5}$$

wobei über alle Zustände summiert wird.

Da es aber für große Systeme oftmals nicht möglich ist, alle Zustände zu enumerieren, wendet man das Monte-Carlo-Verfahren an. Dessen Idee ist es, die Gewichte der Zustände $p(\vec{x})$ als stationäre Wahrscheinlichkeiten $p^*(\vec{x})$ eines geeigneten Markov-Prozesses zu interpretieren.

Dabei verläuft ein Schritt des Algorithmus wie folgt: Befindet sich der Algorithmus im Zustand $\vec{x}$, so wird mit Wahrscheinlichkeit $q(\vec{x} \to \vec{x}')$ der Zustand $\vec{x}'$ als potentieller nächster Zustand ausgewählt. Anschließend wird dieser Zustand mit Wahrscheinlichkeit $r(\vec{x} \to \vec{x}')$ akzeptiert und der Schritt ausgeführt. Die Übergangswahrscheinlichkeiten haben deshalb die Form

$$w(\vec{x} \to \vec{x}') = q(\vec{x} \to \vec{x}') r(\vec{x} \to \vec{x}') \qquad \forall \vec{x} \neq \vec{x}'. \tag{2.6}$$

Wenn der Schritt nicht akzeptiert wurde, bleibt der Algorithmus im Zustand $\vec{x}$, d. h. $w(\vec{x} \to \vec{x}) = 1 - \sum_{\vec{x} \neq \vec{x}'} w(\vec{x} \to \vec{x}')$.

Die ausgeführten Schritte sind in der Regel lokal, d. h. $q(\vec{x} \to \vec{x}')$ ist für einen Zustand $\vec{x}$ nur für eine Teilmenge $\{\vec{x}'\}$ aller Zustände verschieden von Null.

Setzt man die gewünschten Gewichte aus (2.4) in die hinreichende Bedingung (2.3) für stationäres Gleichgewicht ein, so ergibt sich

$$w(\vec{x}' \to \vec{x}) \frac{1}{\Xi} \exp\left(\mu n'\right) = w(\vec{x} \to \vec{x}') \frac{1}{\Xi} \exp\left(\mu n\right) \tag{2.7}$$

$$\Leftrightarrow \qquad \frac{w(\vec{x}' \to \vec{x})}{w(\vec{x} \to \vec{x}')} = \frac{\exp\left(\mu n\right)}{\exp\left(\mu n'\right)} \tag{2.8}$$

$$= \exp\left(\mu (n - n')\right) \tag{2.9}$$

Eine mögliche Wahl der Übergangswahrscheinlichkeiten für einen Markov-Prozess, welcher detailliertes Gleichgewicht erfüllt, ist also

$$q(\vec{x} \to \vec{x}') = q(\vec{x}' \to \vec{x}) \tag{2.10}$$

und

$$r(\vec{x} \to \vec{x}') = \begin{cases} e^{\mu(n'-n)} & \text{wenn } n' < n, \\ 1 & \text{sonst.} \end{cases} \tag{2.11}$$

Ein potentieller Schritt von Zustand $\vec{x}$ nach Zustand $\vec{x}'$ wird also auf jeden Fall ausgeführt, wenn die Teilchenanzahl im System zunimmt, ansonsten mit Wahrscheinlichkeit $e^{\mu(n'-n)}$.

Das Verfahren lässt sich ganz analog für ein System im kanonischen Ensemble anwenden, bei dem das Gewicht der Zustände proportional zu $\exp(-\beta H(S))$ gegeben ist, wobei $H(S)$ die Energie und β die inverse Temperatur ist.

Praktisch wird man nicht in jedem Schritt eine zu $\bar{O}$ beitragende Messung der Observablen vornehmen. Zunächst wartet man zu Beginn einige Equilibrationsschritte (typischerweise in der Ordnung der Systemgröße, hier also $\mathcal{O}(N)$ Schritte), bis das System aus der (zufälligen) Startkonfiguration hinreichend nah an den stationären Zustand gelangt ist. Ebenso wird man zwischen den einzelnen Messungen so lange warten, bis aufeinander folgende Messungen dekorrelieren und so eine Fehlerabschätzung möglich ist.

In dieser Arbeit werden wir das Monte-Carlo-Verfahren insbesondere zur Untersuchung der Grundzustandslandschaft verwenden. Dies werden hier die Zustände sein, die in einer Menge von erlaubten Zuständen maximale Teilchenzahl haben. Formal sind dies die Zustände mit endlichem Gewicht im Grenzwert $\mu \to \infty$, numerisch kann aber nur mit endlichem μ gearbeitet werden.

Bei der algorithmischen Umsetzung steht man deshalb vor folgendem Dilemma: Wird μ dabei zu groß gewählt, dann sind die Übergangsraten in 2.11 zu klein. Der Algorithmus hält sich dann sehr lange in einem – möglicherweise sogar nur lokalen – Minimum auf und es werden in realistischer Zeit nur sehr wenige Grundzustände gefunden.

Wählt man μ dagegen zu klein, dann erreicht der Algorithmus Grundzustände ebenfalls nur sehr selten, da die Wahrscheinlichkeit, die Teilchenanzahl zu erniedrigen, sich also weg von einem Grundzustand zu bewegen, zu groß ist.

Um dieses Problem zu vermeiden, werden wir *Parallel Tempering* verwenden, eine Erweiterung des MC-Verfahrens [10, 11]. Idee des in Algorithmus 2.2 dargestellten Verfahrens ist es, gleichzeitig mehrere Kopien des Systems bei unterschiedlichem chemischem Potential zu simulieren.

```
algorithm Parallel_tempering(N_MC, m)
begin
    Initialisierung von m Kopien des Systems mit
        zufälligen Zuständen x^(1) ... x^(m)
    for t = 1 ... N_MC do
    begin
        for k = 1 ... m do          /* Schleife über die m Kopien */
            N Monte-Carlo-Schritte für System x^(k) bei chem. Potential μ_k
        for k = 1 ... m − 1 do
            begin                   /* PT-Moves */
                ΔE_k := (μ_k − μ_{k+1}) · (n^(k) − n^(k+1))
                mit Wahrscheinlichkeit exp(− min(ΔE_k,0))
                    vertausche x^(k) ↔ x^(k+1)
            end
    end
end
```

Algorithmus 2.2: Parallel-Tempering-Verfahren. Insgesamt werden N_{MC} MC-Sweeps für jede der m verschiedenen Kopien des Systems durchgeführt.

Genau wie beim MC-Verfahren wird jede der m Kopien $\vec{x}^{(k)}$ zunächst zufällig initialisiert. Weiterhin werden m chemische Potentiale $\mu_1 < \mu_2 < \cdots < \mu_m$ festgelegt. Anschließend werden für jede Kopie unabhängig N MC-Schritte (d. h. ein *MC-Sweep*) bei chemischem Potential μ_k ausgeführt.

Danach wird für Paare benachbarter Kopien $(\vec{x}^{(k)}, \vec{x}^{(k+1)})$ geprüft, ob diese Kopien miteinander vertauscht werden (*Parallel-Tempering-Move*), d. h. ab dem nächsten Schleifendurchlauf MC-Sweeps beim chemischem Potential der benachbarten Kopie ausführen. Dazu definiert man

$$\Delta E_k := (\mu_k - \mu_{k+1}) \cdot (n^{(k)} - n^{(k+1)}) \qquad \forall k = 1, \ldots, m-1. \tag{2.12}$$

Benachbarte Zustände werden ausgetauscht

- mit Wahrscheinlichkeit $e^{-\Delta E_k}$, wenn $\Delta E_k > 0$,
- immer, wenn $\Delta E_k \leq 0$, d. h. wenn das System mit niedrigerem chemischem Potential in einem Zustand größerer Teilchenanzahl ist, d. h. einem Zustand mit geringerem Gewicht.

Der Vorteil des PT-Verfahrens besteht darin, dass Konfigurationen zum einen bei höherem chemischem Potential lokale Minima durch ihr höheres Gewicht wahrscheinlicher finden, zum anderen bei niedrigerem chemischem Potential größere Energiebarrieren häufiger überwunden werden.

3 Grundzustandsbestimmung endlich-dimensionaler Ising-Spin-Systeme

3.1 Endlich-dimensionale Spingläser

Spingläser sind ungeordnete magnetische Vielteilchensysteme [1, 2, 12, 13]. Ihre magnetischen Eigenschaften werden bestimmt durch Spins, die durch zufällige ferro- und antiferromagnetische Wechselwirkungen gekoppelt sind. Wir wollen in diesem Abschnitt einige Grundlagen dieser Systeme erläutern, da sie Ausgangspunkt für die in den späteren Abschnitten betrachteten Probleme sind. Dabei werden wir immer das Modell der *Ising*-Spins betrachten, bei dem Spins binäre Variablen $S_i \in \{\pm 1\}$ sind.

Im *Edwards-Anderson-Modell* sind diese Spins auf einem d-dimensionalen Gitter angeordnet. Auf diesem Gitter benachbarte Spins sind durch zufällige, eingefrorene Wechselwirkungen J_{ij} gekoppelt [14]. Das System wird durch die Hamiltonfunktion

$$H_{\{J_{ij}\}}(\{S_i\}) = -\sum_{\langle ij \rangle} J_{ij} S_i S_j. \tag{3.1}$$

beschrieben, $\langle ij \rangle$ bezeichnet die Summe über die nächsten Nachbarn auf dem Gitter. Die Wechselwirkungen werden unabhängig voneinander zufällig festgelegt. Dabei verwendet man vielfach entweder eine bimodale $\pm J$ Verteilung

$$P_{\pm J}(J) = \frac{1}{2}\delta(J+1) + \frac{1}{2}\delta(J-1) \tag{3.2}$$

oder eine Gaußverteilung

$$P_G(J) = \frac{1}{\sqrt{2\pi}} \exp(-J^2/2). \tag{3.3}$$

Ein wesentlicher Unterschied der beiden Verteilungen besteht darin, dass für eine $\pm J$-Verteilung der Grundzustand stark entartet ist, während man für gaußsche Wechselwirkungen wegen der kontinuierlichen Werte der Wechselwirkungen genau einen Grundzustand erwartet. Hier werden wir nur Systeme mit gaußschen Wechselwirkungen betrachten.

Ein derartiges System ist im Allgemeinen *frustriert*, da sich nicht alle Paare benachbarter Spins können sich gleichzeitig entsprechend der durch ihre Wechselwirkung bevorzugten Ausrichtung einstellen können (vgl. Kapitel 1). Oberhalb einer kritischen Temperatur T_c verhält sich das System paramagnetisch. Obwohl für $d > 2$ und $T < T_c$ die Gesamtmagnetisierung ebenfalls verschwindet, ist im Zeitmittel $\sum_i \langle S_i \rangle^2$ verschieden von null.

Bislang ist es nicht gelungen, eine einheitliche Beschreibung der Phasenraumstruktur niedrigenergetischer Zustände der Hamiltonfunktion (1.1) zu finden.

Ein mögliches Modell der Phasenraumstruktur ist das Auftreten von *kontinuierlicher Replika-Symmetriebrechung* (RSB-Modell). Diese findet man im Mean-Field-Äquivalent von (3.1), dem Sherrington-Kirkpatrick-Modell (*SK-Modell*) [15]. Dort sind alle Spins miteinander gekoppelt (Wechselwirkungen werden mit $1/\sqrt{N}$ reskaliert, damit Größen wie Energie etc. extensiv bleiben). Für dieses Modell wurde 1979 durch den Parisi-Ansatz eine Lösung gefunden [16], ca. 25 Jahre später wurde die Korrektheit der dabei gefundenen freien Energie von Talagrand mathematisch bewiesen [17]. Die von Parisi verwendete Replika-Methode ist nicht rigoros, wird aber erfolgreich für viele spinglasartige Modelle verwendet. Im Rahmen dieser Arbeit werden wir in Kapitel 4 auf die Ergebnisse ihrer Anwendung auf das Vertex-Cover-Problem eingehen.

Die für dieses Modell vorausgesagte Energielandschaft hat einige interessante Eigenschaften:

- Es gibt Zustände mit $\mathcal{O}(1)$-Energiedifferenz zum Grundzustand, die diesem im Phasenraum sehr unähnlich sind, d. h. sie unterscheiden sich in $\mathcal{O}(N)$ Spins.
- Die niedrigenergetischen Zustände sind nicht gleichmäßig über den Phasenraum verteilt, sondern bilden zusammenhängende Bereiche, sogenannte *Cluster*. Diese Cluster selbst sind wiederum in Clustern organisiert, diese Hierarchie setzt sich unendlich fort (siehe Abbildung 4.10 auf Seite 51).
- Ultrametrizität (siehe Abschnitt 4.5.5)

Um die Gültigkeit des RSB-Modells für dreidimensionale Spingläser numerisch zu testen, untersucht man in Systemen mit N Spins die Verteilung $P(q)$ des Überlapps $q^{\alpha\beta} = 1/N \sum_i S_i^{(\alpha)} S_i^{(\beta)}$ zwischen zwei Zuständen $\vec{S}^{(\alpha)}$ und $\vec{S}^{(\alpha)}$, gemittelt über alle Zustände einer Realisierung (d. h. einer konkreten Wahl der J_{ij}) und anschließend gemittelt über alle Realisierungen [18–20]; des Weiteren untersucht man das Spektrum der Spin-Spin-Korrelationsmatrix [21, 22] oder versucht, direkt Cluster [23] bzw. Ultrametrizität zu detektieren [23–26]. Einige dieser Methoden werden wir in Kapitel 4 auf das Vertex-Cover-Problem anwenden.

Mit dem RSB-Modell konkurriert vor allem das *Droplet-Bild* [27–29]. In diesem Modell erwartet man eine viel einfachere Energielandschaft: Im Phasenraum gibt es

(bis auf Symmetrie) nur einen Bereich niedrigenergetischer Zustände. Man betrachtet dabei sogenannte *Droplet*-Anregungen, d. h. eine Anregung eines im Realraum kompakten Bereichs von Spins vorgegebener Ausdehnung l, die zu einem Zustand mit lokal minimaler Energie führt. Im Droplet-Bild steigt die Anregungsenergie eines solchen Droplets mit einer positiven Potenz ihres Volumens: Wird der Wert von $\mathcal{O}(l^3)$ Spins invertiert, ändert sich die Energie mindestens um l^θ mit $\theta \approx 0.2$ [30–32]. Folglich kann es im Droplet-Bild keine systemweiten Anregungen mit $\mathcal{O}(1)$ Energiedifferenz geben, wie vom RSB-Modell vorausgesagt.

Ebenfalls unterschiedliche Voraussagen werden für die Verteilung der Überlapps gemacht: Wegen der einfachen Clusterstruktur erwartet man im Droplet-Bild im thermodynamischen Limes eine Deltafunktion, im RSB-Modell dagegen eine kontinuierliche Verteilung.

3.2 Grundzustandsbestimmung durch Genetische Renormalisierung (GRA)

Die außerordentliche numerische Schwierigkeit der Grundzustandsbestimmung von endlich-dimensionalen Spingläsern für $d > 2$ ist eine Ursache, dass die im vorherigen Abschnitt beschriebene Frage „Gilt das RSB-Modell, das Droplet-Bild oder ist möglicherweise keines der beiden Szenarios vollständig gültig?“ noch immer ungeklärt ist.

Im Rahmen dieser Arbeit wurde ein heuristischer Algorithmus weiterentwickelt, mit dem sich für Systeme bis zu 14^3 Spins Grundzustände bestimmen lassen, was für dreidimensionale Spingläser im Bereich der größten erreichten Systemgrößen liegt.

Das Verfahren basiert auf dem *Genetischen Renormalisierungs-Algorithmus* (GRA) von Houdayer und Martin [33]. Der Algorithmus gehört zur Klasse der unvollständigen Algorithmen (vgl. Abschnitt 2.2.2).

Die Bausteine des Algorithmus sind

- lokale Optimierung,
- Verwendung der Techniken genetischer Algorithmen,
- Renormalisierung.

Wir werden zunächst das ursprüngliche Verfahren der Genetischen Renormalisierung von Houdayer und Martin vorstellen und dabei insbesondere auf die Renormalisierung eingehen, welche diesen Algorithmus von anderen Spinglas-Grundzustandsalgorithmen unterscheidet. Für eine detaillierte Beschreibung der anderen beiden Bausteine des Verfahrens siehe [34].

3.2.1 Lokale Optimierung der Konfigurationen

Wir kennen bereits aus Abschnitt 2.2.2 das generelle Vorgehen eines unvollständigen Suchalgorithmus: Man startet zunächst in einem zufällig initialisierten Zustand $\vec{S}$. Anschließend ändert man diesen Zustand lokal, indem man den Wert einzelner Spins umkehrt, bevorzugt in die Richtung, die (3.1) für die gegebene Konfiguration minimiert.

Hier wird das Verfahren von Kerninghan und Lin verwendet [35]. Ausgehend von einem Spin S^* wird durch Hinzufügen benachbarter Spins ein Cluster $\mathcal{C}(t)$ zusammenhängender Spins erzeugt. In jedem Schritt t wird die Energiedifferenz $\Delta E(t)$ bestimmt zwischen dem Ausgangszustand und dem Zustand, in dem alle Spins aus $\mathcal{C}(t)$ entgegengesetzte Werte haben. Dies wird so lange fortgesetzt, bis für eine vorgegebene Anzahl von Schritten kein neues globales Minimum von $\Delta E(t)$ gefunden wurde. Anschließend wird der Wert aller Spins des zum dabei gefundenen Minimum gehörenden Clusters geändert.

S^* iteriert über alle Spins, die nicht entsprechend ihres lokalen Feldes ausgerichtet sind. Dies sind alle diejenigen Spins, bei denen schon ein Ändern des Wertes von S^* allein die Energie zumindest nicht erhöhen würde.

Die Suche wird so lange fortgesetzt, bis kein solcher Spin mehr vorhanden ist, die Konfiguration ist dann in einem lokalen Minimum der Energie.

3.2.2 Genetische Algorithmen

Genetische Algorithmen [36] sind inspiriert durch evolutionsbiologische Begriffe und werden auch von anderen Spinglas-Grundzustandsalgorithmen verwendet, beispielsweise bei der Cluster-exakten Approximation [37].

Man betrachtet gleichzeitig mehrere Kopien des Systems (genannt *Individuen*), die Gesamtheit der Individuen ist die *Population*. Zunächst wird jedes Individuum für sich optimiert. In regelmäßigen Abständen fügt man durch Kombination vorhandener Individuen neue zur Population hinzu (*Crossover*). Dies wird hier durch die im nächsten Abschnitt beschriebene Renormalisierung geschehen. Anschließend entfernt man Individuen, beispielsweise diejenigen mit der größten (d. h. ungünstigsten) Energie.

3.2.3 Renormalisierung

Die ursprüngliche Idee der Verwendung von Renormalisierung zur Bestimmung von Spinglas-Grundzuständen geht zurück auf Kawashima und Suzuki [38].

Aus den k Individuen der Population werden zufällig l Individuen $\vec{S}^{(1)}, \vec{S}^{(2)}, \ldots, \vec{S}^{(l)}$ (*Eltern*) ausgewählt. Für jeden Spin S_i definieren wir die *Signatur*

$$\vec{q}_i = (S_i^{(1)} S_i^{(2)}, S_i^{(1)} S_i^{(3)}, \ldots, S_i^{(1)} S_i^{(l)}). \tag{3.4}$$

Alle Spins mit gleicher Signatur haben dieselbe Ausrichtung in allen Individuen. Wir können deshalb benachbarte Spins mit gleicher Signatur zusammenfassen und durch einen *Ising-Blockspin* $S_I \in \pm 1$ beschreiben und damit l neue, renormalisierte Konfigurationen $\vec{S}_I'^{(1)}, \vec{S}_I'^{(2)}, \ldots, \vec{S}_I'^{(l)}$ erzeugen (*Kinder*, vgl. Abbildung 3.1).

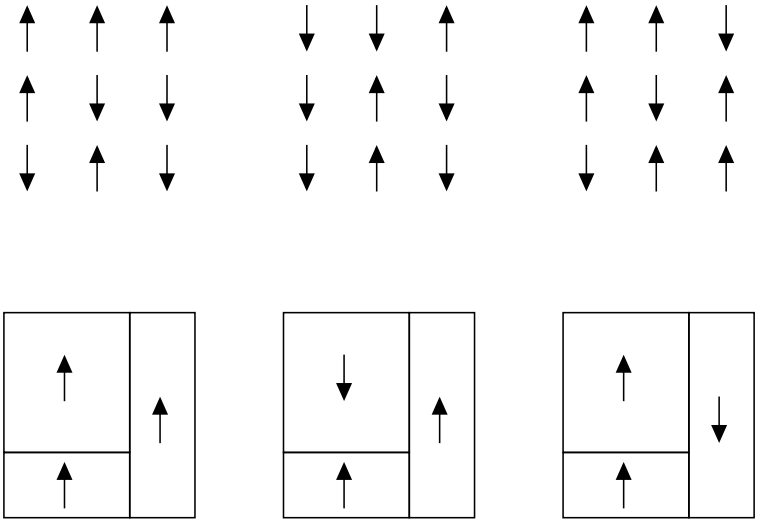


Abbildung 3.1: Genetische Renormalisierung für eine Population aus drei Eltern-Individuen (oben). Es werden drei neue, renormalisierte Kinder-Individuen erzeugt (unten), indem benachbarte Spins, die in allen Eltern dieselbe relative Ausrichtung haben, zu einem Blockspin zusammengefasst werden. Alle Kinder werden durch dieselbe renormalisierte Hamiltonfunktion $H^R_{\{J'_{IJ}\}}(\{S'_I\})$ beschrieben. Diese wird so gewählt, dass beim ersten Kind alle Blockspins den Wert $S'_I = 1$ besitzen, in den anderen Kindern ergibt sich der Wert aus der Signatur.

Dabei setzen wir für $1 \leq j \leq l$

$$S_I'^{(j)} = S_i^{(1)} S_i^{(j)} \text{ mit } i \in I \text{ beliebig}, \tag{3.5}$$

d. h. in $S'^{(1)}$ haben alle Blockspins den Wert $+1$, für die anderen renormalisierten Individuen ergibt sich der Wert aus der Signatur.

Damit ergibt sich als Hamiltonfunktion für das renormalisierte System

$$H^R_{\{J'_{IJ}\}}(\{S'_I\}) = -\sum_{\langle i,j \rangle} J'_{IJ} S'_I S'_J \tag{3.6}$$

mit

$$J'_{IJ} = \sum_{i \in I} \sum_{j \in J} J_{ij} S_i^{(1)} S_j^{(1)}, \tag{3.7}$$

wobei der zusätzliche konstante Anteil der Wechselwirkungen innerhalb der Blockspins weggelassen wurde. Die Energien $H_{\{J_{ij}\}}(\{S_i^{(j)}\})$ der Ausgangskonfiguration und der zugehörigen renormalisierten Konfiguration $H^R_{\{J'_{IJ}\}}(\{S_I^{\prime(j)}\})$ stimmen bis auf diese Konstante überein.

Jede renormalisierte Konfiguration kann nun mit lokaler Suche für sich optimiert werden. Die Anzahl der Spins des renormalisierten Systems ist gegenüber den ursprünglichen Individuen reduziert. Die lokale Suche ist deshalb effizienter, da durch die Renormalisierung gezielt die Gruppen von Spins ausgewählt werden, für die durch die bisherige Optimierung noch keine relative Übereinstimmung der Orientierung gefunden wurde.

Da die Abbildung eineindeutig ist, können die optimierten renormalisierten Konfigurationen zurück auf das Gitter transformiert werden. Ein optimiertes Kind ersetzt sein Eltern-Individuum in der Population, wenn es eine bessere oder die gleiche Energie besitzt. Wenn schon ein identisches Individuum in der Population vorhanden ist, wird nur das Eltern-Individuum entfernt.

Die Prozedur wird dann mit einer anderen Auswahl von Eltern neu gestartet. Wenn nur noch ein Individuum in der Population vorhanden ist oder sich die Durchschnittsenergie der Population über einen vorgegebenen Zeitraum nicht mehr ändert, wird die beste Konfiguration als Ergebnis ausgegeben.

3.2.4 Verbesserung durch Mutation und Abstimmung

Mit der im vorigen Abschnitt vorgestellten Version des Genetischen Renormalisierungs-Algorithmus lassen sich Grundzustände für Systeme bis 12^3 Spins bestimmen. Im Rahmen dieser Arbeit wurde der Algorithmus an zwei Stellen modifiziert:

- *„Abstimmung“:*
  Typischerweise startet der Algorithmus mit einer Population von 50 bis 100 Individuen, von denen bei jeder Renormalisierung bis zu $l = 30$ ausgewählt werden. Die ursprüngliche Definition der Signatur verlangt dann, dass die zu einem Blockspin gehörenden Spins in allen l Individuen übereinstimmen. Dieses Kriterium wird gelockert: Zunächst werden wie in der ursprünglichen Version die Signaturen berechnet. Ein Blockspin wird iterativ erzeugt, indem zunächst ein Spin ausgewählt wird. Alle benachbarten Spins, deren Signatur sich von einem bereits ausgewählten Spins an nicht mehr als d Positionen unterscheidet, werden zum Blockspin hinzugefügt, bis keine weiteren Spins hinzugefügt werden können. Diese Prozedur wird wiederholt, bis alle Spins einem Blockspin zugeordnet wurden.

Der ursprüngliche Algorithmus entspricht dem Fall $d = 0$. Wir verwenden $d = 1$ oder $d = 2$, dadurch wird die Auswahl der Blockspins deutlich verbessert.

- *Mutation vor der lokalen Optimierung:*
  Ausgangspunkt der Clusterbildung bei der lokalen Optimierung der renormalisierten Konfigurationen mit dem Kerninghan-Lee-Algorithmus (KL-Algorithmus, vgl. Abschnitt 3.2.1) sind alle Spins, die nicht in ihrem lokalen Feld ausgerichtet sind. Das renormalisierte System ist aber kein regelmäßiges Gitter mehr. Die Verteilung der Anzahl der Nachbarn ist sehr breit, insbesondere gibt es auch Spins mit sehr vielen Nachbarn.

  Im späteren Verlauf des Algorithmus werden die meisten Spins bereits entsprechend ihres lokalen Feldes ausgerichtet sein. Ändert man vor der lokalen Optimierung den Wert der pN größten Spins, so werden diese mit großer Wahrscheinlichkeit als Ausgangspunkt eines Clusters des KL-Algorithmus gewählt. Empirisch findet man, dass durch diese Störung größere Cluster generiert werden und dadurch andere lokale Minima der Energielandschaft erreicht werden, die Energielandschaft wird durch Mutation besser erforscht.

In unserer Implementierung finden wir die besten Ergebnisse für $d = 1$ oder $d = 2$ und $p \approx 0.1$. Damit lassen sich Grundzustände für Systeme bis 14^3 bestimmen. Zur Illustration ist in Abbildung 3.2 der Verlauf der Differenz zwischen mittlerer Energie der Population und minimaler Energie für eine zufällig ausgewählte Instanz mit 14^3 Spins und gaußschen Wechselwirkungen im Vergleich zwischen urspünglicher und verbesserter Version des Algorithmus angegeben. Nach 10 bzw. 20 Schritten wurde die Population durch Entfernen der Individuen mit größter Energie halbiert. Man sieht, dass die verbessere Version deutlich schneller in Richtung minimaler Energie strebt und dabei einen Zustand geringerer Energie findet.

Als Konsistenztest wurde erfolgreich für 100 Zustände der Größe 14^3 mit $\pm J$-Wechselwirkungen überprüft, ob ein anderer Algorithmus (Cluster-exakte Approximation) dieselbe Grundzustandsenergie findet.

3.3 Untersuchung der Grundzustandslandschaft mit dem Genetischen Renormalisierungs-Algorithmus

Mit Hilfe des modifizierten GRA wurde die Grundzustandslandschaft dreidimensionaler Ising-Spin-Gläser mit gaußverteilten Wechselwirkungen betrachtet. Dabei sollte untersucht werden, wie die Energie von systemweiten Anregungen mit der Größe des Systems steigt.

Replika-Symmetriebrechung und Dropletbild treffen unterschiedliche Voraussagen für die Landschaft der Zustände niedriger Energie, wie im Abschnitt 3.1 beschrieben. Während man bei RSB systemweite Anregungen erwartet, die nur eine endliche

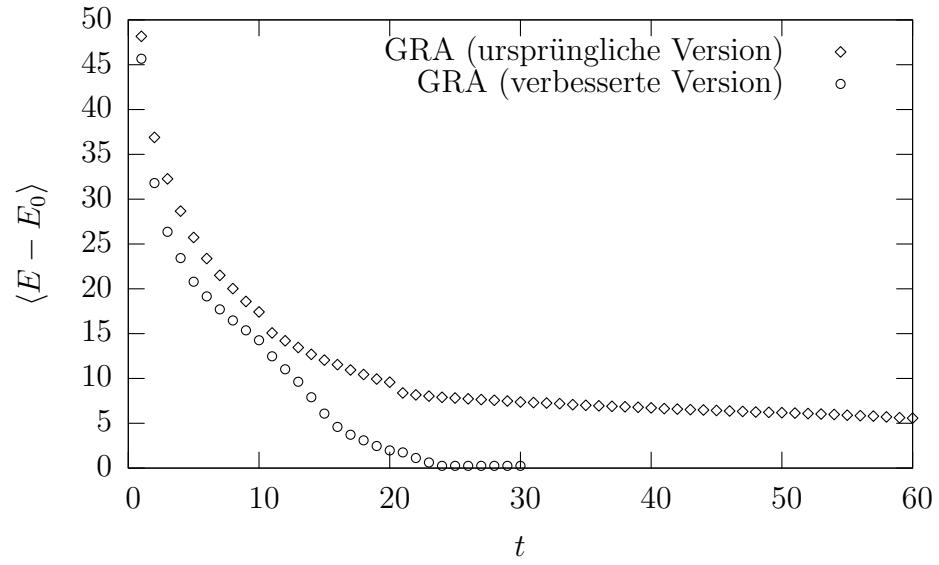


Abbildung 3.2: Verlauf der mittleren Energie der Population in Abhängigkeit von der Anzahl der Renormierungsschritte beispielhaft für eine Instanz mit $N = 14^3$.

(d. h. $\mathcal{O}(1)$) Energiedifferenz ΔE zum Grundzustand besitzen, sind solche Zustände im Dropletbild unwahrscheinlich. Genauer gesagt erwartet man für eine kompakte Anregung der Ausdehnung L eine Skalierung der Energie wie L^θ mit $\theta \approx 0.2$ im Dropletbild und $\theta = 0$ bei RSB.

Um eine Anregung als systemweite zu klassifizieren, verwenden wir ein topologisches Kriterium. Vergleicht man zwei beliebige Zustände, dann unterteilt sich die Menge der Spins in zwei Teilmengen. In der einen sind die Spins mit übereinstimmender, in der anderen die mit entgegengesetzter Ausrichtung. Wir betrachten das System mit periodischen Randbedingungen und definieren zwei Grundzustände als durch eine systemweite Anregung verbunden, wenn es in jeder der beiden Teilmengen eine verbundene Komponente gibt, die sich in mindestens einer Richtung komplett um das System schlingt. Andernfalls wird die Anregung *trivial* genannt.

Wir speichern während der Genetischen Renormalisierung nicht nur die Grundzustände, sondern in einer Liste auch alle angeregten Zustände im Intervall $[E_0,E_0+8]$, wobei E_0 die Energie des Grundzustandes ist. Bei den hier betrachteten Systemen mit gaußschen Wechselwirkungen sind alle Energieniveaus nicht entartet. Beginnend mit dem Grundzustand wird für jeden Zustand überprüft, ob er durch eine triviale Anregung mit einem Zustand niedrigerer Energie verbunden ist. In diesem Fall wird der Zustand aus der Liste entfernt (vgl. Abbildung 3.3). Alle verbleibenden Zustände können wir als lokale Minima der Energielandschaft betrachten.

Mit dem Algorithmus lassen sich diese Niveaus bis zu einer Größe von 12^3 Spins verlässlich bestimmen, was wir durch wiederholte (unterschiedlich randomisierte)

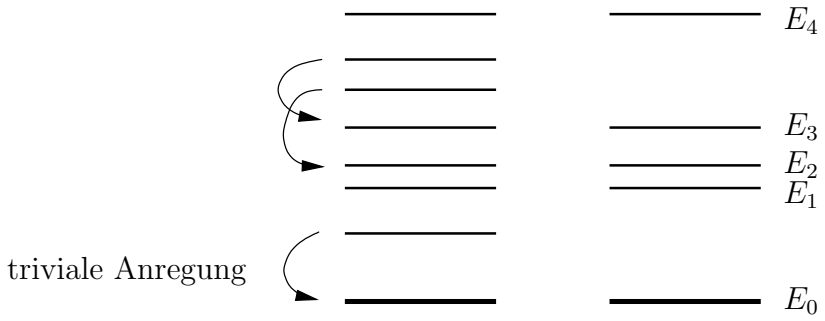


Abbildung 3.3: Alle durch triviale Anregung mit einem Zustand niedriger Energie verbundenen Zustände werden herausgefiltert. Die verbleibenden Zustände sind lokale Minima der Energielandschaft.

Anwendung des Algorithmus auf dieselbe Instanz überprüft haben. Für jede Systemgröße wurde über 1000 Realisierungen gemittelt.

Die Verteilung der Energiedifferenzen zwischen Grundzustand mit Energie E_0 und dem ersten mit ihm durch eine systemweite Anregung verbundenen Zustand E_1 ist in Abbildung 3.4 dargestellt. Man sieht, dass es keine Tendenz der Verteilung gibt, für kleine Werte von ΔE geringer zu werden, so wie man es im Droplet-Bild erwartet hätte. Fast jede Realisierung hat eine Anregung im Intervall $[E_0, E_0 + 4]$. Man erkennt darüber hinaus im Inset, dass auch der Mittelwert der Verteilung annähernd konstant ist bzw. eher geringer wird, Anregungen geringerer Energie werden also wahrscheinlicher. Dies heißt zusammengefasst, dass die Wahrscheinlichkeit für eine systemweite Anregung nicht verschwindet, in Übereinstimmung mit dem von RSB vorausgesagten Verhalten.

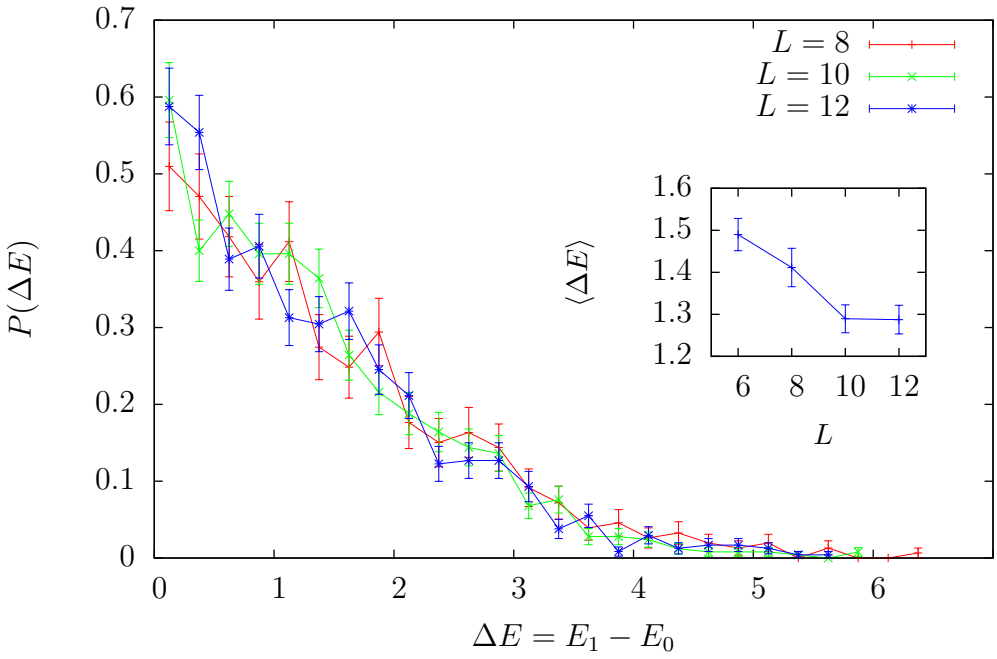


Abbildung 3.4: Verteilung der Energiedifferenzen $\Delta E = E_1 - E_0$ zwischen Grundzustand und erster nicht-trivialer Anregung. Im Inset ist der Mittelwert der Verteilung in Abhängigkeit von der Systemgröße dargestellt.

4 Numerische Analyse der Grundzustandsstruktur des Vertex-Cover-Problems

Das Vertex-Cover-Problem (VC-Problem) gehört zur Klasse der NP-vollständigen Probleme. In der Formulierung als Optimierungsproblem ist für einen gegebenen Graphen eine möglichst kleine Untermenge von Knoten gesucht (genannt *minimales Vertex-Cover*), die von jeder Kante mindestens einen ihrer beiden Endpunkte enthält.

Wie in der Einleitung beschrieben, sollen typische Eigenschaften dieses Problems untersucht werden. Im Fall des VC-Problems betrachtet man dazu das Ensemble von *Zufallsgraphen endlicher Konnektivität* c, d. h. jeder Knoten ist im Mittel mit einer endlichen Anzahl c anderer Knoten durch eine Kante verbunden. Viele Eigenschaften von Graphen dieses Ensembles hängen im thermodynamischen Limes nicht wie erwartet vom Graphen selbst ab, sondern mit Wahrscheinlichkeit 1 nur von seiner mittleren Konnektivität.

Eine der wichtigsten derartigen Eigenschaften ist die Größe des minimalen Vertex-Covers. Diese ist für Konnektivitäten $c < e \approx 2{,}71\ldots$ als Funktion von c exakt analytisch bestimmt [39, 40], für größere Konnektivitäten ist eine analytische Lösung bislang unbekannt.

Minimale Vertex-Cover werden wir in diesem Zusammenhang als Grundzustände eines Systems mit geeignet definierter Energie auffassen. Interessanterweise beobachtet man am Punkt der Instabilität der analytischen Lösung bei $c = e$ gleichzeitig auch eine Veränderung der Grundzustandslandschaft, die eng mit dieser Instabilität zusammenzuhängen scheint:

- Für $c < e$ ist die Menge der minimalen Vertex-Cover verbunden im Phasenraum.
- Für $c > e$ findet man das Phänomenen der *Clusterung*: Grundzustände sind in mehreren Gruppen organisiert, die für sich verbundene Mengen darstellen, aber voneinander im Phasenraum getrennt sind. Darüber hinaus können diese Gruppen selbst wieder unregelmäßig im Phasenraum verteilt sein und auf diese Weise eine Art hierarchische Clusterstruktur formen.

Dieses Verhalten ist in der Statistischen Physik bei Spingläsern [1, 2] bekannt (vgl. Abschnitt 3.1).

Auch bei anderen kombinatorischen Optimierungsproblemen wie dem K-Erfüllbarkeitsproblem (siehe Kapitel 5) findet man einen Zusammenhang zwischen Instabilitäten der analytischen Lösung und einer Änderung der Grundzustandslandschaft [41]. Allerdings ist die analytische Untersuchung der Clusterstruktur sehr schwierig, beispielsweise konnte die Existenz von Clustern überhaupt bisher nur für $K \geq 8$ gezeigt werden [42].

Die Grundzustandslandschaft des VC-Problems ist auch aus algorithmischer Sicht interessant. Für Konnektivitäten $c < e$ kennt man Algorithmen, die typischerweise in linearer Zeit minimale VC finden [40], für größere c ist nicht bekannt, ob ein solcher Algorithmus existieren kann. Vom K-Erfüllbarkeitsproblem weiß man aber, dass das Finden von Lösungen für bestimmte lokale Algorithmen wie der in Kapitel 5 dieser Arbeit untersuchte Walk-SAT-Algorithmus sehr viel schwieriger ist, wenn die Grundzustandslandschaft in Cluster zerfällt. Dies scheint mit dem gleichzeitigen Auftreten metastabiler Zustände zusammenzuhängen, in denen sich diese Algorithmen dann lange Zeit aufhalten. Die Anzahl dieser metastabilen Zustände selbst hängt wiederum sehr von der genauen Form der Clusterung ab [41].

In diesem Teil der Arbeit wollen wir deshalb die Organisation der Grundzustandslandschaft des VC-Problems numerisch untersuchen. Wir werden dabei die im Abschnitt 3.1 für Spingläser aufgeführten Methoden verwenden und im Folgenden noch näher beschreiben. Insbesondere wird uns der Bereich $c > e$ interessieren, für den die analytische Lösung noch offen ist und bislang kein effizienter Algorithmus bekannt ist.

4.1 Definition des Vertex-Cover-Problems

Ein *ungerichteter Graph* G ist ein Paar (V, E) von zwei Mengen. Die Elemente $v \in V$ bezeichnet man als *Knoten*. Die Menge E besteht aus zweielementigen Teilmengen $\{v, w\}$ von V, die als *Kanten* bezeichnet werden. Jede Kante verbindet somit genau zwei Knoten, keine zwei Knoten werden durch mehr als eine Kante verbunden. Ist für $v, w \in V$ das Paar $\{v, w\} \in E$, so heißen die beiden Knoten *benachbart*.

Wir bezeichnen mit $N = |V|$ die Anzahl der Knoten und mit $M = |E|$ die Anzahl der Kanten. Der *Grad* eines Knotens ist die Anzahl seiner Nachbarn. Das Verhältnis $c = 2M/N$, die mittlere Konnektivität, entspricht der durchschnittlichen Anzahl von Nachbarn eines Knotens.

Ein Graph heißt *Untergraph* $G' = (V', E')$ von G, wenn $V' \subset V$ und $E' \subset E$ gilt. Ein Untergraph heißt *Pfad* der Länge l, wenn er die Form $V' = \{v_0, v_2, \ldots, v_l\}$, $E' = \{\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{l-1}, v_l\}\}$ besitzt.

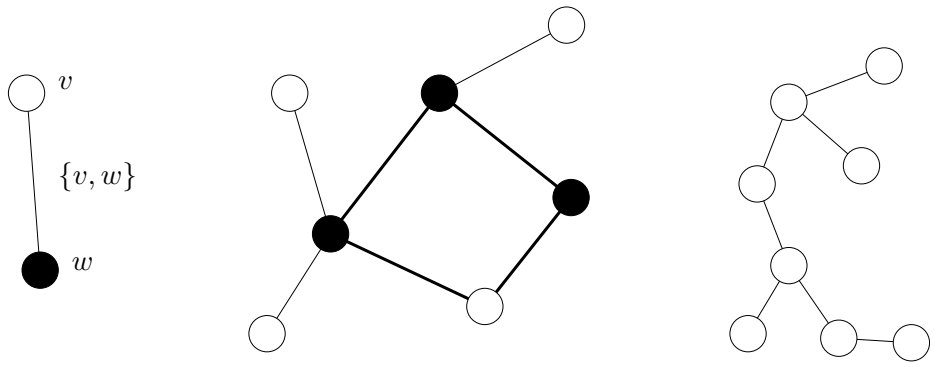


Abbildung 4.1: Grundbegriffe der Graphentheorie und des Vertex-Cover-Problems: *Links:* v und w sind zwei durch eine Kante $\{v, w\}$ verbundene Knoten, v und w sind benachbart, v ist im Zustand „unbedeckt“, w ist im Zustand „bedeckt“; *Mitte:* Graph mit $N = 7$ Knoten, $M = 7$ Kanten und mittlerer Konnektivität $c = 2M/N = 2$. Der Graph enthält einen Kreis der Länge vier (fett gezeichnete Kanten) und hat ein minimales Vertex-Cover der Größe 3 (bedeckte Knoten); *Rechts:* Baum (= Graph ohne Kreis).

Ein Pfad der Länge l heißt *Kreis* der Länge l, wenn $v_0 = v_l$ gilt und jeder Knoten $v_i \in V'$ in genau zwei Kanten aus E' enthalten ist. Ein *Baum* ist ein Graph, der keinen Kreis enthält.

Jeder Knoten des Graphen ist in einem von zwei möglichen Zuständen: *bedeckt* oder *unbedeckt*. Eine Kante heißt bedeckt, wenn mindestens einer ihrer beiden Endpunkte bedeckt ist. Eine Menge U von bedeckten Knoten bildet ein *Vertex-Cover*, wenn dadurch alle Kanten des Graphen bedeckt sind.

Ein VC ist demnach eine Untermenge $U \subset V$ der Knoten, die von jeder Kante mindestens einen Endpunkt enthält, d. h.

$$\forall\ \{v, w\} \in E : v \in U \text{ oder } w \in U. \tag{4.1}$$

Aus physikalischer Sicht kann man für eine beliebige Untermenge $U \subset V$ von bedeckten Knoten die Größe der Menge

$$E(G, U) := \left|\left\{\ \{v, w\} \in E : v \notin U \text{ und } w \notin U\right\}\right|, \tag{4.2}$$

d. h. die Anzahl der von U nicht bedeckten Kanten, als *Energie* $E(G, U)$ auffassen. In diesem Sinne sind Vertex-Cover $U \subset V$ *Grundzustände* mit der Energie $E(G, U) = 0$.

Die Größe $X = |U|$ eines VC ist die Anzahl der bedeckten Knoten. Für jeden beliebigen Graphen bildet die Menge aller Knoten trivialerweise immer ein VC, es gibt somit immer ein Vertex-Cover der Größe N.

Als VC-Problem bezeichnet man folgendes Entscheidungsproblem:

(E1) Gegeben sei ein Graph G und eine natürliche Zahl $K = xN$. Gibt es ein Vertex-Cover mit $|U| \leq K$?

Hier werden wir insbesondere zwei zugehörige Optimierungsprobleme betrachten:

(O1) Gegeben sei ein Graph G. Man bestimme ein VC minimaler Größe $X_{\min}(G)$!

(O2) Gegeben sei ein Graph G und eine natürliche Zahl $K = xN$. Man bestimme $U \in V$ mit $|U| = K$ und $E(G, U)$ minimal!

Das erste Optimierungsproblem wird als *Problem des minimalen VC* bezeichnet. Das zweite entspricht der Frage, für einen Graphen $G(V, E)$ Grundzustände eines Systems mit der Hamiltonfunktion (4.2) und den Untermengen $U \in V$ der Größe K als Zustände zu bestimmen. Die zugehörige minimale Energie bezeichnen wir deshalb als Grundzustandsenergie $E_0(G, K)$ bzw. $E_0(G, x)$. Ist für ein solches System $E_0(G, x) = 0$, so sind dessen Grundzustände Vertex-Cover des Graphen.

4.2 Erdös-Rényi-Zufallsgraphen und einige ihrer Eigenschaften

In dieser Arbeit werden wir das VC-Problem für Instanzen aus einem Ensemble von Zufallsgraphen betrachten, das auf zwei Arbeiten von Paul Erdös und Albert Rényi zurückgeht [43, 44]. Die Graphen werden deshalb als Erdös-Rényi-Zufallsgraphen bezeichnet.

Das Ensemble $\mathcal{G}(N, p)$ mit $0 \leq p \leq 1$ enthält alle Graphen mit N Knoten. In einem Graphen $G \in \mathcal{G}(N, p)$ ist jede potentielle Kante mit Wahrscheinlichkeit p enthalten. Einen solchen Graphen kann man erzeugen, indem man jeder Kante, d. h. jedem ungeordneten Paar $e = \{v, w\} \in V \times V$, eine gleichverteilte Zufallszahl $q(e) \in [0,1]$ zuordnet. Als Menge der Kanten des Graphen E wählt man dann alle Kanten e mit $q(e) < p$.

In den extremalen Fällen $p = 0$ bzw. $p = 1$ ist der Graph völlig ohne Kanten bzw. vollständig vernetzt. Dazwischen variiert die Anzahl der Kanten M bei festem p. Im Mittel erwartet man eine Kantenanzahl

$$\overline{M} = p \binom{N}{2} = \frac{N(N-1)}{2} p. \tag{4.3}$$

Einige typische Eigenschaften des Ensembles $\mathcal{G}(N, p)$ kann man gut veranschaulichen, indem man die Grapherzeugung als Evolutionsprozess ablaufen lässt. Zunächst legt man für einen Graphen mit N Knoten $q(e)$ für alle ungeordneten Paare von Knoten fest. Man startet bei $p(t) = 0$ mit einem Graphen ohne Kanten und fügt zur Zeit

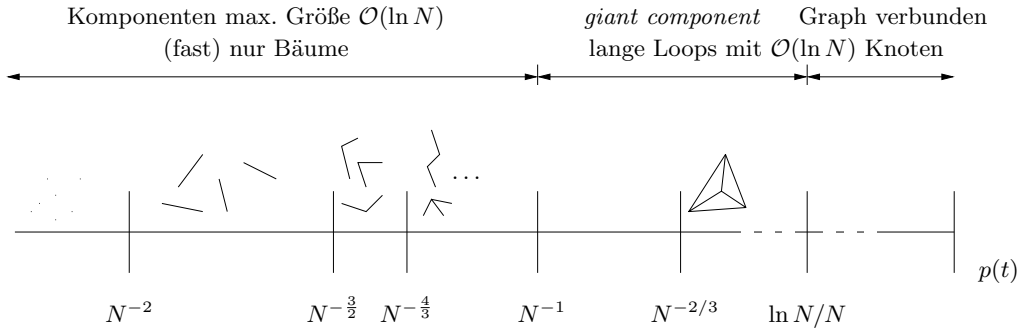


Abbildung 4.2: Evolution von Erdös-Rényi-Zufallsgraphen

t alle Kanten mit $q(e) = p(t)$ hinzu, d. h. zur Zeit t gehört der Graph zu $\mathcal{G}(N, p(t))$. Das typische Aussehen des Graphen für verschiedene Intervalle ist in Abbildung 4.2 dargestellt [45].

Bei $p(t) \sim 1/N^2$ treten die ersten unverbundenen Kanten auf. Allgemeiner findet man bei $p(t) \sim N^{-(k+1)/k}$ erstmals Bäume der Größe k. Für $p(t) < 1/N$ besteht der Graph aus kleinen, unverbunden Komponenten mit maximal $\mathcal{O}(\ln N)$ Knoten, die fast alle Bäume sind.

Die Struktur des Graphen ändert sich deutlich bei $p(t) \sim 1/N$, der *Perkolationsschwelle*. Viele dieser kleinen Komponenten wachsen zu einer großen Struktur, der *giant component*, zusammen. Diese enthält extensiv (d. h. $\mathcal{O}(N)$) viele Knoten, während alle übrigen Komponenten weiterhin maximal Größe $\mathcal{O}(\ln N)$ haben. Bei $p(t) \sim N^{-(k-1)/k}$ findet man erstmals *Cliquen* der Größe k, d. h. vollständig vernetzte Untergraphen mit k Knoten. Durch Hinzufügen weiterer Kanten verschwinden nach und nach die kleinen Komponenten, bis schließlich der Graph bei $p(t) \simeq \ln N/N$ nur noch aus der giant component besteht.

Wir werden in dieser Arbeit Zufallsgraphen betrachten, bei denen $M = \mathcal{O}(N)$ ist, d. h. die mittlere Konnektivität $c = 2M/N$ ist endlich. In der obigen Notation entspricht dies dem Ensemble $\mathcal{G}(N, c/N)$ mit endlichem c.

Die Wahrscheinlichkeit für eine Kante, im Graph enthalten zu sein, beträgt demzufolge c/N. Ein im Graph enthaltener Kreis der Länge k besteht aus k Knoten und k Kanten. Die Anzahl kleiner Kreise mit $k = \mathcal{O}(1)$ ist also gegeben durch

$$N(N-1)\cdots(N-k+1)\left[\frac{c}{N}\right]^k = N^k\left[\frac{c}{N}\right]^k + \mathcal{O}(1) = \mathcal{O}(1), \tag{4.4}$$

d. h. die Anzahl bleibt im thermodynamischen Limes endlich. Dies bedeutet, dass die Struktur eines Zufallsgraphen lokal baumartig ist. Allerdings sind Nachbarn eines Knotens v auch bei Entfernung von v aus dem Graphen nicht unabhängig, da große Kreise der Länge $\mathcal{O}(\ln N)$ existieren [45].

Eine weitere wesentliche Eigenschaft des Ensembles in diesem Bereich besteht darin, dass die Grade d der Knoten einer Poissonverteilung unterliegen, d. h. dass bei

Konnektivität c

$$p(d) = e^{-c}\frac{c^d}{d!} \tag{4.5}$$

gilt.

Alternativ zum Ensemble $\mathcal{G}(N, p)$ kann man das ebenfalls von Erdös und Rényi vorgeschlagene Ensemble $\mathcal{G}(M, N)$ untersuchen, bei dem die Anzahl der Knoten N und Kanten M fest vorgegeben ist. Asymptotisch unterscheiden sich die beiden Ensembles bei endlicher Konnektivität in den später betrachteten Größen nicht, so dass wir wegen der besseren analytischen Handhabbarkeit meist $\mathcal{G}(N, p)$ vorziehen werden.

4.3 Algorithmen zur Bestimmung minimaler Vertex-Cover

Das Vertex-Cover-Problem ist NP-vollständig, die Rechenzeit selbst der besten Algorithmen steigt im ungünstigsten Fall exponentiell mit der Systemgröße.

Um minimale Vertex-Cover eines gegebenen Graphen bestimmen zu können, werden wir deshalb auf heuristische Methoden (vgl. Abschnitt 2.2.2) zurückgreifen müssen. Mit diesen lassen sich auch für größere Systeme ($N \approx 1000$) Vertex-Cover konstruieren. Allerdings ist a priori nicht garantiert, dass es sich dabei tatsächlich um minimale VC handelt.

Für kleinere Systeme kann man VC minimaler Größe (d. h. Lösungen des Optimierungsproblems (O1)) exakt finden. In diesem Abschnitt werden die dafür in der Arbeit verwendeten Algorithmen vorgestellt.

4.3.1 Vollständige Enumeration durch Branch-and-Bound

Das naive vollständige Verfahren für ein derartiges Problem ist das in Abschnitt 2.2.1 dargestellte vollständige Durchsuchen des binären Baums aller möglichen 2^N Konfigurationen. Jeder der N Knoten nimmt dabei einen der beiden Werte

- *bedeckt* oder
- *unbedeckt*

an. In jeder Ebene des Suchbaums wird der Wert eines Knotens festgelegt, alle noch nicht festgelegten Knoten nennen wir *frei*.

Zur Beschleunigung des Algorithmus ist folgende Eigenschaft des VC-Problems von Vorteil: Alle Lösungen müssen die Bedingung erfüllen, dass sie Vertex-Cover sind, d. h. in keinem Falle dürfen die beiden Eckpunkte einer Kante unbedeckt sein. Wird

der Wert eines Knotens auf *unbedeckt* festgelegt, so müssen alle Nachbarknoten folglich den Wert *bedeckt* haben und können sofort auf diesen Wert gesetzt werden.

Eine weitere wesentliche Reduzierung des Suchbaums erhält man durch die Abschätzung der minimal erreichbaren Vertex-Cover-Größe für alle Blätter eines von einem Knoten ausgehenden Unterbaums. Die Grundidee besteht darin, dass durch Festlegen eines Knotens auf den Wert *bedeckt* alle von ihm ausgehenden Kanten zu *unbedeckten* oder *freien* Knoten bedeckt werden.

Diese Anzahl der von einem Knoten i ausgehenden noch nicht bedeckten Kanten wird mit d_i^f bezeichnet. Der Algorithmus führt eine Liste $\big(j(z)\big)$ der noch freien Knoten, die nach fallenden d_i^f geordnet ist.

Außerdem sei $X_{\text{opt}}^{\text{akt}}$ die Größe des bisher kleinsten gefundenen VC. Zu Beginn wird $X_{\text{opt}}^{\text{akt}} = N - 1$ gesetzt, da die Menge aller N Knoten bis auf einen beliebigen immer ein minimales VC darstellt.[1]

Sind nun aktuell X Knoten bedeckt, dann dürfen maximal noch

$$k = X_{\text{opt}}^{\text{akt}} - X - 1 \tag{4.6}$$

weitere noch freie Knoten bedeckt werden, um ein kleineres VC als das bisher bekannte Optimum zu finden. Möchte man dagegen die Menge *aller* minimalen VC bestimmen, dann dürfen maximal noch

$$k' = X_{\text{opt}}^{\text{akt}} - X \tag{4.7}$$

Knoten bedeckt werden, damit in einer möglichen weiteren Lösung höchstens ebenso viele Knoten bedeckt sind wie im aktuellen Optimum.

Durch das Bedecken von k (bzw. k') weiteren Knoten können aber maximal

$$D = \sum_{z=1}^{k} d_{j(z)}^f \tag{4.8}$$

Kanten bedeckt werden, wobei zur Summe die k oberen Elemente der Liste der freien Knoten beitragen. Sollte es Kanten zwischen zwei dieser k Knoten geben, dann ist die tatsächliche Anzahl bedeckter Kanten geringer.

Ist nun D kleiner als die Anzahl der noch nicht bedeckten Kanten U, so braucht die Suche im Unterbaum nicht weitergeführt zu werden, da dieser keine bessere als die bisher optimale Lösung enthalten kann.

Bei dieser Beschneidung des Suchbaums werden umso mehr Knoten des Suchbaums ausgelassen, je eher ein VC mit kleiner Größe gefunden wird, da dann die Bedingung

[1] Man kann den Algorithmus leicht modifizieren, um stattdessen das Entscheidungsproblem (E1) (vgl. Seite 32) zu lösen. Dazu setzt man zu Beginn $X_{\text{opt}}^{\text{akt}} = K + 1$. Findet der Algorithmus kein kleineres VC, so ist die Antwort „nein“, andernfalls „ja“.

$D < U$ häufiger bzw. näher an der Wurzel des Suchbaums wahr wird. Es ist deshalb entscheidend, in welcher Reihenfolge die Blätter durchlaufen werden, d. h. welcher Suchpfad bevorzugt durchlaufen wird.

In [46] werden verschiedene *Heuristiken* analytisch untersucht: Knoten mit großem Grad sind in einem minimalen VC wahrscheinlicher bedeckt, Knoten mit kleinem Grad unbedeckt. Eine mögliche Strategie ist deshalb, Knoten mit vielen freien Nachbarn bevorzugt zu bedecken. Alternativ wählt man einen Knoten mit besonders wenigen freien Nachbarn bevorzugt, setzt dessen Wert auf *unbedeckt* und somit den Wert aller Nachbarn auf *bedeckt*. Diese Vorgehensweise wird als *Generalized Leaf-Removal* bezeichnet.

Durch Analyse des Laufzeitverhaltens dieser Algorithmen erhält man Schranken für die typische Größe des minimalen VC. Ein wichtiger Spezialfall ist der *Leaf-Removal-Algorithmus*[47], welchen der nachfolgende Abschnitt beschreibt.

4.3.2 Leaf-Removal – für $c < e$ ein polynomiales Verfahren

Der Leaf-Removal-Algorithmus ist eine spezielle Variante des Algorithmus von Tarjan und Trojanowski [48]. Grundidee ist, Knoten vom Grad 1 („Blätter“ bzw. „leaves“) zu betrachten und deren einzigen benachbarten Knoten zu überdecken.

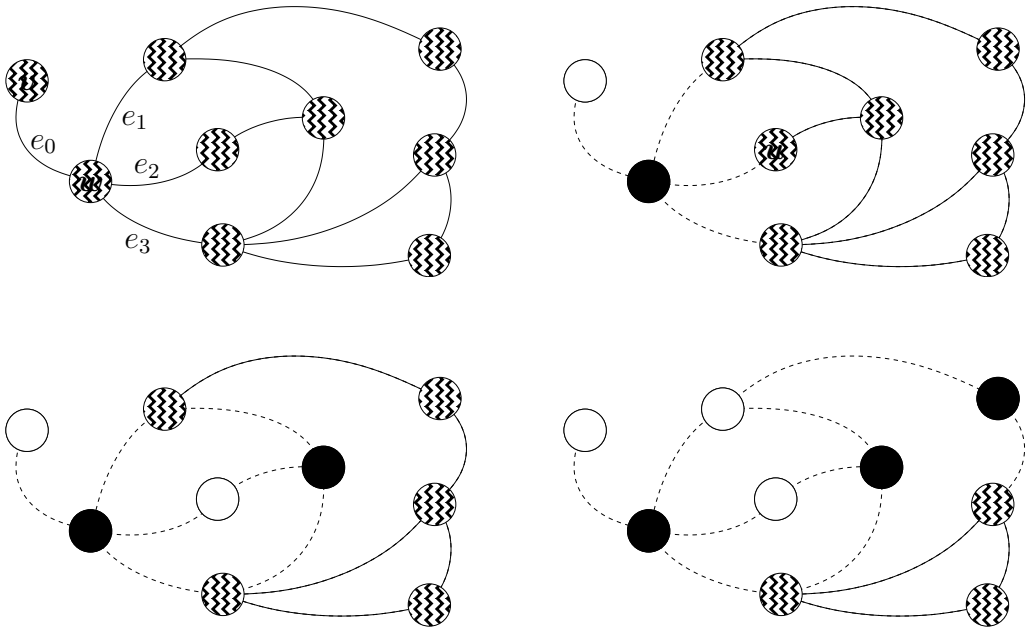


Abbildung 4.3: Leaf-Removal-Algorithmus, dargestellt in vier Schritten von oben links nach unten rechts. Dunkle Knoten sind bedeckt, weiße bleiben unbedeckt und schraffierte sind noch frei, d. h. noch nicht entschieden. Gestrichelte Kanten sind bereits bedeckt.

Betrachten wir den Graph in Abbildung 4.3 oben links. Der Knoten v hat Grad 1. Die einzige von ihm ausgehende Kante e_0 muss bedeckt werden, d. h. v oder w müssen in einem minimalen Vertex-Cover enthalten sein. Wird v bedeckt, so wird nur die Kante e_0 bedeckt. Wird w bedeckt, so werden zusätzlich noch die Kanten e_1, e_2 und e_3 bedeckt. Durch Bedeckung von w erhält man also ein mindestens ebenso kleines VC wie durch Bedeckung von v, deshalb kann Knoten w bedeckt werden.

Die Kanten $\{e_i\}_{0\leq i\leq 3}$ sind damit überdeckt und brauchen bei der weiteren Suche nicht mehr berücksichtigt zu werden. Durch die Entfernung des Blattes v hat nun u den Grad 1 und die Prozedur kann wiederholt werden, bis keine Blätter mehr vorhanden sind.

Der verbleibende Untergraph, in dem nach Konstruktion alle Knoten mindestens Grad 2 haben, wird *Kern* (*core*) genannt. Karp und Sipser [49] sowie Bauer und Golinelli [40] haben durch Analyse dieses Algorithmus gezeigt, dass die Größe des Kerns für Zufallsgraphen mit Konnektivität $c < e \approx 2.71$ im thermodynamischen Limes von der Ordnung $o(N)$ ist. Bei geeigneter Implementierung der Graphenstruktur benötigt Leaf-Removal für Zufallsgraphen mit Konnektivität $c < e$ deshalb nur lineare Rechenzeit, d. h. in diesem Bereich der Konnektivität ist VC typischerweise ein *einfaches Problem*. Für Graphen aller Konnektivitäten können wir Leaf-Removal verwenden, um die Enumeration bei der Bestimmung eines minimalen VC zu beschleunigen. Denn wenn ein Knoten mit einem freien Nachbarn existiert, kann dieser Knoten sofort auf „unbedeckt“ gesetzt werden. Der zu „bedeckt“ gehörende Unterbaum braucht nicht mehr berücksichgt zu werden, da dort keine bessere Lösung vorhanden sein kann.

4.3.3 Bestimmung des Backbones

Eine Instanz des VC-Problems hat im Allgemeinen mehr als ein minimales VC. Die Menge der Knoten V unterteilt sich dadurch disjunkt und vollständig in drei Teilmengen V_+, V_- und V_0, die folgendermaßen definiert sind:

- $V_+ := \{v \in V$ und v ist bedeckt in allen minimalen VC$\}$ (*bedeckter Backbone*).
- $V_- := \{v \in V$ und v ist unbedeckt in allen minimalen VC$\}$ (*unbedeckter Backbone*).
- $V_0 := V \setminus (V_+ \cup V_-)$

Mit Hilfe des Backbones lässt sich die Bestimmung der Menge aller minimalen VC deutlich beschleunigen. Dazu nehmen wir zunächst an, die Menge V_+ sei bereits bestimmt. Entfernt man alle in V_+ enthaltenen Knoten und die von ihnen ausgehenden Kanten aus dem Graph, so ändert sich die Anzahl der Grundzustände nicht. Denn durch Entfernen der Knoten wird einerseits keine weitere Kante unbedeckt, andererseits sind alle entfernten Kanten durch die Knoten in V_+ bedeckt.

Alle Knoten in V_- sind nach Entfernen von V_+ isolierte Knoten, d. h. sie haben Grad 0 und können deshalb ebenfalls aus dem Graphen entfernt werden, ohne die Anzahl der Grundzustände zu ändern.

Da der Anteil der zum Backbone gehörigen Knoten in der Regel groß ist [50], zerfällt der Graph durch seine Entfernung unter Umständen in mehrere Komponenten. Die Menge der Grundzustände kann dann für jede Komponente unabhängig bestimmt werden.

In Abbildung 4.4 sieht man, wie dadurch die Enumeration aller Grundzustände erheblich beschleunigt werden kann. Dargestellt ist der Median der Anzahl der Grundzustände der jeweils größten Komponente des Graphen vor und nach Entfernen des Backbones. Da die Anzahl der Grundzustände exponentiell mit der Größe des Graphen steigt, dominiert diese größte Komponente die Rechenzeit. Man sieht, dass für alle Konnektivitäten der Exponent des Anstiegs kleiner wird. Der Gewinn ist umso größer, je kleiner c ist. Dies liegt einerseits daran, dass die Backbonegröße selbst für kleine c größer ist. Andererseits ist der Graph weniger verbunden, d. h. es werden mehr Komponenten von der giant component abgetrennt. Die Abtrennung einer einzelnen kleinen Komponente aus zwei Knoten reduziert die Anzahl der Grundzustände dabei bereits um den Faktor zwei.

Die Bestimmung des bedeckten Backbones selbst ist möglich, ohne vorher alle Grundzustände zu kennen. Dazu bestimmt man zunächst *ein* minimales VC des Graphen. Kandidaten für V_+ sind alle in diesem VC bedeckten Knoten. Für jeden dieser $\mathcal{O}(N)$ Knoten prüft man nun, ob ein VC der gleichen minimalen Größe existiert, in dem dieser Knoten unbedeckt ist. Dazu setzt man den Wert des Knotens in der obersten Ebene des Entscheidungsbaums 2.1 auf „unbedeckt“ und durchsucht nur den dazu gehörenden Unterbaum. Der Knoten gehört genau dann zu V_+, wenn sich dort kein VC mit gleicher minimaler Größe befindet.

Die $\mathcal{O}(N)$-fache Bestimmung von minimalen VC ist exponentiell schneller als die Enumeration aller Grundzustände. Zum einen kann nur im ersten Fall Leaf-Removal verwendet werden. Zum anderen ist die Anzahl k der noch bedeckbaren Knoten im Branch-and-Bound-Algorithmus bei der Suche nach *einem* minimalen VC (Gleichung 4.6) um eins geringer als bei der Suche nach *allen* VC (Gleichung 4.7). Damit wird die Schranke (4.8) kleiner und es können deshalb größere Unterbäume des Entscheidungsbaums ausgeschlossen werden. Eine weitere Beschleunigung ergibt sich daraus, dass der Wert der Schranke zu Beginn nicht $N - 1$ ist, sondern sofort die Größe des bereits bestimmten minimalen VC verwendet werden kann.

Zusammenfassend ist das verwendete Verfahren zur Bestimmung aller minimalen VC eines Graphen in Algorithmus 4.1 dargestellt. Die minimalen VC des Gesamtgraphen sind alle Kombinationen der minimalen VC der einzelnen Komponenten.

Für Graphen bis zur maximalen Größe zwischen $N = 200$ (bei $c = 6$) und $N = 2000$ (bei $c = 3$) kann ein minimales VC bestimmt werden. Die Enumeration aller Grundzustände ist bis zu $N \approx 150$ Knoten möglich.

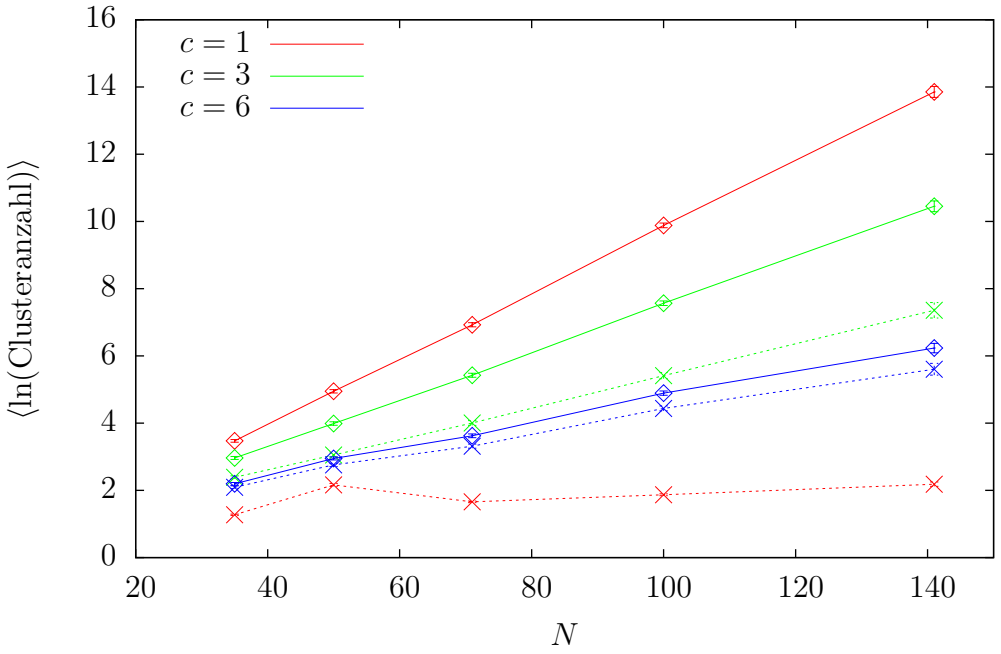


Abbildung 4.4: Reduzierung der Anzahl der Grundzustände der größten Komponente des Graphen. Die durchgezogenen Linien geben die Median-Werte vor der Entfernung der Backboneknoten an, die gestrichelten Linien diejenigen danach. Je kleiner c, desto stärker reduziert sich die Anzahl der Knoten in der größten Komponente und desto größer ist damit die Beschleunigung des Algorithmus.

algorithm enumerate_min_vc(G)
begin
    Teile den Graph in verbundene Komponenten
    for jede Komponente **do**
    begin
        Bestimme *ein* minimales VC mit *Leaf-Removal* und *Branch-and-Bound*
        Bestimme den bedeckten Backbone V_+
        Entferne V_+ und die zugehörigen Kanten sowie V_-
        Teile den Graph in verbundene Komponenten
        for jede Komponente **do**
            Bestimme *alle* minimalen VC der Komponente mit *Branch-and-Bound*
    end
end

Algorithmus 4.1: Bestimmung aller Grundzustände eines Graphen G

4.3.4 Der Parallel-Tempering-Algorithmus

Für größere Systeme werden wir Grundzustände mit *Parallel Tempering* (PT) bestimmen. Dieses Verfahren wurde bereits allgemein im Abschnitt 2.2.2 eingeführt (Algorithmus 2.2). Hier wird beschrieben, wie wir dieses Verfahren auf das VC-Problem anwenden.

Erlaubte Zustände einer gegebenen Instanz sollen alle – nicht notwendigerweise minimalen – VC sein. Die Rolle der Teilchenanzahl n übernimmt hier die Anzahl der unbedeckten (!) Knoten, da diese Anzahl für minimale VC maximal wird. Das Gewicht $p(V')$ eines Vertex-Covers V' ist somit durch $\exp\{\mu(N - |V'|)\}$ gegeben. Da aber nur relative Gewichte relevant sind, definieren wir

$$p(V') = \exp\{-\mu|V'|\}. \tag{4.9}$$

Das PT-Verfahren simuliert gleichzeitig mehrere Kopien des Systems bei unterschiedlichem chemischem Potential μ. Für jede Kopie werden unabhängig N MC-Schritte ausgeführt, die in Algorithmus 4.2 beschrieben sind. Dabei wird in jedem Schritt ein Knoten x_i zufällig ausgewählt und mit gleicher Wahrscheinlichkeit eine der beiden folgenden Operationen ausgeführt [50]:

(M) Wenn der Knoten x_i bedeckt ist und genau einer seiner Nachbarknoten unbedeckt ist, dann wird dieser stattdessen bedeckt. In diesem Fall wird die Eigenschaft „bedeckt“ also entlang einer Kante des Graphen verschoben (*„Move“*).

(E) Ist x_i unbedeckt, dann wird der Knoten mit Wahrscheinlichkeit $\exp\{-\mu \cdot 1\}$ bedeckt, die Größe des VC erhöht sich um eins. Sind x_i und alle seine Nachbarn bedeckt, dann wird x_i unbedeckt, die Größe des VC erniedrigt sich um eins (*„Exchange with reservoir“*). Diese Operation entspricht dem MC-Schritt, für den in Abschnitt 2.2.2 detaillierte Balance gezeigt wurde.

Die PT-Simulation wird hier für jede Instanz mit $m = 26$ Kopien des Systems mit chemischem Potential von $\mu_1 = 1$ bis $\mu_{26} = 12$ durchgeführt. In einer Zeit von $N_{\mathrm{MC}} = 10^6$ MC-Sweeps werden jeweils 500 Zustände bei $\mu = 9$ gespeichert. Wie man in Abbildung 4.5 sieht, sind bei diesem Wert des chemischem Potentials die meisten Zustände Grundzustände. Für die Systemgrößen, bei denen mit vollständigen Algorithmen ein minimales VC bestimmt werden kann (vgl. Abschnitt 4.3.3), wurde zusätzlich nachgeprüft, dass die von PT gefundenen Zustände tatsächlich minimale VC sind.

algorithm Monte-Carlo-Schritt(μ, $\vec{x} \equiv (x_1, \ldots, x_N)$)
begin
  do N-mal
  begin
    wähle x_i zufällig
    mit Wahrscheinlichkeit 1/2 führe (M) oder (E) aus
      (M) **if** x_i ist bedeckt und genau ein Nachbar x_j ist unbedeckt
          then
            setze x_i unbedeckt, setze x_j bedeckt
      (E) **if** x_i ist unbedeckt **then**
            bedecke x_i mit Wahrscheinlichkeit $e^{-\mu}$
          else
            if alle Nachbarn von x_i sind bedeckt **then**
              Setze x_i unbedeckt
  end
end

Algorithmus 4.2: Monte-Carlo-Schritt bei chemischem Potential μ. Ein Zustand $\vec{x}$ ist gegeben durch die Werte der N Knoten $x_i \in \{\text{bedeckt, unbedeckt}\}$.

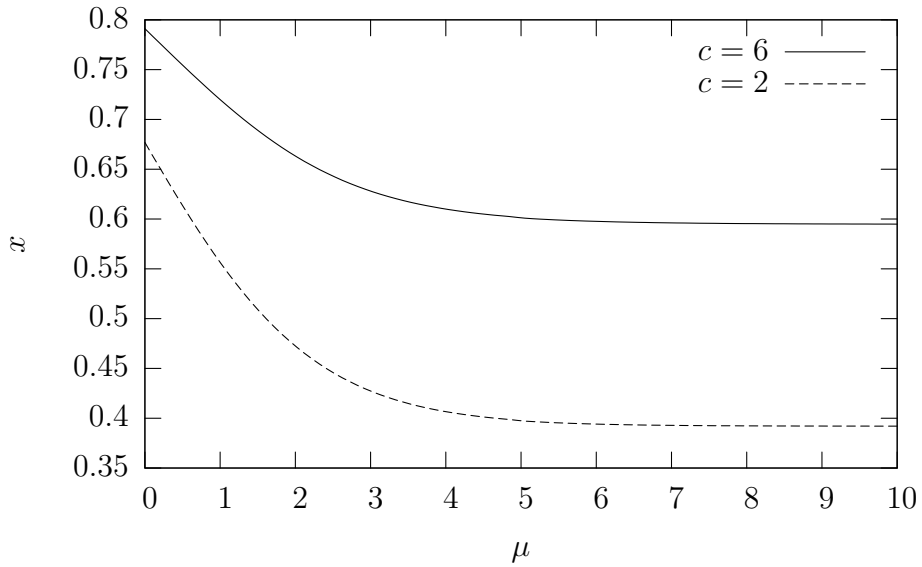


Abbildung 4.5: Mittlerer Anteil x der im minimalen VC bedeckten Knoten abhängig vom Wert des chemischen Potentials für Graphen der Größe $N = 64\,000$

4.4 Statistische Mechanik des VC-Problems

4.4.1 Der COV-UNCOV-Phasenübergang

Numerische Beobachtungen

Bestimmt man für Graphen aus dem in Abschnitt 4.2 vorgestellten Ensemble $\mathcal{G}(N, c/N)$ numerisch Vertex-Cover der Größe xN, so ist klar, dass für $x = 0$ die Wahrscheinlichkeit $p_{\mathrm{cov}}(x, c, N)$ einen Graphen zu überdecken verschwindet. Umgekehrt ist $p_{\mathrm{cov}}(1, c, N) = 1$, denn die Auswahl aller Knoten ist immer ein VC.

Ein insbesondere auch aus physikalischer Sicht sehr interessantes Phänomen findet man bei Betrachtung des Verlaufs von $p_{\mathrm{cov}}(x, c, N)$ in Abhängigkeit von x zwischen diesen beiden Extremen (beispielhaft für $c = 2$ in Abbildung 4.6 dargestellt) [39]: Mit steigender Systemgröße N strebt die Kurve asymptotisch gegen eine Stufenfunktion. Demzufolge existiert ein kritischer Wert $x_c(c)$, so dass für $p_{\mathrm{cov}}(x, c) = \lim_{N\to\infty} p_{\mathrm{cov}}(x, c, N)$ gilt:

$$p_{\mathrm{cov}}(x, c) = \begin{cases} 0 & \text{für } x < x_c(c) \\ 1 & \text{für } x > x_c(c) \end{cases} \tag{4.10}$$

Man findet also bei Veränderung des Parameters x am Punkt $x_c(c)$ einen Übergang von einer Phase, in der fast kein Graph ein VC der Größe xN besitzt (*UNCOV-Phase*), zu einer Phase, wo dies für fast alle Graphen der Fall ist (*COV-Phase*). Wegen der Ähnlichkeit mit physikalischen Phasenübergängen wird dieses Verhalten als COV-UNCOV-Phasenübergang bezeichnet. Insbesondere hängt die Größe des minimalen VC für fast alle Graphen nicht mehr vom konkreten Graph selbst, sondern nur von dessen Konnektivität ab.

Aus algorithmischer Sicht bemerkenswert ist, dass die Rechenzeit in der Umgebung des Phasenübergangs einen Peak hat: Denn für $x \gg x_c$ wird schon eine sehr einfache Heuristik, die ohne Backtracking (vgl. Abschnitt 2.2.1) auskommt, ein VC finden. Die benötigte Rechenzeit steigt dann nur linear mit der Systemgröße. Löst man andererseits das Entscheidungsproblem in der UNCOV-Phase beispielsweise mit einem Algorithmus nach Abschnitt 4.3.1, so wird der durch xN gegebene Startwert der Schranke $X_{\mathrm{min}}^{\mathrm{akt}}$ mit kleinerem x geringer, damit auch die Höhe des Suchbaums. Die Rechenzeit steigt dann zwar exponentiell, der Exponent nimmt aber mit wachsendem Abstand zu x_c ab.

Mittelt man die in Abschnitt 4.1 eingeführte Energie $E_0(G, x)$ für festes x über das Ensemble $\mathcal{G}(N, c/N)$ (Notation: $\overline{\cdot}$), so erhält man den ebenfalls in Abbildung 4.6 dargestellten Verlauf der mittleren Energiedichte [51]

$$e(x, c, N) = \frac{1}{N}\overline{E_0(G, x)}. \tag{4.11}$$

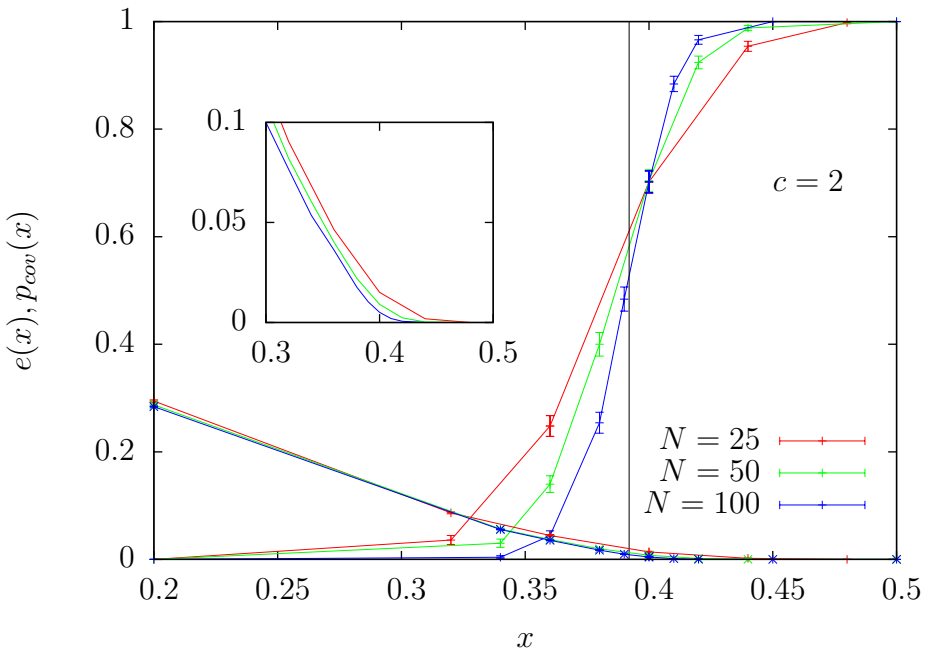


Abbildung 4.6: Wahrscheinlichkeit, dass ein Graph mit N Knoten und Konnektivität 2 ein Vertex-Cover der Größe xN besitzt. Mit wachsender Systemgröße nähern sich die numerischen Kurven einer Stufenfunktion mit Sprung bei $x_c(2) \approx 0.392$ (senkrechte Linie). Außerdem dargestellt ist die mittlere Energiedichte $e(x, 2, N)$ [51].

Es ist $e(0, c, N) = 2c$, mit zunehmendem x fällt die Energiedichte, bis sie bei einem x oberhalb von x_c fast verschwindet. Mit steigender Graphengröße tendiert x in Richtung x_c, der Verlauf entspricht dem Verhalten bei einem Phasenübergang zweiter Ordnung.

Finite-Size-Verhalten und Korrelationsvolumen

Zur näheren Beschreibung des Phasenübergangs wurde im Rahmen dieser Arbeit das kritische Verhalten untersucht.

In Abbildung 4.7 ist das Skalierungsverhalten der mittleren Größe eines minimalen VC $x_c(N)$ in Abhängigkeit von der Systemgröße dargestellt. Dazu wurden minimale VC für je 5000 Graphen für jede Systemgröße und Konnektivität bestimmt. Versucht man, die Abhängigkeit durch eine Gleichung der Form $x_c(N) = x_c(\infty) + aN^{-b}$ zu beschreiben, so gelingt die Anpassung oberhalb der Graph-Perkolationsschwelle, d. h. für $c > 1$ (Geraden in Abb. 4.7), während es am Perkolationsübergang des Graphen bei $c = 1$ signifikante Abweichungen gibt. Die numerisch bestimmten Konstanten sind in Tabelle 4.1 angegeben.

c	$x_c(\infty)$	a	b
2	0.3919 (1)	0.18 (7)	0.91 (9)
3	0.4664 (1)	0.26 (4)	0.88 (4)
4	0.5193 (2)	0.27 (4)	0.82 (4)
6	0.5942 (9)	0.52 (2)	0.92 (11)

Tabelle 4.1: Numerische Werte für die Konstanten des angenommenen Skalengesetzes $x_c(N) = x_c(\infty) + aN^{-b}$ der Geraden in Abbildung 4.7 in Abhängigkeit von der Konnektivität

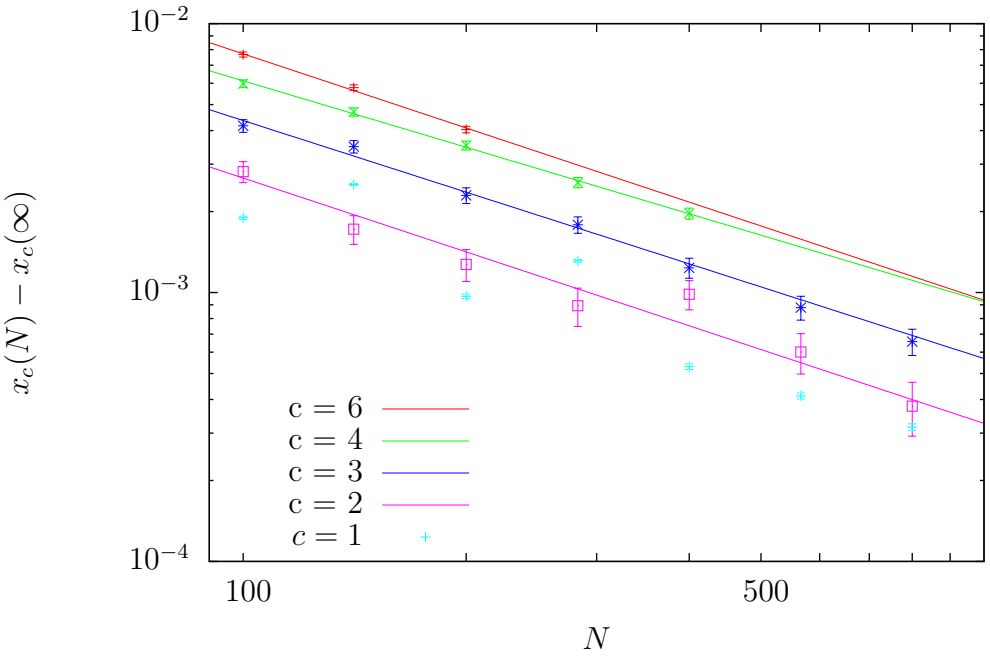


Abbildung 4.7: Skalierungsverhalten der mittleren minimalen VC-Größe. Die Geraden für $c > 1$ sind die besten Anpassungen an ein Skalengesetz der Form $x_c(\infty) + aN^{-b}$.

Ähnliches *Finite-Size*-Skalierungsverhalten kann man auch bei anderen Phasenübergängen 2. Ordnung beobachten [52]. Dabei hängt der Exponent $b = 1/\nu$ mit dem kritischen Exponenten ν der Divergenz der Korrelationslänge zusammen.

Beim VC finden wir für alle Konnektivitäten $c > 1$ im Rahmen der Fehlerbalken denselben Wert für b, der Exponent scheint also universell zu sein. Da der dem VC-Problem zugrunde liegende Graph als mean-field-artiges System keine Längenskalen besitzt, betrachten wir hier statt einer Korrelationslänge ein *Korrelationsvolumen*: Im Allgemeinen ist die Menge der Grundzustände von Knotenuntermengen gegebener Größe xN und minimaler Energiedichte entartet. Für Knoten v, welche nicht zum Backbone (vgl. Abschnitt 4.3.3) gehören, gibt es also sowohl Grundzustände mit v bedeckt und v unbedeckt. Die Anzahl der Knoten, in denen sich zwei Grundzustände unterscheiden, bezeichnen wir als deren (Hamming-)Abstand. Als *Korrelationsvolumen* definieren wir den minimalen Abstand zwischen zwei Grundzuständen mit v bedeckt und v unbedeckt, gemittelt über alle Knoten v, die nicht zum Backbone gehören.

Grundzustände mit v bedeckt unterscheiden sich in mindestens einem weiteren Knoten von Grundzuständen mit v unbedeckt, da die Anzahl der bedeckten Knoten konstant ist, d. h. dieser minimale Abstand 2 zwischen zwei Grundzuständen ist eine untere Schranke für das Korrelationsvolumen. In Abbildung 4.8 ist das Korrelationsvolumen für $c = 2$ gemittelt über 100 Graphen als Funktion von x für verschiedene Systemgrößen dargestellt. Weit weg von x_c ist das Korrelationsvolumen unabhängig von N. Knapp unterhalb von x_c beobachtet man einen Peak, welcher mit wachsender Systemgröße deutlicher ausgeprägt ist; ähnlich zur Divergenz der Korrelationslänge in anderen Phasenübergängen zweiter Ordnung.

Die numerische Bestimmung des Korrelationsvolumens erfordert die Berechnung vieler Grundzustände, beim hier verwendeten Verfahren die Berechnung aller Grundzustände. Deshalb reichen die erreichbaren Systemgrößen nicht aus, um über die qualitativen Aussagen hinaus auch einen Zusammenhang zum kritischen Exponenten b herzustellen.

Analytische Schranken für $x_c(c)$

In Abbildung 4.9 sind die numerischen Werte für $x_c(c)$ aus Simulationen mit $N = 250$ ($c \leq 4$) bzw. $N = 100$ ($c > 4$) dargestellt (Kreissymbole). Eine untere Schranke erhält man durch Abzählung der Vertex-Cover mit xN Knoten, die ein Graph im Ensemble $\mathcal{G}(N, c/N)$ im Mittel besitzt. Dabei findet man eine Schranke $x_{\mathrm{an}}(c) < x_c(c)$, gegeben durch

$$0 = x_{\mathrm{an}}(c) \ln x_{\mathrm{an}}(c) - (1 - x_{\mathrm{an}}(c)) \ln(1 - x_{\mathrm{an}}(c)) + \frac{c}{2} \ln \{x_{\mathrm{an}}(c)[2 - x_{\mathrm{an}}(c)]\}\,, \quad (4.12)$$

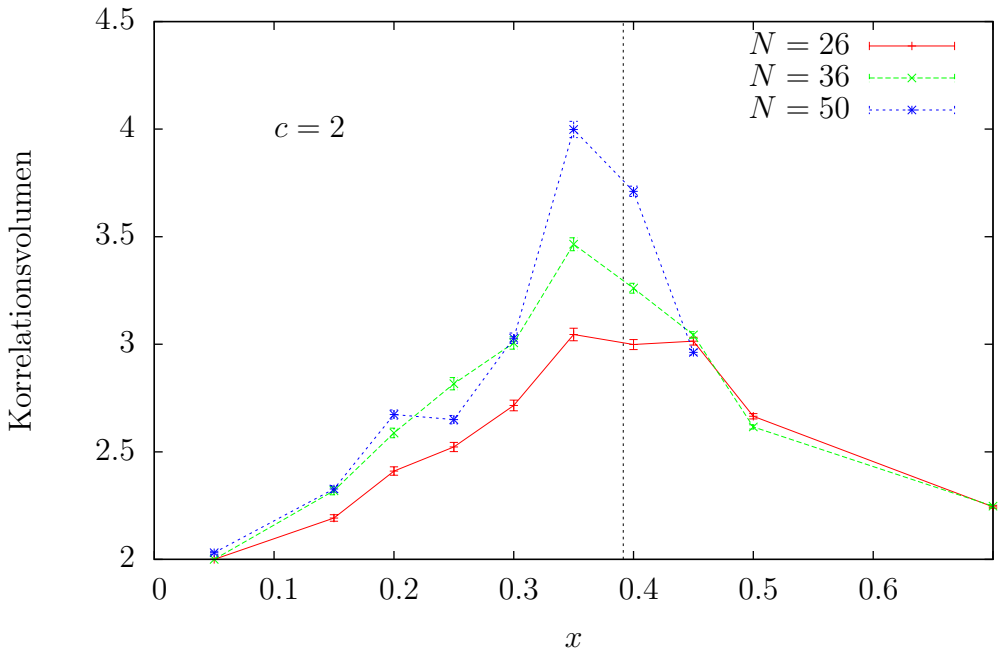


Abbildung 4.8: Das Korrelationsvolumen bei Konnektivität $c = 2$ für verschiedene Systemgrößen abhängig vom Anteil x bedeckter Knoten. Man findet einen Peak knapp unterhalb von x_c (gestrichelte senkrechte Linie).

unterhalb derer diese Anzahl im thermodynamischen Limes exponentiell verschwindet [53]. Für endliche c lässt sich diese Gleichung nur numerisch lösen (untere gestrichelte Linie in Abb. 4.9). Das asymptotische Verhalten für große c ist analytisch gegeben durch [54]

$$x_c(c) = 1 - \frac{2}{c}(\ln c - \ln\ln c - \ln 2 + 1) + o(c^{-1}). \tag{4.13}$$

Eine obere Schranke (obere gestrichelte Linie) [53]

$$x_c(c) < 1 - \ln(c)/c \tag{4.14}$$

erhält man durch Analyse eines naiven Algorithmus: Solange noch Knoten vorhanden sind, wird in jedem Schritt zufällig ein Knoten ausgewählt, seine Nachbarn werden bedeckt und alle von diesen Knoten ausgehenden Kanten entfernt.

In den folgenden Abschnitten werden die Methoden der statistischen Physik beschrieben, mit denen man $x_c(c)$ für $c < e$ analytisch bestimmen kann und als Ergebnis die durchgezogene Linie in Abb. 4.9 erhält.

4.4.2 Repräsentation als Harte-Kugeln-Modell

Zur analytischen Betrachtung des VC-Problems wird dieses auf ein äquivalentes physikalisches Modell abgebildet, und zwar auf das *Harte-Kugeln-Modell*. Dazu de-

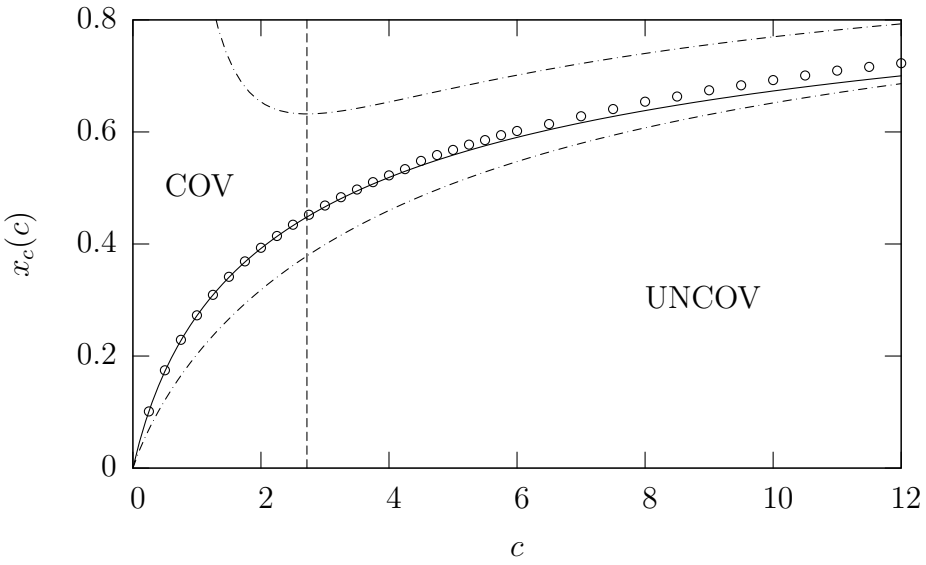


Abbildung 4.9: Phasendiagramm des VC-Problems [39]: Fast alle Graphen mit Konnektivität c besitzen im thermodynamischen Limes ein Vertex-Cover der Größe $x_c(c)N$, während für $x < x_c(c)$ fast kein Graph mit xN bedeckten Knoten vollständig überdeckt werden kann. Die durchgezogene Linie ist die Replika-symmetrische Voraussage für $x_c(c)$, die Strich-Punkt-Linien rigorose obere und untere Schranken von Gazmuri. Die Symbole sind Resultate numerischer Simulationen mit $N = 100$ bzw. $N = 250$. Für $c > e$ (Bereich rechts der senkrechten gestrichelten Linie $c = e$) liegen diese oberhalb der RS-Lösung.

finiert man als Abstand zweier Knoten die Anzahl des kürzesten sie verbindenden Pfades. Zwei Knoten haben somit genau dann Abstand 1, wenn sie durch eine Kante verbunden sind.

Jeder Knoten des Graphen darf von einer Kugel mit Radius 1 besetzt werden, allerdings dürfen sich diese Kugeln nicht überlappen. Eine Untermenge $S \in V$ von mit Kugeln besetzten Knoten ist somit genau dann eine mögliche Konfiguration, wenn S *höchstens* einen Endpunkt jeder Kante enthält.

Das Problem des minimalen VC ist dann äquivalent zur Maximierung der Anzahl platzierter Kugeln: Ist $V' \subset V$ ein VC, so enthält es von jeder Kante mindestens einen Endpunkt. Das Komplement $S' = V \setminus V'$ ist somit mögliche Konfiguration des Systems harter Kugeln. Ist V' minimales VC, dann enthält S' die maximale Kugelanzahl. Denn gäbe es eine Konfiguration S'' mit $|S''| > |S'|$, dann wäre $V'' = V \setminus S''$ ein VC mit $|V''| < |V'|$ im Widerspruch zur Mimimalität von V'. Analoges gilt umgekehrt.

Mit Hilfe dieses Systems kann das VC-Problem in der Sprache der statistischen Mechanik formuliert werden. Dazu werden für eine Untermenge $V' \subset V$ die binären

Variablen

$$\nu_i = \begin{cases} 0 & \text{wenn } i \in V' \\ 1 & \text{wenn } i \notin V' \end{cases} \tag{4.15}$$

definiert. ν_i entspricht gerade der Besetzungszahl des Knotens i und ist deshalb 1 in der zu V' komplementären Menge, d. h. für die mit Kugeln besetzten Knoten. Eine Konfiguration ist genau dann zulässig, wenn von keiner Kante beide Endpunkte mit je einer Kugel besetzt sind (d. h. $\nu_i = 1$), also genau dann, wenn

$$1 = \chi(\nu_1, \ldots \nu_N) \equiv \chi(\vec{\nu}) := \prod_{(i,j)\in E} (1 - \nu_i\nu_j) = \prod_{i<j} (1 - J_{ij}\nu_i\nu_j) \tag{4.16}$$

gilt, wobei (J_{ij}) die Adjazenzmatrix des Graphen G ist, d. h.

$$J_{ij} = \begin{cases} 1 & \text{wenn } (i,j) \in E \\ 0 & \text{sonst.} \end{cases} \tag{4.17}$$

Die großkanonische Zustandssumme des Harte-Kugeln-Modells lässt sich somit schreiben als

$$\Xi = \sum_{\{\nu_i=0,1\}} \exp\left(\mu \sum_{i=1}^{N} \nu_i\right) \chi(\vec{\nu}), \tag{4.18}$$

wobei das *chemische Potential* μ die Teilchenanzahl (und damit die Größe des komplementären VC) kontrolliert. Im Limes $\mu \to \infty$ tragen nur die Zustände mit maximaler Teilchenanzahl bei. Jeder dieser Zustände korrespondiert zu einem minimalen VC.

Der großkanonische Erwartungswert einer Observable $f(\vec{\nu})$ für einen gegebenen Graphen (also fest vorgegebene J_{ij}) ist definiert als

$$\langle f(\vec{\nu})\rangle_\mu := \Xi^{-1} \sum_{\{\nu_i=0,1\}} \exp\left(\mu \sum_{i=1}^{N} \nu_i\right) \chi(\vec{\nu}) f(\vec{\nu}). \tag{4.19}$$

Uns wird insbesondere die mittlere Teilchendichte

$$\rho(\vec{\nu}) := \frac{1}{N} \left\langle \sum_{i=1}^{N} \nu_i \right\rangle_\mu = \frac{\partial}{\partial \mu} \frac{\ln \Xi}{N} \tag{4.20}$$

interessieren, ihr Komplement entspricht dem Anteil der Knoten in minimalen VC des gegebenen Graphen.

4.4.3 Replika-symmetrische Lösung

Zur Berechnung der typischen Größe $x_c(c)$ minimaler VC von Graphen mit Konnektivität c muss (4.20) über das Ensemble $\mathcal{G}(N, c/N)$ gemittelt werden. Dieses *Unordnungsmittel* bezeichnen wir wie in Abschnitt 4.4.1 mit $\overline{\cdot}$. $x_c(c)$ ergibt sich dann als Komplement der Teilchendichte durch Grenzwertbildung

$$x_c(c) = 1 - \lim_{N\to\infty} \lim_{\mu\to\infty} \overline{\rho(\vec{\nu})} = 1 - \lim_{N\to\infty} \lim_{\mu\to\infty} \frac{\partial}{\partial\mu} \frac{\overline{\ln \Xi}}{N}. \tag{4.21}$$

Die Schwierigkeit bei der analytischen Lösung dieser Gleichung besteht in der Berechnung des Unordnungsmittels $\overline{\ln \Xi}$. Einfacher wäre die Berechnung von $\ln \overline{\Xi}$, dies würde aber ein thermodynamisches System beschreiben, bei dem sowohl die Besetzungszahlen ν_i als auch die den Graphen beschreibenden Wechselwirkungen J_{ij} gleichberechtigte thermodynamische Variablen sind (*annealed average*). Allgemein gilt wegen der Konkavität des Logarithmus $\ln \overline{\Xi} \leq \overline{\ln \Xi}$. Als Resultat ergäbe sich hier die in Gleichung (4.12) angegebene untere Schranke $x_{\mathrm{an}}(c)$.

Das korrekte Unordnungsmittel kann durch Verwendung des *Replika-Tricks* [1] mit Hilfe der Identität

$$\overline{\ln \Xi} = \lim_{n\to 0} \frac{\overline{\Xi^n} - 1}{n} \tag{4.22}$$

berechnet werden, bei der man die Potenz auf der rechten Seite der Gleichung zunächst für ganzzahlige Werte auswertet und als Produkt von n identischen Systemen (*Replikas*) auffasst. Anschließend wird durch analytische Fortsetzung der Grenzwert $n \to 0$ gebildet.

Die Replikamethode führt nicht zu streng mathematischen Resultaten, ist aber im Bereich ungeordneter Systeme ein vielfach erfolgreich angewendetes Werkzeug, beispielsweise bei der Lösung der in Kapitel 3 betrachteten Ising-Spingläser im unendlich-dimensionalen Fall (SK-Modell) durch den Parisi-Ansatz [16].

Die Besetzungszahl von Knoten i in Replika $a \in \{1, \ldots, n\}$ bezeichnen wir mit ν_i^a. Man führt den Ordnungsparameter

$$c(\vec{\xi}) = \frac{1}{N} \sum_i \prod_a \delta_{\xi^a, \nu_i^a} \tag{4.23}$$

ein, welcher beschreibt, wie groß der Anteil der Knoten mit replizierter Besetzungszahl $\vec{\xi} \in \{0,1\}^n$ ist.

Um Gleichung (4.21) zu lösen, nimmt man *Replikasymmetrie* an, d. h. der Ordnungsparameter $c(\vec{\xi})$ bleibt bei einer Permutation der Replikas unverändert. Da die Einträge von $\vec{\xi}$ nur 0 oder 1 sind, kann $c(\vec{\xi})$ deshalb nur von der Anzahl der „0“- bzw. „1“-Einträge von ξ abhängen, d. h. $c(\vec{\xi}) = c\left(\sum_{\alpha=1}^n \xi^{(\alpha)}\right)$. Damit verschwindet die Vektorabhängigkeit und man kann den Limes $n \to 0$ ausführen.

Man erhält [50]

$$x_c(c) = 1 - \frac{2W(c) + (W(c))^2}{2c}, \tag{4.24}$$

wobei $W(c)$ die Lambert-W-Funktion ist, gegeben durch

$$W(x) = xe^{-W(x)} \tag{4.25}$$

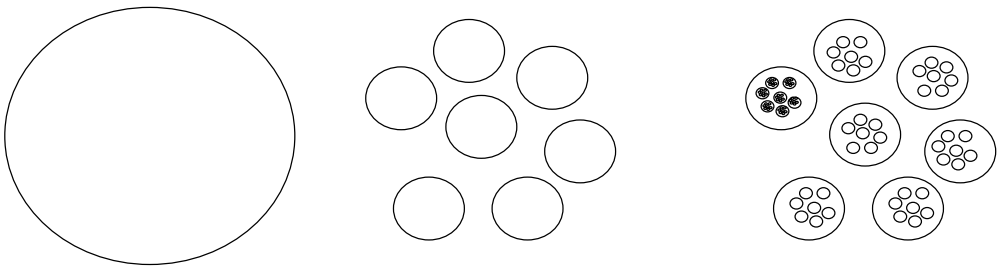
4.4.4 Gültigkeit der RS-Lösung und Zusammenhang zur Grundzustandsstruktur

Die analytische Lösung (4.24) für $x_c(c)$ ist als durchgezogene Linie in Abb. 4.9 dargestellt. Die numerischen Werte stimmen für $c < e \approx 2.718\ldots$ überein. Dies ist der Bereich, in dem der in Abschnitt 4.3.2 beschriebene Leaf-Removal-Algorithmus ein minimales Vertex-Cover in linearer Zeit findet. Wie in diesem Abschnitt beschrieben, lässt sich durch Analyse dieses Algorithmus mittels Ratengleichung rigoros zeigen, dass die analytische Lösung (4.24) für $c < e$ korrekt ist [40, 49]. Der Beweis lässt sich nicht auf $c > e$ übertragen, da dort der Leaf-Removal Algorithmus kein minimales VC für fast den gesamten Graphen findet, sondern ein Kern aus $\mathcal{O}(N)$-Knoten zurückbleibt.

Für diese Konnektivitäten $c > e$ stimmen auch die numerisch gefundenen Werte in Abb. 4.9 nicht mit der Replika-symmetrischen (RS) Lösung überein. Für sehr große Werte von c verletzt die Replika-symmetrische (RS) Lösung sogar die bewiesenen unteren Schranken (4.12) und (4.13) von Gazmuri bzw. Frieze.

Dies deutet darauf hin, dass die RS-Lösung für Werte $c > e$ nicht die korrekte ist, die Annahme von Replika-Symmetrie also nicht überall zutrifft. Dieses Phänomen findet man auch in vergleichbaren Systemen, beispielsweise beim Erfüllbarkeitsproblem für binäre Variablen, das in Abschnitt 5 betrachtet wird. Damit verbunden ist in diesen Systemen eine Veränderung der Grundzustandsstruktur. Im Bereich der Gültigkeit der Replika-symmetrischen Lösung, der RS-Phase, sind alle Grundzustände in einem Cluster organisiert. Dies bedeutet, dass man im Phasenraum $\{0,1\}^N$ zwischen zwei beliebigen Grundzuständen einen Pfad finden kann, dessen aufeinander folgende Elemente sich nur in $\mathcal{O}(1)$ vielen Bits unterscheiden. Für einen kritischen Wert des Kontrollparameters α (dem hier die Konnektivität c entspricht), tritt *spontane Replika-Symmetriebrechung* auf [55, 56]: Der Zustandsraum zerfällt in eine exponentielle Anzahl von Clustern (siehe Abb. 4.10). Zustände innerhalb eines Clusters können wieder durch Pfade verbunden werden, Zustände aus verschiedenen Clustern unterscheiden sich dagegen in $\mathcal{O}(N)$ Bits.

Darüber hinaus werden dort auch die Energiebarrieren zwischen den einzelnen Clustern extensiv, d. h. das System ist nicht mehr *ergodisch*. Unterschiedliche Bereiche, die wir auch als *reine Zustände* bezeichnen werden, können sich auch in ihren makroskopischen Eigenschaften unterscheiden. Die Replika-symmetrische Annahme der



Replika-Symmetrie (RS) | 1-Schritt-Replika Symmetriebrechung (1-RSB) | kontinuierliche Replika-Symmetriebrechung (∞-RSB)

Abbildung 4.10: Verschiedene mögliche Szenarios für die Clusterung der Grundzustandslandschaft.

Invarianz unter Permutation der Repliken ist verletzt, da diese zu verschiedenen reinen Zuständen gehören können.

Im Fall des VC-Problems kann man einen einfachen Ansatz für dieses *Ein-Schritt-Brechung* (*1-RSB*) genannte Verhalten lösen [50, 57]. Für $c < e$ ergibt dieser Ansatz wieder die RS-Lösung, was die Annahme von Replika-Symmetrie für diesen Bereich bestätigt. Bei größeren Werten von c gibt sie nur obere und untere Schranken für x_c. Die obere Schranke entspricht der in Gleichung (4.14) angegebenen Schranke von Gazmuri, die untere Schranke ist etwas oberhalb der RS-Lösung, liegt aber noch immer deutlich unter den numerischen Werten.

Beim Erfüllbarkeitsproblem findet man eine weitere Replika-symmetriegebrochene Phase: Zusätzlich zu den Zuständen sind auch die Cluster selbst wieder in Familien organisiert, diese wiederum in Familien etc., mit einer charakteristischen Verteilung der Abstände zwischen Clustern auf verschiedenen Ebenen der hierarchischen Struktur. Diese *kontinuierliche Replika-Symmetriebrechung* wurde zuerst bei der Lösung des Mean-Field-Spinglasmodells gefunden [16]. Ein ähnliches Szenario wäre auch hier für das VC-Problem für $c > e$ denkbar. Analytisch ist es bisher nicht gelungen, einen entsprechenden Ansatz zu übertragen.

In den folgenden Abschnitten soll deshalb numerisch die Grundzustandslandschaft des Vertex-Cover-Problems eingehend betrachtet werden und dabei insbesondere untersucht werden, ob man Merkmale der verschiedenen in diesem Abschnitt vorgestellten Szenarios finden kann.

4.5 Numerische Analyse der Grundzustandsstruktur

4.5.1 Clusterung mittels Hamming-Abstand

Die einfachste Definition eines Clusters beruht auf dem *Hammingabstand* zwischen zwei Zuständen, der durch die Anzahl der sie unterscheidenden Knoten gegeben ist:

$$d_{\text{ham}}(\vec{x}^{(\alpha)}, \vec{x}^{(\beta)}) \equiv d_{\alpha\beta} = \sum_{i=1}^{N} |x_i^{(\alpha)} - x_i^{(\beta)}|. \tag{4.26}$$

Grundzustände haben die gleiche Anzahl bedeckter und unbedeckter Knoten, damit ist der minimale Hammingabstand zwischen zwei verschiedenen Grundzuständen 2. Wir nennen deshalb zwei Grundzustände *benachbart*, wenn sie diesen minimalen Abstand haben. Zwei benachbarte Zustände sollen zum selben Cluster gehören. Durch Transitivität gehören so zwei Zustände $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ zum selben Cluster, wenn es eine Folge $M = (\vec{x}_0, \ldots, \vec{x}_n)$ von Zuständen mit den beiden folgenden Eigenschaften gibt:

- $\vec{x}_0 = \vec{x}^{(\alpha)}, \vec{x}_n = \vec{x}^{(\beta)}$
- $d_{\text{ham}}(\vec{x}_i, \vec{x}_{i+1}) = 2 \quad \forall i : 0 < i < n$

Ähnliche Clusterdefinitionen wurden beispielsweise für zufälliges p-XOR-SAT [58] oder endlich-dimensionale Spingläser verwendet [59].

Die Wahl der Konstanten 2 in der Definition ist zunächst willkürlich. Im thermodynamischen Limes müssen sich gemäß Abschnitt 4.4.4 verschiedene Cluster in extensiv vielen (d. h. $\mathcal{O}(N)$) Bits unterscheiden. Für die hier untersuchten Systeme endlicher Größe ist diese Definition aber zweckmäßig. Qualitativ ändern sich die Ergebnisse nicht, wenn man Cluster durch einen etwas größeren Abstand trennt, indem beispielsweise noch Zustände mit Hammingabstand 4 als benachbart definiert werden.

Der in Abbildung 4.11 dargestellte Graph aus vier Knoten hat drei Grundzustände, die alle zum selben Cluster gehören. Für den Graph in Abbildung 4.12, welcher ebenfalls aus vier Knoten besteht, aber einen Kreis besitzt, gehören die beiden Zustände zu verschiedenen Clustern. Da die Anzahl dieser kleinen Komponenten auch für $N \to \infty$ endlich bleibt (vgl. Abschnitt 4.2), ist ihr Beitrag zur Grundzustandslandschaft trivial. Deshalb werden sie bei der Analyse der Clusterstruktur nicht berücksichtigt, sondern nur die jeweils größte Komponente des Graphen untersucht.

Das naive Verfahren besteht somit aus den folgenden drei Schritten:

1. Bestimmen der größten Komponente des Graphen,

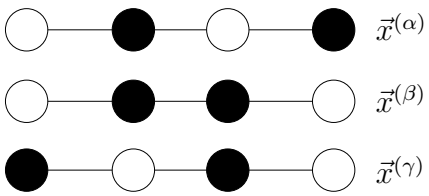


Abbildung 4.11: Es ist $d_{\alpha\beta} = d_{\beta\gamma} = 2$, also gehören $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ sowie $\vec{x}^{(\beta)}$ und $\vec{x}^{(\gamma)}$ zum selben Cluster. Wegen Transitivität gehören somit alle drei Zustände zu einem einzigen Cluster.

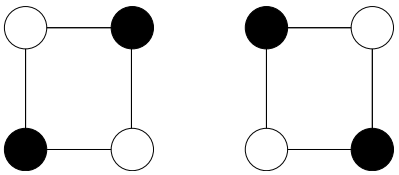


Abbildung 4.12: Ein Loop aus vier Knoten hat zwei Grundzustände, die den Hammingabstand 4 haben.

2. Berechnung aller Grundzustände dieses Untergraphen,
3. Clusterung der Grundzustände.

Der im 3. Schritt zur Unterteilung der Menge S der Grundzustände in Cluster verwendete Algorithmus 4.3 ist quadratisch in der Anzahl der Grundzustände, Es können damit Systeme mit bis ca. 10 000 Grundzuständen behandelt werden. Entscheidend ist aber, dass tatsächlich alle Grundzustände des Graphen bestimmt werden müssen, dadurch ist die behandelbare Systemgröße auf $N \approx 70$ beschränkt. Für jeden betrachteten Wert von N wurden 10^4 Systeme untersucht. Die mittlere Anzahl (Median) der Cluster als Funktion der Konnektivität c sind als Kreissymbole für $N \in \{25, 35, 50, 71\}$ in Abbildung 4.16 auf Seite 59 dargestellt. Für $c = 1$ gibt es in fast allen Systemen genau einen Cluster. Dies stimmt mit der im folgenden Abschnitt 4.5.2 bewiesenen Aussage überein, dass für baumartige Graphen (welche typisch für $c < 1$ sind) immer genau ein Cluster existiert.

Für sehr große Werte wie $c = 6$ ist bereits in diesem Bereich kleiner Systeme der Anstieg der Clusteranzahl erkennbar. Hauptsächlich wird die Methode aber dazu dienen, die Gültigkeit ihrer im Abschnitt 4.5.3 beschriebenen Erweiterung zu rechtfertigen, bei der für größere Systeme statt aller Grundzustände nur eine Auswahl berücksichtigt wird.

4.5.2 Clusterstruktur baumartiger Graphen

In diesem Abschnitt werden wir zeigen, dass die Grundzustände von baumartigen Graphen immer in genau einem Cluster liegen, d. h. man kann zwischen zwei beliebigen Grundzuständen immer eine Folge M benachbarter Grundzustände finden.

```
begin
    S:= Menge der Grundzustände
    i = 0        /* Anzahl der bisher gefundenen Cluster */
    while S ≠ ∅ do
    begin
        i = i + 1
        entferne ein beliebiges Element x⃗^(α) aus S
        C_i = (x⃗^(α))
        setze Zeiger x⃗^(β) auf das erste Element in C_i
        while x⃗^(β) ≠ NULL do
        begin
            for all Elemente x⃗^(γ) ∈ S
                if d_ham(x⃗^(β), x⃗^(γ)) = 2 then
                begin
                    entferne x⃗^(γ) aus S
                    setze x⃗^(γ) an das Ende von C_i
                end
            setze Zeiger x⃗^(β) auf das nächste Element in C_i
            oder auf NULL, wenn kein weiteres existiert
        end
    end
end
```

Algorithmus 4.3: Algorithmus zur Bestimmung der Cluster von Zuständen mit minimalem Hammingabstand: Wenn zu einem schon in einen Cluster eingeordneten Zustand $\vec{x}^{(\beta)}$ ein noch nicht eingeordneter Zustand $\vec{x}^{(\gamma)}$ Hammingabstand 2 hat, so wird $\vec{x}^{(\gamma)}$ ebenfalls diesem Cluster zugeordnet, andernfalls wird ein neuer Cluster begonnen.

Der Beweis erfolgt durch Induktion nach der Anzahl N der Knoten des Baumes.

Für $N = 2$ gibt es genau zwei Grundzustände, diese haben Hammingabstand 2.

Angenommen, die Behauptung sei für alle Bäume mit maximal N Knoten gezeigt. Wir betrachten nun einen Baum G der Größe $N + 1$.

Dieser Graph hat als Baum mindestens einen Knoten (Blatt) v_0 mit genau einem Nachbarknoten v. Mittels *Leaf-Removal* können wir einen Grundzustand $\vec{x}^{(\delta)}$ konstruieren, in dem v bedeckt ist. Den Grad von v bezeichnen wir mit k.

Wir zeigen getrennt, dass $\vec{x}^{(\delta)}$ im selben Cluster wie alle Grundzustände $\{\vec{x}^{(\delta')}\}$ mit v bedeckt und v unbedeckt ist.

Sei also zunächst $\vec{x}^{(\delta')}$ ein beliebiger weiterer Grundzustand mit v bedeckt. Falls kein solcher Zustand existiert, ist in diesem Fall nichts zu zeigen. Durch Entfernung des Knotens v zerfällt der baumartige Graph in k Untergraphen $G_1 = \{v_0\}, G_2, \ldots, G_k$. Sowohl $\vec{x}^{(\delta)}$ als auch $\vec{x}^{(\delta')}$ induzieren auch auf jedem dieser Untergraphen ein minimales VC, da sie andernfalls entgegen der Voraussetzung auch keine Grundzustände von G wären. Sämtliche Untergraphen haben aber höchstens N Knoten (genauer gesagt sogar nur $N - 1$), demnach kann nach Induktionsvoraussetzung für jeden Untergraphen unabhängig eine Folge M_k gefunden werden, die die induzierten Grundzustände miteinander verbindet. Damit lassen sich auch $\vec{x}^{(\delta)}$ und $\vec{x}^{(\delta')}$ durch eine Folge benachbarter Grundzustände von G verbinden und gehören mithin zum selben Cluster.

Sei nun $\vec{x}^{(\delta')}$ ein beliebiger Grundzustand mit v unbedeckt. Falls kein solcher Zustand existiert, ist nichts weiter zu zeigen. Wie oben konstruieren wir Untergraphen G_k durch Entfernung von v (siehe Abb. 4.13).

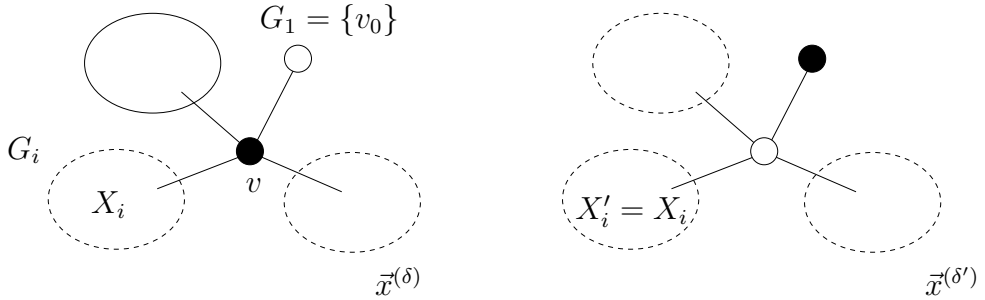


Abbildung 4.13: Links: Beliebiger Grundzustand $\vec{x}^{(\delta)}$ mit v bedeckt. Die Anzahl der bedeckten Knoten X_i im Untergraphen G_i ist für alle derartigen Grundzustände identisch. Rechts: Beliebiger Grundzustand $\vec{x}^{(\delta)'}$ mit v unbedeckt. Für alle Untergraphen $G_i \neq G_1$ stimmt die Anzahl der bedeckten Knoten X_i' mit X_i überein.

Sei X_i die Größe der von $\vec{x}^{(\delta)}$ im Untergraphen G_i erzeugten Überdeckung, ebenso X_i' die Größe der von $\vec{x}^{(\delta')}$ erzeugten Überdeckung. Da sowohl $\vec{x}^{(\delta)}$ als auch $\vec{x}^{(\delta')}$ beides

Grundzustände von G sind und bei $\vec{x}^{(\delta)}$ ein bedeckter Knoten entfernt wurde, gilt $1 + \sum_{i=1}^{k} X_i = \sum_{i=1}^{k} X_i'$. Dies ist äquivalent zu

$$1 = \sum_{i=1}^{k} (X_i' - X_i). \tag{4.27}$$

Alle Summanden auf der rechten Seite sind nicht negativ, mithin gibt es genau einen Untergraphen G_j mit $X_j' - X_j = 1$. Da der Untergraph $G_1 = \{v_0\}$ diese Eigenschaft erfüllt, muss $G_j = G_1$ sein.

Für alle anderen Untergraphen sind die durch $\vec{x}^{(\delta)}$ bzw. $\vec{x}^{(\delta')}$ erzeugten Überdeckungen gleich groß. Da sie für $\vec{x}^{(\delta)}$, wie im anderen Fall gezeigt, Grundzustände der Untergraphen G_k sein müssen, sind sie das somit auch für $\vec{x}^{(\delta')}$. Wieder kann man durch Induktion eine die beiden Grundzustände verbindende Folge M separat für jeden Untergraphen G_i (mit $i \neq j$) und für den Untergraphen $\{v\} \cup \{v_0\}$ konstruieren, somit gehören auch in diesem Fall beide Grundzustände zum selben Cluster.

Da wir für jeden Zustand $\vec{x}^{(\delta')}$ eine Folge benachbarter Grundzustände konstruiert haben, die ihn mit $\vec{x}^{(\delta)}$ verbindet, können wir auch für zwei beliebige Grundzustände eine solche Folge konstruieren. Mithin liegen für baumartige Graphen alle Grundzustände in einem einzigen Cluster.

4.5.3 Ballistische Suche

In diesem Abschnitt soll beschrieben werden, wie ausgehend nur von einer repräsentativen Auswahl von Grundzuständen die Clusterstruktur rekonstruiert werden kann. Zwei Grundzustände $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ gehören dabei wieder zum selben Cluster, wenn eine Folge M im Phasenraum benachbarter Grundzustände existiert, die diese beiden Grundzustände verbindet. Wenn zwar eine solche Folge in der Menge aller Grundzustände existiert, wird ihre Existenz im Allgemeinen nicht mehr gewährleistet sein, wenn nur noch eine Auswahl der Grundzustände zur Verfügung steht. Mit der hier beschriebenen *ballistischen Suche* [60] wird es möglich sein, die dazwischen liegenden Grundzustände zu konstruieren. Das Verfahren wurde bereits bei der Grundzustandsanalyse endlich-dimensionaler Spingläser verwendet [59]. Der Algorithmus startet in einem der beiden Zustände, z. B. in $\vec{x}^{(\alpha)}$, und findet eine Folge von Grundzuständen, geführt von einer lokalen Austauschdynamik, die die Energie konstant lässt. Wenn auf diese Weise der andere Zustand $\vec{x}^{(\beta)}$ erreicht wird, so gehören – genau wie im vorherigen Abschnitt 4.5.1 – beide Zustände zum selben Cluster.

Die Dynamik selbst hängt von den beiden zu untersuchenden Zuständen $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ ab: Sei $cov^{(\alpha)}$ die Untermenge der Knoten des Graphen G, die in Zustand $\vec{x}^{(\alpha)}$ bedeckt sind, nicht aber in $\vec{x}^{(\beta)}$. Analog definieren wir $cov^{(\beta)}$. Da beides Grundzustände sind, gilt $|cov^{(\alpha)}| = |cov^{(\beta)}|$. Sei $v \in cov^{(\alpha)}$ ein beliebiger im Zustand $\vec{x}^{(\alpha)}$

bedeckter Knoten. Mindestens einer der Nachbarn von v muss in $\vec{x}^{(\alpha)}$ unbedeckt sein, da dies sonst kein minimales VC wäre. Nennen wir diesen Nachbarn w. Umgekehrt ist v im Zustand $\vec{x}^{(\beta)}$ unbedeckt, d. h. alle seine Nachbarn müssen in diesem Zustand bedeckt sein, sonst wäre $\vec{x}^{(\beta)}$ kein VC. Daraus folgt: $w \in cov^{(\beta)}$.

Wie soeben gezeigt, hat jeder Knoten in $cov^{(\alpha)}$ einen Nachbarn in $cov^{(\beta)}$, während keine zwei Knoten aus $cov^{(\alpha)}$ bzw. $cov^{(\beta)}$ untereinander benachbart sind. Der Untergraph G' von G, der alle Knoten aus $cov^{(\alpha)}$ und $cov^{(\beta)}$ und die sie in G verbindenden Kanten enthält, ist somit ein bipartiter Graph, d. h., seine Knoten können in der beschriebenen Weise auf zwei Mengen so aufgeteilt werden, dass alle Kanten nur Knoten aus verschiedenen Mengen verbinden. Jede dieser beiden Mengen $cov^{(\alpha)}$ und $cov^{(\beta)}$ ist ein minimales VC von G'.

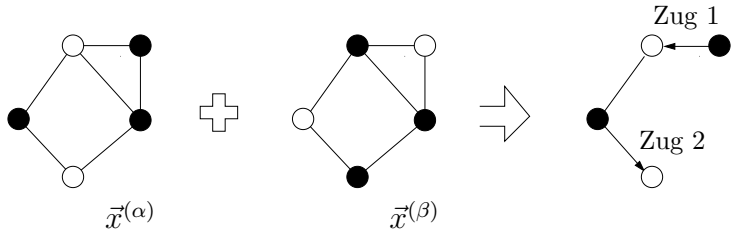


Abbildung 4.14: Ballistische Suche: Bildung des bipartiten Untergraphen (rechts) aus den Knoten, in denen sich die Zustände $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ (rechts) unterscheiden. Alle in Zustand $\vec{x}^{(\alpha)}$ bedeckten Knoten werden mit einem Markierungsstein versehen. Anschließend wird die Eigenschaft „bedeckt“ jeweils entlang einer Kante so verschoben, dass nach jedem Zug die bedeckten Knoten ein VC bilden.

Nach der Bestimmung dieses bipartiten Graphen G' verläuft die ballistische Suche wie folgt:

1. Wähle einen Knoten $v \in cov^{(\alpha)} \subset G'$ mit genau einem Nachbarn w in G'; wenn kein solches v existiert: STOP.
2. $G' := G' \setminus \{v, w\}$.
3. Gehe zu Schritt 1.

Die Anzahl der Knoten in G' ist endlich und in jedem Durchlauf werden zwei Knoten entfernt. Nach endlich vielen Schritten sind damit alle Knoten entfernt, spätestens dann terminiert der Algorithmus. Sind am Ende alle Knoten entfernt, ist also $G' = \emptyset$, dann gehören $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ zum selben Cluster.

Anschaulich kann man den Algorithmus verstehen, in dem man sich jeden Knoten aus $cov^{(\alpha)}$ mit einem Markierungsstein bedeckt denkt. Alle Nachbarn (die Knoten aus $cov^{(\beta)}$) sind unbedeckt. Jeder Stein wird höchstens einmal entlang einer Kante bewegt (siehe Abbildung 4.14). Ein Zug ist nur dann erlaubt, wenn der benachbarte

Knoten der einzige unbedeckte Nachbar ist. Damit sind nach jedem Zug die von Steinen bedeckten Knoten ein (minimales) VC von G'. Anschließend werden Anfangs- und Endpunkt und die sie verbindende Kante aus G' entfernt.

Wurde jeder Stein genau einmal gezogen, dann sind (wegen der bipatiten Zerlegung von G' in $cov^{(\alpha)}$ und $cov^{(\beta)}$) genau alle Knoten aus $cov^{(\beta)}$ bedeckt. Dies ist genau dann der Fall, wenn $G' = \emptyset$ nach Beendigung des Algorithmus gilt. Führt man die Züge ohne Entfernen der Knoten auf dem Ausgangsgraphen G selbst durch, so erhält man eine Folge M von Zuständen mit Hammingabstand 2, die $\vec{x}^{(\alpha)}$ und $\vec{x}^{(\beta)}$ verbindet. Beide gehören in diesem Fall also zum selben Cluster. Wenn nicht jeder Stein genau einmal bewegt werden kann, dann bricht der Algorithmus mit $G \neq \emptyset$ ab und die beiden Zustände werden verschiedenen Clustern zugeordnet.

Abhängig von der Systemgröße wurden für ca. 100 bis 500 Realisierungen mittels *Parallel Tempering* (vgl. Abschnitt 2.2.2) während einer Zeit von 10^6 MC-Schritten jeweils 500 Zustände bei $\mu = 9$ bestimmt. Diese Anzahl reicht aus, um sicherzustellen, dass zwei zum selben Cluster gehörige Zustände nicht fälschlicherweise in verschiedene Cluster eingeordnet werden, da für größere Cluster zwei weiter entfernt liegende Zustände auch dann demselben Cluster zugeordnet werden, wenn sie sich mit einem gemeinsamen dritten Zustand in einem Cluster befinden [60].

Für kleine Cluster kann es passieren, dass aus ihnen während des Parallel Tempering gar kein Zustand ausgewählt wird und sie deshalb nicht erkannt werden können. Dies wird in Abbildung 4.15 deutlich: Während für $c = 2$ kein Unterschied in der Anzahl der Cluster zwischen der Auswahl von 32 und 500 Zuständen messbar ist, steigt für $c > e$ die Anzahl an, da Zustände aus bisher nicht gezählten Clustern hinzukommen.

Die Ergebnisse der Clusterung mit ballistischer Suche sind in Abbildung 4.16 dargestellt. Im Übergangsbereich bei $N \approx 50$ stimmen die Ergebnisse mit denen aus Abschnitt 4.5.1 überein. Für kleine Konnektivitäten $c = 1$ wird, wie im RS-Bereich erwartet, ziemlich genau ein Cluster gefunden, auch für $c = 2$ bleibt die mittlere Clusteranzahl konstant und nahe eins. Für große Werte von c steigt die Anzahl der Cluster, dies ist vereinbar mit der Veränderung der Grundzustandslandschaft bei $c = e$. Für $c > e$ sind, wie oben beschrieben, die Werte nur untere Schranken, so dass auch ein stärkerer als der in der Abbildung erkennbare logarithmische Anstieg denkbar ist. Allerdings werden selbst bei der größten betrachteten Systemgröße $N = 800$ bei $c = 4$ mit 500 Zuständen im Mittel weniger als drei Cluster gefunden, so dass ein exponentieller Anstieg eher unwahrscheinlich ist.

4.5.4 Extensive Eigenwerte und Clusteranzahl

In diesem Abschnitt soll die Clusterstruktur mit einem ganz anderen Ansatz untersucht werden. Sinova et al. [21, 22] beschreiben ein Verfahren zur Bestimmung der Anzahl reiner Zustände (vgl. Abschnitt 4.4.4) in Ising-Spingläsern. Dieses Verfahren

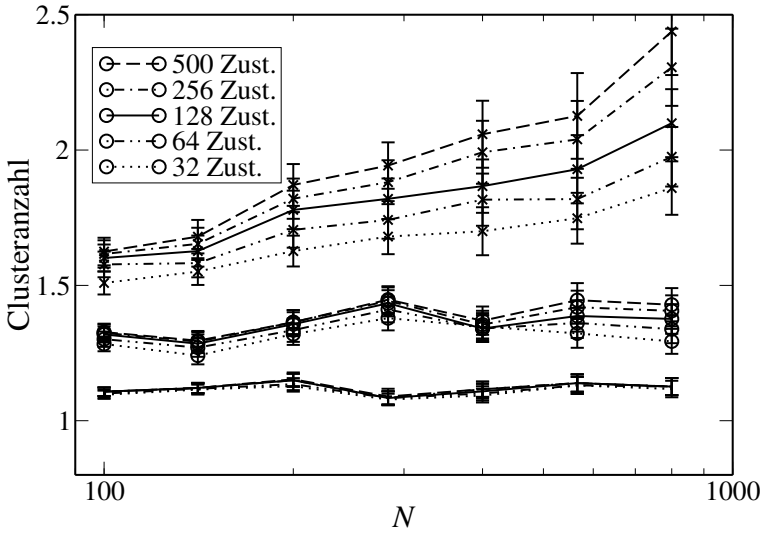


Abbildung 4.15: Abhängigkeit der mittleren Anzahl gefundener Grundzustandcluster von der Anzahl der ausgewählten Zustände bei verschiedenen Konnektivitäten ($c = 2$: Striche, $c = 3$: Kreise, $c = 4$: Kreuze). Für $c < e$ werden im Rahmen der Fehlerbalken mit wachsender Zustandsanzahl keine neuen Cluster gefunden, während für $c > e$ deren Anzahl mit der Anzahl ausgewählter Zustände ansteigt.

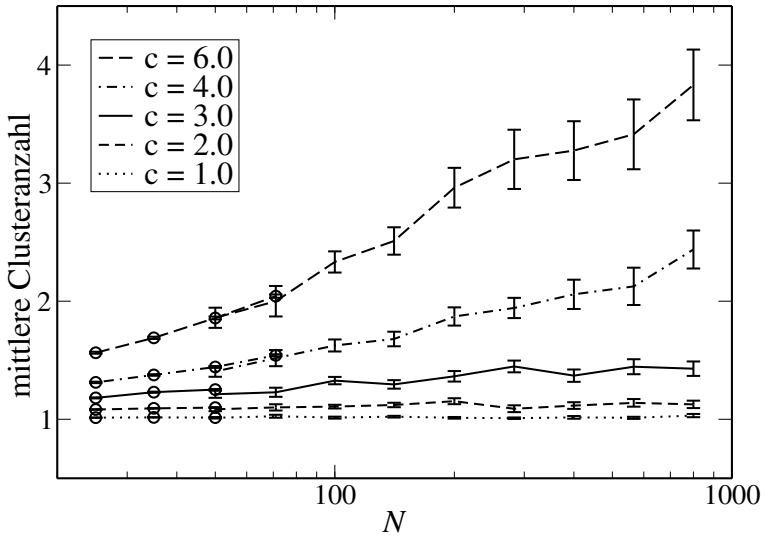


Abbildung 4.16: Mittlere Anzahl der Grundzustands-Cluster in Abhängigkeit der Systemgröße für verschiedene Konnektivitäten. Für kleine Systeme (Kreis-Symbole) wurde jeweils wie in Abschnitt 4.5.1 beschrieben die Menge aller Grundzustände betrachtet. Bei größeren Systemen (Strich-Symbole) erfolgte die Clusterung durch ballistische Suche mit 500 ausgewählten Zuständen (Abschnitt 4.5.3), so dass für $c > e$ die dargestellten Werte untere Schranken der tatsächlichen Clusteranzahl bilden.

wird hier auf das VC-Problem übertragen. Die grundlegende Idee ist, die spektralen Eigenschaften der Spin-Spin-Korrelationsmatrix $\langle S_i S_j \rangle \equiv C_{ij}$ zu betrachten, wobei $\langle \cdot \rangle$ wie in Abschnitt 4.4.2 das thermische Mittel bezeichnet. Diese Matrix C hat folgende Eigenschaften:

1. C ist semidefinit, denn für $\vec{v}$ beliebig ist $\sum_{i,j} v_i \langle S_i S_j \rangle v_j = \langle (\sum_i S_i v_i)^2 \rangle \geq 0$.
2. C hat extensive Spur N, denn $Tr\, C = \sum_i \langle S_i^2 \rangle = N$.
3. Aus 1. und 2. folgt: C hat N positive Eigenwerte mit Summe N.

Im Unterschied zum VC-Problem treten in Ising-Spingläsern wegen der Up-down-Symmetrie die reinen Zustände immer als Paare auf. Oberhalb der kritischen Temperatur T_c, also in der paramagnetischen Phase, ist für Mean-Field-Spingläser $C_{ij} = \delta_{ij}$, alle Eigenwerte sind somit 1. Unterhalb T_c ist die Situation anders: Wenn genau ein Paar reiner Zustände existiert, dann gilt $C_{ij} \to \pm 1$ für $T \to 0$ und der Zustand ist Eigenvektor zum Eigenwert N; alle Eigenwerte gehen gegen 0. Man kann also am Spektrum der Spin-Spin-Korrelationsmatrix den Übergang bei T_c und die dabei einsetzende langreichweitige Ordnung erkennen.

Die Autoren [21, 22] zeigen darüber hinaus, dass die Anzahl der extensiven Eigenwerte der Anzahl der reinen Zustände (bzw. Zustandspaare) entspricht, während alle anderen Eigenwerte mit einer Potenz N^α, $\alpha < 1$, gegen null gehen. Eventuelle Hierarchieebenen einer Clusterstruktur kann man allerdings nicht im Spektrum erkennen, da nur etwas über die *Anzahl* reiner Zustände ausgesagt wird. Ebenso gibt es keinen direkten Zusammenhang zur Clusteranzahl. Somit kann man die Replika-symmetrische-Phase, in der ein einziger reiner Zustand und damit genau ein extensiver Eigenwert existiert, von Replika-Symmetriebrechung unterscheiden (vgl. Abschnitt 4.4.4), nicht aber die Art der RSB (z. B. 1-RSB oder kontinuierliche RSB).

Hier wird die Methode für das VC-Problem angewendet, in dem wir wie im vorherigen Abschnitt mit Parallel Tempering niedrigenergetische Zustände erzeugen und die Knoten-Knoten-Korrelationsmatrix C_{ij} berechnen (wobei wir $S_i := 2\nu_i - 1$ setzen).

Der größte Eigenwert ist immer extensiv, da das System für den betrachten Wert des chemischen Potentials $\mu = 9$ nicht paramagnetisch ist. In Abbildung 4.17 sind der zweit- und drittgrößte Eigenwert dargestellt. Für $c = 1$ und $c = 2$ gehen beide Eigenwerte wie N^α gegen null, wie man das für die RS-Phase erwartet.

Für große Konnektivitäten findet man ein ganz anderes Verhalten. Der zweitgrößte Eigenwert erreicht ein Plateau, für $c = 12$ bei $N \approx 200$. Je näher die Konnektivität an $c = e$ ist, desto später wird dieses Plateau erreicht. Insbesondere für $c = 3$ ist der Trend mit den erreichbaren Systemgrößen nicht eindeutig erkennbar. Für den drittgrößten Eigenwert sieht man ebenfalls einen deutlichen Unterschied zwischen kleinen c, bei denen dieser Eigenwert mit steigender Systemgröße verschwindet, und

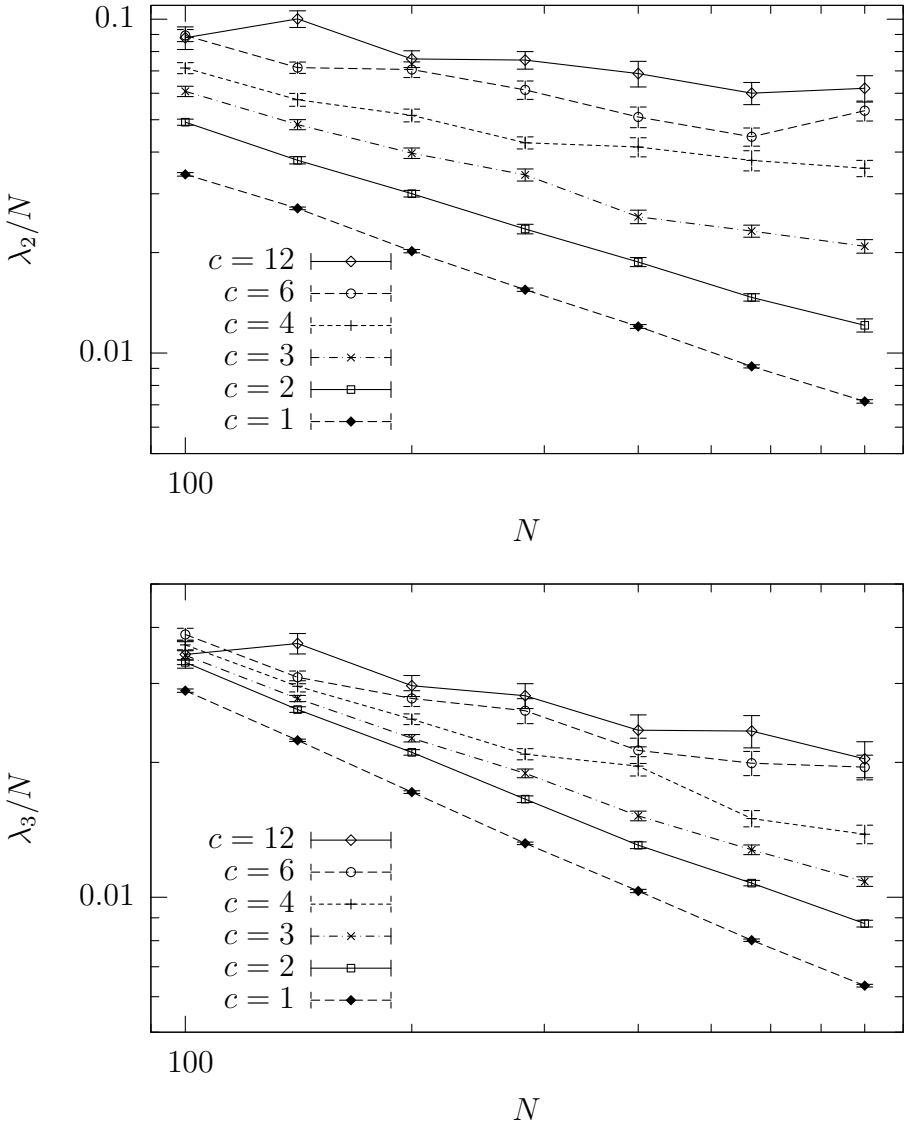


Abbildung 4.17: Skalierung des zweitgrößten Eigenwertes λ_2 (oben) und des drittgrößten Eigenwertes λ_3 (unten) von C_{ij}. Linien dienen nur der Orientierung.

großen Konnektivitäten. Allerdings ist hier das Plateau schwächer ausgeprägt und man kann mit den gegebenen Daten nicht ausschließen, ob dieser Eigenwert nicht auch für größere Konnektivitäten gegen null geht.

Insgesamt lässt sich aber sagen, dass oberhalb eines c' ReplikaSymmetriebrechung auftritt (wobei der Wert von c' zwischen 2 und 4 liegt), da mehr als ein Eigenwert extensiv ist. Wie bereits erläutert, kann man die Art der RSB nicht am Spektrum ablesen, da nur die Anzahl der Zustände, nicht aber die Anzahl der Hierarchieebenen bestimmt wird. Dies soll mit der Methode des nächsten Abschnittes erfolgen.

4.5.5 Hierarchische Clusterung

Kontinuierliche Replika-Symmetriebrechung (∞-RSB) findet man im Mean-Field-Modell für Spingläser, dem SK-Modell [15] (siehe Kapitel 3). Eine wichtige Eigenschaft der Parisi-Lösung für dieses System [16] ist eine hierarchisch organisierte Grundzustandsstruktur (vgl. Abschnitt 4.4.4), welche für die Abstände (bzw. Überlapps) von je drei Grundzuständen *Ultrametrizität* impliziert [61]: Es gilt für die Hammingabstände dreier Zustände, von denen o. B. d. A. $d_{\beta\gamma}$ der kürzeste sei:

$$d_{\alpha\beta} \geq d_{\alpha\gamma} \geq d_{\beta\gamma} \Longrightarrow d_{\alpha\beta} = d_{\alpha\gamma} \geq d_{\beta\gamma} \tag{4.28}$$

Diese Eigenschaft ist stärker als die Dreiecksungleichung $d_{\beta\gamma}+d_{\alpha\gamma} \geq d_{\alpha\beta}$ und besagt, dass im metrischen Raum $(\{\vec{x}\}, d_{\mathrm{ham}}(\vec{x}^{(\alpha)}, \vec{x}^{(\beta)}))$ jeweils drei beliebige Zustände ein gleichschenkliges Dreieck mit der Basis als höchstens gleich langer Seite bilden.

Mit der hier vorgestellten Methode soll untersucht werden, ob auch bei VC eine derartige ultrametrische Struktur als Indiz für Replika-Symmetriebrechung gefunden werden kann. In diesem Abschnitt bezeichnen wir als Hammingabstand immer den normierten Hammingabstand, d. h.

$$d_{\mathrm{ham}}(\vec{x}^{(\alpha)}, \vec{x}^{(\beta)}) = \frac{1}{N} \sum_{i=1}^{N} |x_i^{(\alpha)} - x_i^{(\beta)}|. \tag{4.29}$$

Wir werden nun diesen Begriff des Hammingabstands zwischen zwei Zuständen auf einen Abstand $d_{\mathrm{proxim}}(\mathcal{C}_\alpha, \mathcal{C}_\beta)$ zwischen zwei aus mehreren Zuständen bestehenden Clustern erweitern und diesen Abstand *Proximity Matrix* nennen. Da er für zwei aus nur je einem Zustand bestehenden Cluster mit dem Hammingabstand übereinstimmen soll, werden wir auch hier $d_{\mathrm{proxim}}(\mathcal{C}_\alpha, \mathcal{C}_\beta) = d_{\alpha\beta}$ abkürzen.

Eine hierarchische Struktur wird gemäß folgendem Algorithmus konstruiert:

1. Initialisiere die Menge der Cluster $\{\mathcal{C}_\alpha\}$ mit $\mathcal{C}_\alpha := \{\vec{x}^{(\alpha)}\}$, d. h. mit genau einem Zustand je Cluster.
2. $d_{\mathrm{proxim}}(\mathcal{C}_\alpha, \mathcal{C}_\beta) := d_{\mathrm{ham}}(\vec{x}^{(\alpha)}, \vec{x}^{(\beta)}) \equiv d_{\alpha\beta}$.

3. Wähle Cluster $\mathcal{C}_\alpha$, $\mathcal{C}_\beta$ mit minimalem Abstand $d_{\alpha\beta}$.
4. Ersetze $\mathcal{C}_\alpha$ und $\mathcal{C}_\beta$ durch $\mathcal{C}_\gamma = \mathcal{C}_\alpha \cup \mathcal{C}_\beta$ in der Menge der Cluster.
5. $d_{\text{proxim}}(\mathcal{C}_\gamma, \mathcal{C}_\delta) := f(d_{\alpha\delta}, d_{\beta\delta}, d_{\alpha\beta}, \dots) \quad \forall \delta \neq \gamma$.
6. Gehe zu Schritt 3., wenn noch mehr als ein Cluster vorhanden ist.

Der Algorithmus fasst also in jedem Schleifendurchlauf die zwei ähnlichsten Cluster zu einem größeren zusammen und aktualisiert anschließend die Abstände zu diesem neuen Cluster. Die entstehende hierarchische Struktur wird in einem Dendogramm dargestellt (siehe Abbildung 4.18). Dies ist ein Baum mit den nur aus je einem Grundzustand bestehenden Clustern als Blättern. Jeder Knoten entspricht einem Cluster $\mathcal{C}_\gamma$, die Höhe des Knotens im Dendogramm ist gleich dem Abstand $d_{\text{proxim}}(\mathcal{C}_\alpha, \mathcal{C}_\beta)$ der beiden Cluster $\mathcal{C}_\alpha$ und $\mathcal{C}_\beta$, die zu $\mathcal{C}_\gamma$ vereinigt wurden.

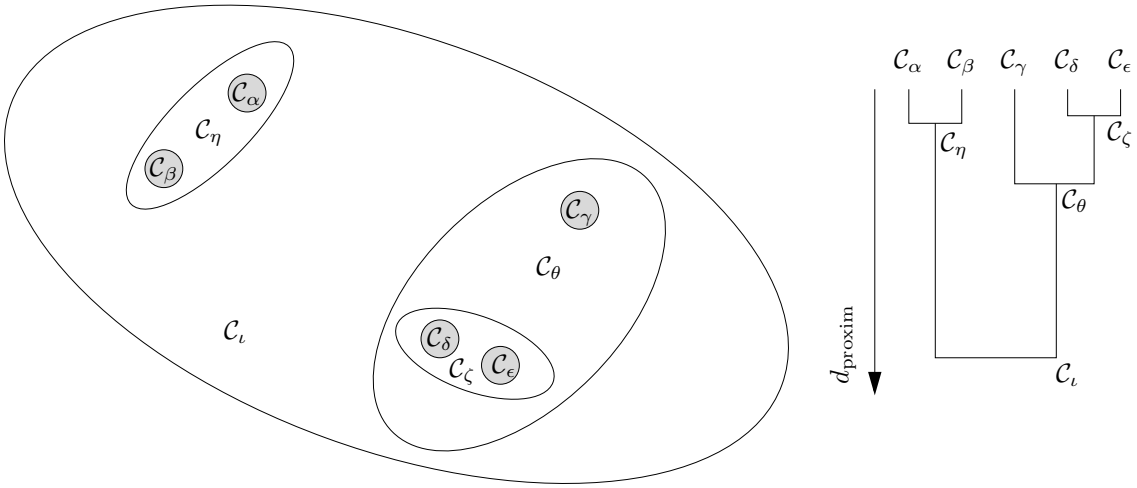


Abbildung 4.18: Schematische Darstellung eines hierarchischen Clusterungsverfahrens (links) und des dabei entstehenden Dendogramms (rechts).

In Schritt 5 des Algorithmus wird die Proximity Matrix um die Abstände zwischen den bereits vorhandenen und dem neu hinzugekommenen Cluster erweitert. Für die genaue Form der Funktion f gibt es verschiedene gebräuchliche Möglichkeiten (Übersicht dazu siehe [62]). Die meisten genügen der allgemeinen Form

$$d_{\gamma\delta} = k_1 d_{\delta\alpha} + k_2 d_{\delta\beta} + k_3 d_{\alpha\beta} + k_4 |d_{\delta\alpha} - d_{\delta\beta}|, \tag{4.30}$$

in der $\mathcal{C}_\gamma$ der aus der Vereinigung von $\mathcal{C}_\alpha$ und $\mathcal{C}_\beta$ entstandende Cluster und $\mathcal{C}_\delta$ ein beliebiger weiterer Cluster ist [63]. Die Koeffizienten k_i ergeben sich im Allgemeinen aus gewissen Parametern der Cluster. Mit $k_1 = k_2 = 1/2, k_3 = 0, k_4 = -1/2$ beispielsweise wird der Abstand zweier Cluster als Minimum aller Abstände zwischen jeweils einem (Ausgangs-)Zustand aus jedem dieser beiden Cluster definiert, mit $k_1 = k_2 = 1/2, k_3 = 0, k_4 = +1/2$ dagegen als Maximum aller dieser Abstände.

Abhängig von der Form von f können sich somit unterschiedliche Clusterungen ergeben. Idealerweise sollte das Ergebnis des Algorithmus aber die in den Daten implizit bereits vorhandene Clusterstruktur wiederspiegeln und nicht eine durch die Wahl der Funktion künstlich erzeugte.

Wir werden hier das Verfahren von Ward verwenden [64], welches empirisch gute Clusterungsergebnisse liefert [62]. Im Rahmen der statistischen Mechanik wurde es für die Untersuchung der Grundzustandslandschaft von Spingläsern verwendet [23–26], um dort ebenfalls eine ultrametrische Struktur im Zusammenhang mit RSB festzustellen.

Die Clusterung mit dem Verfahren von Ward hat folgende Eigenschaft: Jeder Cluster $\mathcal{C}_\delta$ hat ein Zentrum

$$\vec{m}^{(\delta)} = \frac{1}{n_\delta} \sum_{\vec{x}^{(\alpha)} \in \mathcal{C}_\delta} \vec{x}^{(\alpha)}, \tag{4.31}$$

wobei n_δ die Anzahl der Ausgangszustände in Cluster $\mathcal{C}_\delta$ ist und die Summe über diese Zustände läuft. Die Summe der quadratischen Abstände vom Zentrum der Zustände eines Clusters wird definiert durch

$$e^{(\delta)} = \sum_{\vec{x}^{(\alpha)} \in \mathcal{C}_\delta} \sum_{i=1}^{N} \left(\vec{x}_i^{(\alpha)} - \vec{m}_i^{(\delta)} \right)^2. \tag{4.32}$$

In jedem Schritt des Algorithmus ist die Änderung der quadratischen Abstände innerhalb des Clusters, summiert über alle Cluster, minimal. Das Verfahren wird deshalb auch „*minimum variance method*“ genannt.

Die Koeffizienten in der kombinatorischen Formel (4.30) hängen für das Verfahren von Ward von den Größen n_α, n_β und n_δ der entsprechenden Cluster $\mathcal{C}_\alpha, \mathcal{C}_\beta, \mathcal{C}_\delta$ ab:

$$k_1 = \frac{n_\alpha + n_\delta}{n_\alpha + n_\beta + n_\delta} \tag{4.33}$$

$$k_2 = \frac{n_\beta + n_\delta}{n_\alpha + n_\beta + n_\delta} \tag{4.34}$$

$$k_3 = -\frac{n_\alpha + n_\beta}{n_\alpha + n_\beta + n_\delta} = -\frac{n_\gamma}{n_\alpha + n_\beta + n_\delta} \tag{4.35}$$

$$k_4 = 0. \tag{4.36}$$

Damit ergibt sich der Abstand $d_{\gamma\delta}$ zwischen dem neu erzeugten Cluster $\mathcal{C}_\gamma$ und einem weiteren Cluster $\mathcal{C}_\delta$ zu

$$d_{\text{proxim}}(\mathcal{C}_\gamma, \mathcal{C}_\delta) = d_{\gamma\delta} = \frac{(n_\alpha + n_\delta) d_{\alpha,\delta} + (n_\beta + n_\delta) d_{\beta,\delta} - (n_\alpha + n_\beta) d_{\alpha,\beta}}{n_\alpha + n_\beta + n_\delta}\,. \tag{4.37}$$

Für das Vertex-Cover-Problem wurde die Clusterstruktur der mit Parallel Tempering erzeugten Zustände aus Abschnitt 4.5.4 untersucht. Typische Ergebnisse sind in Abb. 4.19 dargestellt. Die Abstandsmatrix $d_{\mathrm{ham}}(\vec{x}^{(\alpha)},\vec{x}^{(\beta)})$ wird jeweils als Quadrat veranschaulicht. Dunkle Farben entsprechen kleinen Hammingabständen, d. h. ähnlichen Zuständen. Die Bilder (a) und (e) zeigen zwei Beispiele unsortierter Abstandsmatrizen vor Anwendung des Clusterungsalgorithmus. In den Bildern (b) bis (d) und (f) sind unterhalb der sortierten Abstandsmatrizen die jeweiligen Dendogramme angegeben. Sie geben Aufschluss darüber, ab welcher Ebene zwei Zustände zum selben Cluster gehören. Durch das Ward-Verfahren wurden die Zustände entsprechend dieser Dendogramme umsortiert, d. h. alle Zustände sind jetzt so angeordnet, dass zum gleichen Cluster gehörende Zustände benachbart sind, da sich keine Äste in der Darstellung des Dendogramms kreuzen.

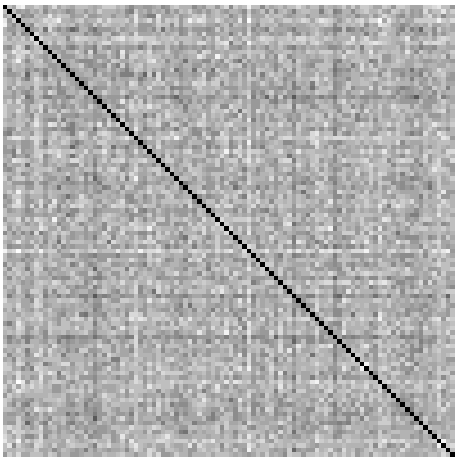
Für kleine Konnektivität $c = 1$ wird durch das Verfahren von Ward keine Clusterstruktur sichtbar (Abb. 4.19 (a)–(c)). Die Abstände zwischen den Zuständen sind annähernd gleich, sie liegen also alle in einem einzigen Cluster. Lediglich entlang der Diagonalen ist eine schwache Struktur erkennbar, die von den direkt benachbarten Zuständen herrührt. Vergleicht man die Fälle $\mu = 2$ und $\mu = 9$ miteinander, so zeigt sich, dass mit zunehmendem chemischem Potential die Abstände wie erwartet kleiner werden, da Grundzustände ein höheres Gewicht erhalten und damit während des PT-Algorithmus häufiger erreicht werden.

Für $c = 3$ ist bei $\mu = 2$ ebenfalls noch keine Clusterstruktur sichtbar (Abb. 4.19 (d)). Bei hohem chemischem Potential $\mu = 9$ sieht man dagegen im Vergleich von Abb. 4.19 (e) und (f) deutlich, dass eine Clusterstruktur vorhanden ist. Untermengen von Zuständen haben untereinander einen geringeren Abstand als zu Zuständen außerhalb. Bei genauerem Hinsehen erkennt man auch eine zweite und dritte Ebene der Clusterung, wie man sie bei kontinuierlicher RSB erwartet. Die Hierarchie erkennt man ergänzend im jeweiligen Dendogramm am Wert von $d_{\mathrm{proxim}}(\mathcal{C}_\alpha, \mathcal{C}_\beta)$, welcher als Höhe des Knotens im Dendogramm dargestellt wird (vgl. Abb. 4.18): Bei der Vereinigung von Clustern innerhalb einer Hierarchieebene ist die Zunahme des Abstands deutlich geringer als bei der Vereinigung zu einem Cluster einer höheren Hierarchieebene.

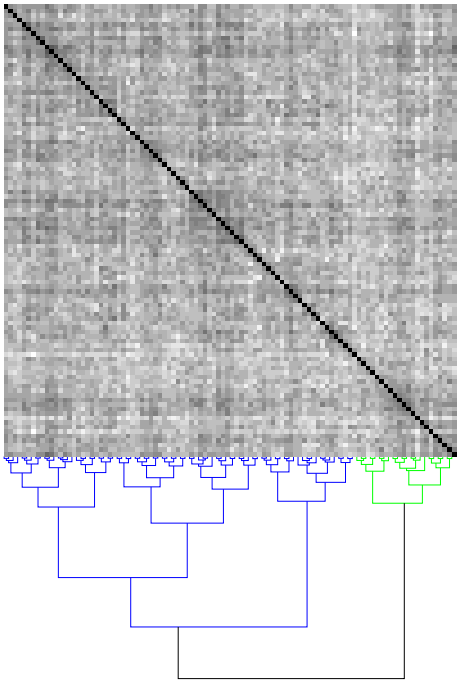
In den Dendogrammen erkennt man auch, dass das Verfahren von Ward immer eine „Clusterung“ findet, auch wenn in den zu Grunde liegenden Daten gar keine Clusterung vorhanden ist. Erst bei Betrachtung der Abstandsmatrizen erkennt man, ob die durch die Clusterung erfolgte Umsortierung tatsächlich zu einer Clusterstruktur korrespondiert.

Die Entfernung zweier Grundzustände auf dem Dendogramm bezeichen wir als ihren *kophenetischen Abstand* d_{coph} (*cophenetic distance*). Er wird folgendermaßen definiert: Seien $\vec{x}^{(\alpha)} \in \mathcal{C}_\gamma, \vec{x}^{(\beta)} \in \mathcal{C}_\delta$ und $\mathcal{C}_\varepsilon = \mathcal{C}_\gamma \cup \mathcal{C}_\delta$ mit $\mathcal{C}_\gamma \neq \mathcal{C}_\varepsilon \neq \mathcal{C}_\delta$, d. h. $\mathcal{C}_\varepsilon$ ist der Cluster, in dem die beiden Zustände $\vec{x}_\alpha$ und $\vec{x}_\beta$ das erste Mal gemeinsam enthalten sind. Dann ist

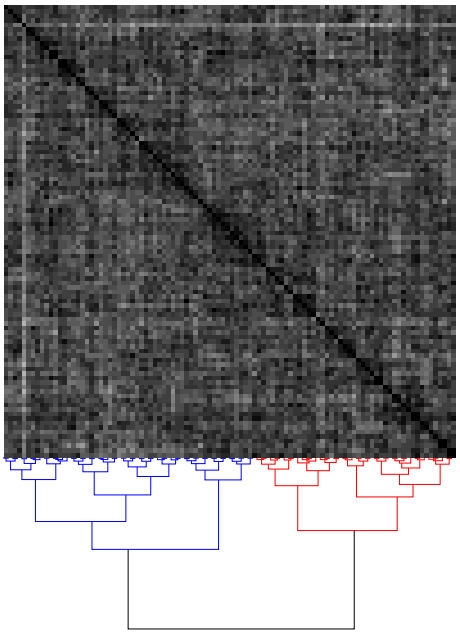
$$d_{\mathrm{coph}}(\vec{x}^{(\alpha)}, \vec{x}^{(\beta)}) = d_{\mathrm{proxim}}(\mathcal{C}_\gamma, \mathcal{C}_\delta) \tag{4.38}$$



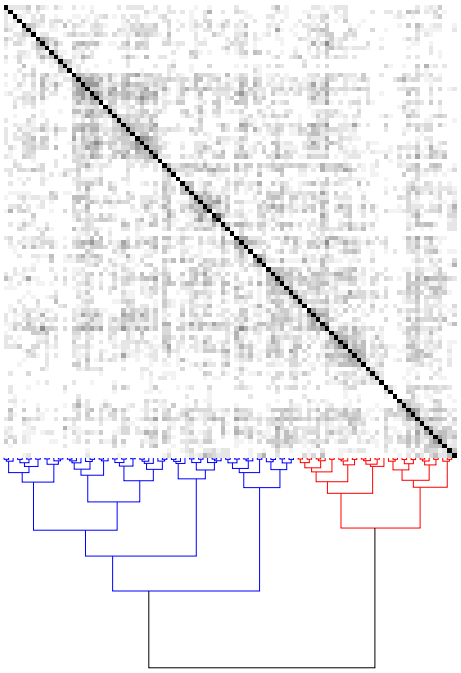
(a) $c = 1$, $\mu = 2$, unsortiert



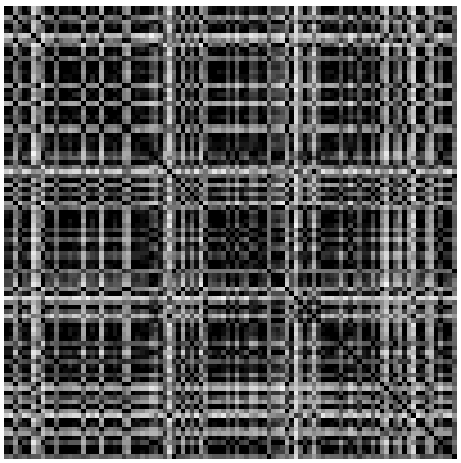
(b) $c = 1$, $\mu = 2$



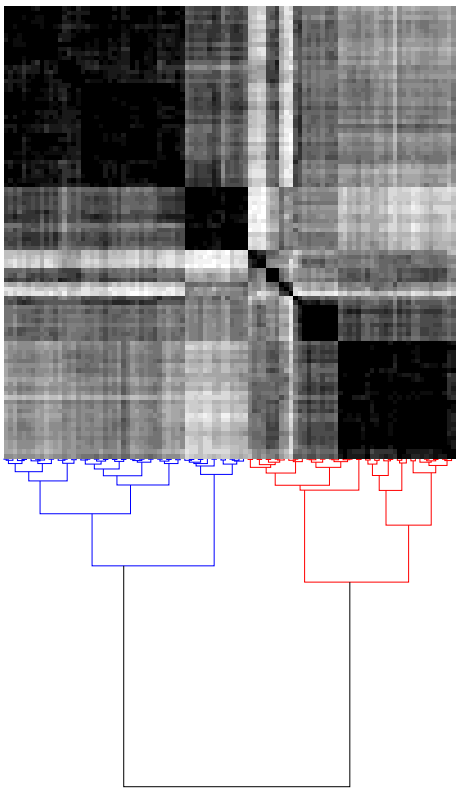
(c) $c = 1$, $\mu = 9$



(d) $c = 3$, $\mu = 2$



(e) $c = 3$, $\mu = 9$, unsortiert



(f) $c = 3$, $\mu = 9$

Abbildung 4.19: Typische Resultate der Anwendung des hierarchischen Clusterungsverfahrens auf 100 Zustände von Beispielgraphen mit $N = 400$ Knoten. Die Quadrate zeigen $d_{\text{ham}}(\vec{x}^{\alpha}, \vec{x}^{\beta})$. Dunkle Farben entsprechen kleinen Hammingabständen, also sehr ähnlichen Zuständen. Darunter dargestellt das durch den Algorithmus erzeugte Dendogramm. (a) zeigt die Abstandsmatrix vor Umordnung der Zustände (vgl. Text) durch Anwendung des Verfahrens von Ward. Für (b und c) kleine Konnektivität $c = 1$ ist nur ein Cluster erkennbar. Das Gleiche (d) gilt bei Konnektivität $c = 3$ (d. h. $c > e$) bei geringem chemischem Potential $\mu = 2$. Bei $c = 3$ und $\mu = 9$ sind die mit PT bestimmten Zustände fast ausschließlich Grundzustände. Vor der Anwendung (e) des Algorithmus von Ward ist die Clusterstruktur nicht erkennbar. Danach (f) erkennt man mehrere Ebenen einer Clusterhierarchie.

d_{coph} entspricht also der Höhe von $\mathcal{C}_\varepsilon$ im Dendogramm. Per Definition ist für drei Grundzustände mit diesem Abstand Ultrametrizität erfüllt.

Durch Vergleich dieses Abstands mit dem Hammingabstand kann man eine quantitative Aussage über die Gültigkeit der Clusterung gewinnen. Für die quantitative Analyse definiert man den *kophenetischen Korrelationskoeffizienten*

$$\mathcal{K} := \overline{d_{\text{ham}} \cdot d_{\text{coph}}} - \overline{d_{\text{ham}}} \cdot \overline{d_{\text{coph}}}, \tag{4.39}$$

wobei $\overline{\cdot}$ wieder das Mittel über verschiedene Realisierungen bezeichnet.

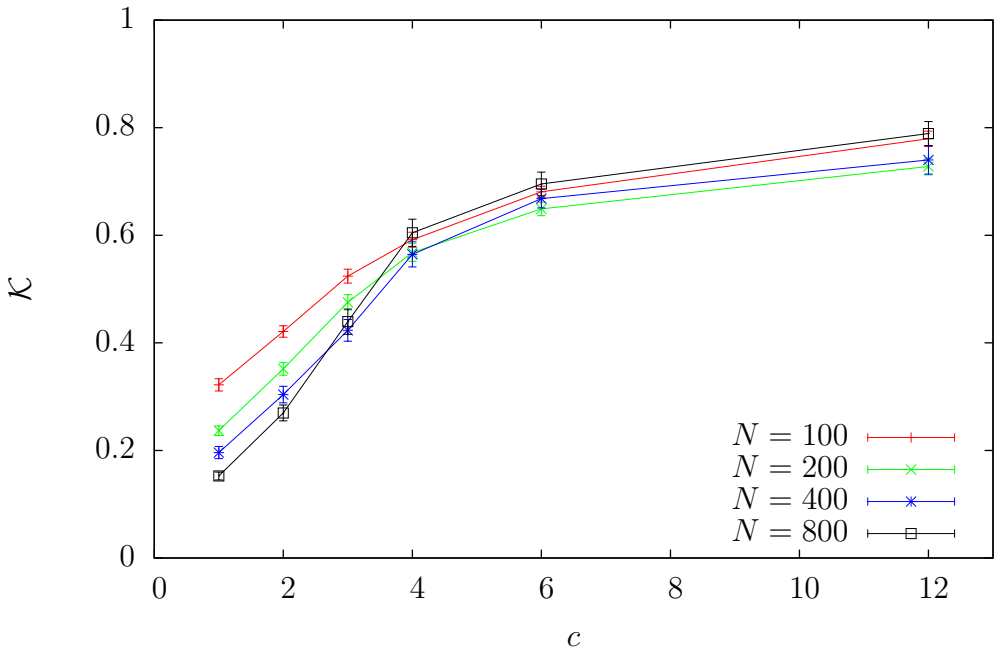


Abbildung 4.20: Der kophenetische Korrelationskoeffizient zwischen Hammingabstand und dem ultrametrischen Abstand auf dem Dendogramm.

Die Ergebnisse dieses Tests sind in Abbildung 4.20 dargestellt, wobei abhängig von der Systemgröße jeweils über ca. 100 bis 500 mit Parallel Tempering bestimmte Zustände gemittelt wurde. Für kleine Konnektivitäten $c < e$ ist wie erwartet die Korrelation klein, denn da keine ultrametrische Struktur vorhanden ist, kann die Clusterstruktur nur durch das Verfahren von Ward künstlich erzeugt worden sein. Mit steigender Konnektivität wächst auch $\mathcal{K}$. Insbesondere kreuzen sich die Kurven für verschiedene Systemgrößen bei $c \approx e$: Unterhalb e sinkt $\mathcal{K}$ mit wachsendem N für jedes feste c, oberhalb steigt $\mathcal{K}$ mit wachsendem N. Dies deutet darauf hin, dass für größere c als $c \approx e$ eine ultrametrische Struktur vorhanden ist. Allerdings sättigt sich der kophenetische Korrelationskoeffizient bei $\mathcal{K} \approx 0.8$, die durch das Ward-Verfahren erzeugte Clusterung gibt also möglicherweise nicht vollständig die Struktur minimaler VC wieder.

5 Laufzeit-Analyse einer einfachen Walk-SAT-Variante

Gegeben sei eine Menge boolescher Variablen und eine Menge von Bedingungen, die jeweils K dieser Variablen logisch verknüpft. Gibt es eine Belegung der Variablen, die alle Bedingungen gleichzeitig erfüllt?

Dieses aus der Theoretischen Informatik stammende Entscheidungsproblem wird als *K-Erfüllbarkeitsproblem* (*satisfiability problem*, kurz: SAT) bezeichnet. Werden die Bedingungen *zufällig* erzeugt, so spricht man vom Ensemble zufälliger K-SAT-Formeln. Dieses Ensemble besitzt – wie das im vorherigen Teil 4 betrachtete Vertex-Cover-Problem – Ähnlichkeiten zu verdünnten Spinglas-Modellen und ist deshalb auch aus Sicht der Statistischen Physik sehr interessant. Auch hier findet man einen Phasenübergang von einer lösbaren zu einer nicht lösbaren Phase sowie einen dynamischen Phasenübergang, der durch spontane Clusterung der Grundzustände gekennzeichnet ist [65–67].

In diesem Teil der Arbeit werden wir das *typische Laufzeitverhalten von Lösungsalgorithmen* für das K-SAT-Problem untersuchen. Die NP-Vollständigkeit des Erfüllbarkeitsproblems impliziert, dass bisher kein Algorithmus bekannt ist, dessen Rechenzeit im *Worst-Case* höchstens polynomial mit der Anzahl der Variablen steigt (vgl. Abschnitt 2.1). Betrachtet man aber die typische Rechenzeit für Ensembles zufälliger K-SAT-Formeln, so stellt man fest, dass in bestimmten Bereichen des Ensembles Lösungen sehr effizient, d. h. in linearer Rechenzeit gefunden werden, während in anderen Bereichen die typische Rechenzeit exponentiell steigt. Ähnliches Verhalten hatten wir bereits beim Vertex-Cover-Problem in Kapitel 4 beobachtet. Für *vollständige* Algorithmen, basierend auf Enumerationsverfahren wie in Abschnitt 2.2.1 beschrieben, konnte man diesen algorithmenabhängigen Phasenübergang teilweise verstehen (für eine Übersicht siehe [68]).

Hier soll nun ein unvollständiges Verfahren aus der Klasse der *stochastischen lokalen Suchalgorithmen* untersucht werden. Wie in Abschnitt 2.2.2 beschrieben, durchsuchen derartige Algorithmen nicht den gesamten Lösungsraum, sondern folgen einer stochastisch getriebenen Dynamik. Wir wollen das Laufzeitverhalten des *Walk-SAT*-Algorithmus verstehen, einem der bedeutendsten und effizientesten „SAT-Solver“ [69]. Dabei wollen wir besonders die folgenden Fragen betrachten: Wie findet

der Algorithmus zu einer Lösung? Warum gibt es Bereiche linearer und exponentieller Lösungszeit? Wie kann man die Parameter des Algorithmus wählen, damit er besonders effizient ist?

Nach einer Einführung des K-SAT-Problems (Abschnitt 5.1) und der Zusammenfassung bekannter Ergebnisse speziell zum Auftreten von Phasenübergängen und zur Struktur der Grundzustandslandschaft (Abschnitt 5.2) wird das Walk-SAT-Verfahren selbst beschrieben (Abschnitt 5.3). Anschließend zeigen wir im Abschnitt 5.4 numerische Ergebnisse der Anwendung des Algorithmus. Dabei werden wir zwei Bereiche mit unterschiedlicher typischer Laufzeit unterscheiden, abhängig von der Anzahl der Einschränkungen pro Variable: Während in einem Bereich eine Lösung typischerweise in linearer Rechenzeit gefunden wird, equilibriert im anderen Bereich der Algorithmus auf ein Plateau, von dem aus eine Lösung durch eine große, exponentiell seltene Fluktuation erreicht wird. Diese numerischen Beobachtungen werden – getrennt für die beiden Bereiche – in den Abschnitten 5.5 und 5.6 analytisch beschrieben.

5.1 Das Erfüllbarkeitsproblem

5.1.1 Boolesche Variablen und boolesche Ausdrücke

Eine boolesche Variable x ist eine Variable, die nur die Werte *wahr* oder *falsch* annehmen kann. Für diese Variablen definiert man den unären Operator „nicht“ ($\overline{x}$) sowie die binären Operatoren „oder“ ($x \vee y$), „und“ ($x \wedge y$) und „exklusiv-oder“ („XOR“, $x \oplus y = (x \vee y) \wedge (\overline{x} \vee \overline{y})$). Die Operatoren sind folgendermaßen erklärt:

x	y	$\overline{x}$	$x \vee y$	$x \wedge y$	$x \oplus y$
wahr	wahr	falsch	wahr	wahr	falsch
wahr	falsch		wahr	falsch	wahr
falsch	wahr	wahr	wahr	falsch	wahr
falsch	falsch		falsch	falsch	falsch

5.1.2 Definition des K-SAT-Problems

Gegeben seien N boolesche Variablen $x_1, x_2, \ldots, x_N$. Als *K-Klausel* C_α bezeichnen wir eine Verknüpfung von K Literalen C_{α_r} durch logisches „oder“:

$$C_\alpha = \bigvee_{r=1}^{K} C_{\alpha_r}, \tag{5.1}$$

wobei jedes Literal entweder eine Variable x_i oder ihre Negierung $\overline{x_i}$ ist. Die Anzahl K der Literale einer Klausel ist deren *Länge*. So ist zum Beispiel $(x_1 \vee x_2 \vee \overline{x_3} \vee x_4)$

eine Klausel der Länge vier, $(x_1 \vee \overline{x_1})$ eine Klausel der Länge zwei und (x_7) eine Klausel der Länge eins.

Eine *Formel* $F(\vec{x})$ ist eine beliebige Verknüpfung von Literalen durch „und“ oder „oder“, z. B.

$$F(\vec{x}) = (x_1 \vee x_2) \wedge (x_1 \vee (\overline{x_2} \wedge x_3)) \wedge \overline{x_1} \wedge (x_2 \vee (\overline{x_1} \wedge x_3)) \tag{5.2}$$

Die Formel F ist in *konjunktiver Normalform* (K-CNF), wenn sie als Verknüpfung von M Klauseln C_α der Länge K mit „und“ dargestellt ist, d. h. in der Form

$$F = \bigwedge_{\alpha=1}^{M} C_\alpha. \tag{5.3}$$

Ein Beispiel für eine 2-CNF-Formel ist

$$(x_1 \vee x_2) \wedge (\overline{x_4} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2). \tag{5.4}$$

Das K-Erfüllbarkeitsproblem können wir nun analog zur Definition des Vertex-Cover-Problems in Abschnitt 4.1 zunächst als *Entscheidungsproblem* formulieren:

(E2) Sei $F(\vec{x})$ eine K-CNF-Formel. Gibt es eine Belegung der Variablen $x_1, \ldots, x_n$, für die $F(\vec{x}) = \mathit{wahr}$ gilt?

Eine solche Belegung nennen wir *Lösung* der Formel. Eine Formel heißt *erfüllbar*, wenn sie mindestens eine Lösung besitzt. Da die Klauseln einer Formel durch „und“ verknüpft sind, ist das genau dann der Fall, wenn jede Klausel für sich erfüllt ist. Wenn eine nicht negiert in einer Klausel auftretende Variable den Wert „wahr“ hat, dann sagen wir, dass die Variable in dieser Klausel *korrekt gesetzt* ist. Das gleiche gilt für eine negiert auftretende Variable mit Wert „falsch“. Eine Klausel ist nur dann nicht erfüllt, wenn keine der K Variablen der Klausel korrekt gesetzt ist, d. h. alle nicht negierten Variablen den Wert „falsch“ und alle negierten den Wert „wahr“ haben. Obige Beispielformel (5.4) ist erfüllbar, denn mit $x_1 = \mathit{wahr}$, $x_2 = \mathit{wahr}$ ist der gesamte Ausdruck wahr. Nimmt man aber noch die Klausel $(\overline{x_1} \vee \overline{x_2})$ hinzu, so lässt sich keine Belegung der Variablen mehr finden, mit der die Formel zu einer wahren Aussage wird. Eine solche Formel nennen wir *nicht erfüllbar*.

Das K-Erfüllbarkeitsproblem lässt sich als verdünntes Spinglas-Problem formulieren [66]. Dazu definieren wir Wechselwirkungen

$$J_{\alpha_r,i} = \begin{cases} 1 & \text{wenn } C_{\alpha_r} = x_i \\ -1 & \text{wenn } C_{\alpha_r} = \overline{x_i} \\ 0 & \text{sonst} \end{cases} \tag{5.5}$$

und ersetzen die booleschen Variablen x_i durch Spins S_i

$$S_i = \begin{cases} 1 & \text{wenn } x_i = \mathit{wahr} \\ -1 & \text{wenn } x_i = \mathit{falsch}. \end{cases} \tag{5.6}$$

In der Hamiltonfunktion

$$E(\vec{x}) = E(\vec{S}) = \sum_{\alpha=1}^{M} \left(\prod_{r=1}^{K} \frac{1 - \sum_{i=1}^{N} J_{\alpha_r,i} S_i}{2} \right) \tag{5.7}$$

beschreibt jeder Summand eine Plakette aus K wechselwirkenden Spins (entsprechend einer Klausel) und liefert genau dann einen Beitrag 1 zur Summe, wenn alle Spins gleichzeitig entgegen der durch die Klausel vorgegebenen Ausrichtung zeigen. Die Hamiltonfunktion zählt somit die Anzahl der nicht erfüllten Klauseln bei der zu $\vec{S}$ korrespondierenden Belegung der Variablen $\vec{x}$.

Mit Hilfe von (5.7) können wir das Erfüllbarkeitsproblem als Optimierungsproblem formulieren:

(O3) Sei $F(\vec{x})$ eine K-CNF-Formel.
Man bestimme die Grundzustandsenergie $\min_{\vec{x}} E(\vec{x})$.

Ist $\min_{\vec{x}} E(\vec{x}) = 0$, so sind die Grundzustände Lösungen der Formel und die Formel ist erfüllbar.

5.1.3 Worst-Case-Komplexität

K-SAT kann für $K = 1$ in linearer Zeit gelöst werden: Jede Klausel legt unmittelbar eine Variable fest. Gibt es mindestens eine Variable, die sowohl negiert als auch nicht negiert vorkommt, so ist die Formel nicht erfüllbar. Denn egal, ob man diese Variable mit wahr oder falsch belegt, es wird immer eine Klausel verletzt. 1-SAT gehört also zur Klasse P.

Auch für 2-SAT existiert ein polynomialer Lösungsalgorithmus [4]. Die Grundidee ist, dass in einer Klausel $(x_i \vee x_j)$ durch Festlegen des Wertes von x_i die Klausel entweder erfüllt ist (und deshalb nicht mehr berücksichtigt werden muss) oder den Wert von x_j festlegt. Auf diese Weise erhält man eine Kette von Implikationen. Führt für eine Variable x_i sowohl die Annahme $x_i = \textit{wahr}$ als auch $x_i = \textit{falsch}$ zu einem Widerspruch, so ist die Formel nicht erfüllbar.

Das Worst-Case-Laufzeitverhalten ändert sich für $K \geq 3$: Alle bisher bekannten Algorithmen benötigen eine Rechenzeit, die im Worst-Case exponentiell mit der Anzahl der Variablen N steigt. K-SAT ist für $K \geq 3$ ebenso wie Vertex-Cover NP-vollständig (vgl. Abschnitt 2.1) [70].

Da damit K-SAT für alle $K \geq 3$ aus Sicht der Komplexitätstheorie äquivalente Probleme sind, beschäftigt man sich oftmals hauptsächlich mit dem Fall $K = 3$.

Der naive Branch-and-Bound-Algorithmus (vgl. Abschnitt 2.2.1) lässt sich auch zum Ermitteln von Lösungen des SAT-Problems verwenden. Durch eine geschicktere Wahl der Reihenfolge der Festlegung der Variablenwerte verbessert sich seine Laufzeit für $K = 3$ von $\mathcal{O}(2^N)$ auf $\mathcal{O}(1.619^N)$ [71].

Der in diesem Teil der Arbeit analysierte Walk-SAT-Algorithmus kann so modifiziert werden, dass die Laufzeit im Worst-Case sogar nur wie $\mathcal{O}(1.324^N)$ steigt. Dies ist die derzeit beste bekannte Schranke [72, 73]. Wir werden auf diesen Algorithmus im Abschnitt 5.3 noch etwas genauer eingehen.

5.2 Das Ensemble zufälliger K-SAT-Formeln

5.2.1 SAT-UNSAT-Phasenübergang

In diesem Abschnitt werden wir das Ensemble zufälliger K-SAT-Formeln betrachten. Das Phasenverhalten und die Grundzustandsstruktur wurden für dieses Ensemble bereits sehr ausführlich mit Methoden der Statistischen Physik untersucht [67, 68]. Einige bekannte Ergebnisse werden in diesem Abschnitt näher beschrieben. Dabei werden wir Parallelen zu den typischen Eigenschaften des Vertex-Cover-Problems im Ensemble der Zufallsgraphen finden (vgl. Abschnitt 4.4.1).

Eine zufällige K-SAT-Formel mit N Variablen und M Klauseln wird erzeugt, indem für jede Klausel zufällig K verschiedene Variablen gezogen werden und anschließend jede Variable mit Wahrscheinlichkeit 1/2 nicht negiert oder negiert zur Klausel hinzugefügt wird.

Man definiert

$$\alpha := \frac{M}{N} \tag{5.8}$$

als das Verhältnis der Anzahl M der Klauseln zur Anzahl N der Variablen. Dieser Parameter wird eine ähnliche Rolle spielen wie die Konnektivität c und auch wie der Anteil der bedeckten Knoten x im VC-Problem.

Die Struktur von Zufallsformeln ist ähnlich der Struktur von Zufallsgraphen aus Abschnitt 4.2. Da die K Variablen in jeder der M Klauseln unabhängig zufällig ausgewählt werden, ist die Anzahl der Klauseln $d(x_i)$, in denen eine Variable x_i (oder ihre Negation) auftritt, poissonverteilt mit Mittelwert $K \cdot M/N = K\alpha$. Wir bezeichnen die Variablen, die gemeinsam mit x_i bzw. $\overline{x_i}$ in einer Klausel auftreten, als die *Nachbarn* von x_i. Eine Variable hat somit im Mittel $K \cdot (K-1)\alpha$ Nachbarn. Diese sind mit Wahrscheinlichkeit 1 voneinander verschieden, da Zufallsformeln genau wie Zufallsgraphen lokal baumartig sind (vgl. Abschnitt 4.2).

In einer Zufallsformel mit kleinem α gibt es nur sehr wenige Bedingungen pro Variable, fast jede Belegung der $\vec{x}$ wird die Formel erfüllen. Gibt es dagegen sehr viele Klauseln, dann sind die $\vec{x}$ so stark eingeschränkt, dass keine Lösung mehr existiert (im Extremfall könnten beispielsweise für K Variablen alle 2^K möglichen Klauseln auftreten). Für die Wahrscheinlichkeit $p(\alpha, K, N)$, dass eine zufällige K-SAT-Formel

mit N Variablen und αN Klauseln erfüllbar ist, gilt somit hinsichtlich der Abhängigkeit von α: $p(0, K, N) = 1$, $\lim_{\alpha\to\infty} p(\alpha, K, N) = 0$ und $p(\alpha, K, N)$ ist monoton fallend mit steigendem α.

Numerisch findet man für $p(\alpha, K, N)$ einen Übergang von $p(\alpha, K, N) = 1$ zu $p(\alpha, K, N) = 0$, der mit wachsender Variablenanzahl schärfer wird [74]. Man findet einen kritischen Wert $\alpha_c(K)$ so, dass für $p(\alpha, K) = \lim_{N\to\infty} p(\alpha, K, N)$

$$p(\alpha, K) = \begin{cases} 1 & \text{für } \alpha < \alpha_c(K) \qquad (\textit{SAT-Phase}) \\ 0 & \text{für } \alpha > \alpha_c(K) \qquad (\textit{UNSAT-Phase}) \end{cases} \tag{5.9}$$

gilt. Man beachte die Ähnlichkeit mit (4.10).

Die Erfüllbarkeit einer Formel hängt also im thermodynamischen Limes $N \to \infty$ fast nicht mehr von der konkreten Wahl der Klauseln ab, sondern nur noch vom Wert des Parameters α. Analytisch konnte für das K-SAT-Problem die Existenz eines solchen SAT/UNSAT-Phasenübergangs für große N gezeigt werden [75], allerdings lässt der Beweis offen, ob der kritische Wert α_c tatsächlich unabhängig von N ist.

Aus algorithmischer Sicht ist der Phasenübergang interessant, weil – wie beim VC-Problem – die Rechenzeit nahe α_c ihr Maximum besitzt (vgl. Abb. 5.1 für einen Branch-and-Bound-Algorithmus im Fall $K = 3$) [76].

Bei genauerer Betrachtung erkennt man drei Bereiche [68]:

- Unterhalb einer (algorithmenabhängigen) Schranke α_w ist die Formel lösbar und die Lösungen sind leicht zu finden, da es nur sehr wenige Klauseln im Verhältnis zur Anzahl der Variablen gibt. Ein Algorithmus, der jede vorkommende Variable so belegt, dass eine beliebige Klausel, in der die Variable auftritt, erfüllt ist, wird fast nie auf einen Widerspruch stoßen. Schon der erste Abstieg im Entscheidungsbaum (Abb. 2.1) führt fast sicher zu einer Lösung, diese wird deshalb in annähernd linearer Zeit gefunden und die Kurven für $\langle \ln(t/N) \rangle$ liegen in diesem Bereich übereinander.

- Für große Werte $\alpha > \alpha_c$ sind die Formeln nicht erfüllbar. Um dies festzustellen, muss der gesamte Entscheidungsbaum durchsucht werden, die Suche dauert deshalb exponentiell lange. Mit wachsendem α wird der Algorithmus Widersprüche näher an der Wurzel finden. Das Durchlaufen des Suchbaums kann also eher abgebrochen werden, so dass der Exponent der Rechenzeit abnimmt.

- Im Bereich $\alpha_w < \alpha < \alpha_c$ existieren Lösungen, diese sind aber schwer zu finden. Die Formeln sind *kritisch eingeschränkt*, so dass ein großer Teil des Entscheidungsbaums durchsucht werden muss, ehe ein zu einer Lösung gehörendes Blatt erreicht wird. Die Rechenzeit steigt deshalb in diesem Bereich ebenfalls exponentiell mit N.

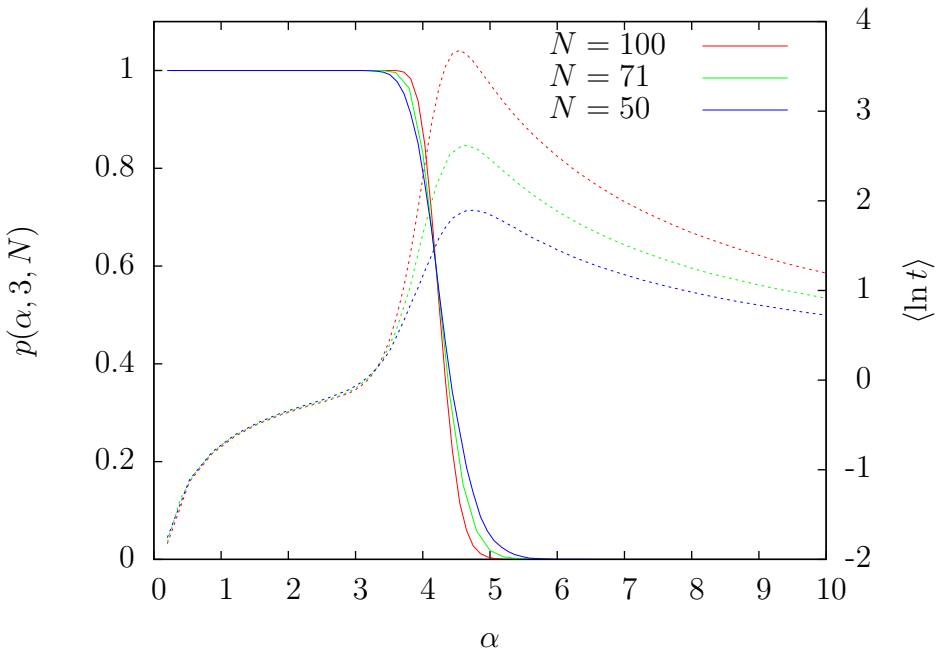


Abbildung 5.1: Ensemble zufälliger 3-SAT-Formeln: Anteil lösbarer Probleme $p(\alpha, 3, N)$ (durchgezogene Linien) und mittlere Rechenzeit eines Branch-and-Bound-Algorithmus (gestrichelte Linien) für verschiedene Systemgrößen N. Die Rechenzeit hat ein Maximum nahe des SAT/UNSAT-Übergangs bei $\alpha_c \simeq 4.267$.

5.2.2 Grundzustandsstruktur von K-SAT

Da sich das K-SAT-Problem wie in Abschnitt 5.1.2 gesehen als Spinglas-Problem formulieren lässt, wurden zum Verständnis des SAT/UNSAT-Phasenübergangs die dort bekannten Methoden [77], insbesondere zunächst die Replika-Theorie (vgl. Abschnitt 4.4.3), angewendet [66]. Dabei konnten Veränderungen der Grundzustandslandschaft der Hamiltonfunktion (5.7) gefunden werden (siehe Abbildung 5.2), welche mitverantwortlich für das im vorherigen Abschnitt 5.2.1 beschriebene algorithmische Verhalten *einfach–hart–einfach* sein könnten.

Für $K \geq 3$ gibt es in der SAT-Phase bei $\alpha_d < \alpha_c$ einen dynamischen Phasenübergang [78]. Für $\alpha < \alpha_d$ ist das System replikasymmetrisch, alle Grundzustände gehören zu einem einzigen Cluster. Im Bereich $\alpha_d < \alpha < \alpha_c$ zerfällt der Grundzustandsraum in eine exponentielle Anzahl von Clustern mit Hammingabstand $\mathcal{O}(N)$. Innerhalb dieses Intervalls ist die zugehörige 1-RSB-Lösung für $\alpha_s < \alpha < \alpha_c$ lokal stabil. Im unteren Bereich $\alpha_d < \alpha < \alpha_s$ sind die Cluster selbst wieder in Familien organisiert [41], mit typischen Abständen zwischen Clustern innerhalb einer Familie einerseits bzw. zwischen Clustern aus zwei verschieden Familien andererseits. Analytisch wird die 1-RSB-Lösung instabil gegenüber 2-RSB. Die genaue Grund-

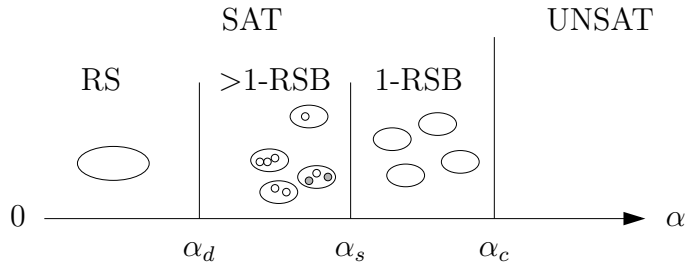


Abbildung 5.2: Qualitatives Phasendiagramm des Ensembles zufälliger K-SAT-Formeln (nach [67])

zustandsstruktur in diesem Bereich ist – ähnlich wie bei VC für $c > e$ – unklar, denkbar sind insbesondere kontinuierliche RSB verbunden mit einer hierarchischen Clusterstruktur der Grundzustände.

Aus algorithmischer Sicht ist bedeutsam, dass gleichzeitig mit dem Auftreten einer exponentiellen Anzahl von Grundzustandsclustern im Intervall (α_d, α_c) auch eine exponentielle Anzahl *metastabiler* Cluster auftritt. Zustände $\vec{x}$ innerhalb dieser Cluster besitzen Energien $E(\vec{x}) = D > 0$ und kein Zustand in der Nähe des Clusters (d. h. mit $\mathcal{O}(1)$ Hammingabstand) hat eine Energie kleiner als D. Dabei unterscheidet sich der Bereich (α_d, α_s) noch vom Bereich (α_s, α_c) in den Energien D, bis zu denen man solche metastabilen Cluster findet [41].

Diese lokalen Minima der Energielandschaft führen möglicherweise dazu, dass lokale Suchalgorithmen lange Zeit in diesen metastabilen Clustern verweilen und deshalb die Rechenzeit oberhalb von α_d bzw. α_s stark ansteigt. α_d oder α_s könnten also für lokale Algorithmen eine algorithmenunabhängige obere Schranke für die im vorherigen Abschnitt eingeführte algorithmenabhängige Schranke α_w des Beginns exponentiell steigender Rechenzeiten darstellen.

Im Stabilitätsbereich der 1-RSB-Lösung kann man mit Hilfe des *Cavity-Ansatzes* [55, 56] eine selbstkonsistente Lösung finden, aus der man (nicht-rigoros) Werte für α_d, α_s und α_c gewinnt [79]. Für 3-SAT ergeben sich $\alpha_d = 3.925 \pm 0.004$, $\alpha_s = 4.15$ und $\alpha_c = 4.267$. Umgekehrt lassen sich damit nicht nur Aussagen über typische Eigenschaften von zufälligen K-SAT-Formeln treffen, sondern auch für einzelne Instanzen Grundzustände bestimmen. Dieser *Survey-Propagation* genannte Algorithmus kann 3-SAT-Formeln bis zu $N = 10^7$ noch nahe des SAT/UNSAT-Übergangs lösen [55, 80].

5.2.3 Das K-XOR-SAT-Problem

Verknüpft man die Literale innerhalb einer Klausel statt mit „oder“ durch „exklusiv-oder“ ($\oplus$), so erhält man Instanzen des K-XOR-SAT-Problems.

Aus Sicht der Komplexitätstheorie ist dieses Problem *einfach*, es gehört zur Klasse P: Die $\oplus$-Verknüpfung entspricht einer Addition von 0/1-Variablen modulo 2. Jede Klausel ist somit äquivalent zu einer linearen Gleichung (modulo 2) und die gesamte Formel ist äquivalent zu einem System von $M = \alpha N$ linearen Gleichungen. Dieses lässt sich in $\mathcal{O}(N^3)$ Schritten, also in polynomialer Zeit lösen.

Es gibt damit zwar einen effizienten globalen Algorithmus, die lokale Struktur ist aber ähnlich zu K-SAT. Dies zeigt sich auch im Phasendiagramm der Grundzustandslandschaft: Man findet einen dynamischen Phasenübergang, bei dem die Menge der Lösungen spontan in eine exponentielle Anzahl von Clustern zerfällt, sowie einen SAT/UNSAT-Übergang [58, 81].

Auf Grund dieser gemeinsamen lokalen Eigenschaften sollten auch die hier zu untersuchenden lokalen Algorithmen ähnliches Verhalten für die beiden Modelle zeigen. Da beim K-XOR-SAT-Problem die Änderung des Wertes einer Variablen immer auch den Wert aller Klauseln ändert, die diese Variable enthalten, ist das dynamische Verhalten von Algorithmen einfacher zu behandeln. Wir werden deshalb die analytischen Methoden zunächst auf K-XOR-SAT anwenden und anschließend auf K-SAT übertragen.

5.3 Walk-SAT – ein stochastischer lokaler Suchalgorithmus

Vollständige Suchalgorithmen wie der in 2.2.1 beschriebene Branch-and-Bound-Algorithmus durchsuchen systematisch den gesamten Phasenraum nach einer Lösung. *Stochastische lokale Suchalgorithmen* dagegen starten in einem (zumindest teilweise) zufällig gewählten Zustand $\vec{x}$ und folgen einer Dynamik durch den Phasenraum. Die Konsequenzen für nicht lösbare Instanzen wurden bereits in Abschnitt 2.2.2 erläutert.

Für lösbare Instanzen des Erfüllbarkeitsproblems sind stochastische lokale Suchalgorithmen aber sehr erfolgreiche Lösungsverfahren, insbesondere das von Selman, Kautz und Cohen entwickelte *Walk-SAT* [69].

Hier werden wir die Dynamik einer einfachen Variante dieses Verfahrens analysieren, welche als Pseudocode in Algorithmus 5.1 angegeben ist.

Zu Beginn werden die N Variablen $\vec{x}$ zufällig belegt. In jedem Schritt wird zufällig eine noch nicht erfüllte Klausel C_α ausgewählt und aus dieser Klausel wiederum eine Variable x^* wie folgt:

- Mit Wahrscheinlichkeit q die in den wenigsten erfüllten Klauseln auftretende Variable aus C_α (*gieriger Schritt*),
- mit Wahrscheinlichkeit $1 - q$ zufällig (*zufälliger Schritt*).

algorithm walksat(C, q)
begin
  Belege die Variablen $\vec{x}$ zufällig mit *wahr* oder *falsch*
  while Formel ist nicht erfüllt
  begin
    Wähle zufällig eine verletzte Klausel $C_\alpha \in C$
    Mit Wahrscheinlichkeit q
      Wähle die Variable $x^* \in C_\alpha$, die in den wenigsten bisher erfüllten Klauseln auftritt
    else
      Wähle zufällig eine Variable $x^* \in C_\alpha$
    Negiere x^*, d. h. $x^* := \overline{x^*}$
  end
  print „Gefundene Lösung: $\vec{x}$“
end

Algorithmus 5.1: Der Walk-SAT-Algorithmus: C ist die Menge aller Klauseln, q die Wahrscheinlichkeit eines gierigen Schrittes.

Anschließend wird der Wert von x^* negiert. Dadurch werden alle bisher unerfüllten Klauseln, die x^* enthalten, erfüllt. Für K-XOR-SAT werden alle bisher erfüllten Klauseln unerfüllt, für K-SAT nur genau die erfüllten Klauseln, in denen x^* die einzige korrekt gesetzte Variable war. Die Prozedur wird so lange wiederholt, bis keine unerfüllte Klausel mehr existiert, also eine Lösung gefunden wurde. Für nicht erfüllbare Formeln terminiert der Algorithmus nie.

Bei geeigneter Wahl der Datenstrukturen lässt sich eine noch nicht erfüllte Klausel in $\mathcal{O}(1)$ finden. Auch die Auswahl der Variablen im gierigen Schritt kann in Zeit $\mathcal{O}(1)$ erfolgen: Die für jede Variable benötigten Größen sind die Anzahlen der erfüllten und nicht erfüllten Klauseln, in denen sie nicht negiert bzw. negiert vorkommt. Beim Negieren einer Variable x^* ändern sich diese Anzahlen für die Variable selbst und für die mit ihr verbundenen, d. h. zusammen mit x^* in einer Klausel auftretenden Variablen. Um diese Anzahlen nach Negieren von x^* zu berechnen, muss man alle Klauseln betrachten, in denen x^* vorkommt. Dies sind im Mittel $K\alpha$, man benötigt also nur $\mathcal{O}(1)$ Zeit. Die Dauer eines Walk-SAT-Schrittes ist somit unabhängig von N.

Bei einer wichtigen Modifikation des Walk-SAT-Algorithmus wird die Iteration bereits abgebrochen, wenn nach einer bestimmten Anzahl t_r von Schritten noch keine Lösung gefunden wurde, um danach mit einer neuen zufälligen Belegung aller N Variablen zu beginnen (Walk-SAT mit *Restarts*). Schöning konnte zeigen, dass für einen Algorithmus mit $t_r = 3N$ die Rechenzeit im Worst-Case nur wie $(4/3)^N$ steigt [72]. Durch kleine Modifikationen des Algorithmus bzw. genauere Abschätzungen lässt sich diese Schranke noch geringfügig auf $\mathcal{O}(1.324^N)$ verbessern [73].

Im Abschnitt 5.6 werden wir das Verhalten dieser Variante mit Restarts noch etwas detaillierter untersuchen. In den Abschnitten davor wird „Walk-SAT“ immer „Walk-SAT ohne Restarts“ bedeuten.

5.4 Numerische Resultate des Laufzeitverhaltens von Walk-SAT

Wir beginnen zunächst mit einigen numerischen Ergebnissen zum Laufzeitverhalten des Walk-SAT-Algorithmus. Da diese für 3-XOR-SAT und 3-SAT qualitativ sehr ähnlich sind, beschränken wir uns hier auf die Ergebnisse für zufällige 3-SAT-Formeln und gehen nur bei Unterschieden gesondert auf 3-XOR-SAT ein.

Die Laufzeit t wird in Anzahl von *Monte-Carlo-Sweeps* (MC-Sweeps) gemessen. Ein MC-Sweep entspricht N Schritten des Walk-SAT-Algorithmus (Auswahl und Negierung einer Variablen x^*). Während eines MC-Sweeps wird jede Variable im Mittel einmal ausgewählt. Lineare Rechenzeit entspricht somit einer konstanten Anzahl von MC-Sweeps, während bei einer exponentiellen Anzahl von Walk-SAT-Schritten auch die Anzahl der MC-Sweeps exponentiell ist.

In Gleichung (5.7) haben wir als Energie $E(\vec{x})$ die Anzahl der durch die Belegung $\vec{x}$ verletzten Klauseln definiert. Hier betrachten wir den zeitlichen Verlauf der Energiedichte $\alpha_u(t) = E(\vec{x}(t))/N$, d. h. die Anzahl nicht erfüllter Klauseln pro Variable.

Zunächst untersuchen wir das typische Verhalten auf linearen Zeitskalen, also für eine endliche Anzahl von MC-Sweeps. Wir werden dabei annehmen, dass die Energiedichte selbstmittelnd ist, d. h. statt einer Mittelung über die Unordnung wenden wir den Algorithmus direkt auf jeweils eine große Formel mit $N = 50\,000$ Variablen an. Dabei konzentrieren wir uns zuerst auf den Fall $q = 0$, d. h. der Algorithmus führt nur Zufallsschritte aus und keine gierigen Schritte.

In Abbildung 5.3 ist $\alpha_u(t)$ für verschiedene Werte von α dargestellt. Nach dem zufälligen Initialisieren der Variablen sind bei 3-SAT nur die Klauseln nicht erfüllt, bei denen alle drei Variablen nicht korrekt gesetzt sind. Damit ist im Mittel

$$\alpha_u(t = 0) = \frac{1}{N} \cdot \frac{M}{2^3} = \frac{\alpha}{8}. \tag{5.10}$$

Bei K-XOR-SAT dagegen ist jede Klausel gleichwahrscheinlich erfüllt oder nicht erfüllt, dort ist dann $\alpha_u(t = 0) = \alpha/2$.

Für kleine Zeiten bis zu ca. 2 MC-Sweeps fällt die Energiedichte zunächst deutlich ab. Das weitere Verhalten ist abhängig von α:

- Für $\alpha < \alpha_w \simeq 2.7$ führt dieser Abfall nach einer endlichen Zeit t_f auf $\alpha_u(t_f) = 0$, d. h. Walk-SAT findet in linearer Zeit eine Lösung.

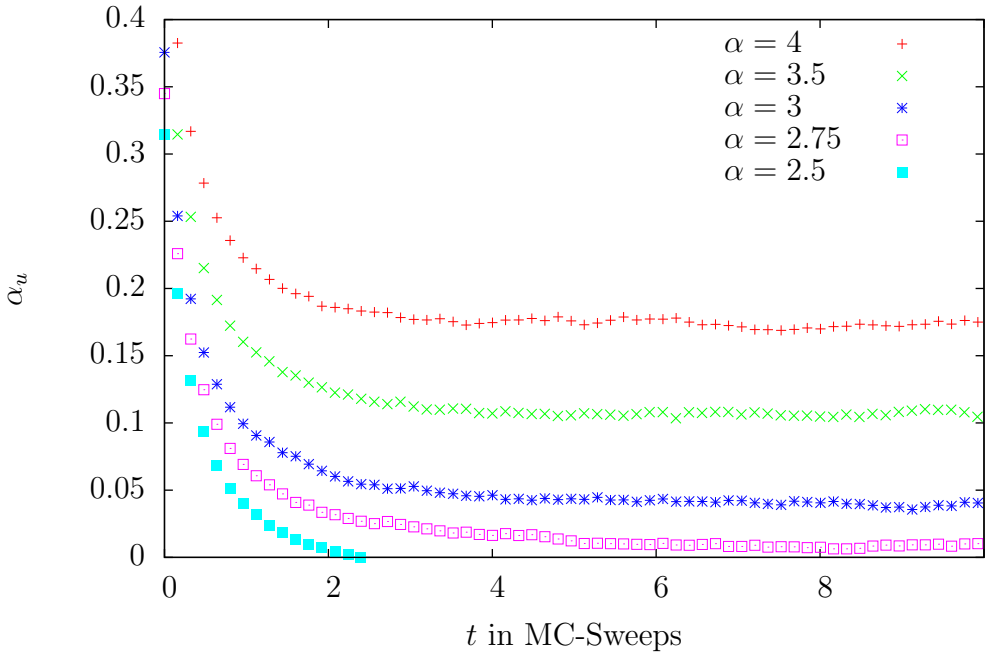


Abbildung 5.3: Zeitlicher Verlauf der Energiedichte, d. h. der mittleren Anzahl $\alpha_u(t)$ nicht erfüllter Klauseln pro Variable, für verschiedene Werte von α und je eine Formel mit $N = 50\,000$ Variablen. Für $\alpha < \alpha_w \simeq 2.7$ wird dabei eine Lösung gefunden (die Energiedichte erreicht in endlicher Zeit den Wert 0), für größere α bleibt die typische Energiedichte auf einem Plateauwert größer null.

- Für $\alpha > \alpha_w$ erreicht die Energiedichte nach kurzer Zeit einen Plateauwert größer null. Anschließend fluktuiert α_u um diesen Wert (siehe Abbildung 5.4). Die Zeitskala, auf der sich diese Fluktuationen abzeichnen, hängt von N ab. Für den dargestellten Bereich bis zu 160 MC-Sweeps erscheinen ausgeprägte Fluktuationen für $N = 140$. Wenn die Formel erfüllbar ist, kann schließlich eine dieser Fluktuationen so groß sein, dass $\alpha_u(t) = 0$ wird und damit eine Lösung der Formel gefunden wird.

Es gibt also eine dynamische Schwelle $a_w \simeq 2.7$, oberhalb derer Walk-SAT nach wenigen MC-Sweeps in ein Gleichgewicht der Energiedichte einläuft. Nur eine makroskopische Fluktuation führt dann schließlich zum Finden einer Lösung. Solche Fluktuationen treten aber nur mit exponentiell kleiner Wahrscheinlichkeit auf, deshalb steigt die Rechenzeit im Bereich $\alpha > 2.7$ exponentiell mit N.

Bei 3-XOR-SAT findet man qualitativ gleiches Verhalten. Die dynamische Schwelle und damit der Beginn exponentieller Rechenzeit liegt dort bei $\alpha_w \simeq 0.33$.

In Abbildung 5.5 dargestellt ist ein Histogramm des Logarithmus der Rechenzeit über 5000 zufällige 3-SAT-Formeln bei $\alpha = 3.5$. Für andere $\alpha > \alpha_w$ findet man

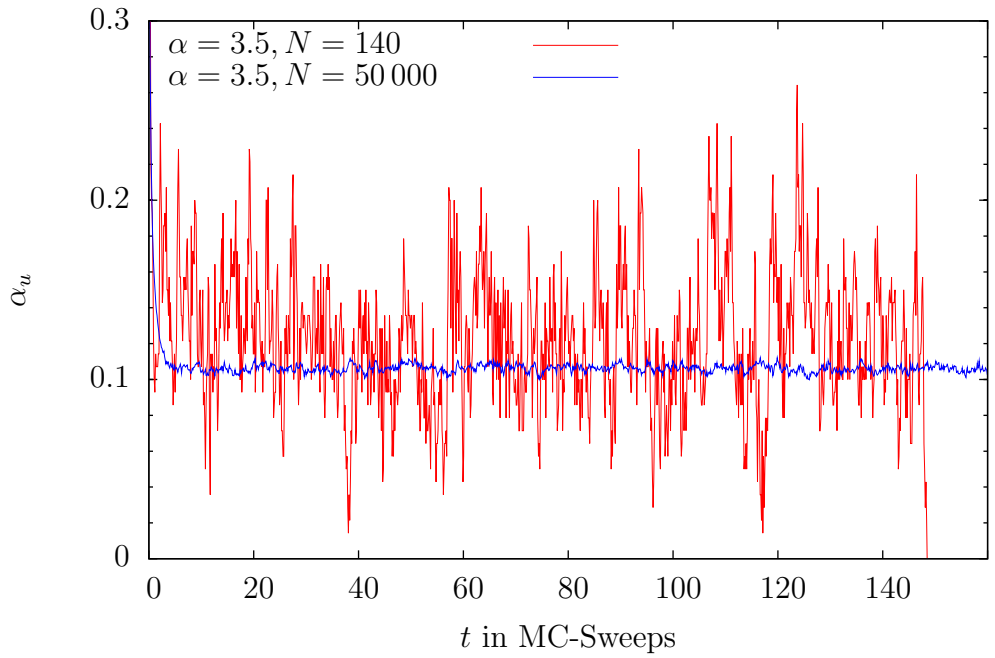


Abbildung 5.4: Fluktuation von α_u um den Plateauwert für zwei Formeln verschiedener Systemgrößen. Für das kleinere System mit $N = 140$ Variablen ist nach ca. 145 MC-Sweeps eine Fluktuation groß genug, um eine Lösung der Formel zu finden.

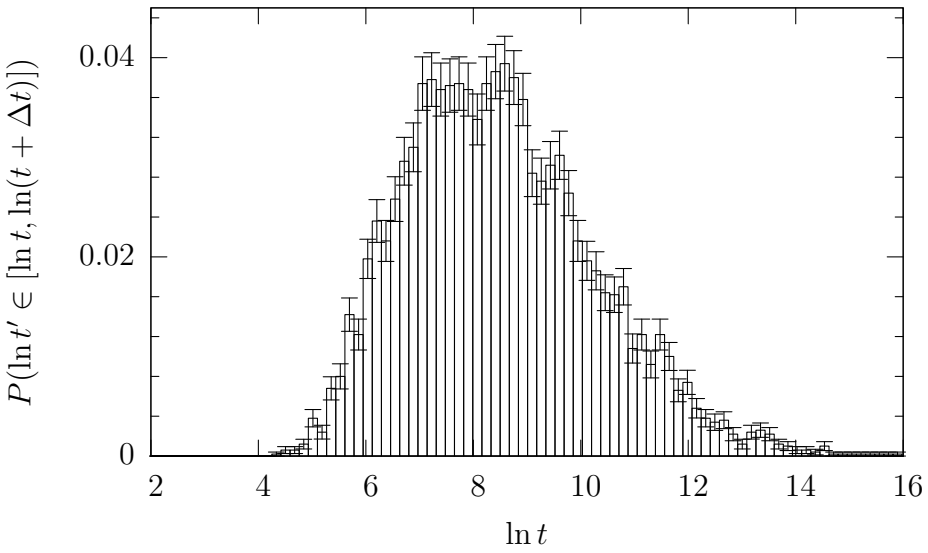


Abbildung 5.5: Histogramm des Logarithmus der Rechenzeit in MC-Sweeps für Walk-SAT über 5000 3-SAT-Formeln mit $\alpha = 3.5$ und $N = 100$.

qualitativ dasselbe Verhalten. Man erkennt, dass die Verteilung gut durch den Mittelwert des Logarithmus $\langle \ln t \rangle$ beschrieben werden kann. Dies entspricht hier einer Charakterisierung durch den Median der Rechenzeit, während der Mittelwert durch exponentiell seltene Formeln mit exponentiell längerer Rechenzeit dominiert würde. Dieser Median ist in Abbildung 5.6 in Abhängigkeit von α für verschiedene Systemgrößen dargestellt. Für $\alpha > \alpha_w$ steigt die Rechenzeit exponentiell mit N, während für kleinere α die Rechenzeit linear steigt, gekennzeichnet durch die Konvergenz der Kurven von $\langle \ln t \rangle$ für $N \to \infty$.

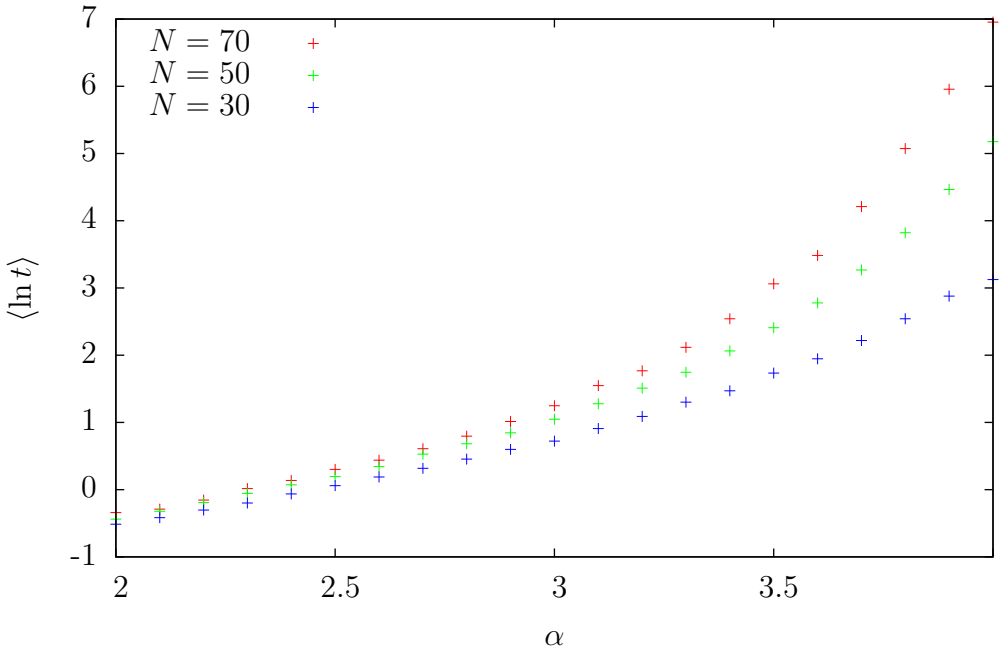


Abbildung 5.6: 3-SAT: Median der Rechenzeit des Walk-SAT-Algorithmus abhängig vom Verhältnis α von Klauseln zu Variablen. Man erkennt den Bereich linearer Rechenzeit für $\alpha < \alpha_w \simeq 2.7$; für größere α steigt die Rechenzeit exponentiell mit N.

Nahe liegend ist eine Verbesserung des Algorithmus durch Absenkung des Plateauwertes, um eine höhere Wahrscheinlichkeit für Fluktuation auf $\alpha_u(t) = 0$ zu erhalten. Dies kann zum Beispiel durch Hinzunahme von gierigen Schritten mit Wahrscheinlichkeit $q > 0$ geschehen. In Abbildung 5.7 ist die über 20 MC-Sweeps gemittelte Plateauenergie für je eine $N = 50\,000$-Instanz bei verschiedenen Werten von $\alpha > \alpha_w$ in Abhängigkeit des Anteils q gieriger Schritte dargestellt. Tatsächlich verringert sich die Plateauenergie signifikant, ihr Minimum ist bei großen Werten von q. Numerisch findet man auch ein leichtes Ansteigen der dynamischen Schwelle α_c selbst. Sie hat ihr Maximum bei $q \simeq 0.85$, dort können noch Formeln bis $\alpha \simeq 2.8$ in linearer Zeit gelöst werden.

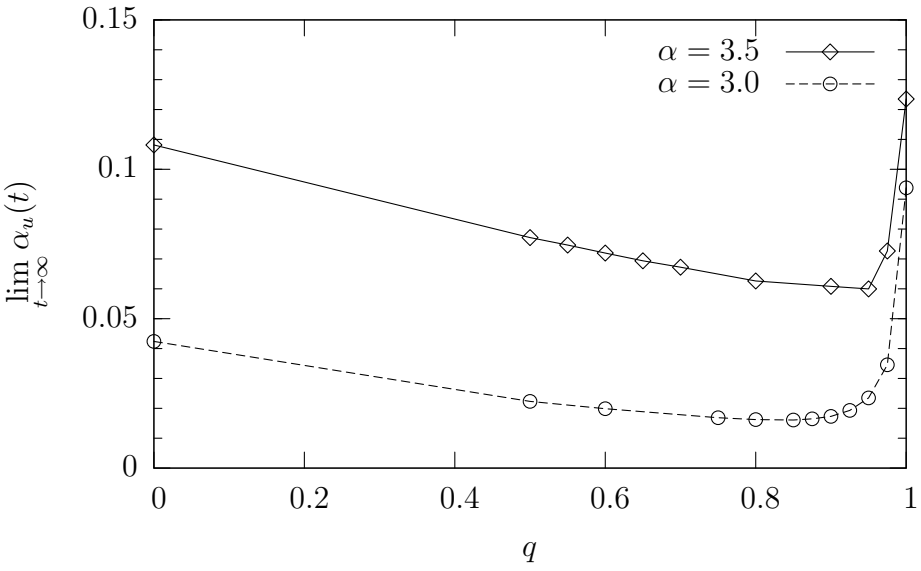


Abbildung 5.7: Abhängigkeit des Plateauwertes der Energiedichte vom Anteil q der gierigen Schritte. Man findet Minima der Energiedichte des Plateaus bei $q \simeq 0.95$ ($\alpha = 3.5$) bzw. $q \simeq 0.85$ ($\alpha = 3.0$).

5.5 Analyse des Laufzeitverhaltens im Bereich linearer Rechenzeit

5.5.1 Auswahl der zu negierenden Variablen

Zunächst wollen wir das Verhalten des Algorithmus auf linearen Zeitskalen verstehen. Dazu charakterisieren wir den Zustand zur Zeit t durch die Anzahl $N_t(s,u)$ der Variablen, die zur Zeit t in s erfüllten und u unerfüllten Klauseln vorkommen. Für jede einzelne Variable bleibt die Summe $s+u$ konstant, ebenfalls nach Definition ist $\sum_{s=0}^{\infty}\sum_{u=0}^{\infty} N_t(s,u) \equiv \sum_{s,u} N_t(s,u) = N$. Die Anzahl der zur Zeit t nicht erfüllten Klauseln $N \cdot \alpha_u(t)$ erhalten wir durch

$$N \cdot \alpha_u(t) = \frac{\sum_{s,u} u \cdot N_t(s,u)}{K}, \tag{5.11}$$

da jede unerfüllte Klausel bei der Summierung K-fach gezählt wird.

Eine zufällig gewählte Variable tritt mit Wahrscheinlichkeit $p_t(s,u) = N_t(s,u)/N$ in s bzw. u Klauseln auf. Damit können wir die Energiedichte $\alpha_u(t)$ schreiben als

$$\alpha_u(t) = \frac{\sum_{s,u} u \cdot N_t(s,u)}{KN} = \frac{\sum_{s,u} u \cdot p_t(s,u)}{K} =: \frac{\langle u \rangle_t}{K}. \tag{5.12}$$

Wir bezeichnen dabei das Mittel über die Verteilung $p_t(s,u)$ zur Zeit t mit

$$\langle \cdot \rangle_t := \sum_{s,u} (\cdot) p_t(s,u). \tag{5.13}$$

Analog ergibt sich die mittlere Anzahl erfüllter Klauseln pro Variable $\alpha_s(t) = \langle s \rangle_t / K$ und tatsächlich ist $\alpha_u(t) + \alpha_s(t) = \sum_{s,u} (u+s) \cdot p_t(s,u)/K = \langle u+s \rangle_t / K = \alpha$, da $u+s$ für jede Variable konstant bleibt.

Wir werden für die analytische Untersuchung folgende Annahme verwenden: Die Wahrscheinlichkeit $p_t(s_1, u_1, s_2, u_2, \ldots, s_K, u_K)$, eine Klausel zu finden, deren r-te Variable in s_r bzw. u_r Klauseln auftritt, faktorisiert, es soll also

$$p_t(s_1, u_1, s_2, u_2, \ldots, s_K, u_K) = \prod_{r=1}^{K} p_t(s_r, u_r) \tag{5.14}$$

gelten. Dies ist exakt nur für $t = 0$ erfüllt, da durch die Dynamik im Verlauf des Algorithmus Korrelationen zwischen den Variablen aufgebaut werden. Wir werden aber sehen, dass die Annahme auch für $t > 0$ eine gute Näherung ist.

Walk-SAT wählt wie in Algorithmus 5.1 dargestellt zunächst eine unerfüllte Klausel C_α und daraus die zu negierende Variable v^* aus. Unter obiger Annahme hängt die Wahrscheinlichkeit $p_t^{(\text{flip})}(s,u)$, dass v^* in s bzw. u Klauseln auftritt, nicht von den anderen Variablen in C_α ab, sondern nur von $p_t(s,u)$.

Im zufälligen Schritt (*walk step*) wird v^* zufällig unter allen Variablen in C_α ausgewählt. Die Wahrscheinlichkeit $p_t^{(\text{flip-walk})}$ hängt damit nur von der Anzahl der unerfüllten Klauseln u ab, in denen v^* auftritt. Sie ist proportional zu dieser Anzahl und zur Anzahl der Variablen mit s bzw. u Klauseln überhaupt. Es gilt somit $p_t^{(\text{flip-walk})}(s,u) \sim u$ und $p_t^{(\text{flip-walk})}(s,u) \sim p_t(s,u)$, außerdem muss die Verteilung normiert sein. Damit erhalten wir mit (5.12)

$$p_t^{(\text{flip-walk})}(s,u) = \frac{u p_t(s,u)}{\langle u \rangle_t} =: p_t^{(u)}(s,u). \tag{5.15}$$

Im gierigen Schritt (*greedy step*) wird nur die Klausel C_α selbst zufällig bestimmt. Unter den K Variablen dieser Klausel wird diejenige als v^* gewählt, die in den wenigsten erfüllten Klauseln auftritt. Treten mehrere gleich selten auf, dann wird unter diesen eine zufällig ausgesucht. Analog zum zufälligen Schritt ergibt sich, wieder mit der Annahme der Unabhängigkeit der Variablen, $p_t^{(u)}(s_r, u_r)$ als Wahrscheinlichkeit, dass die Variable an Position r dieser Klausel in s_r bzw u_r Klauseln auftritt.

Für $K = 2$ erhalten wir mit der Heavyside-Funktion

$$\Theta(x) = \begin{cases} 0 & \text{für } x < 0 \\ 1/2 & \text{für } x = 0 \\ 1 & \text{für } x > 0 \end{cases} \tag{5.16}$$

als Verteilung $p_t^{(\text{flip-2-greedy})}(s,u)$ der zu negierenden Variablen

$$p_t^{(\text{flip-2-greedy})}(s,u) = \sum_{s_1,u_1,s_2,u_2} p_t^{(u)}(s_1,u_1) p_t^{(u)}(s_2,u_2) \cdot \left[\delta_{(s_1,u_1),(s,u)} \cdot \Theta(s_2 - s_1) + \delta_{(s_2,u_2),(s,u)} \cdot \Theta(s_1 - s_2)\right]$$

$$
\begin{aligned}
&= 2 \sum_{s',u',s'',u''} p_t^{(u)}(s',u') p_t^{(u)}(s'',u'') \cdot \delta_{(s'',u''),(s,u)} \cdot \Theta(s'-s'') \\
&= 2 p_t^{(u)}(s,u) \sum_{s',u'} p_t^{(u)}(s',u') \Theta(s'-s) \\
&= 2 p_t^{(u)}(s,u) \sum_{u'=0}^{\infty} \left(-1/2 p_t^{(u)}(s,u') + \sum_{s'=s}^{\infty} p_t^{(u)}(s',u') \right)
\end{aligned}
$$

und da $p_t^{(u)}(s,u)$ normiert ist

$$
\begin{aligned}
p_t^{(\text{flip-2-greedy})}(s,u) &= p_t^{(u)}(s,u) \left\{ \left(- \sum_{u'=0}^{\infty} p_t^{(u)}(s,u') \right) + 2 \left(1 - \sum_{u'=0}^{\infty} \sum_{s'=0}^{s-1} p_t^{(u)}(s',u') \right) \right\} \\
&= p_t^{(u)}(s,u) \left[2 - \sum_{u'=0}^{\infty} \left(p_t^{(u)}(s,u') + 2 \sum_{s'=0}^{s-1} p_t^{(u)}(s',u') \right) \right]. \qquad (5.17)
\end{aligned}
$$

Der Fall $K = 3$ kann ganz analog betrachtet werden. Zu beachten ist, dass, wenn alle Variablen der zufällig gewählten Klausel in gleich vielen erfüllten Klauseln enthalten sind ($s_1 = s_2 = s_3$), die Wahrscheinlichkeit für eine der drei Variablen, ausgewählt zu werden, nur 1/3 beträgt und nicht $(1/2)^2 = 1/4$. Mit dem Korrekturterm $\delta_{s,s'}\delta_{s,s''}/12$ erhalten wir

$$
\begin{aligned}
p_t^{(\text{flip-3-greedy})}(s,u) &= 3 p_t^{(u)}(s,u) \sum_{s',u',s'',u''} p_t^{(u)}(s',u') p_t^{(u)}(s'',u'') \\
&\qquad \cdot \left[\Theta(s'-s)\Theta(s''-s) + \frac{1}{12} \delta_{s,s'} \delta_{s,s''} \right] \\
&= 3 p_t^{(u)}(s,u) \left[1 - \sum_{u'=0}^{\infty} \left(1/2 p_t^{(u)}(s,u') + \sum_{s'=0}^{s-1} p_t^{(u)}(s',u') \right) \right]^2 \\
&\qquad + \frac{1}{4} p_t^{(u)}(s,u) \left[\sum_{u'=0}^{\infty} p_t^{(u)}(s,u') \right]^2 . \qquad (5.18)
\end{aligned}
$$

Walk-SAT führt mit Wahrscheinlichkeit q einen gierigen Schritt aus, mit Wahrscheinlichkeit $1-q$ einen zufälligen. Damit ist die Wahrscheinlichkeit $p^{(\text{flip})}(s,u)$ als Linearkombination der beiden Fälle durch

$$
p_t^{(\text{flip})}(s,u) = (1-q) p_t^{(\text{flip-walk})}(s,u) + q p_t^{(\text{flip-K-greedy})}(s,u) \qquad (5.19)
$$

gegeben.

5.5.2 Poisson-Abschätzung für den Algorithmus ohne gierige Schritte

Zunächst möchten wir eine Abschätzung für (5.15) im einfacheren Fall $q = 0$ gewinnen, wenn der Algorithmus nur zufällige Schritte ausführt. Dazu nehmen wir an,

dass s und u für alle Zeiten unabhängig poissonverteilt mit Mittelwert $K\alpha_s(t)$ bzw. $K\alpha_u(t)$ sind, d. h.

$$p_t(s,u) = e^{-K\alpha}\frac{(K\alpha_s(t))^s(K\alpha_u(t))^u}{s!u!}. \tag{5.20}$$

Wieder ist diese Näherung nur korrekt in der Ausgangskonfiguration bei $t = 0$, in der die Werte der Variablen unabhängig zufällig belegt sind. Eingesetzt in (5.15), ergibt sich mit Gleichung (5.12) für $u \geq 1$

$$p_t^{(\text{flip-walk})}(s,u) = \frac{ue^{-K\alpha}(K\alpha_s(t))^s(K\alpha_u(t))^u}{K\alpha_u(t)s!u!} = e^{-K\alpha}\frac{(K\alpha_s(t))^s(K\alpha_u(t))^{u-1}}{s!(u-1)!}, \tag{5.21}$$

also erneut ein Produkt von Poissonverteilungen von s und $(u-1)$. Eine zufällig aus einer unerfüllten Klausel ausgewählte Variable v^* tritt unter dieser Näherung somit im Mittel in $K\alpha_s(t)$ erfüllten und $K\alpha_u(t)+1$ unerfüllten Klauseln auf. Mit Hilfe dieser Mittelwerte können wir nun zunächst für K-XOR-SAT und dann für K-SAT eine erste Abschätzung des Laufzeitverhaltens und des dynamischen Phasenübergangs bei α_w geben.

Abschätzung für zufälliges K-XOR-SAT

Durch Negieren der Variablen v^* werden bei K-XOR-SAT alle s erfüllten Klauseln unerfüllt und alle u unerfüllten Klauseln erfüllt. Mit den Ergebnissen der vorhergehenden Betrachtung für die Mittelwerte von s und u einer zufällig gewählten Variablen v^* werden im Mittel also $K\alpha_s(t)$ Klauseln neu unerfüllt, während $K\alpha_u(t)+1$ vorher unerfüllte Klauseln nunmehr erfüllt werden. Der Erwartungswert der Anzahl unerfüllter Klauseln $N_t^{(u)}$ ändert sich damit während eines Schrittes des Algorithmus um

$$\Delta N_t^{(u)} = K\alpha_s(t) - (K\alpha_u(t)+1) = K(\alpha - \alpha_u(t)) - K\alpha_u(t) - 1 = K\alpha - 2K\alpha_u(t) - 1. \tag{5.22}$$

Dieser Erwartungswert bestimmt die *typische Dynamik* von $\alpha_u(t) = N_t^{(u)}/N$, welcher der Algorithmus im thermodynamischen Limes $N \to \infty$ mit Wahrscheinlichkeit 1 folgt.

Wir messen die Zeit t in MC-Sweeps, so dass die Zeitdauer eines Schrittes des Algorithmus $\Delta t = 1/N$ beträgt. Damit lässt sich die linke Seite in (5.22) schreiben als

$$\Delta N_t^{(u)} = \Delta(N\alpha_u(t)) = N\Delta\alpha_u(t) = \frac{\Delta\alpha_u(t)}{\Delta t}. \tag{5.23}$$

Für $N \gg 1$ können wir zum Differentialquotienten übergehen und erhalten eingesetzt in (5.22) eine einfache Differentialgleichung für die Änderung der Energiedichte

$$\dot{\alpha}_u(t) = K\alpha - 2K\alpha_u(t) - 1. \tag{5.24}$$

Diese Differentialgleichung lässt sich leicht durch Trennung der Variablen lösen. Wir erhalten als allgemeine Lösung

$$\alpha_u(t) = \frac{1}{2K}\left(K\alpha - 1 + Ce^{-2Kt}\right). \tag{5.25}$$

Bei der Initialisierung der Variablen zur Zeit $t = 0$ werden die Werte der Variablen unabhängig voneinander zufällig gesetzt. Der Wert einer Klausel wird erst durch den Wert der K-ten Variablen festgesetzt, mit Wahrscheinlichkeit $1/2$ wird sie erfüllt bzw. unerfüllt. Im Mittel erwartet man also zu Beginn $\alpha_u(0) = \frac{M/2}{N} = \alpha/2$ unerfüllte Klauseln pro Variable. Mit dieser Anfangsbedingung erhalten wir als Resultat

$$\alpha_u(t) = \frac{1}{2K}\left(K\alpha - 1 + e^{-2Kt}\right). \tag{5.26}$$

In Abbildung 5.8 auf Seite 91 ist Gleichung (5.26) für verschiedene Werte von α dargestellt (gestrichelte Linien). Im Vergleich mit den numerischen Ergebnissen (Kreuzsymbole) sieht man eine sehr gute Übereinstimmung für kleine Zeiten. Der Abfall der Energiedichte zu Beginn wird also sehr gut durch die Näherung beschrieben. Für größere Zeiten erreicht der Algorithmus einen etwas niedrigeren Plateauwert der Energiedichte, als man unter Annahme der Poissonverteilung erwarten würde. Die Abweichungen des Plateauwertes sind dabei bei größerem α etwas geringer ($1.5 \cdot 10^{-2}$ bei $\alpha = 0.5$ gegenüber $1.15 \cdot 10^{-2}$ bei $\alpha = 1.5$).

Gleichung (5.26) erlaubt auch eine Abschätzung der dynamischen Schwelle α_w zwischen linearer und exponentieller Lösungszeit. Diese ergibt sich aus der Bedingung

$$0 = \lim_{t\to\infty} \frac{1}{2K}\left(K\alpha_w - 1 + e^{-2Kt}\right) \tag{5.27}$$

zu

$$\alpha_w = \frac{1}{K}. \tag{5.28}$$

Für $\alpha < \alpha_w$ fällt die Energiedichte bereits nach wenigen MC-Sweeps auf den Wert 0 ab, während sie sich für $\alpha > \alpha_w$ nach einiger Zeit auf einem endlichen Plateauwert stabilisiert.

Speziell für $K = 3$ erhalten wir $\alpha_w = 1/3$, was ziemlich genau mit dem in Abschnitt 5.4 angegebenen numerischen Wert übereinstimmt.

Abschätzung für zufälliges K-SAT

Die Rechnung für zufälliges K-SAT verläuft analog zu K-XOR-SAT. Durch Negieren von v^* werden die u vorher unerfüllten Klauseln erfüllt. Umgekehrt werden aber von den vorher erfüllten Klauseln nur die unerfüllt, in denen v^* die einzige korrekt gesetzte Variable ist. Dies ist der wesentliche Unterschied zu K-XOR-SAT.

Für die Belegung der K Variablen einer Klausel C_α gibt es 2^K Möglichkeiten. Bei genau einer Belegung ist keine Variable korrekt gesetzt und die Formel ist unerfüllt, es gibt also $2^K - 1$ Belegungen, bei denen C_α erfüllt ist. Von diesen ist genau in einer Belegung v^* die einzige korrekt gesetzte Variable. Wenn wir Unabhängigkeit der Klauseln annehmen, dann wird somit jede der s erfüllten Klauseln mit Wahrscheinlichkeit

$$\mu = 1/(2^K - 1) \tag{5.29}$$

unerfüllt. Damit erhalten wir für die Änderung des Erwartungswerts der Anzahl unerfüllter Klauseln

$$\Delta N_t^{(u)} = \frac{1}{2^K - 1} K\alpha_s(t) - (K\alpha_u(t) + 1) = \frac{K\alpha}{2^K - 1} - \frac{2^K K}{2^K - 1}\alpha_u(t) - 1. \tag{5.30}$$

Wieder für $N \to \infty$ zu kontinuierlichen Zeiten übergehend erhalten wir die Differentialgleichung

$$\dot{\alpha}_u(t) = \frac{K\alpha}{2^K - 1} - \frac{2^K K}{2^K - 1}\alpha_u(t) - 1, \tag{5.31}$$

mit allgemeiner Lösung analog zu (5.25)

$$\begin{aligned} \alpha_u(t) &= \frac{2^K - 1}{2^K K}\left\{\frac{K\alpha}{2^K - 1} - 1 + C\exp\left(-\frac{2^K K}{2^K - 1}\,t\right)\right\} \\ &= \frac{\alpha}{2^K} + \frac{2^K - 1}{2^K K}\left\{-1 + C\exp\left(-\frac{2^K K}{2^K - 1}\,t\right)\right\}. \end{aligned} \tag{5.32}$$

In der typischen Anfangskonfiguration sind $\alpha_u(0) = \alpha/2^K$ Klauseln pro Variable verletzt, damit ergibt sich wieder $C = 1$ und somit als Resultat für das zeitliche Verhalten der Energiedichte

$$\alpha_u(t) = \frac{\alpha}{2^K} + \frac{2^K - 1}{2^K K}\left\{-1 + \exp\left(-\frac{2^K K}{2^K - 1}\,t\right)\right\}. \tag{5.33}$$

Der Verlauf ist für verschiedene Werte von α in Abbildung 5.10 auf Seite 95 dargestellt (gestrichelte Linien). Im oberen Teil sieht man im Vergleich mit der Simulation (Kreuzsymbole), dass für kleine Zeiten die Annahme der Poissonverteilung eine sehr gute Approximation liefert. Die Plateauenergie

$$\lim_{t\to\infty} \alpha_u(t) = \frac{K\alpha - 2^K + 1}{2^K K} \tag{5.34}$$

wird aber deutlich zu groß vorausgesagt, deshalb sind im unteren Teil von Abb. 5.10 auf Seite 95 zur besseren Übersicht auch nur die Kurven für $\alpha = 2.85$, $\alpha = 3.5$ und $\alpha = 4.0$ angegeben. Die sich als Nullstelle von (5.34) ergebende dynamische Schwelle α_w wird folglich zu klein vorausgesagt. Es ergibt sich hier allgemein

$$\alpha_w = \frac{2^K - 1}{K} \tag{5.35}$$

und damit $\alpha_w = 7/3$ für $K = 3$, deutlich unter dem in Abschnitt 5.4 gefundenen numerischen Wert $\alpha_w \simeq 2.7$ für den Algorithmus mit nur Zufallsschritten. Für größere Zeiten sind also größere Abweichungen von einer Poissonverteilung für $p_t(s, u)$ zu vermuten.

5.5.3 Beschreibung durch Ratengleichungen

Mit der Poisson-Näherung des vorherigen Abschnittes ließ sich das Verhalten von Walk-SAT für kleine Zeiten sehr gut beschreiben. Für K-XOR-SAT stimmten auch die Werte des Plateaus und der dynamischen Schwelle recht gut mit den numerischen Werten überein, für K-SAT dagegen gab es bei diesen Werten deutliche Abweichungen.

Wir wollen deshalb in diesem Abschnitt die Annahme der Poissonverteilung fallen lassen und nur, wie schon in Abschnitt 5.5.1, die Unabhängigkeit der Variablen einer Klausel, d. h. die Faktorisierung von $p_t(s_1, u_1, s_2, u_2, \ldots, s_K, u_K)$, voraussetzen. Unter dieser Annahme werden wir – der Prozedur in [46] folgend – das Verhalten von $p_t(s, u)$ auf linearen Zeitskalen beschreiben. Daraus folgt dann mit (5.12) über die mittlere Energiedichte das typische Verhalten des Algorithmus.

Zunächst werden wir wieder den einfacheren Fall K-XOR-SAT betrachten, anschließend K-SAT.

Ratengleichung für K-XOR-SAT

Im Abschnitt 5.5.2 haben wir mit Mittelwerten $K\alpha_u(t) + 1$ bzw. $K\alpha_s(t)$ über die Poissonverteilungen der erfüllten Klauseln gearbeitet, in denen die zu negierende Variable v^* enthalten ist. Jetzt gehen wir von der tatsächlichen Verteilung $p_t(s, u)$ aus. Die Wahrscheinlichkeit $p_t^{(\mathrm{flip})}(s^*, u^*)$, dass die zu negierende Variable v^* in s^* erfüllten und u^* unerfüllten Klauseln vorkommt, lässt sich aus dieser Verteilung über (5.19) bestimmen.

Die Anzahl $N_t(s, u) = Np_t(s, u)$ der Variablen, die in s erfüllten und u unerfüllten Klauseln auftreten, ändert sich abhängig von s^* und u^*. Mittels einer Ratengleichung wollen wir den Erwartungswert dieser Änderung während eines Schrittes beschreiben. Jeder Schritt dauert dabei wieder $\Delta t = 1/N$.

Es gibt drei verschiedene Beiträge zur Änderung von $N_t(s, u)$:

- *Beitrag durch v^*:*
  $N_t(s^*, u^*)$ erniedrigt sich um eins, wenn v^* vor der Negierung in s^* bzw. u^* Klauseln vorkommt. Dies passiert wie oben geschrieben mit Wahrscheinlichkeit $p_t^{(\mathrm{flip})}(s^*, u^*)$ und führt im Mittel zu einem negativen Beitrag von $-p_t^{(\mathrm{flip})}(s, u)$ zu $N_t(s, u)$.
  Alle s^* erfüllten Klauseln werden unerfüllt, alle u^* unerfüllten Klauseln werden erfüllt. v^* kommt nach Ausführung des Schrittes in u^* erfüllten und s^* unerfüllten Klauseln vor, damit erhöht sich also $N_t(u^*, s^*)$ um eins. Dies passiert mit Wahrscheinlichkeit $p_t^{(\mathrm{flip})}(s^*, u^*)$, d. h. im Mittel gibt es einen positiven Beitrag $p_t^{(\mathrm{flip})}(u, s)$ zu $N_t(s, u)$.

Der gesamte Beitrag zur Änderung von $N_t(s,u)$ beträgt somit $-p_t^{(\text{flip})}(s,u) + p_t^{(\text{flip})}(u,s)$.

- *Beitrag durch die Nachbarn von v^* in erfüllten Klauseln:*
  Im Mittel ist v^* in $\sum_{s,u} sp_t^{(\text{flip})}(s,u) =: \langle s \rangle_t^{(\text{flip})}$ vorher erfüllten Klauseln enthalten, die während des Schrittes unerfüllt werden. Da Zufallsformeln lokal baumartig sind (siehe Abschnitt 5.2.1) und jede Klausel K Variablen enthält, hat v^* in erfüllten Klauseln im Mittel insgesamt $(K-1)\langle s \rangle_t^{(\text{flip})}$ Nachbarn. Für jeden dieser Nachbarn erniedrigt sich die Anzahl der erfüllten Klauseln um 1, während sich die Anzahl der unerfüllten Klauseln um 1 erhöht.
  Nun müssen wir noch wissen, zu welchen $N_t(s,u)$ dieser Beitrag hinzukommt, d. h. mit welcher Wahrscheinlichkeit $p_t^{(s)}(s,u)$ eine Variable aus einer erfüllten Klausel in s erfüllten und u unerfüllten Klauseln vorkommt. Diese ist aber analog zu (5.15) gegeben durch

$$p_t^{(s)}(s,u) = \frac{sp_t(s,u)}{\langle s \rangle_t}. \tag{5.36}$$

  Dadurch erniedrigt sich der Erwartungswert von $N_t(s,u)$ um $(K-1)\langle s \rangle_t^{(\text{flip})} p_t^{(s)}(s,u)$. Gleichzeitig erhöht er sich analog um $(K-1)\langle s \rangle_t^{(\text{flip})} p_t^{(s)}(s+1,u-1)$, da Nachbarn von v^* in erfüllten Klauseln, die vor der Negierung von v^* in $s+1$ bzw. $u-1$ Klauseln vorkamen, danach in s bzw. u Klauseln vorkommen.

- *Beitrag durch die Nachbarn von v^* in unerfüllten Klauseln:*
  Dieser Beitrag ergibt sich ganz analog zum vorherigen.

Summieren wir alle drei Beiträge auf, so erhalten wir die gesamte Änderung von $N_t(s,u)$ während eines Schrittes des Algorithmus. Es ist

$$\begin{aligned}
\Delta N_t(s,u) &= N_{t+\Delta t}(s,u) - N_t(s,u) \\
&= -p_t^{(\text{flip})}(s,u) + p_t^{(\text{flip})}(u,s) \\
&\quad + (K-1)\langle s \rangle_t^{(\text{flip})} \left(-p_t^{(s)}(s,u) + p_t^{(s)}(s+1,u-1) \right) \\
&\quad + (K-1)\langle u \rangle_t^{(\text{flip})} \left(-p_t^{(u)}(s,u) + p_t^{(u)}(s-1,u+1) \right)
\end{aligned} \tag{5.37}$$

Wir schreiben analog zu (5.23) mit $\Delta t = 1/N$

$$\begin{aligned}
N_{t+\Delta t}(s,u) - N_t(s,u) &= N\left(p_{t+\Delta t}(s,u) - p_t(s,u)\right) \\
&= \frac{p_{t+\Delta t}(s,u) - p_t(s,u)}{\Delta t}
\end{aligned} \tag{5.38}$$

und gehen für $N \to \infty$ zum Differentialquotienten über. Damit erhalten wir das

folgende System gekoppelter Differentialgleichungen:

$$\begin{aligned}\dot{p}_t(s,u) = &-p_t^{(\text{flip})}(s,u) + p_t^{(\text{flip})}(u,s) \\ &+ (K-1)\langle s\rangle_t^{(\text{flip})}\left(-\frac{sp_t(s,u)}{\langle s\rangle_t} + \frac{(s+1)p_t(s+1,u-1)}{\langle s\rangle_t}\right) \\ &+ (K-1)\langle u\rangle_t^{(\text{flip})}\left(-\frac{up_t(s,u)}{\langle u\rangle_t} + \frac{(u+1)p_t(s-1,u+1)}{\langle u\rangle_t}\right). \end{aligned} \tag{5.39}$$

In der typischen Anfangskonfiguration ist die Hälfte der Klauseln erfüllt bzw. unerfüllt, außerdem sind ihre Werte bei $t = 0$ unabhängig voneinander. Die Anfangsbedingung $p_0(s,u)$ ist somit gegeben durch die Poissonverteilung in (5.20) mit $\alpha_s(0) = \alpha_u(0) = \alpha/2$.

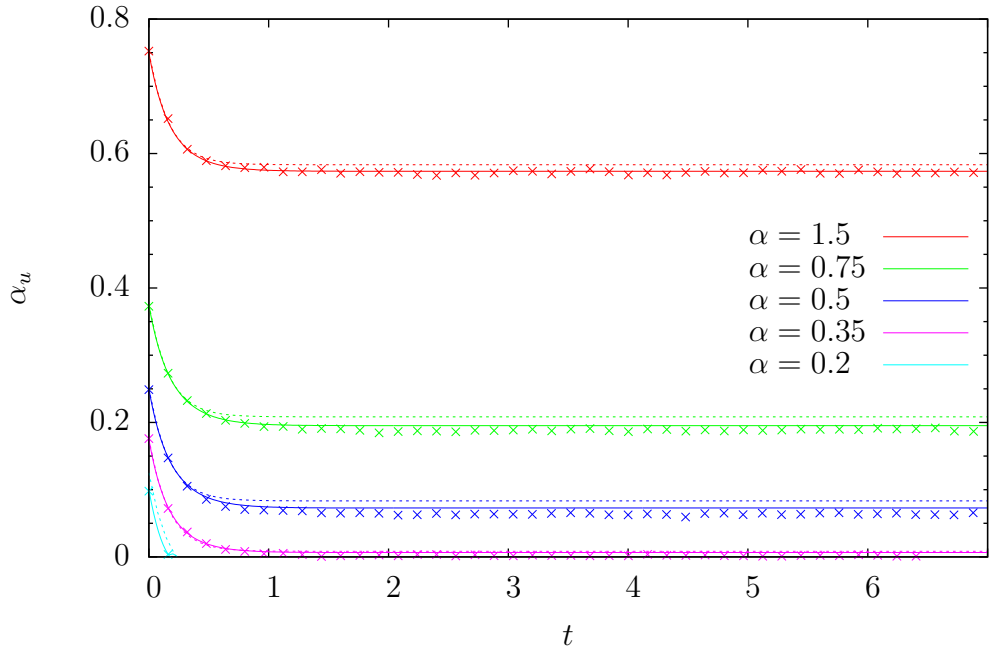


Abbildung 5.8: 3-XOR-SAT: Typischer Verlauf der Energiedichte bei Walk-SAT (ohne gierige Schritte) als Funktion der Zeit (in MC-Sweeps) für verschiedene Werte von α. Die Kreuzsymbole sind numerische Ergebnisse jeweils einer Instanz mit $N = 50\,000$. Die gestrichelten Linien sind die Ergebnisse der Poissonnäherung aus Abschnitt 5.5.2 (Gleichung (5.26)), die durchgezogenen Linien sind aus der integrierten Ratengleichung (5.39) unter Verwendung von (5.12) gewonnen.

Eine exakte analytische Lösung von (5.39) zu suchen erscheint hier wenig aussichtsreich, stattdessen integrieren wir numerisch (Vorwärts-Euler-Verfahren). Zunächst betrachten wir den Algorithmus ohne gierige Schritte. Wir setzen für $p_t^{(\text{flip})}(s,u)$ (5.19) mit $q = 0$ ein, d. h. $p_t^{(\text{flip})}(s,u) = p_t^{(\text{flip-walk})}(s,u) = p_t^{(u)}(s,u)$. Die dann aus der

Verteilung $p_t(s, u)$ resultierende Energiedichte ist in Abbildung 5.8 dargestellt, zusammen mit der Poissonnäherung aus Abschnitt 5.5.2 und den numerischen Daten aus der Anwendung des Walk-SAT-Algorithmus ohne gierige Schritte auf eine große Instanz mit $N = 50\,000$ Variablen.

Für kleine Zeiten liegen Numerik und beide analytischen Kurven übereinander. Bei größeren Zeiten erkennt man, dass die Ratengleichung die schon gute Näherung durch den Poissonansatz noch deutlich verbessert. Der Plateauwert der Energiedichte wird um eine Größenordnung besser approximiert, der Fehler beträgt $4.4 \cdot 10^{-3}$ bei $\alpha = 0.5$ bzw. $1.8 \cdot 10^{-3}$ bei $\alpha = 1.5$. Für die dynamische Schwelle finden wir hier ebenfalls einen Wert von $\alpha_w = 0.33$.

Insgesamt scheint bei 3-XOR-SAT die Annahme der Unabhängigkeit der Variablen für den Verlauf des Walk-SAT-Algorithmus mit $q = 0$ (nur Zufallsschritte) sehr gut erfüllt zu sein.

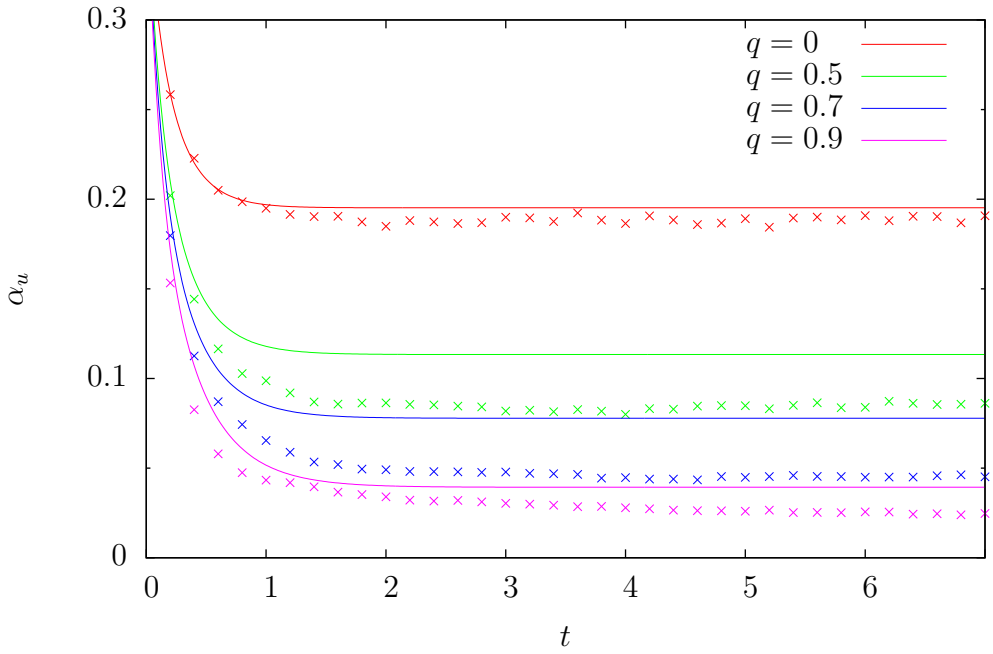


Abbildung 5.9: 3-XOR-SAT: Einfluss des Anteils q gieriger Schritte auf die Energiedichte für $\alpha = 0.75$. Die durchgezogenen Linien sind die Ergebnisse der integrierte Ratengleichung, die Symbole zeigen den Verlauf für eine Instanz mit $N = 50\,000$ Variablen.

Dies ist nicht mehr der Fall, wenn wir auch gierige Schritte zulassen. In die Ratengleichung (5.39) setzen wir dafür $p_t^{(\text{flip})}(s, u) = (1 - q)p_t^{(\text{flip-walk})}(s, u) + qp_t^{(\text{flip-greedy})}(s, u)$ mit $q > 0$ ein. In Abbildung 5.9 ist der Verlauf der resultierenden Energiedichte beispielhaft für $\alpha = 0.75$ dargestellt, wieder zusammen mit numerischen Daten einer typischen Instanz mit $N = 50\,000$. Qualitativ sieht man sowohl numerisch als auch analytisch, dass für die dargestellten Werte von q mit wachsendem Anteil gieriger

Schritte zum einen der Abfall der Energiedichte für kleine Zeiten steiler wird, zum anderen der Wert des Plateaus kleiner wird. Man sieht aber, dass schon für kleine Zeiten $t = 0.5$ die von der Ratengleichung vorausgesagte Energiedichte wie auch der Plateauwert für $t \to \infty$ oberhalb der numerischen Werte liegen. Da die Auswahlbedingung für v^* im gierigen Schritt explizit von den $K-1$ weiteren Variablen der ausgewählten Klausel abhängt, bauen sich sehr schnell Korrelationen zwischen den Variablen auf, so dass die Annahme der Unabhängigkeit der Variablen verletzt wird.

Ratengleichung für K-SAT

Das Vorgehen für K-SAT ist analog zu K-XOR-SAT. Der wesentliche Unterschied wird wieder sein, dass erfüllte Klauseln durch Negieren einer Variable nur dann unerfüllt werden, wenn dies die einzige vorher korrekt gesetzte Variable der Klausel war. Wie in Abschnitt 5.5.2 nehmen wir an, dass diese Wahrscheinlichkeit unabhängig von der Klausel durch den Erwartungswert $\mu = 1/(2^K - 1)$ gegeben ist (vgl. Gleichung (5.29)).

Für die drei Beiträge zur Änderung von $N_t(s,u)$ ergibt sich hier Folgendes:

- *Beitrag durch v^*:*
  Wie bei K-XOR-SAT erniedrigt sich $N_t(s^*,u^*)$ um eins, wenn v^* vor der Negierung in s^* bzw. v^* Klauseln vorkam. Dies passiert mit Wahrscheinlichkeit $p_t^{(\mathrm{flip})}(s^*,u^*)$ und führt im Mittel zu einem negativen Beitrag von $-p_t^{(\mathrm{flip})}(s,u)$ zu $N_t(s,u)$.

  Alle u^* vorher unerfüllten Klauseln werden erfüllt. Von den s^* vorher erfüllten Klauseln bleiben l erfüllt und s^*-l werden unerfüllt, d. h. $N_t(u^*+l, s^*-l)$ erhöht sich um eins. Die Wahrscheinlichkeit für eine Möglichkeit, l der s^* Klauseln auszuwählen, beträgt $\mu^{s^*-l}(1-\mu)^l$, insgesamt gibt es $\binom{s^*}{l}$ Möglichkeiten. Es gibt also für alle $0 \le l \le s^*$ zu $N_t(u^*+l, s^*-l)$ im Mittel einen Beitrag $\binom{s^*}{l}\mu^{s^*-l}(1-\mu)^l p_t^{(\mathrm{flip})}(s^*,u^*)$.

  Damit ergibt sich umgekehrt zu $N_t(s,u)$ im Mittel ein positiver Beitrag

$$\sum_{l=0}^{s} \binom{u+l}{l} \mu^u (1-\mu)^l p_t^{(\mathrm{flip})}(u+l, s-l) \tag{5.40}$$

- *Beitrag durch die Nachbarn von v^* in vorher erfüllten Klauseln*
  Im Unterschied zu K-XOR-SAT werden die $\langle s \rangle_t^{(\mathrm{flip})}$ vorher erfüllten Klauseln, in denen v^* enthalten ist, nur mit Wahrscheinlichkeit μ unerfüllt. Bis auf diesen zusätzlichen Vorfaktor ist der Beitrag identisch.

- *Beitrag durch die Nachbarn von v^* in vorher unerfüllten Klauseln*
  Da sowohl in K-XOR-SAT als auch in K-SAT alle vorher unerfüllten Klauseln durch Negierung einer ihrer Variablen erfüllt werden, ist dieser Beitrag identisch.

Durch Summierung und Übergang zu $p_t(s,u)$ wie in (5.38) erhalten wir folgendes DGL-System:

$$\begin{aligned}\dot{p}_t(s,u) = &-p_t^{(\text{flip})}(s,u) + \mu^u \sum_{l=0}^{s} \binom{u+l}{l} (1-\mu)^l p_t^{(\text{flip})}(u+l, s-l) \\ &+ \mu(K-1)\langle s\rangle_t^{(\text{flip})} \left(-\frac{s p_t(s,u)}{\langle s\rangle_t} + \frac{(s+1)p_t(s+1,u-1)}{\langle s\rangle_t} \right) \\ &+ (K-1)\langle u\rangle_t^{(\text{flip})} \left(-\frac{u p_t(s,u)}{\langle u\rangle_t} + \frac{(u+1)p_t(s-1,u+1)}{\langle u\rangle_t} \right). \end{aligned} \tag{5.41}$$

In der typischen Anfangskonfiguration sind bei zufälligem K-SAT $\alpha/2^K$ Klauseln pro Variable unerfüllt, d. h. die Anfangsbedingung $p_{t=0}(s,u)$ ist durch (5.20) mit $\alpha_s(0) = \frac{2^K-1}{2^K}\alpha$ und $\alpha_u(0) = \frac{1}{2^K}\alpha$ gegeben.

Die Ergebnisse der numerischen Integration nach Einsetzen von (5.19) für $K = 3$ und $q = 0$ (nur zufällige Schritte) sind in Abbildung 5.10 dargestellt, zusammen mit der numerischen Simulation einer großen Instanz und den Voraussagen des Poissonansatzes. In der oberen Hälfte ist der Ausschnitt kleiner Zeiten zusätzlich vergrößert dargestellt.

Für kleine Zeiten weichen die Ergebnisse der Ratengleichung deutlich nach unten von den numerischen Ergebnissen ab, sie sind sogar deutlich schlechter als die Poissonnäherung.

Für große Zeiten dagegen wird das Verhalten sehr gut wiedergegeben, der Plateauwert der Energiedichte $\alpha_u(t = \infty)$ wird sehr gut vorausgesagt. Eine Ursache für diese Übereinstimmung sieht man in Abbildung 5.11. Dort sind die Verteilungen $p_t(s,u)$ beispielhaft für $\alpha = 3.5 > \alpha_w$ dargestellt, zum einen für kleine Zeiten beispielhaft bei $t = 1.6$, zum anderen nach Erreichen des Plateaus beispielhaft bei $t = 6.0$. Man sieht, dass die Übereinstimmung in der Verteilung $p_t(s,u)$ keine notwendige Bedingung für eine Übereinstimmung von $\alpha_u(t)$ ist. Denn obwohl $p_t(s,u)$ für kleine Zeiten nicht der Poissonverteilung aus (5.20) entspricht, wird die Energiedichte dort gut von dieser Näherung wiedergegeben.

Umgekehrt erkennt man, dass die Ratengleichung die Verteilung $p_t(s,u)$ für große Zeiten ausgezeichnet wiedergibt, was die gute Übereinstimmung des Plateauwertes erklärt. Damit können wir auch präzise die dynamische Schwelle α_w bestimmen, oberhalb derer Walk-SAT auf linearen Zeitskalen keine Lösung mehr findet, sondern um eine endliche Plateauenergie equilibriert. In Abbildung 5.12 ist die typische Rechenzeit dargestellt, die für $\alpha \to \alpha_w$ divergiert. Gleichzeitig beginnt bei α_w das lineare Wachstum des Energieplateaus. Solch ein lineares Wachstum (mit Anstieg

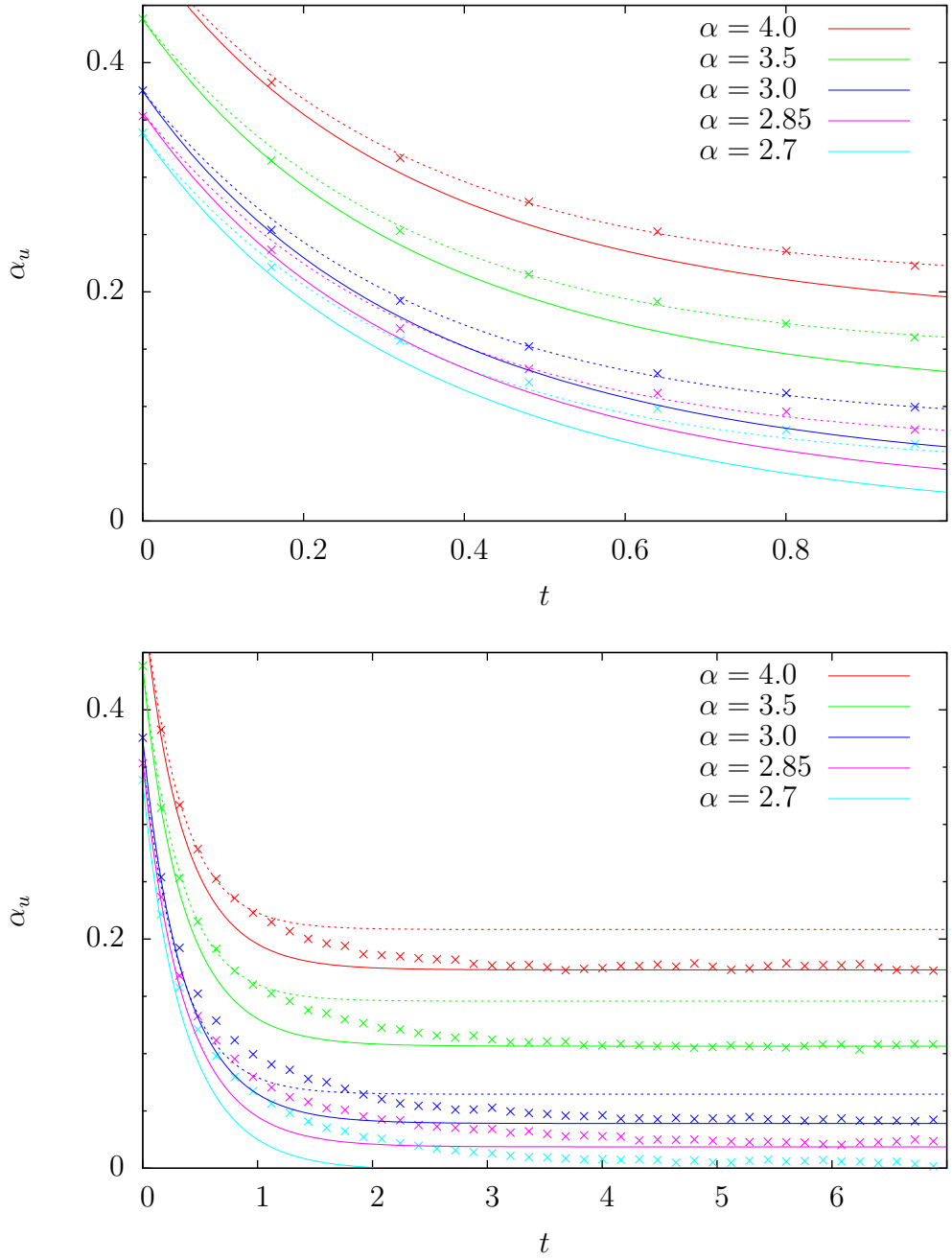


Abbildung 5.10: 3-SAT: Typischer Verlauf der Energiedichte bei Walk-SAT (ohne gierige Schritte) als Funktion der Zeit für verschiedene Werte von α. Der obere Ausschnitt zeigt eine Vergrößerung für kleine Zeiten. Die Kreuzsymbole sind numerische Ergebnisse jeweils einer Instanz mit $N = 50\,000$. Die gestrichelten Linien sind die Ergebnisse der Poissonnäherung aus Abschnitt 5.5.2 (Gleichung (5.33)), im unteren Bild der Übersichtlichkeit halber nur für $\alpha = 4.0, 3.5, 2.85$ eingezeichnet. Die durchgezogenen Linien sind aus der integrierten Ratengleichung (5.41) zusammen mit (5.12) gewonnen.

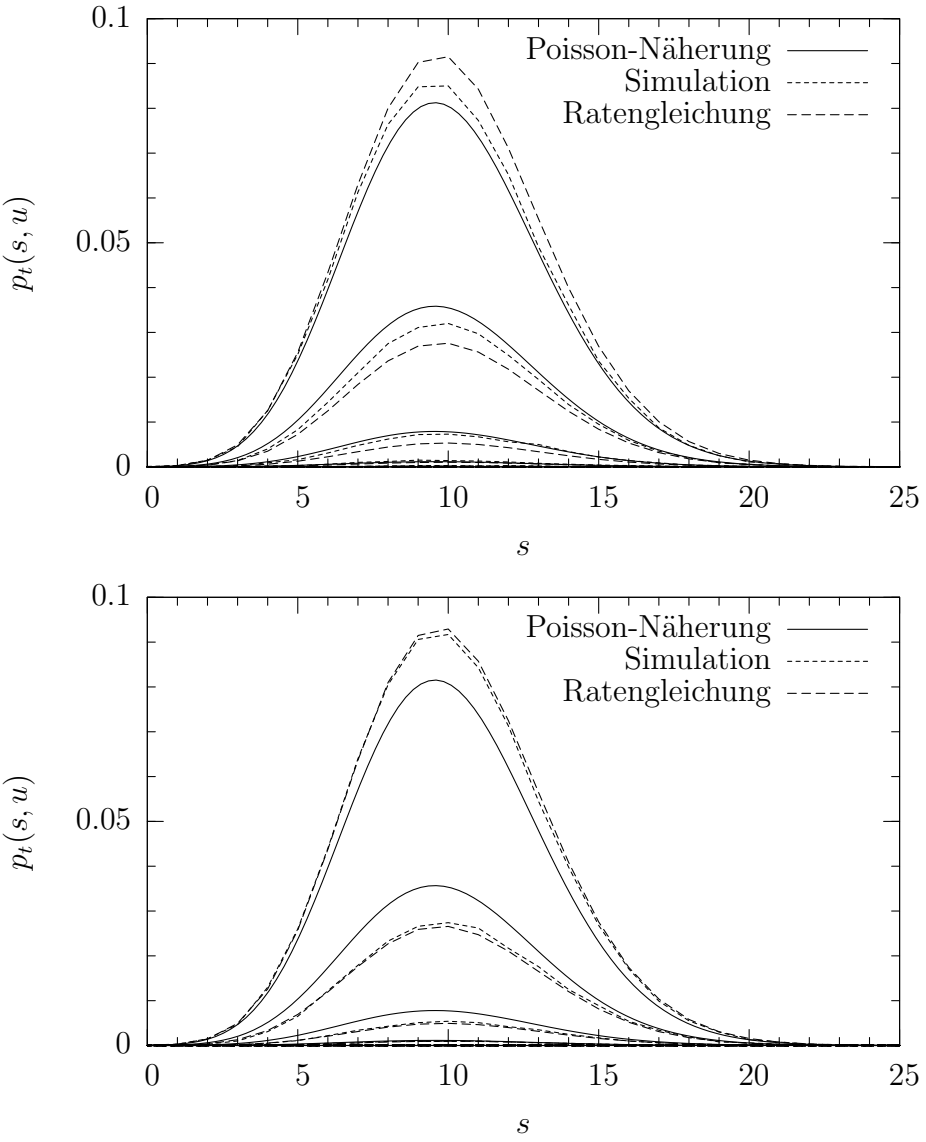


Abbildung 5.11: 3-SAT: Verteilung $p_t(s,u)$ für $\alpha = 3.5$ bei $t = 1.6$ (oberer Graph) und $t = 6.0$ (unterer Graph), dargestellt jeweils als Funktion von s für $u = 0, 1, 2, 3$ (von oben nach unten). Man erkennt, dass bei größere Zeiten, d. h. nach Erreichen des Plateauwertes, die Ratengleichung die tatsächliche Verteilung sehr viel besser wiedergibt als bei kleineren Zeiten.

$1/7 \simeq 0.142$ für $K = 3$) wurde auch schon von der Poissonnäherung durch Gleichung (5.33) vorausgesagt. Hier finden wir einen Anstieg von 0.134, in guter Übereinstimmung mit den numerischen Werten. Für die dynamische Schwelle ergibt sich aus der Ratengleichung $\alpha_w \simeq 2.71$, ebenfalls in guter Übereinstimmung mit den numerischen Werten und deutlich besser als der Poisson-Ansatz in Abschnitt 5.5.2.

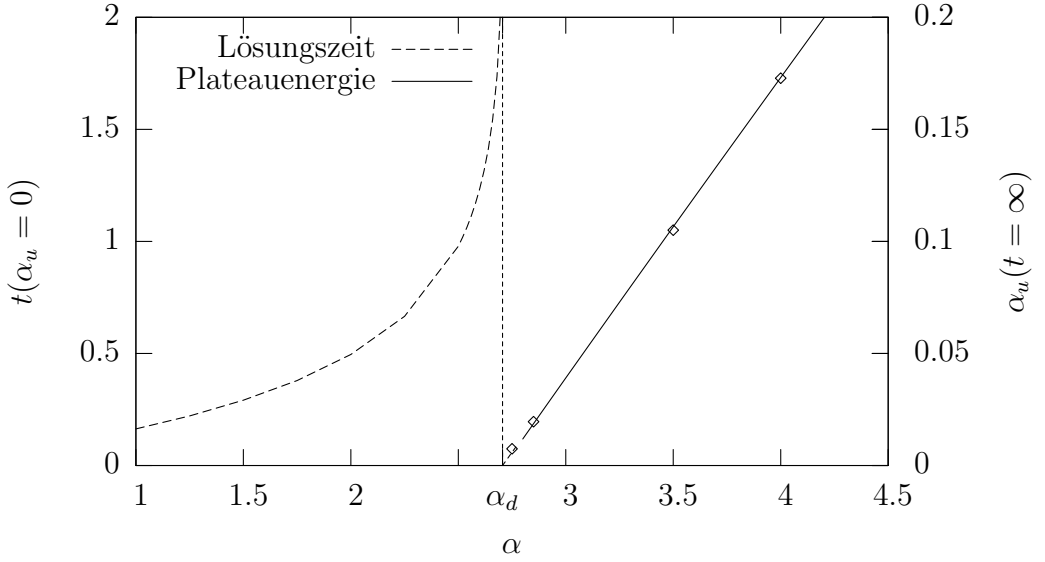


Abbildung 5.12: 3-SAT: Dynamischer Phasenübergang bei $\alpha_d \simeq 2.71$ (vertikale gestrichelte Linie). Dargestellt sind in Abhängigkeit von α die aus den Ratengleichungen resultierenden Werte für die typischen Lösungszeiten (für $\alpha < \alpha_d$, linker Bereich) und für den Plateauwert der Energiedichte $\alpha_u(t = \infty)$ zusammen mit numerischen Plateauwerten je einer Instanz mit $N = 50\,000$ (für $\alpha > \alpha_d$, rechter Bereich). Auf linearer Zeitskala divergiert die Lösungszeit für $\alpha \to \alpha_d$. Oberhalb von α_d wächst die Plateauenergie wie $0.134 \cdot (\alpha - \alpha_d)$.

Da sich schon im einfacheren Fall von K-XOR-SAT durch die Hinzunahme gieriger Schritte nach kurzer Zeit Korrelationen aufbauen, ist nicht zu erwarten, dass wir mittels einer Ratengleichung für K-SAT bei einem Anteil gieriger Schritte $q > 0$ quantitative Aussagen gewinnen können.

Nahe liegend zur Verbesserung der Ergebnisse ist die Untersuchung der Verteilungen anderer Observablen. Semerjian und Monasson [82] beispielsweise betrachten die Anzahl der Klauseln mit $0, 1, 2, \ldots, K$ korrekt gesetzten Variablen. Sie erhalten für die Energiedichte

$$\alpha_u(t) = \frac{\alpha}{2^K} - \frac{1}{K}\left[1 - \frac{1}{(1 + \tanh t)^K}\right]. \tag{5.42}$$

Damit wird das Zeitverhalten der Energiedichte im Bereich kleiner Zeiten gut beschrieben, für $t \to \infty$ findet man aber denselben (zu großen) Plateauwert wie durch die Poissonnäherung in (5.34) gegeben.

Auch weiter gehende Verteilungen wären denkbar, beispielsweise die Kombination des Ansatzes von Semerjian und Monasson mit dem hier betrachteten durch eine Verteilung $p_t(s_0, s_1, \ldots, s_K)$. Man würde Ratengleichungen mit ähnlicher Form wie in (5.41) erwarten, die gleichzeitig aber deutlich komplizierter wären. Letztendlich würde man eine verbesserte Näherung erhalten, ohne aber das exakte Verhalten auf diese Weise beschreiben zu können. Wir wollen es deshalb bei den hier untersuchten Ansätzen belassen, da diese die Dynamik des Algorithmus auf linearer Zeitskala und die Lage des dynamischen Phasenübergangs sowohl qualitativ als auch quantitativ befriedigend charakterisieren konnten.

5.6 Analyse des Laufzeitverhaltens im Bereich exponentieller Rechenzeit

Im vorigen Abschnitt 5.5 haben wir das typische Verhalten auf linearen Zeitskalen von Walk-SAT für zufällige K-XOR-SAT- bzw. K-SAT-Formeln analysiert. Dabei haben wir den Wert α_w der Schwelle bestimmt, an der die Lösungszeit auf dieser Skala divergiert. Gleichzeitig haben wir gesehen, dass die typische Trajektorie für $\alpha > \alpha_w$ nach wenigen MC-Sweeps einen Plateauwert erreicht.

In diesem Abschnitt wollen wir die typische Lösungszeit im Bereich $\alpha > \alpha_w$ bestimmen. Zunächst skizzieren wir die Idee unserer Vorgehensweise:

Die numerischen Ergebnisse aus Abschnitt 5.4 zeigen, dass die typische Rechenzeit $t_{\mathrm{sol}}(\alpha)$ für $\alpha > \alpha_w$ exponentiell ansteigt, d. h.

$$t_{\mathrm{sol}}(\alpha) \sim \exp(N\tau(\alpha)). \tag{5.43}$$

In diesem Bereich werden Lösungen durch *atypische* Fluktuationen weg vom Plateauwert der typischen Trajektorie gefunden (vgl. Abbildung 5.4). Wir nehmen an, dass diese Fluktuationen unabhängige Zufallsereignisse sind, d. h. insbesondere ist die Größe einer Fluktuation unabhängig vom vorherigen Verlauf des Algorithmus. Dies entspricht den numerischen Beobachtungen (vgl. Abb. 5.4).

Wie wir sehen werden, ist die Wahrscheinlichkeit, in einem bestimmten Schritt des Algorithmus eine Fluktuation auf null zu finden, exponentiell klein. Deshalb benötigt der Algorithmus typischerweise eine exponentielle Anzahl von Schritten, um eine solche Fluktuation zu finden, was zu den beobachteten exponentiellen Rechenzeiten für $\alpha > \alpha_w$ führt.

Um die Wahrscheinlichkeit einer solchen Fluktuation zu *berechnen*, bestimmen wir die Wahrscheinlichkeit

$$P(\alpha_u(t = 0) \to \alpha_u(t_f) = 0), \tag{5.44}$$

dass der Algorithmus zu einer bestimmten vorgegeben Zeit t_f mit $1 \ll t_f \ll N$ eine solche Fluktuation ausführt, genauer gesagt, dass der Algorithmus einer atypischen

Trajektorie folgt, die vom Startwert $\alpha_u(t = 0)$ ausgeht und nach einer vorgegeben Zeit t_f mit $1 \ll t_f \ll N$ eine Lösung erreicht. Wir werden den Verlauf einer solchen Trajektorie bestimmen und dabei sehen, dass sie in Übereinstimmung mit den numerischen Beobachtungen nach wenigen MC-Sweeps ein Plateau erreicht, dort verweilt und schließlich zur Zeit t_f durch eine einzige große Fluktuation zu einer Lösung führt.

Wir werden feststellen, dass die Wahrscheinlichkeit $P(\alpha_u(t = 0) \to \alpha_u(t_f) = 0)$ für kleine Lösungszeiten $t_f > 2$ schon annähernd den Plateauwert erreicht. Die Rechenzeit des Algorithmus für eine einzelne Instanz ist dann in führender Ordnung durch

$$t_{\mathrm{sol}}(\alpha) \sim \lim_{t_f \to \infty} \lim_{N \to \infty} \frac{1}{P(\alpha_u(0) \to \alpha_u(t_f) = 0)} \tag{5.45}$$

gegeben. Da aber der Mittelwert der Rechenzeit durch exponentiell seltene Instanzen mit exponentiell längerer Rechenzeit dominiert wird (vgl. Abbildung 5.5), betrachten wir hier

$$\tau(\alpha) = - \lim_{t_f \to \infty} \lim_{N \to \infty} \frac{1}{N} \ln(P(\alpha_u(0) \to \alpha_u(t_f) = 0)), \tag{5.46}$$

die typische Rechenzeit (der Median) ergibt sich dann aus Gleichung (5.43).

Die Reihenfolge der Grenzwerte ist entscheidend. Denn für lösbare Instanzen (und das ist für $\alpha < \alpha_c$ fast jede Instanz) ist $\lim_{t_f \to \infty} P(\alpha_u(0) \to \alpha_u(t_f) = 0) = 1$, da nach exponentiellen Schritten der Algorithmus fast immer eine Lösung findet (vgl. Abb. 5.5); die rechte Seite von (5.46) würde dann also einfach verschwinden.

Für einen Algorithmus *mit Restarts*, wie er am Ende von Abschnitt 5.3 beschrieben wurde, ist die Situation etwas anders. Hier müssen wir zusätzlich berücksichtigen, dass der Algorithmus schon atypisch nahe an einer Lösung starten könnte. Dies passiert mit einer gewissen Wahrscheinlichkeit

$$\rho(\alpha(0)) \sim \exp N s(\alpha_u(0)), \tag{5.47}$$

wobei $s(\alpha_u(0))$ die mikrokanonische Entropie der Zustände mit Energiedichte $\alpha_u(0)$ ist.

Nehmen wir an, dass der Algorithmus nach jeweils t_f Sweeps neu mit einer zufällig initialisierten Konfiguration startet. Die exponentiell kleinere Wahrscheinlichkeit einer atypischen Startkonfiguration kann dabei unter Umständen kompensiert werden durch eine exponentiell größere Wahrscheinlichkeit, von ihr aus innerhalb t_f Sweeps eine Lösung zu finden. Deshalb ist hier die typische Anzahl der Restarts gegeben durch

$$t_{\mathrm{sol}}^{\mathrm{restart}}(\alpha) \sim e^{N\tau^{\mathrm{restart}}(\alpha)} \tag{5.48}$$

mit

$$\tau^{\mathrm{restart}}(\alpha) = - \max_{\alpha_u(0)} \lim_{N \to \infty} \frac{1}{N} \rho(\alpha_u(0)) \max_{0 \le t \le t_f} P(\alpha_u(0) \to \alpha_u(t_f) = 0), \tag{5.49}$$

d. h. die Wahrscheinlichkeit, innerhalb t_f Sweeps eine Lösung zu finden, wird mit der Wahrscheinlichkeit der Anfangskonfiguration gewichtet. Für $N \to \infty$ dominiert dabei das Maximum über alle Anfangsbedingungen.

Um $P(\alpha_u(0) \to \alpha_u(t_f) = 0)$ tatsächlich analytisch bestimmen zu können, werden wir $p_t(s,u)$ wie in Gleichung (5.20) durch eine Poissonverteilung approximieren. In diesem Abschnitt sei also wieder

$$p_t(s,u) = e^{-K\alpha} \frac{(K\alpha_s(t))^s (K\alpha_u(t))^u}{s!u!} \tag{5.50}$$

vorausgesetzt. Außerdem werden wir immer den Algorithmus ohne gierige Schritte betrachten, d. h. $q = 0$, so dass die Annahme einer Poissonverteilung annähernd erfüllt ist (vgl. Abschnitt 5.5).

Zufälliges K-XOR-SAT

Im Gegensatz zum vorherigen Abschnitt 5.5 reicht es hier nicht mehr aus, nur die typische Trajektorie zu bestimmen, die im thermodynamischen Limes fast sicher angenommen wird.

Wir werden zunächst die Wahrscheinlichkeit $P_T^{T+1}(\Delta e)$ ausrechnen, dass sich die Energie, d. h. die Anzahl der unerfüllten Klauseln, in einem algorithmischen Schritt um Δe ändert, wobei $T = Nt$ die Zeit in algorithmischen Schritten misst.

Mit Wahrscheinlichkeit $p_t^{(\text{flip})}(s,u)$, gegeben durch (5.15), kommt die zu negierende Variable v^* in s erfüllten und u unerfüllten Klauseln vor. Durch Negieren von v^* werden die s Klauseln unerfüllt und die u Klauseln erfüllt, d. h. $\Delta e = s - u$ und

$$\begin{aligned}
P_T^{T+1}(\Delta e) &= \sum_{s,u=0}^{\infty} p_t^{(\text{flip})}(s,u) \delta_{\Delta e, s-u} \\
&= \sum_{s,u=0}^{\infty} \frac{u p_t(s,u)}{K\alpha_u(t)} \delta_{\Delta e, s-u} \\
&= \sum_{s,u=0}^{\infty} e^{-K\alpha} \frac{(K\alpha_s(t))^s (K\alpha_u(t))^u}{s!u!} \frac{u}{K\alpha_u(t)} \delta_{\Delta e, s-u} \\
&= \sum_{s,u'=0}^{\infty} e^{-K\alpha} \frac{(K\alpha_s(t))^s (K\alpha_u(t))^{u'}}{s!u'!} \delta_{\Delta e, s-u'-1}.
\end{aligned} \tag{5.51}$$

Dabei haben wir im dritten Schritt die Annahme (5.50) einer Poissonverteilung für $p_t(s,u)$ verwendet.

Die Wahrscheinlichkeit $P_T^{T+\Delta T}(\Delta E)$ einer Änderung der Energie um ΔE nach $1 \ll \Delta T \ll N$ aufeinander folgenden algorithmischen Schritten ist durch die Faltung der

Wahrscheinlichkeiten für jeden einzelnen Schritt gegeben, d. h. durch

$$P_T^{T+\Delta T}(\Delta E) = \sum_{\{\Delta e_1\}} \cdots \sum_{\{\Delta e_{\Delta T}\}} \left(\prod_{k=1}^{\Delta T} P_{T+k-1}^{T+k}(\Delta e_k) \right) \cdot \delta_{\Delta E, \sum_{k=1}^{\Delta T} \Delta e_k}. \tag{5.52}$$

Wir wollen Änderungen der Verteilung $p_t(s,u) = \frac{1}{N} N_t(s,u)$ während der $\Delta T \ll N$ Schritte vernachlässigen und deshalb $P_{T+k-1}^{T+k}(\Delta e)$ für $k = 1, \ldots, \Delta T$ als konstant annehmen (subdominante $\mathcal{O}(\Delta T) = \mathcal{O}(1/N)$ Terme vernachlässigend, die für $N \to \infty$ irrelevant sind). Damit vereinfacht sich (5.52) zu

$$P_T^{T+\Delta T}(\Delta E) = \sum_{\{\Delta e_1\}} \cdots \sum_{\{\Delta e_{\Delta T}\}} \left(\prod_{k=1}^{\Delta T} P_T^{T+1}(\Delta e_k) \right) \cdot \delta_{\Delta E, \sum_{k=1}^{\Delta T} \Delta e_k}. \tag{5.53}$$

Im Fourierraum erhalten wir mit

$$\hat{P}_T^{T+1}(l) = \sum_{\Delta e} P_T^{T+1}(\Delta e) \exp\{il\Delta e\} \tag{5.54}$$

durch Einsetzen

$$\begin{aligned}
\hat{P}_T^{T+\Delta T}(l) &= \left(\hat{P}_T^{T+1}(l) \right)^{\Delta T} \\
&= \left(\sum_{s,u=0}^{\infty} e^{-K\alpha} \frac{(K\alpha_s(t))^s (K\alpha_u(t))^u}{s!u!} \exp\left\{-il(s-u-1)\right\} \right)^{\Delta T} \\
&= \left(e^{-K\alpha+il} \left[\sum_{s=0}^{\infty} \frac{(K\alpha_s(t))^s}{s!} e^{-ils} \right] \left[\sum_{u=0}^{\infty} \frac{(K\alpha_u(t))^u}{u!} e^{ilu} \right] \right)^{\Delta T} \\
&= \left(e^{-K\alpha+il} \left[\sum_{s=0}^{\infty} \frac{(K\alpha_s(t)e^{-il})^s}{s!} \right] \left[\sum_{u=0}^{\infty} \frac{(K\alpha_u(t)e^{il})^u}{u!} \right] \right)^{\Delta T} \\
&= \left(\exp\{-K\alpha + K\alpha_s(t)e^{-il} + K\alpha_u(t)e^{il} + il\} \right)^{\Delta T} \\
&= \exp\left\{ \Delta T \left(-K\alpha + K\alpha_s(t)e^{-il} + K\alpha_u(t)e^{il} + il \right) \right\}.
\end{aligned} \tag{5.55}$$

Wegen $N\Delta t = \Delta T \ll N$ ist $\Delta t \ll 1$, außerdem können wir durch Übergang zurück zu intensiven Größen $E(t) = N\alpha_u(t)$ und damit $\Delta E(t) = N\dot{\alpha}_u(t)\Delta t + \mathcal{O}(\Delta t^2)$ schreiben. Vor der Rücktransformation von (5.55) schreiben wir l als $l(t)$, da wir den Zeitindex später verwenden werden. Es ergibt sich

$$\begin{aligned}
P_T^{T+\Delta T}(\Delta E(t)) &= \int_{-\pi}^{\pi} \frac{dl(t)}{2\pi} e^{il(t)\Delta E(t)} \left(\hat{P}_t(l(t)) \right)^{\Delta T} \\
&= \int_{-\pi}^{\pi} \frac{dl(t)}{2\pi} e^{N\Delta t i l \dot{\alpha}_u(t)} \\
&\quad \cdot \exp\left\{ N\Delta t \left(-K\alpha + K\alpha_s(t)e^{-il(t)} + K\alpha_u(t)e^{il(t)} + il(t) \right) \right\}
\end{aligned}$$

$$= \int_{-\pi}^{\pi} \frac{dl(t)}{2\pi} \exp\Bigg\{ N\Delta t \cdot \Big(il(t)\dot{\alpha}_u(t) - K\alpha + K(\alpha - \alpha_u(t))e^{-il(t)} + K\alpha_u(t)e^{il(t)} + il(t) \Big) \Bigg\}. \tag{5.56}$$

Gleichung (5.56) beschreibt die Wahrscheinlichkeit, dass der Algorithmus einer Trajektorie folgt, die ihn von $\alpha_u(t) = E/N$ zur Zeit $t = T/N$ nach $\alpha_u(t + \Delta t) = (E + \Delta E)/N$ zur Zeit $t + \Delta t = (T + \Delta T)/N$ führt.

Unsere gesuchte Größe ist die Gesamtwahrscheinlichkeit $P(\alpha_u(0) \to \alpha_u(t_f) = 0)$ für alle bei $t = 0$ in $\alpha_u(0)$ beginnenden und bei t_f in $\alpha_u(t_f) = 0$ endenden Trajektorien. Wir setzen jede einzelne Trajektorie aus vielen aufeinander folgenden Stückchen von $\alpha_u(t)$ nach $\alpha_u(t + \Delta t)$ zusammen. Die Wahrscheinlichkeit einer bestimmten Trajektorie ergibt sich dann wiederum als Faltung der Wahrscheinlichkeiten der sie bildenden Trajektorienstückchen. Wir erhalten also für die Wahrscheinlichkeit einer einzelnen Trajektorie

$$P\Big(\alpha_u(t=0) \to \alpha_u(t=1/N) \to \alpha_u(t=2/N) \to \cdots \to \alpha_u(t=(t_f N)/N)\Big) \tag{5.57}$$

$$= \int_{-\pi}^{\pi} \prod_t \frac{dl(t)}{2\pi} e^{N\Delta t i l(t)\dot{\alpha}_u(t)} \hat{P}_{Nt}^{N(t+\Delta t)}(l(t))$$

$$= \int_{-\pi}^{\pi} \prod_t \frac{dl(t)}{2\pi} \exp\left\{ -N \sum_t \mathcal{L}(l(t), \alpha_u(t), \dot{\alpha}_u(t))\Delta t \right\}, \tag{5.58}$$

wobei jeweils $t \in \left\{ \frac{1}{N}, \frac{2}{N}, \ldots, \frac{t_f N}{N} \right\}$ ist und die Funktion $\mathcal{L}$ durch

$$\mathcal{L}(l(t), \alpha_u(t), \dot{\alpha}_u(t)) = -il(t)\dot{\alpha}_u + K\alpha - K(\alpha - \alpha_u(t))e^{-il(t)} - K\alpha_u(t)e^{il(t)} - il(t) \tag{5.59}$$

gegeben ist.

Anschließend summieren wir über alle möglichen Pfade, welche die Randbedingung erfüllen. Die maximale mögliche Energiedichte ist α, d. h. $\alpha_u(t) \in [0, \alpha]$ für $t \in \left\{ \frac{1}{N}, \frac{2}{N}, \ldots, \frac{t_f N - 1}{N} \right\}$. Es ergibt sich als Gesamtwahrscheinlichkeit

$$\begin{aligned} P\left(\alpha_u(0) \to \alpha_u(t_f) = 0\right) &= \int_0^{\alpha} \prod_{t \in \left\{ \frac{1}{N}, \frac{2}{N}, \ldots, \frac{t_f N - 1}{N} \right\}} d\alpha_u(t) \\ &\quad \cdot P\Big(\alpha_u(t=0) \to \alpha_u(t=1/N) \to \cdots \to \alpha_u(t=(t_f N)/N)\Big) \\ &= \int_0^{\alpha} \prod_{t < t_f} d\alpha_u(t) \\ &\quad \cdot \int_{-\pi}^{\pi} \prod_t \frac{dl(t)}{2\pi} \exp\left\{ -N \sum_t \mathcal{L}(l(t), \alpha_u(t), \dot{\alpha}_u(t))\Delta t \right\} \end{aligned} \tag{5.60}$$

Im Limes $N \to \infty$, d. h. $\Delta t \to dt$, ergibt sich daraus formal ein Pfadintegral

$$P(\alpha_u(0) \to \alpha_u(t_f) = 0) = \int_{\alpha_u(0)}^{\alpha_u(t_f)} \mathcal{D}\alpha_u(t) \int \mathcal{D}l(t) \exp\left\{-N \int_0^t \delta t \mathcal{L}(l(t), \alpha_u(t), \dot{\alpha}_u(t))\right\}. \tag{5.61}$$

Die Integrale werden im thermodynamischen Limes $N \to \infty$ durch ihre Sattelpunkte dominiert, d. h. die Wahrscheinlichkeit der *typischen* Trajektorie von $(t = 0, \alpha_u(0))$ nach $(t = t_f, \alpha_u(t_f) = 0)$ dominiert die Gesamtwahrscheinlichkeit. Wegen der zweiten Randbedingung bei t_f ist diese Trajektorie nicht identisch mit der im vorherigen Abschnitt berechneten typischen Trajektorie des Algorithmus überhaupt, die auf einem Plateauwert der Energiedichte verbleibt.

Der Sattelpunkt des Pfadintegrals (5.61) ist gegeben durch die Euler-Lagrange-Gleichungen

$$0 = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{l}(t)} - \frac{\partial \mathcal{L}}{\partial l(t)} \tag{5.62}$$

$$\text{und } 0 = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\alpha}_u(t)} - \frac{\partial \mathcal{L}}{\partial \alpha_u(t)}. \tag{5.63}$$

Da $l(t)$ keine dynamische Variable ist ($\dot{l}(t)$ kommt nicht in der Lagrange-Funktion vor), ergibt (5.62)

$$0 = -\frac{\partial \mathcal{L}}{\partial l} = i\dot{\alpha}_u(t) - iK(\alpha - \alpha_u(t))e^{-il(t)} + iK\alpha_u(t)e^{il(t)} + i. \tag{5.64}$$

Die Euler-Lagrange-Gleichung (5.63) für $\alpha_u(t)$ ergibt

$$0 = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\alpha}_u} - \frac{\partial \mathcal{L}}{\partial \alpha_u} = i\dot{l}(t) + Ke^{-il(t)} - Ke^{il(t)}. \tag{5.65}$$

Durch Substitution

$$\kappa(t) = e^{il(t)} \tag{5.66}$$

erhalten wir das folgende System gekoppelter Differentialgleichungen erster Ordnung:

$$\dot{\alpha}_u(t) = -1 - K\alpha_u(t)\kappa(t) + K\frac{\alpha - \alpha_u(t)}{\kappa(t)} \tag{5.67}$$

$$\dot{\kappa}(t) = K\kappa^2(t) - K. \tag{5.68}$$

In unserem Fall sind zwei Randbedingungen für $\alpha_u(t)$ gegeben. Zum einen der typische Startwert $\alpha_u(0) = \alpha/2$, zum anderen aus der vorgegebenen Lösungszeit t_f folgend $\alpha_u(t_f) = 0$. Für $\kappa(t)$ gibt es dagegen keine Randbedingungen.

Eine triviale Lösung der zweiten Differentialgleichung (5.68) ist $\kappa(t) \equiv 1$, was durch Einsetzen in die erste Gleichung auf

$$\dot{\alpha}_u(t) = -1 - K\alpha_u(t) + K(\alpha - \alpha_u(t)) = K\alpha - 1 - 2K\alpha_u(t) \tag{5.69}$$

führt. Dies ist dann aber wieder die Differentialgleichung (5.24), welche wir in Abschnitt 5.5.2 unter der Annahme einer Poissonverteilung für $p_t(s, u)$ für die typische Trajektorie hergeleitet hatten, die der Algorithmus mit Wahrscheinlichkeit eins durchläuft und die auf einen Plateauwert der Energiedichte führt. Setzt man diese Lösung in (5.59) ein, so erhält man mit $\kappa(t) \equiv 1 \Leftrightarrow l(t) \equiv 0$

$$\mathcal{L}(\kappa(t) \equiv 1, \alpha_u(t), \dot{\alpha}_u(t)) = K\alpha - K(\alpha - \alpha_u(t)) - K\alpha_u(t) = 0, \tag{5.70}$$

d. h. diese Trajektorie hat wegen (5.61) tatsächlich Wahrscheinlichkeit eins im thermodynamischen Limes.

Allerdings ist diese Lösung instabil, denn für $\kappa(t) < 1$ ist $\dot{\kappa}(t) < 0$ und für $\kappa(t) > 1$ ist $\dot{\kappa}(t) > 0$. Sobald $\kappa(t)$ von 1 abweicht, wird also die typische Trajektorie verlassen.

Für $\kappa(t) < 1$ ist wegen (5.67) $\dot{\alpha}_u(t) > K\alpha - 1 - 2K\alpha_u(t)$, d. h. die zugehörige Lösung $\alpha_u(t)$ liegt über der typischen Trajektorie (5.26) des Algorithmus. Da schon für diese $\lim_{t\to\infty} \alpha_u(t) > 0$ gilt, kann für $\kappa(t) < 1$ keine Lösung zur Randbedingung $\alpha_u(t_f) = 0$ existieren.

Die allgemeine Lösung erhalten wir, indem wir zunächst die zweite Gleichung unter der Bedingung $\kappa(t) > 1$ mit Trennung der Variablen lösen. Es ergibt sich

$$\kappa(t) = \frac{1 + Ae^{2Kt}}{1 - Ae^{2Kt}}, \tag{5.71}$$

wobei die Konstante A erst durch die Randbedingungen von $\alpha_u(t)$ festgelegt wird. Es muss aber $A > 0$ wegen $\kappa(t) > 1$ gelten.

Die DGL (5.67) für $\alpha_u(t)$ bringen wir zunächst in die Form

$$\alpha_u(t) = -K\left(\kappa(t) + \frac{1}{\kappa(t)}\right)\alpha_u(t) - 1 + \frac{K\alpha}{\kappa(t)} \tag{5.72}$$

$$=: f(t)\alpha_u(t) + g(t) \tag{5.73}$$

und erhalten als Lösung

$$\alpha_u(t) = \left[\alpha_u(0) + \int_0^t d\tau g(\tau) e^{-\int_0^\tau d\tau' f(\tau')}\right] e^{\int_0^t d\tau' f(\tau')}, \tag{5.74}$$

wovon man sich durch Ableiten überzeugen kann. Dabei wurde die Randbedingung $\alpha_u(t = 0) = \alpha_u(0)$ bereits eingesetzt. Beide Integrale in (5.74) lassen sich lösen.

Allerdings konnte für die Lösung des Integrals über τ keine kompakte Form gefunden werden, so dass wir nur die Integration über τ' angeben. Als Lösung ergibt sich

$$\alpha_u(t) = \alpha_u(0)e^{-2Kt}\frac{1-A^2e^{4Kt}}{1-A^2} + \int_0^t d\tau \left(-1 + K\alpha\frac{1-Ae^{2Kt}}{1+Ae^{2Kt}}\right) e^{-2K(t-\tau)}\frac{1-A^2e^{4Kt}}{1-A^2e^{4K\tau}}. \tag{5.75}$$

Die zweite Randbedingung $\alpha_u(t_f) = 0$ ist erfüllt, wenn

$$0 = 1 - A^2e^{4Kt_f} \tag{5.76}$$

gilt, d. h.

$$A = e^{-2Kt_f}. \tag{5.77}$$

Setzt man dies in (5.75) ein, so erhält man den in Abbildung 5.13 für $t_f = 3$ dargestellten Verlauf der Energiedichte. Man erkennt, dass zwar $\alpha_u(3) = 0$ ist, allerdings ist dies nicht die erste Nullstelle von $\alpha_u(t)$, man findet eine weitere bei $t'_f \approx 2.817$. Der Algorithmus stoppt aber, sobald er eine Lösung gefunden hat, d. h. der Verlauf von $\alpha_u(t)$ ist unphysikalisch für $t > t'_f$.

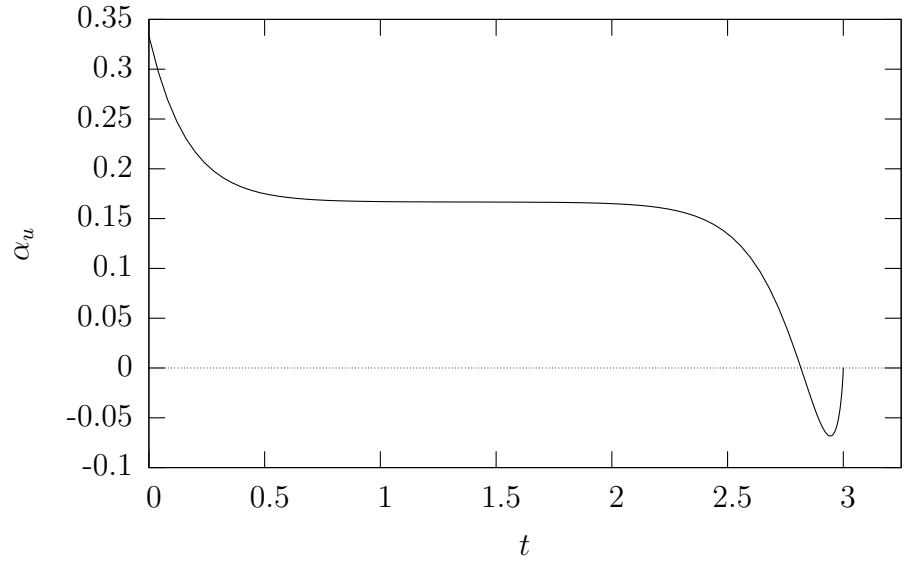


Abbildung 5.13: 3-XOR-SAT ($\alpha = 2/3$): Der Verlauf der unphysikalischen Lösung zur Randbedingung $\alpha_u(t_f = 3) = 0$. Man erkennt, dass die Wahl der Konstanten $A = e^{-2Kt_f}$ nicht die korrekte Trajektorie wiedergibt, da eine weitere Nullstelle $t'_f < t_f$ existiert.

Die erste Nullstelle $\alpha_u(t_f) = 0$ haben numerisch bestimmt und damit den korrekten Wert der Konstanten A festgelegt. Abbildung 5.14 zeigt Trajektorien für $\alpha = 2/3$, Lösungszeiten $t_f \in \{3, 6, 9\}$ und drei verschiedene Anfangsenergiedichten $\alpha_u(0) \in$

$\{0.1, 0.2, 0.3\}$. Insgesamt sind in der Abbildung also 9 Trajektorien dargestellt, jede entspricht einer anderen Kombination von $\alpha_u(0)$ und t_f. Man findet darin den in Abschnitt 5.4 beschriebenen Verlauf wieder: Unabhängig von $\alpha_u(0)$ equilibrieren alle Trajektorien nach ca. 1 MC-Sweep auf einen nahezu identischen Plateauwert und liegen dort übereinander. Numerisch sind die Trajektorien im weiteren Verlauf bis ca. $t_f - 1$ nicht unterscheidbar und erst dann führt eine exponentiell seltene Fluktuation zum Finden einer Lösung. Die in den numerischen Experimenten in Abbildung 5.4 sichtbaren Fluktuationen um den Plateauwert für $1 < t < t_f - 1$ werden von der analytischen Lösung nicht wiedergegeben. Diese beschreibt die typische nach Zeit t_f zur Lösung führende Trajektorie, in dieser mitteln sich die Fluktuationen (abgesehen von der letzten, durch die Ranbedingung vorgegebene) weg.

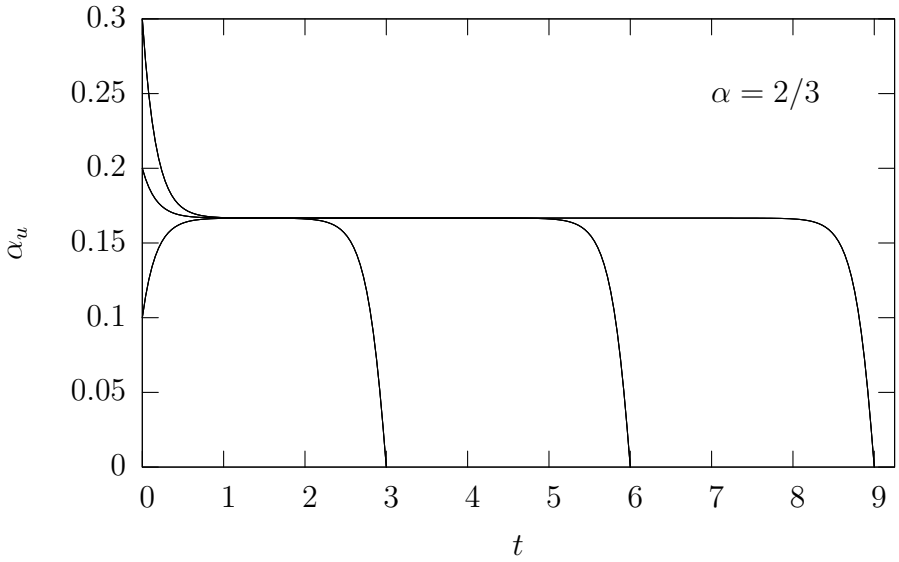


Abbildung 5.14: 3-XOR-SAT: Trajektorien bei $\alpha = 2/3$ für verschiedene Randbedingungen für $\alpha_u(0)$ und $t_f \in \{3, 6, 9\}$. Alle Trajektorien equilibrieren zunächst auf annähernd dasselbe Plateau. Eine Lösung wird durch eine große Fluktuation der Dauer von ca. 1 MC-Sweep gefunden.

Um die exponentiell kleine Wahrscheinlichkeit der zur Lösung führenden Trajektorie zu bestimmen, müssen wir zunächst die Wirkung

$$S(\mathcal{L}(\kappa(t), \alpha_u(t), \dot{\alpha}_u(t))) = \int_0^{t_f} dt \mathcal{L}(\kappa(t), \alpha_u(t), \dot{\alpha}_u(t)) \tag{5.78}$$

bestimmen. Zur Vereinfachung können wir in (5.59) zunächst (5.66) für $l(t)$ und anschließend (5.67) für $\dot{\alpha}_u(t)$ einsetzen und erhalten

$$\begin{aligned} &S(\mathcal{L}(\kappa(t), \alpha_u(t), \dot{\alpha}_u(t))) \\ &\quad = \int_0^{t_f} dt \left(-\ln(\kappa(t))(\dot{\alpha}_u + 1) + K\alpha - K\frac{\alpha - \alpha_u(t)}{\kappa(t)} - K\alpha_u(t)\kappa(t) \right) \end{aligned}$$

$$= K \int_0^{t_f} \left(-\ln(\kappa(t)) \left(-\alpha_u(t)\kappa(t) + \frac{\alpha - \alpha_u(t)}{\kappa(t)} \right) + \alpha - \frac{\alpha - \alpha_u(t)}{\kappa(t)} - \alpha_u(t)\kappa(t) \right). \tag{5.79}$$

Die numerischen Ergebnisse der Integration bei $\alpha = 2/3$ sind in Abbildung 5.15 für verschiedene Anfangsbedingungen $\alpha_u(0)$ dargestellt. Für die typische Anfangsbedingung $\alpha_u(0) = \alpha/2$ fällt die Wirkung zunächst schnell monoton und hat für $t_f > 1$ praktisch ihren asymptotischen Wert erreicht.

Mit Gleichung (5.61) erhalten wir aus der Wirkung die Wahrscheinlichkeit der Trajektorie durch

$$P(\alpha_u(0) \to \alpha_u(t_f) = 0) = \exp\{-NS\} \tag{5.80}$$

und daraus mit (5.46) und (5.43) die typische Lösungszeit

$$t_{\text{sol}} \sim \lim_{t_f \to \infty} e^{NS}. \tag{5.81}$$

Bisher haben wir stets einen Algorithmus *ohne Restarts* untersucht. Betrachten wir aber nun noch einmal Abbildung 5.15, so stellen wir fest, dass für kleinere als die typische Anfangsenergiedichte die Wirkung zunächst ein Minimum bei sehr kleinen Lösungszeiten besitzt. Dieses entspricht Trajektorien, die schon nahe (in Energie, d. h. Anzahl der verletzten Klauseln) an einer Lösung starten und dann mehr oder weniger direkt zu ihr verlaufen.

Ein Algorithmus *mit Restarts* könnte dies möglicherweise ausnutzen, wenn die exponentiell kleine Wahrscheinlichkeit $\rho(\alpha_u(0))$, näher an einer Lösung zu starten, durch die exponentiell schnellere Rechenzeit einer solchen Trajektorie überkompensiert wird. Dazu ist im Inset von Abbildung 5.15

$$\tau^{\text{restart}}(\alpha_u(0), t_f) = -\lim_{N \to \infty} 1/N \ln(\rho(\alpha_u) P(\alpha_u(0) \to \alpha_u(t_f) = 0)) \tag{5.82}$$

aufgetragen, d. h. die Wahrscheinlichkeit der atypischen Trajektorie wurde wie in (5.49) mit der Wahrscheinlichkeit der atypischen Startkonfiguration gewichtet. Letztere ist dabei numerisch über (5.47) aus der in [81] bestimmten Entropie $s(\alpha_u)$ der Zustände mit Energiedichte α_u gegeben.

Man sieht, dass $\tau(\alpha_u(0), t_f)$ minimal wird für die *typische Anfangsbedingung* $\alpha_u(0) = \alpha/2 = 1/3$. Für 3-XOR-SAT ist die Entropie von Startkonfigurationen nahe einer Lösung somit zu gering, die Wahrscheinlichkeit, dass der Algorithmus einer Trajektorie folgt, die nahe an der Lösung startet und direkt zu ihr hinläuft, ist exponentiell geringer, als über eine Trajektorie mit typischer Startkonfiguration zur Lösung zu finden.

Da $P(\alpha_u(0) \to \alpha_u(t_f) = 0)$ schon für $t_f > 1$ seinen Plateauwert erreicht hat und da für die Wahrscheinlichkeit der typischen Anfangsbedingung $\rho(\alpha/2) \to 1$ gilt, unterscheidet sich bei XOR-SAT die Lösungszeit von Algorithmen mit oder ohne

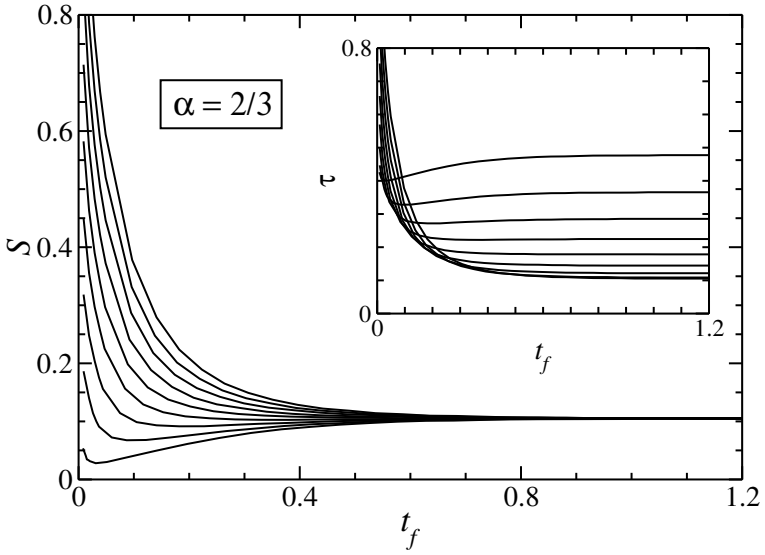


Abbildung 5.15: 3-XOR-SAT: Wirkung für Trajektorien mit unterschiedlichen Anfangsenergiedichten $\alpha_u(0) = 0.02, 0.06, 0.1, \ldots, 0.34$ (von unten nach oben) und unterschiedlichen Lösungszeiten t_f. Im Inset ist der Logarithmus τ der vorausgesagten Lösungszeit t_{sol} für dieselben Anfangsenergiedichten dargestellt, dort aber für große t_f in umgekehrter Reihenfolge.

Restarts nicht und ist gegeben durch (5.81). Sie ist in Abbildung 5.16 zusammen mit numerischen Daten für Walk-SAT mit Restarts nach $Nt_f = 3N$ Schritten (Schönings Algorithmus, [72]) dargestellt. Wegen der exponentiell ansteigenden Rechenzeiten konnten numerisch nur Systeme bis $N = 70$ untersucht werden. Dabei scheinen sich sehr viel kleinere Lösungszeiten zu ergeben. Allerdings sieht man im Inset für $\alpha = 0.4$ und $\alpha = 0.42$, wo wegen des kleineren Exponenten Systemgrößen bis $N = 1000$ betrachtet werden können, dass die Finite-Size-Effekte sehr bedeutend sind und der thermodynamische Limes von Systemgrößen $N = 70$ noch nicht annähernd wiedergegeben werden kann.

Qualitativ stimmen die numerischen Ergebnisse und analytischen Voraussagen überein, insbesondere wird der Anstieg oberhalb der Schwelle bei $\alpha_w \simeq 0.33$ gut wiedergegeben. Ebenso liegt der Wert am SAT/UNSAT-Übergang bei $\alpha_c = 0.918$ mit $1/N \ln t_{\mathrm{sol}} \simeq 0.249$ relativ nahe an der Worst-Case-Schranke von Schönings-Algorithmus, welche $\ln(4/3) \simeq 0.288$ beträgt.

Zufälliges K-SAT

Bei der Analyse für K-SAT verfahren wir wieder analog zu K-XOR-SAT und gehen deshalb hier nur auf die wesentlichen Unterschiede ein. Wie schon in Abschnitt 5.5.2 wird beim Negieren einer Variablen eine erfüllte Klausel im Mittel nur mit

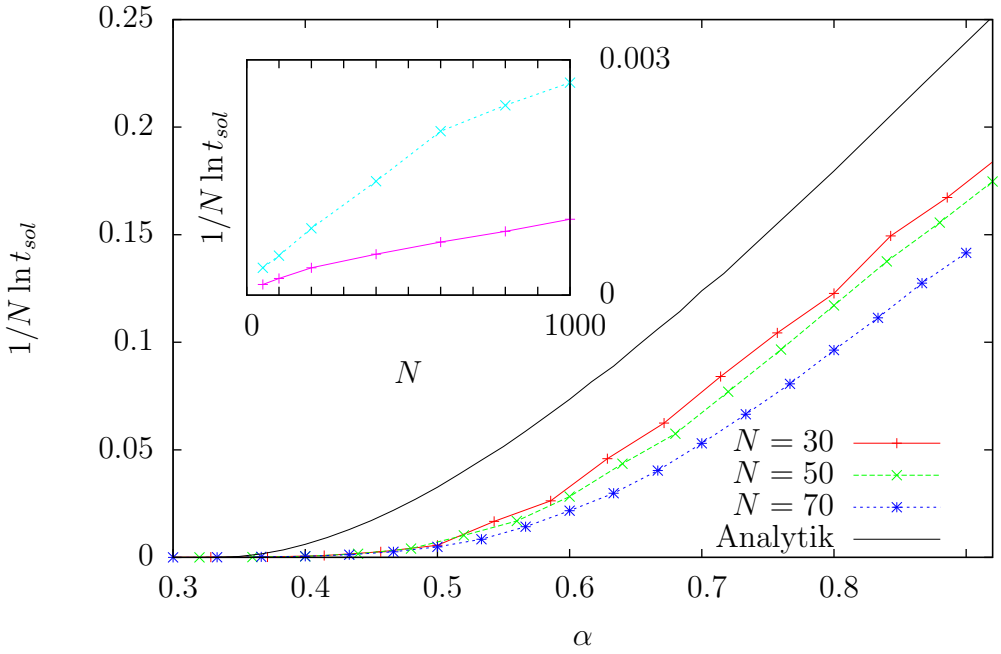


Abbildung 5.16: 3-XOR-SAT: Lösungszeit für Walk-SAT mit Restarts nach $Nt_f = 3N$ Schritten. Im Inset sieht man am Beispiel von $\alpha = 0.4$ (durchgezogene Linie) und $\alpha = 0.42$ (gepunktete Linie), dass die Finite-Size-Effekte noch sehr groß sind (Linien nur zur Orientierung). Die analytisch für diese beiden α dort vorausgesagten Werte sind $\frac{1}{N} \ln t_{\mathrm{sol}} = 0.0061$ bzw. 0.0099.

Wahrscheinlichkeit

$$\mu = 1/(2^K - 1) \tag{5.83}$$

unerfüllt. Ist die negierte Variable in s erfüllten Klauseln enthalten, so ist die Wahrscheinlichkeit, dass genau k Klauseln unerfüllt werden demnach unter Annahme der Unabhängigkeit der Klauseln gegeben durch

$$\sum_{k=0}^{s} \binom{s}{k} \mu^k (1-\mu)^{s-k}. \tag{5.84}$$

Mit Wahrscheinlichkeit $p_t^{(\mathrm{flip})}(s,u)$, gegeben durch (5.15), kommt die zu negierende Variable v^* in s erfüllten und u unerfüllten Klauseln vor. Durch Negieren von v^* werden k der s Klauseln unerfüllt und alle u Klauseln erfüllt, d. h. die Wahrscheinlichkeit für eine Änderung der Energiedichte um Δe ist gegeben durch

$$P_T^{T+1}(\Delta e) = \sum_{s,u=0}^{\infty} p_t^{(\mathrm{flip})}(s,u) \sum_{k=0}^{s} \binom{s}{k} \mu^k (1-\mu)^{s-k} \delta_{\Delta e, k-u}. \tag{5.85}$$

Analog zur Vorgehensweise für K-XOR-SAT und ebenfalls unter Annahme einer Poissonverteilung (5.50) für $p_t(s,u)$ erhalten wir das Pfadintegral (5.61)

$$P(\alpha_u(0) \to \alpha_u(t_f)) = \int_{\alpha_u(0)}^{\alpha_u(t_f)} \mathcal{D}\alpha_u(t) \int \mathcal{D}\kappa(t) \exp\left\{-N \int_0^t \delta t \mathcal{L}(\kappa(t), \alpha_u(t), \dot{\alpha}_u(t))\right\}, \tag{5.86}$$

wobei de Lagrange-Funktion hier durch

$$\begin{aligned}&\mathcal{L}(\kappa(t),\alpha_u(t),\dot{\alpha}_u(t)) \\ &= -\ln(\kappa(t))(\dot{\alpha}_u(t)+1) + K\alpha - K(\alpha-\alpha_u(t))\left(1-\mu+\frac{\mu}{\kappa(t)}\right) - K\alpha_u(t)\kappa(t)\end{aligned} \tag{5.87}$$

gegeben ist.

Als Euler-Lagrange-Gleichungen für den Sattelpunkt ergeben sich daraus

$$\begin{aligned}\dot{\alpha}_u(t) &= -1 - K\alpha_u(t)\kappa(t) + K\mu\frac{\alpha-\alpha_u(t)}{\kappa(t)} \\ \dot{\kappa}(t) &= K\kappa^2(t) - K(1-\mu)\kappa(t) - K\mu\end{aligned} \tag{5.88}$$

mit Lösung

$$\begin{aligned}\kappa(t) &= \frac{1+\mu A e^{2Kt}}{1-Ae^{2Kt}} \\ \alpha_u(t) &= \alpha_u(0)e^{-(1+\mu)Kt}\frac{1-Ae^{(1+\mu)Kt}}{1-A}\cdot\frac{1+\mu Ae^{(1+\mu)Kt}}{1+\mu A} \\ &+ \int_0^t d\tau\left(-1+\mu K\alpha\frac{1-Ae^{(1+\mu)Kt}}{1+\mu Ae^{(1+\mu)Kt}}\right)e^{-(1+\mu)K(t-\tau)}\frac{1-Ae^{(1+\mu)Kt}}{1-Ae^{(1+\mu)K\tau}}\cdot\frac{1+\mu Ae^{(1+\mu)Kt}}{1+\mu Ae^{(1+\mu)K\tau}}.\end{aligned} \tag{5.89}$$

Der noch freie Parameter A ist dabei auch hier durch die Randbedingung $\alpha_u(t_f) = 0$ gegeben und wird numerisch bestimmt. Wir zeigen die Ergebnisse wieder für $K = 3$ exemplarisch bei einem α mit $\alpha_w < \alpha < \alpha_c$. Die aus der numerischen Integration folgenden Trajektorien sind in Abbildung 5.17 dargestellt. Qualitativ findet man dasselbe Verhalten wie für 3-XOR-SAT in Abbildung 5.14 gezeigt. Unabhängig von der Anfangsbedingung wird nach ca. einem MC-Sweep der Plateauwert erreicht, wegen des reduzierten exponentiellen Faktors $e^{-(1+\mu)Kt}$ etwas später als bei 3-XOR-SAT. Knapp 2 MC-Sweeps vor der vorgegebenen Lösungszeit verlässt die Trajektorie das Plateau und führt in der verbleibenden Zeit zur Lösung.

Wie bei XOR-SAT in Gleichung (5.79) berechnen wir die Wirkung als Funktion der Lösungszeit, die Ergebnisse der numerischen Integration sind in Abbildung 5.18 dargestellt. Nach Erreichen des Plateauwerts ist die Wahrscheinlichkeit, eine Lösung zu finden, unabhängig von der Ausgangsenergiedichte.

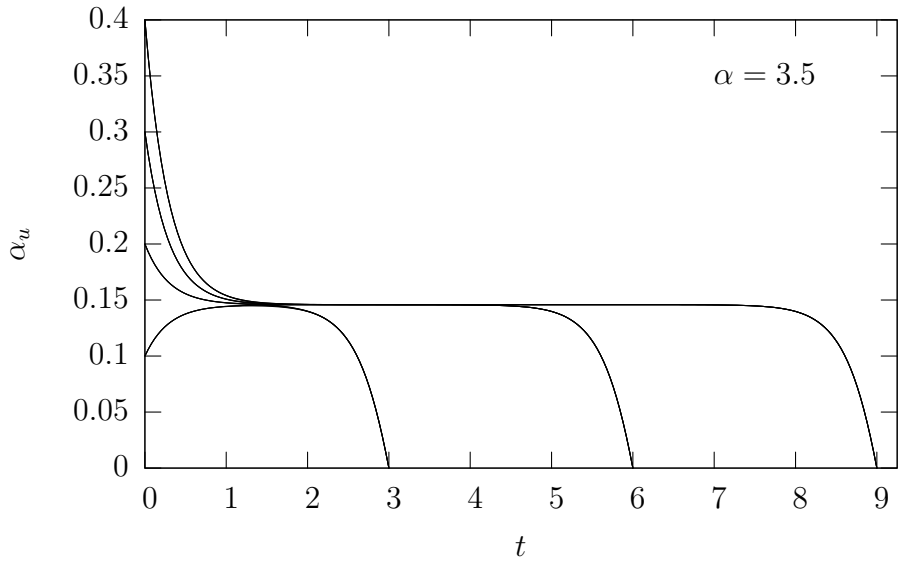


Abbildung 5.17: 3-SAT: Energiedichten $\alpha_u(t)$ für verschiedene Anfangsbedingungen $\alpha_u(0)$ und verschiedene Lösungszeiten $t_f \in \{3, 6, 9\}$.

Hier bestimmen wir die typische Lösungszeit für einen Algorithmus ohne Restart, da die mikrokanonische Entropie der Anfangszustände mit Energiedichte α_u nicht bekannt ist. Der exponentiell dominierende Beitrag ist wiederum gegeben durch $t_{\text{sol}} \sim \lim_{t_f \to \infty} e^{NS}$. Die resultierenden Lösungszeiten zeigen wir in Abbildung 5.19 im Vergleich mit numerischen Daten von Systemen der Größe $N = 30$, $N = 50$ und $N = 70$. Für kleine α stimmen die Kurven annähernd überein, während für $\alpha > \alpha_w \simeq 2.7$ die numerischen Daten tendenziell zunächst unter der analytischen Kurve liegen, im späteren Verlauf dann diese aber sogar schneiden. Qualitativ findet man aber eine gute Übereinstimmung der unter der recht groben Poissonnäherung bestimmten analytischen Lösungszeiten mit den numerischen Werten.

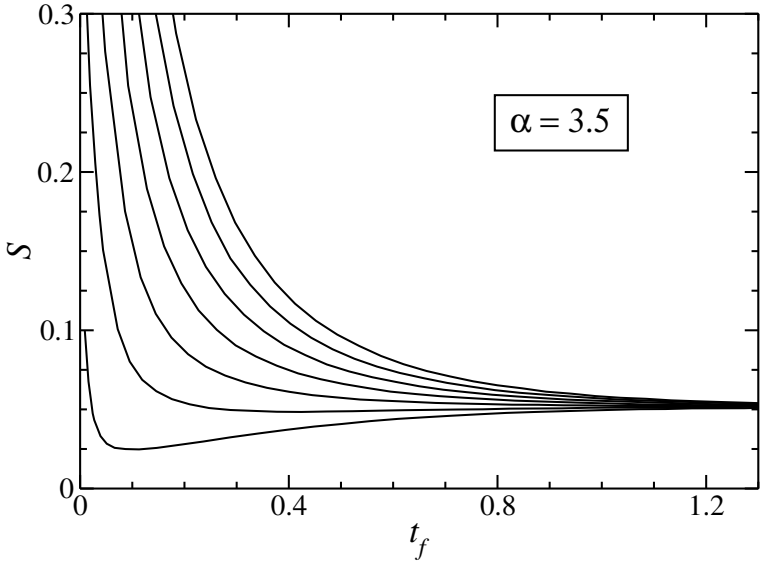


Abbildung 5.18: 3-SAT: Wirkung als Funktion der Lösungszeit t_f für verschiedene Anfangsbedingungen $\alpha_u(0) = 0.1, 0.3, 0.5, \ldots, 1.3$ von oben nach unten.

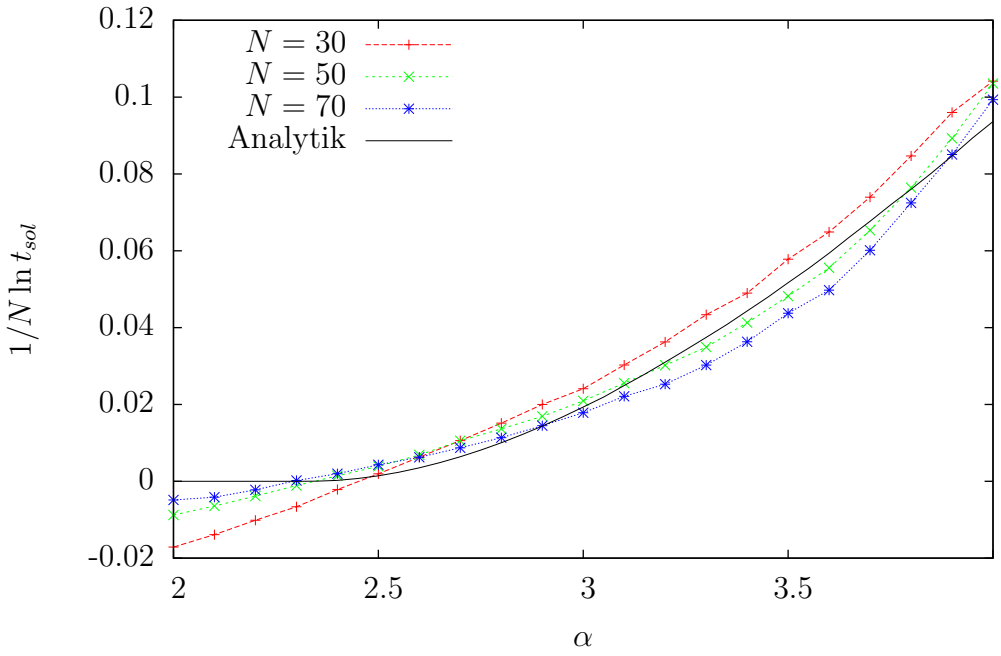


Abbildung 5.19: 3-SAT: Lösungszeiten von Walk-SAT ohne Restarts für verschiedene Systemgrößen im Vergleich zur analytischen Voraussage (Linien bei den numerischen Daten nur zur Orientierung).

6 Zusammenfassung

In dieser Arbeit wurde die Grundzustandsstruktur ungeordneter Systeme und die Dynamik von Grundzustandsalgorithmen betrachtet.

Zunächst wurde ein verbesserter Algorithmus zur Bestimmung von Grundzuständen dreidimensionaler Ising-Spin-Systeme vorgestellt, der auf dem Genetischen Renormalisierungs-Algorithmus von Houdayer und Martin basiert. Mit diesem Verfahren können Grundzustände bis zur Systemgröße $N = 14^3$ bestimmt werden. Als Anwendung wurde die Landschaft niedrigenergetischer Anregungen bei Systemen mit gaußverteilten Wechselwirkungen betrachtet. Dabei wurde in Übereinstimmung mit den Voraussagen des Mean-Field-Modells festgestellt, dass die Wahrscheinlichkeit einer systemweiten Anregung mit endlicher Energie nicht mit wachsender Systemgröße abnimmt.

Als Zweites wurde numerisch die Grundzustandslandschaft des Vertex-Cover-Problems untersucht. Dies ist ein kombinatorisches Optimierungsproblem, dessen Ursprünge in der Theoretischen Informatik liegen. Die typischen Eigenschaften für das Ensemble von Zufallsgraphen sind sehr gut mit Methoden der statistischen Physik, insbesondere der Untersuchung von Spingläsern, beschreibbar.

Für kleine Systeme wurden alle minimalen Vertex-Cover bestimmt und entsprechend ihres Hammingabstands in Cluster eingeteilt. Für $c < e$ liegen die Zustände in einer zusammenhängenden Menge, es gibt also genau einen Cluster von Lösungen. Dies wurde darüber hinaus für baumartige Graphen exakt gezeigt, welche für Zufallsgraphen unterhalb der Perkolationsschwelle typisch sind. Für größere Systeme wurde die Clusterung mit ballistischer Suche durchgeführt und dabei festgestellt, dass die Clusteranzahl für $c > e$ mindestens logarithmisch wächst. Dies stimmt überein mit der analytischen Voraussage, dass die RS-Lösung oberhalb von c instabil wird und die Ebene der Replika-Symmetriebrechung (RSB) höher als 1-RSB ist. Bestätigt wurde dies durch die Analyse des Spektrums der Knoten-Knoten-Korrelation, bei der für Konnektivitäten $c \geq 4$ mehr als ein reiner Zustand gefunden wurde.

Die Anwendung eines hierarchischen Clusterungsverfahrens zeigte qualitativ für Konnektivitäten $c > e$ die Existenz mehrerer Ebenen einer Clusterstruktur, so wie man es bei kontinuierlicher RSB erwartet. Quantitativ entspricht die durch den Algorithmus vorhergesagte Clusterstruktur für $c > e$ mehr und mehr der tatsächlichen Struktur der Zustände im Realraum, ein Indiz dafür, dass in diesem Bereich der Konnektivität wirklich eine ultrametrische Struktur und damit kontinuierliche RSB vorhanden ist.

Schließlich wurde als Drittes die Dynamik eines Algorithmus zur Bestimmung von Lösungen des K-SAT-Problems untersucht. Dieses hat aus Sicht der statistischen Physik vergleichbare Eigenschaften wie das Vertex-Cover-Problem. Der hier verwendete Ansatz für die Beschreibung seltener Trajektorien wurde später auf einen Algorithmus zur Bestimmung minimaler VC übertragen [85].

Abhängig von einem Parameter α, der das Verhältnis der Anzahl der Bedingungen und der Anzahl der booleschen Variablen beschreibt, an die diese Bedingungen gestellt werden, konnten wir den Wert einer dynamischen Schwelle α_w bestimmen. Diese trennt zwei Phasen mit unterschiedlichem typischem Laufzeitverhalten:

- Für $\alpha < \alpha_w \simeq 2.71$ findet der Algorithmus eine Lösung typischerweise schnell, d. h. in einer Zeit linear in der Anzahl der Variablen können fast alle Instanzen gelöst werden. Für diesen Bereich linearer Lösungszeiten konnte das Verhalten des Algorithmus durch eine einfache Ratengleichung beschrieben werden.
- Für $\alpha > \alpha_w$ steigt die typische Lösungszeit exponentiell. Zunächst fällt die Anzahl der verletzten Klauseln in kurzer Zeit auf ein Plateau, anschließend fluktuiert das System um diesen Plateauwert. Diese Fluktuationen wurden als unabhängige Zufallsereignisse betrachtet und ihre Wahrscheinlichkeit durch einen Funktionalintegral-Ansatz bestimmt. Die exponentiell kleinen Wahrscheinlichkeiten einer solchen Fluktuation führen zu den beobachteten exponentiell großen Lösungszeiten.

Der zur Beschreibung der Dynamik verwendete Ansatz ließe sich prinzipiell verfeinern, um insbesondere die Verbesserung durch Heuristiken wie das Ausführen gieriger Schritte besser beschreiben zu können. Ausgehend davon könnten die Laufzeiten verschiedener stochastischer Algorithmen analytisch miteinander verglichen werden und damit Heuristiken systematisch verbessert werden.

Eine wichtige offene Frage ist der Zusammenhang zwischen dem Beginn exponentieller Lösungszeiten an der algorithmenabhängigen Schranke α_w und einer Veränderung der Grundzustandsstruktur des Ensembles durch das Auftreten exponentiell vieler Cluster bei $\alpha_d \approx 3.925$. Eine weitere Veränderung der Grundzustandslandschaft findet man bei $\alpha_s \approx 4.15$. Für den in dieser Arbeit betrachteten Algorithmus ist $\alpha_w < \alpha_d$. Inzwischen kennt man aber Varianten des Walk-SAT-Algorithmus, die in numerischen Experimenten bis zu dieser Schwelle [83] oder darüber hinaus [84] Lösungen typischerweise in linearer Zeit finden. Durch Übertragung des hier beschrieben Ansatzes könnte man versuchen, das Verhalten dieser Heuristiken analytisch zu beschreiben.

Für die betrachten kombinatorischen Optimierungsprobleme ist man dabei, das mean-field-Verhalten der Grundzustandslandschaft vollständig zu verstehen. Für Ising-Spingläser kennt man im mean-field-Fall bereits eine Lösung. Um auch für endlich-dimensionale Spingläser Fortschritte zu erzielen, wären ähnlich große algorithmische Fortschritte wünschenswert, wie sie bei der Weiterentwicklung der Walk-SAT-Lösungsalgorithmen in den letzten Jahren gemacht wurden. Für den in die-

ser Arbeit beschriebenen Genetischen Renormalisierungs-Algorithmus gibt es einige nahe liegende Verbesserungen, wie beispielsweise eine bevorzugte Auswahl von Zuständen mit niedrigerer Energie für die Renormalisierung. Ebenso denkbar wäre eine Übertragung der Ideen der Genetischen Renormalisierung auf kombinatorische Optimierungsprobleme wie das Vertex-Cover-Problem.

Aus Sicht der Informatik mit dieser Arbeit verbundene längerfristige Ziele sind einerseits die Beschreibung anwendungsbezogener Ensemble von Probleminstanzen, andererseits die Klärung des Worst-Case-Verhaltens und des damit verbundenen P-NP-Problems der Komplexitätstheorie.

Literaturverzeichnis

[1] M. Mézard, G. Parisi, M.A. Virasaro, *Spin glass theory and beyond*, (World Scientific, Singapore 1987).

[2] K.H. Fisher and J.A. Hertz, *Spin Glasses*, (Cambridge University Press, Cambridge 1991).

[3] A.K. Hartmann and H. Rieger (eds.), *New Optimization Algorithms in Physics*, (Wiley-VCH, Weinheim 2004).

[4] M.R. Garey and D.S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*, (W.H. Freeman & Company, San Francisco 1979).

[5] C.H. Papadimitriou, *Computational Complexity*, (Addison-Wesley, New York 1994).

[6] `http://www.claymath.org/millennium/P_vs_NP/`

[7] A. H. Land and A. G. Doig, Econometrica, **28**, 497 (1960).

[8] M.E.J. Newman and G.T. Barkema, *Monte Carlo Methods in Statistical Physics*, (Oxford University Press, Oxford 1999).

[9] D.P. Landau and K. Binder *A Guide to Monte Carlo Simulations in Statistical Physics*, (Cambridge University Press, Cambridge 2000).

[10] E. Marinari and G. Parisi, Europhys. Lett. **19**, 451 (1992).

[11] K. Hukushima and K. Nemoto, J. Phys. Soc. Jpn. **65**, 1604 (1996).

[12] K. Binder and A.P. Young, Rev. Mod. Phys. **58**, 801 (1986).

[13] A.P. Young (ed.), Spin glasses and random fields, (World Scientific, Singapore 1998).

[14] S.F. Edwards and P.W. Anderson, J. Phys. F **5**, 965 (1975).

[15] D. Sherrington and S. Kirkpatrick, Phys. Rev. Lett. **35**, 1792 (1975).

[16] G. Parisi, Phys. Rev. Lett. **43**, 1754 (1979); J. Phys. A **13**, 1101 (1980); **13**, 1887 (1980); **13**, L115 (1980); Phys. Rev. Lett. **50**, 1946 (1983).

[17] M. Talagrand, Annals of Mathematics, to appear. Announced in C.R.A.S. **337**, 111 (2003).

[18] A.P. Young, Phys. Rev. Lett. **51**, 1206 (1983).

[19] G. Parisi, F. Ritort and F. Slanina, J. Phys. A **26**, 3775 (1993).

[20] A. Billoire, S. Franz, and E. Marinari, J. Phys. A **36**, 15 (2003).

[21] J. Sinova, G. Canright, and A.H. MacDonald, Phys. Rev. Lett. **85**, 2609 (2000).

[22] J. Sinova, G. Canright, H.E. Castillo, and A.H. MacDonald, Phys. Rev. B **63**, 104427 (2001).

[23] G. Hed, A.P. Young, and E. Domany, Phys. Rev. Lett. **92**, 157201 (2004).

[24] G. Hed, A.K. Hartmann, D. Stauffer, and E. Domany, Phys. Rev. Lett. **86**, 3148 (2001).

[25] E. Domany, G. Hed, M. Palassini, and A.P. Young, Phys. Rev. B. **64**, 224406 (2001).

[26] S. Ciliberti and E. Marinari, J. Stat. Phys. **115**, 557 (2004).

[27] W.L. McMillan, J. Phys. C **17**, 3179 (1984).

[28] M.A. Moore and A.J. Bray, J. Phys. C **18**, L669, (1985).

[29] D.S. Fisher and D.A. Huse, Phys. Rev. Lett. **56**, 1601 (1986).

[30] A.J. Bray and M.A. Moore, J. Phys. C **17**, L463 (1984).

[31] W.L. McMillan, Phys. Rev. B **30**, 476 (1984).

[32] A.K. Hartmann, Phys. Rev. E **59**, 84 (1999).

[33] J. Houdayer and O.C. Martin, Phys. Rev. E **64**, 056704 (2001).

[34] O.C. Martin, in A.K. Hartmann and H. Rieger (eds.), *New Optimization Algorithms in Physics* (Wiley-VCH, Weinheim 2004).

[35] B. Kerninghan and S. Lin, Bell System Technical Journal **49**, 291 (1970).

[36] C. Reeves, in F. Glover and G.A. Kochenberger (eds.), *Handbook of Metaheuristics*, Kluwer, Boston (2003).

[37] A.K. Hartmann and H. Rieger, *Optimization Algorithms in Physics*, (Wiley-VCH, Weinheim 2002).

[38] N. Kawashima und M. Suzuki, J. Phys. A. **25**, 1055 (1992).

[39] M. Weigt and A.K. Hartmann, Phys. Rev. Lett. **84**, 6118 (2000).

[40] M. Bauer and O. Golinelli, Phys. Rev. Lett. **86**, 2621 (2001).

[41] A. Montanari, G. Parisi, F. Ricci-Tersenghi, J. Phys. A **37**, 2073 (2004).

[42] M. Mézard, T. Mora, R. Zecchina, Phys. Rev. Lett. **94**, 197205 (2005).

[43] P. Erdös and A. Rényi, *Publ. Math. Debrecen* **6**, 290 (1959).

[44] P. Erdös and A. Rényi, *Publ. Math. Inst. Hung. Acad. Sci.* **5**, 17 (1960).

[45] B. Bollobás, *Modern Graph Theory*, Springer, New York (1998).

[46] M. Weigt, Eur. Phys. J. B **28**, 369 (2002).

[47] M. Bauer and O. Golinelli, Eur. Phys. J. B **24**, 339 (2001).

[48] R.E. Tarjan and A.E. Trojanowski, SIAM J. Comp. **6**, 537 (1977).

[49] R.M. Karp and M. Sipser, Proc. of the 22nd Ann. IEEE Symp. on Found. of Comp., 364 (1981).

[50] M. Weigt and A.K. Hartmann, Phys. Rev. E **63**, 056127 (2001).

[51] A.K. Hartmann and M. Weigt, Theor. Comp. Sci. **265**, 199 (2001).

[52] M.N. Barber, in: C. Domb, J.L. Lebowitz (Eds.), *Phase Transition and Critical Phenomena*, vol. 8, Academic Press, London, 146 (1983).

[53] P.G. Gazmuri, Networks, **14**, 367 (1984).

[54] A.M. Frieze, Discr. Math. **81**, 171 (1990).

[55] M. Mézard, G. Parisi and R. Zecchina, Science **297**, 812 (2002).

[56] M. Mézard and R. Zecchina, Phys. Rev. E **66**, 056126 (2002).

[57] H. Zhou, Eur. Phys. J. B **32**, 265 (2003).

[58] M. Mézard, F. Ricci-Tersenghi, and R. Zecchina, J. Stat. Phys. **111**, 505 (2003).

[59] A.K. Hartmann, Phys. Rev. E **63**, 016106 (2001).

[60] A.K. Hartmann, J. Phys. A **33**, 657 (2000).

[61] R. Rammal, G. Toulouse, and M.A. Virasoro, Rev. Mod. Phys. **58** , 765 (1986).

[62] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, USA (1988).

[63] G.N. Lance and W.T. Williams, Comp. J. **9**, 373 (1967).

[64] J. Ward, J. of the Am. Stat. Association **58**, 236 (1963).

[65] R. Monasson, S. Kirkpatrick, B. Selman, L. Troyansky, **400**, 133 (1999).

[66] R. Monasson and R. Zecchina, Phys. Rev. Lett. **76**, 3881 (1996); Phys. Rev. E **56**, 1357 (1997).

[67] R. Zecchina, in A.K. Hartmann and H. Rieger (Eds.), *New Optimization Algorithms in Physics* (Wiley-VCH, Weinheim 2004).

[68] S. Cocco, L. Ein-Door, R. Monasson, in A.K. Hartmann and H. Rieger (Eds.), *New Optimization Algorithms in Physics* (Wiley-VCH, Weinheim 2004).

[69] B. Selman, H. Kautz, and B. Cohen, in *Cliques, Coloring and Satisfiability*, ed. by D.S. Johnson and M.A. Trick, DIMACS series vol. 26 (AMS, Providence, 1996).

[70] S.A. Cook, Proceedings 3rd ACM Symposium on Theory of Computing, 151 (1971).

[71] B. Monien, E. Speckenmeyer: Solving satisfiability in less than 2^n steps, *Discrete Applied Mathematics* **10**, 287 (1985).

[72] U. Schöning, Proc. 40th Sympos. on Foundations of Computer Science 1999, 410 (IEEE, New York, 1999); Algorithmica **32**, 615 (2002).

[73] K. Iwama and S. Tamaki: Improved upper bounds for 3-SAT. *Electronic Colloquium on Computational Complexity*, Report No. 53, 2003. Also: Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete algorithms, ACM, 328 (2004).

[74] S. Kirkpatrick and B. Selman, Science **264** 1297 (1994).

[75] E. Friedgut, J. Amer. Math. Soc. **12**, 1017 (1999).

[76] B. Hayes, *Can't get no satisfaction*, Am. Sci. **85** (2) 108 (1997).

[77] O. Dubois, R. Monasson, B. Selman, and R. Zecchina (Eds.), special issue Theoret. Comp. Sci. **265** (2001).

[78] G. Biroli, R. Monasson, and M. Weigt, Eur. Phys. J. B **14**, 551 (2000).

[79] S. Mertens, to appear in Random Struct. Algorithms (2005), online at `cs.CC/0309020`

[80] Survey-Propagation code at `http://www.ictp.trieste.it/~zecchina/SP`

[81] F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, Phys. Rev. E **63**, 026702 (2001).

[82] G. Semerjian and R. Monasson, Phys. Rev. E **67**, 066103 (2003).

[83] E. Aurell, U. Gordon, and S. Kirkpatrick, 18th Annual Conference on Neural Information Processing Systems (2004), online at `cond-mat/0406217`

[84] S. Seitz, M. Alava, P. Orponen, Proc. 8th International Conference on Theory and Applications of Satisfiability Testing, 475 (2005).

[85] A.K. Hartmann and M. Weigt, *Phase Transitions in Combinatorial Optimization Problems*, (Wiley-VCH, Weinheim 2005).

Danksagung

Bei der Entstehung dieser Arbeit haben mich viele Menschen unterstützt, ihnen soll dafür an dieser Stelle herzlich gedankt werden.

Bedanken möchte ich mich an erster Stelle bei Dr. Alexander Hartmann für seine intensive Betreuung und die Begeisterung, die er mir für dieses Arbeitsgebiet vermittelt hat. Sehr gerne erinnere ich mich an die Zusammenarbeit mit Dr. Martin Weigt, bei der die Ergebnisse zur Analyse des Walk-SAT-Algorithmus entstanden, ebenso an den Aufenthalt am LPTMS in Paris bei Prof. Olivier Martin. Mit ihm zusammen wurde die verbesserte Variante des Spinglas-Grundzustandsalgorithmus entwickelt und angewendet.

Frau Prof. Annette Zippelius danke ich für das Interesse an der Arbeit und die Übernahme des Korreferats, ebenso für die freundliche Arbeitsatmosphäre in der alten Villa und in der neuen Physik. Zahlreiche Anregungen zur Arbeit erhielt ich durch Diskussionen mit Dr. Bernd Burghardt, Dr. Timo Aspelmeier und Till Kranz.

Ganz besonderen Dank gilt meinen Eltern Maria und Jochen Barthel sowie Jörg Hahn, Karsten Roeseler und Katrin Radenbach für das sorgfältige Korrekturlesen, desweiteren Christian Wald für die organisatorische Unterstützung. Unbedingt erwähnen möchte ich die Sekretärinnen des Instituts für Theoretische Physik, die stets freundlich und mit vielen kleinen Dingen für ein gutes Klima am Institut gesorgt haben. Und nicht zuletzt geht der Dank an alle weiteren Freunde und Kollegen am Institut, an Axel, Carlo, Emmanuel, Garrit, Henning, Peter, Stefan, Stefan, Stephan, ...

Finanziell unterstützt wurde die Arbeit von der DFG durch Mittel aus dem Leibniz-Programm, AZ Zi209/6-1, durch die Volkswagenstiftung im Rahmen des Programms „Nachwuchsgruppen an Universitäten“ und durch den DAAD im Rahmen des Programms „Kurzstipendien für Doktoranden“.

Lebenslauf

Wolfgang Barthel
Humboldtallee 14
37073 Göttingen

Biographische Daten

Geburtsdatum:	19. August 1977
Geburtsort:	Halle (Saale)
Staatsangehörigkeit:	Deutsch

Akademischer Werdegang

07/1996	Abitur am Georg-Cantor-Gymnasium Halle (Saale) (Note 1,0)
10/1997–02/2002	Studium an der Georg-August-Universität Göttingen
04/1999	Vordiplom in Mathematik (Note 1,4)
07/1999	Vordiplom in Physik (Note 1,4)
01/2001–02/2002	Diplomarbeit zum Thema „Statistische Mechanik harter, lösbarer Erfüllbarkeitsprobleme“ unter Anleitung von Dr. Martin Weigt und Prof. Dr. Annette Zippelius
02/2002	Diplom in Physik (Note 1,1)
03/2002–11/2005	Promotion unter Anleitung von PD Dr. Alexander K. Hartmann am Institut für Theoretische Physik der Georg-August-Universität Göttingen
07/2000–09/2000	Aufenthalt am ICTP Trieste/Italien bei Prof. Riccardo Zecchina
10/2004–02/2005	Aufenthalt am LPTMS Orsay/Frankreich bei Prof. Olivier Martin

Göttingen, 8. November 2005