Monte-Carlo-Methoden I:
Der Wolff-Algorithmus
- Ausarbeitung -

Tobias Haas

19. Februar 2010

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation

Monte-Carlo-Methoden Bekanntermaßen ist in der statistischen Mechanik der statistische Mittelwert einer Zustandsgröße B definiert als

$$\langle B \rangle = \sum_{x \in \Omega} \Pi(x) B(x)$$

bzw. allgemeiner

$$\langle B \rangle = \int_{x \in \Omega'} \tilde{\Pi}(x) B(x) d\mu$$

wobei Ω der fragliche Phasenraum und Π bzw. $\tilde{\Pi}$ das normierte statistische Gewicht des Zustandes $x \in \Omega, \Omega'$ ist und μ eine geeignete Maßfunktion von Ω' (für viele physikalische Systeme z.B. das Lebesgue-Borel-Maß).

Leider ist diese Definition in der Praxis meist unbrauchbar, da der Phasenraum meist zu groß ist, als dass die Summe oder das Integral ausgewertet werden können. Einen möglichen Ausweg bietet hier die Statistik über das Gesetz der großen Zahlen:

$$\langle B \rangle \approx \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} B(x_k)$$

mit einer geeigneten Folge $(x_k)_{k \in \mathbb{N}}$. Was das genau bedeutet, soll in Abschnitt 2.1 etwas genauer beleuchtet werden. Interessant daran ist, dass somit $\langle B \rangle$ über eine endliche Summe approximiert werden kann, die insbesondere numerisch ausgewertet werden kann. Dies ist die Grundlage der sogenannten Monte-Carlo-Methode, der Name rührt von der Benützen zufallsbasierten Methode (wie im Casino) her.

Clusteralgorithmen Clusteralgorithmen sind eine Weiterentwicklung lokal operierdender (im Sinne von Operationen an einzelnen Partikel etc.) Algorithmen. Die Motivation leitet sich von folgendem Problem ab: Im Bereich von beispielsweise Phasenübergängen können die Autokorrelationszeiten sehr groß werden, was bedeutet, dass der Rechenaufwand zum Finden obiger Folge $(x_k)_{k \in \mathbb{N}}$

stark ansteigt. Damit wird das Verfahren unpraktikabel. Als Lösung entwickelt man einen Algorithmus, der nicht mehr lokal arbeitet, sondern ganze Bereiche (Cluster) auf einmal manipuliert. Damit wirkt man der Verlangsamung der Konvergenzgeschwindigkeit entgegen.
Ein weiterer Vorteil ist, dass Clusteralgorithmen häufig gut parallelisierbar sind.

1.2 Typen

Im Folgenden soll eine kurze weder vollständige noch repräsentative Auswahl von Algorithmen kurz vorgestellt werden.

- Swendsen-Wang-Algorithmus [12]
- Wolff-Algorithmus [14]
- Invaded Cluster [8]
- Probability-Change-Clusteralgorithmus [13]

Die Reihung ist chronologisch vorgenommen. Der Swendsen-Wang-Algorithmus [12] kann als Prototyp für viele andere Algorithmen verstanden werden. In diesem Zusammenhang soll nicht weiter auf die genaue Funktionsweise eingegangen werden, nur erwähnt, dass er auf dem Metropolis-Algorithmus aufbaut und Cluster mit einer festen Wahrscheinlichkeit p aufgebaut werden.
Der Wolff-Algorithmus basiert im Prinzip auf dem Swendsen-Wang-Algorithmus und nutzt eine interessante Besonderheit des betrachteten Spingittersystems aus, sodass er keine Zurückweisungen der vom Algorithmus vorgeschlagenen Züge aufweist (im Gegensatz zum Swendsen-Wang-Algorithmus). Auf diesen soll im Folgenden noch speziell in Kapitel 3.1 eingegangen werden.
Die beiden anderen Algorithmen sind für die Fragestellung nach kritischen Punkten optimiert, wobei der Probability-Change-Algorithmus mehr oder weniger auf dem Swendsen-Wang-Algorithmus aufbaut (mit variierender Wahrscheinlichkeit p). Der Invaded-Cluster-Algorithmus arbeitet nach einem anderen Prinzip, das hier aber nicht weiter erläutert werden soll.

Diese Algorithmen sind alle für Spinsysteme (Gitter) gebaut, für andere Systeme (mit kontinuierlichen Parametern beispielsweise) und Fragestellungen soll später in Kapitel 4 eingegangen werden.

Kapitel 2

Theorie

2.1 Monte-Carlo-Methoden

2.1.1 Mittelwerte

Wie schon in der Einleitung beschrieben, ist in der statistischen Mechanik der statistische Mittelwert einer Zustandsgröße B definiert als

$$\langle B \rangle = \sum_{x \in \Omega} \Pi(x) B(x)$$

bzw. allgemeiner

$$\langle B \rangle = \int_{x \in \Omega'} \tilde{\Pi}(x) B(x) d\mu$$

wobei Ω der fragliche Phasenraum und Π bzw. $\tilde{\Pi}$ das normierte statistische Gewicht des Zustandes $x \in \Omega$ bzw. $x \in \Omega'$ ist und μ eine geeignete Maßfunktion von Ω' (für viele physikalische Systeme z.B. das Lebesgue-Borel-Maß).
Das Problem ist nun, dass man zum Beispiel bei einem Mol Teilchen, was ca. $20l$ Luft oder $18ml$ Wasser entspricht, 10^{23} Teilchen mit $6 \cdot 10^{23}$ Impuls- und Ortskoordinaten im Phasenraum hat. Diese Summen oder Integrale sind nicht mehr durch stuhres (numerisches) Aufsummieren oder Integrieren lösbar. Man behilft sich daher mit folgendem Resultat:

Satz 2.1.1 (Gesetz der großen Zahlen)**.** *Für den statistischen Mittelwert einer Größe B im Phasenraum Ω*

$$\langle B \rangle = \sum_{x \in \Omega} \Pi(x) B(x) \tag{2.1}$$

bzw. allgemeiner

$$\langle B \rangle = \int_{x \in \Omega'} \tilde{\Pi}(x) B(x) d\mu$$

mit dem normierten statistischen Gewicht $\Pi(x)$ bzw. $\tilde{\Pi}$ (also z.B. der Boltzmannfaktor) gilt für N hinreichend groß:

$$\langle B \rangle \approx \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} B(x_k) \tag{2.2}$$

mit einer geeigneten Folge $(x_k)_{k\in\mathbb{N}}$, die insbesondere ergodisch sein sollte und die „richtige" Gewichtung der x_k für den Gleichgewichtsfall garantieren muss.

Prizipiell ist dabei zu sagen, dass Integrale nicht einfach in Summen umgeschrieben werden können (Riemann-integrierbare Funktionen sind zwar definiert über konvergierende Ober- und Untersummen und somit auch konvergierender Riemann-Summe, d.h. man kann solche mit einer Riemann-Summe approximieren. Allerdings muss man dazu iterierte Riemann-Integrale in entsprechende Summen umwandeln und somit immernoch über alle $6 \cdot 10^{23}$ Dimensionen summieren.). Die eigentliche Krux des Problem verbirgt sich hinter dem Begriff „geeignet".

Vom mathematischen Standpunkt (Statistik) aus bewegt man sich hier im Feld der Markovketten und Markovprozesse. Darauf soll an dieser Stelle aber nur verwiesen werden, da es den Rahmen sprengen würde darauf genauer einzugehen und für die mathematische Formulierung der Physik zwar essentiell ist, aber zuerst einmal keine physikalischen Resultate liefert.

Bemerkt werden soll hier noch, dass im Allgemeinen experimentell meistens auch nicht der gesamte Zustandsraum zugänglich ist und von diesem Standpunkt aus gesehen eine Simulation ein Experiment durchaus ersetzten kann.

2.1.2 Detailliertes Gleichgewicht

Die hier vorgestellten Algorithmen arbeiten alle mit dem Konzept des detaillierten Gleichgewichts („detailed balance"), daher soll hier kurz darauf eingegangen werden. Erforderlich dafür, dass die Bedingung des detaillierten Gleichgewichts erfüllt ist, ist, dass die nachfolgende Gleichung (2.3) gilt.

Definition 2.1.2 (Detailliertes Gleichgewicht)**.**

$$\pi(a)P(a \to b) = \pi(b)P(b \to a) \tag{2.3}$$

Mit:

- $\pi(x)$ *der Wahrscheinlichkeit der Konfiguration x im Gleichgewicht.*
- $P(x \to y)$ *der Übergangswahrscheinlichkeit von der Konfiguration x zur Konfiguration y.*

Das Konzept wird dadurch motiviert, dass die Algorithmen Gleichgewichtszustände berechnen sollen. Soll es sich um einen Gleichgewichtszustand handeln mit $\pi(a)$ groß und $\pi(b)$ klein, so muss für einen Übergang $a \to b$ mit Übergangswahrscheinlichkeit $P(a \to b)$ klein gelten, dass $P(b \to a)$ groß ist, sonst würde die Gleichgewichtsbedingung verletzt.

In einem kanonischen Ensemble (Vorgabe von Teilchenzahl N, Volumen V und Temperatur T; es ist nur eine Energieaustausch mit einem Wärmebad möglich) ist das statistische Gewicht $\pi(x)$ der Boltzmannfaktor, also $\pi(x) = e^{-\beta E(x)}$, mit $\beta = \frac{1}{k_B T}$ und der Boltzmannkonstanten k_B.

2.1.3 a-priori-Wahrscheinlichkeit

Ein weiteres wichtiges Konzept in diesem Zusammenhang, ist das der a-priori-Wahrscheinlichkeit („a-priori probability“), welches wie folgt definiert ist:

Definition 2.1.3.

$$P(a \to b) = A(a \to b)\tilde{P}(a \to b) \tag{2.4}$$

Mit:

- $A(a \to b)$ *der Wahrscheinlichkeit den Übergang (im Folgenden auch als „Zug“ bezeichnet) zu betrachten.*
- $\tilde{P}(a \to b)$ *der Wahrscheinlichkeit den Übergang (Zug) zu akzeptieren.*

Damit lässt sich dann die Bedingung des detaillierten Gleichgewichts (2.3) wie folgt umformulieren:

$$\frac{\tilde{P}(a \to b)}{\tilde{P}(b \to a)} = \frac{A(b \to a)}{A(a \to b)} \frac{\pi(b)}{\pi(a)}. \tag{2.5}$$

Dies ist in der im Folgenden benutzen klassischen Darstellung meist leicht berechenbar (da hier $\pi(x) = e^{-\beta E(x)}$ gilt, ist $\frac{\pi(b)}{\pi(a)} = e^{-\beta \Delta E}$). Weil auch die Wahrscheinlichkeit $A(a \to b)$ den Übergang zu betrachten durch den Algorithmus (bzw. die konkrete Implementierung) vorgegeben ist, wird nur noch ein Algorithmus benötigt, der unter der Berücksichtigung der detaillierten Gleichgewichtsbedinung die Wahrscheinlichkeit den Übergang zu akzeptieren $\tilde{P}(a \to b)$ angibt. Diese liefert der anschließend kurz besprochene Metropolisalgorithmus.

2.2 Metropolis-Algorithmus

Metropolis et al. [9] schlugen 1953 einen Algorithmus vor, der die unter Abschnitt 2.1.1 geforderten Eigenschaften für eine Folge $(x_k)_{k \in \ltimes}$ erfüllt (also z.B. das richtige statistische Gewicht hat). Dieser sieht wie folgt aus:

Definition 2.2.1 (Metropolis-Algorithmus)**.**

$$\tilde{P}(a \to b) = min\{1, \frac{\pi(b)}{\pi(a)}\}. \tag{2.6}$$

Mit Hilfe des detaillierten Gleichgewichts und von a-priori-Wahrscheinlichkeiten schreibt sich die Vorschrift so:

$$\tilde{P}(a \to b) = min\{1, \frac{A(b \to a)}{A(a \to b)} \frac{\pi(b)}{\pi(a)}\} \tag{2.7}$$

Da in den von uns betrachteten klassischen Fällen $\pi(x)$ einfach auswertbar sein wird, muss nur noch ein Algorithmus gefunden werden, der „gute“ Vorschläge für Züge ($A(a \to b)$) macht, also Züge mit großer Akzeptanz ($\tilde{P}(a \to b)$ groß). Dies ist das nun eigentlich zu lösende Problem. Man kann trivialerweise von einem lokalen Algorithmus ausgehen (am Beispiel des Ising-Modells: Es wird zufällig ein Spin ausgewählt und „geflippt“ (Zug). Dieser wird mit der vom Metropolis-Algorithmus gegebenen Wahrscheinlichkeit akzeptiert.) und einfach für mehrere (n) lokale Operationen als eine Clusteroperation definieren. Allerdings führt das zu mit größerem n stark abnehmenden Akzeptanzraten für den

Zug. Daher müssen sinnvolle Clusteralgorithmen im Allgemeinen besser auf das Problem und den Algorithmus angepasst werden.

Bemerkt werden soll hier noch, dass der eigentliche Metropolisalgorithmus nicht für Gitter konzipiert war, sondern für kontinuierliche Systeme im kanonischen Ensemble mit einer bestimmten „Vorschlagsfunktion“ $P(a \to b)$. Hastings [3] hat eine Verallgemeinerung davon gebildet für Algorithmen, die mit obigem Konzept der a-priori-Wahrscheinlichkeit arbeiten, und ansonsten eine beliebige Vorschlagsdichte $A(x_i \to y)$ und einem beliebigen statistischen Gewicht $W(x)$ arbeiten gemäß:

$$\tilde{P}(x_i \to y) = min\{1, \frac{A(x_i \to y)}{A(y \to x_i)} \frac{W(y)}{W(x_i)}\}, \tag{2.8}$$

wobei x_i der aktuelle Ort und y der betrachtete potentiel zukünftige Ort ist. Daher wird der Algorithmus auch als Metropolis-Hastings-Algorithmus bezeichnet.
Der Metropolis-Algorithmus ist auch nicht der einzige, der die unter Abschnitt 2.1.1 geforderten Bedingungen erfüllt. Ein anderer Algorithmus stammt beispielsweise von Barker [2] und ist wie folgt definiert:

Definition 2.2.2.

$$\tilde{P} = \frac{z}{1+z} \tag{2.9}$$

mit

$$z = \frac{A(b \to a)}{A(a \to b)} \frac{\pi(b)}{\pi(a)}. \tag{2.10}$$

2.3 Monte-Carlo: Ein erster Schluss

Für unsere Monte-Carlo-Metropolis-Algorithmen sollen also folgende Bedingungen erfüllt sein:

1. Algorithmus muss Ergodizität gewährleisten (im Sinne, dass der ganze Phasenraum erreicht werden kann).
2. Algorithmus soll detailliertes Gleichgewicht gewährleisten und mit a-priori-Wahrscheinlichkeit arbeiten.
3. Die Quotienten $\frac{A(b \to a)}{A(a \to b)}$ und $\frac{\pi(b)}{\pi(a)}$ müssen berechenbar sein.

Der zweite Punkte ist über das Schema des Metropolis-Algorithmus abgehakt und der Quotient $\frac{\pi(b)}{\pi(a)}$ ist für die betrachteten „klassischen“ kanonischen Systeme (also den Boltzmannfaktor) normalerweise einfach berechenbar. Damit haben wir das Ausgangsproblem des hochdimensionalen Phasenraums auf folgendes andere Problem verschoben: Die Aufgabe einen Algorithmus zu finden, der ergodisch ist und bei dem der Quotient $\frac{A(b \to a)}{A(a \to b)}$ gut berechenbar ist und möglichst schnell konvergiert.

2.4 Ising-Modell

Dieser Abschnitt soll kurz das benützte Ising-Modell vorstellen und die Hauptideen rekapitulieren. Keinesfalls soll das Ising-Modell erschöpfend wiedergegeben werden.

2.4.1 Das Original

Ising betrachtete in seiner Publikation 1925 [4] eine einfache lineare Kette von Spins, die nur zwei Einstellungen haben können ($\{+,-\}$) in einem äußeren Magnetfeld. Seine Ausführungen, dass es dort zu keinem Phasenübergang bei der Magnetisierung kommen kann sind korrekt, sein Übertrag von der linearen Kette zu einem zweidimensionalen Modell nicht. Dies kann später auch mittels der Simulation durch den Wolff-Algorithmus gezeigt werden.
Nichts destotrotz hat sich das Modell als recht erfolgreich erwiesen, schon deshalb, weil es ein- und zweidimensional (mit gewissen Einschränkungen) analytisch lösbar ist und gewisse Eigenschaften (Phasenübergang) vorhersagt, die tatsächlich beobachtet werden. Heut meint man mit einem Ising-Modell meist ein n-dimensionales Gitter mit Spins, die zwei Einstellungen (entspricht quantenmechanische der Projektion des Spins z.B. eines Elektrons auf eine Vorzugsrichtung, o.B.d.A. werde diese als z-Richtung bezeichnet) einnehmen können. Der Energieterm habe dann folgende Form:

$$E = -\frac{1}{2}\sum_{(i,j)} J_{ij} s_i s_j - H_z \sum_i s_i \tag{2.11}$$

wobei H_z die z-Komponente eines externen magnetischen Feldes H darstellt, J_{ij} die Kopplungskonstante des Paares (i,j) ist und die Summe normalerweise nur über nächste Nachbarn (i,j) läuft.
Quantenmechanisch wären dies natürlich die Erwartungswerte der $\hat{s}_z^i$ Operatoren, in der Quantenfeldtheorie würde zusätzlich noch das externe Magnetfeld quantisiert werden.

2.4.2 Modell

Hier soll im Folgenden nur eine weit vereinfachte Variante des Ising-Modell verwendet werden, welches aber für die untersuchten Effekte voll genügt. Wenn also von einem Ising-Modell gesprochen wird, ist nachfolgendes gemeint:

Definition 2.4.1. *Es werden von N^2 Spins auf einem 2 dimensionalen Gitter L mit Plätzen (i,j) die Komponenten in einer ausgezeichneten Richtung (o.B.d.A z-Richtung) betrachtet. Die (klassische) Wechselwirkungsenergie ist dabei durch*

$$E = -J\sum_{(i,j)} s_i s_j \tag{2.12}$$

gegeben mit $s_i, s_j \in \{-1,1\} \cong \{+,-\}$ und der Summe über alle nächsten Nachbarn (i,j). Die Kopplungskonstante J soll für alle Paarungen denselben Wert haben.

Eine Beispielkonfiguration dafür ist mit Abbildung 2.1 gegeben, ein äußeres Magnetfeld wird nicht zugelassen. Da im Folgenden Randeffekte soweit möglich

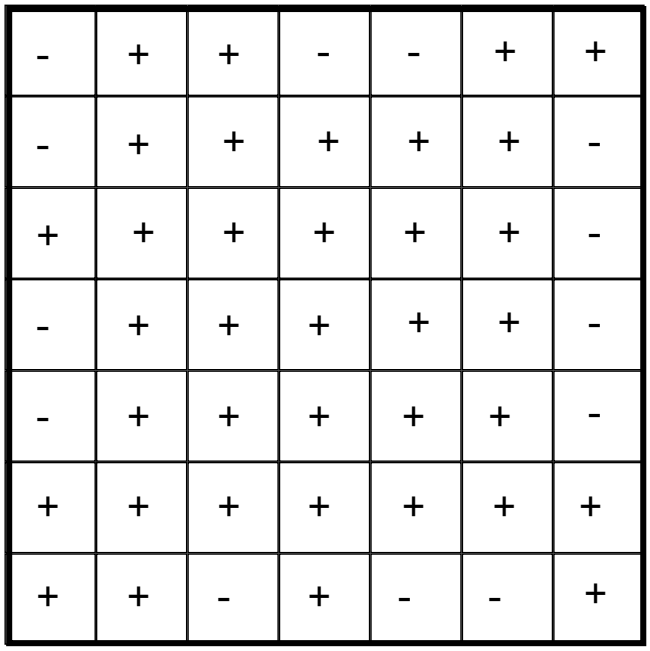


Abbildung 2.1: Beispiel eines quadratischen Spingitters mit 49 Plätzen

ignoriert werden sollen, werden immer periodische Randbedingungen angenommen.

Kapitel 3

Der Algorithmus

Nachfolgende Abschnitte orientieren sich an einem Buchkapitel von Krauth aus [5]. Wie Krauth in diesem Kapitel die Algorithmen nicht mittels des Pseudokodes erklärt, sollen auch hier Bilder den Algorithmus und seine Probleme veranschaulichen. Den Pseudokode dazu findet man bei Krauth.

3.1 Der Wolff-Algorithmus

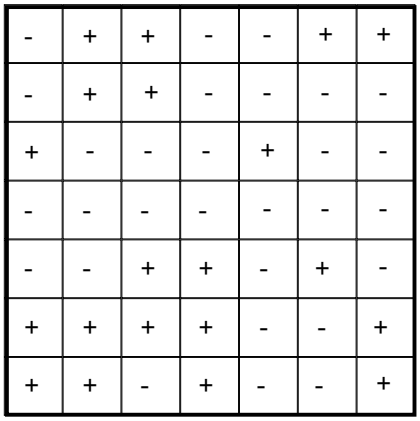
Der Wolff-Algorithmus wurde von Ulli Wolff in seiner Publikation von 1989 [14] für ein etwas anderes Modell betrachtet, zur Erklärung genügt aber unser Ising-Modell vollauf.

Abbildung 3.1 illustriert wie der Algorithmus arbeitet. Es handelt sich um folgende Schritte:

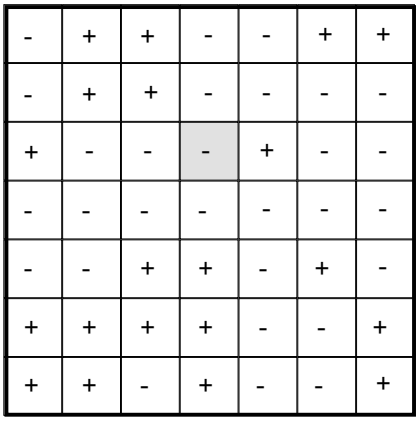
Algorithmus 3.1.1 (Wolff-Algorithmus)**.**

1. *Abbildung 3.1(a): Startkonfiguration gegeben.*
2. *Abbildung 3.1(b): Suche Platz* (α_0, β_0) *zufällig aus (Im Beispiel* $(\alpha_0, \beta_0) = (4, 3)$*).*
3. *Abbildung 3.1(c): Bilde ausgehend von* (α_0, β_0) *mit Wahrscheinlichkeit* p *„Verbindungen“ zwischen zwei benachbarten Plätzen (wenn* $s_i = s_j$*).*
4. *Abbildung 3.1(d): Flippe Cluster (wird für bestimmtes* p *immer akzeptiert,* $\tilde{P} = 1$*, der Kern des Algorithmus). Man erhält die neue Konfiguration* b*.*
5. *Abbildung 3.1(e): Wiederhole dies für neu gewählten Startplatz* (α_1, β_1)*.*

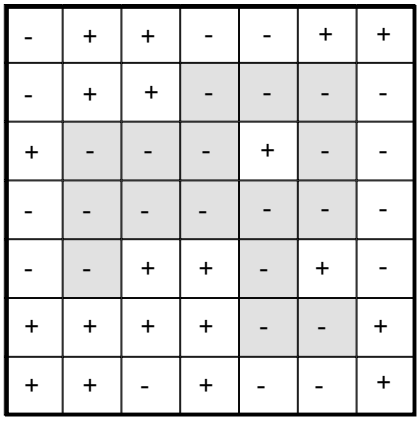
Die Effizienz des Algorithmus rührt von Punkt 4 her. Für eine bestimmte Wahl von p wird $\tilde{P} = 1$ und der Zug immer akzepiert, der Algorithmus rückweisungsfrei („rejection free“). Warum dies so ist, soll im nächsten Abschnitt erläutert werden.



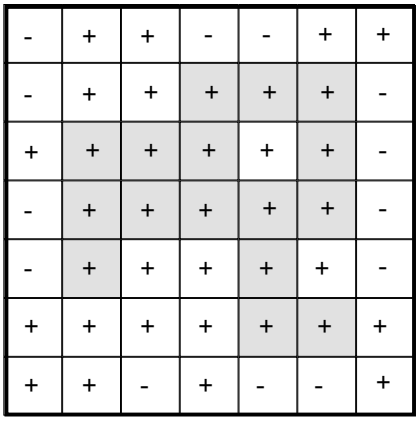
(a) Ausgangskonfiguration a.



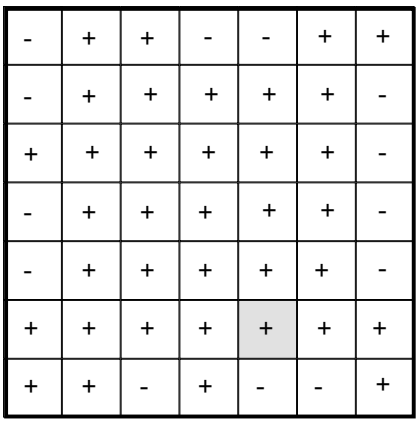
(b) Startplatz $(\alpha_0, \beta_0) = (4, 3)$ ausgewählt.



(c) Clusterbildung um $(4, 3)$ mit Bindungswahrscheinlichkeit p.



(d) „Geflippter“ Cluster, Konfiguration b



(e) Neu gewählter Startplatz.

Abbildung 3.1: Der Wolff-Algorithmus graphisch dargestellt.

3.1.1 Der Kern des Wolff-Algorithmus: Warum kann $\tilde{P} = 1$ werden?

Die Frage warum $\tilde{P} = 1$ wird für eine bestimmtes p kann man leicht über den verwendeten Metropolis-Algorithmus klären. Betrachte hierzu die Vorschlagswahrscheinlichkeiten der Züge $(a \to b)$ und $(b \to a)$ und die Energien der Konfigurationen E_a, E_b:

- Konfiguration a in Abbildung 3.1 hat:

$$\begin{aligned} A(a \to b) &= A_{innen} \cdot (1-p)^{n_{gleich}} && (3.1)\\ E_a &= E_{innen} + E_{aussen} - n_{gleich} \cdot J + n_{diff} \cdot J && (3.2)\\ \pi_a &= e^{-\beta E_a} && (3.3) \end{aligned}$$

- Konfiguration b in Abbildung 3.1 hat:

$$\begin{aligned} A(b \to a) &= A_{innen} \cdot (1-p)^{n_{diff}} && (3.4)\\ E_b &= E_{innen} + E_{aussen} + n_{gleich} \cdot J - n_{diff} \cdot J && (3.5)\\ \pi_b &= e^{-\beta E_b} && (3.6) \end{aligned}$$

Hierbei meint „innen“ und „aussen“ jeweils innerhalb des Clusters und außerhalb des Clusters, jedoch nie den Rand des Clusters. Der Rand wird n_{diff} (Anzahl der am Rand zusammentreffenden Stellen mit verschiedenem s_i) und n_{gleich} (Anzahl der Stellen mit gleichem s_i) mitgenommen, wobei der Bezug immer der ursprüngliche, nicht „geflippte“ Cluster ist.
Folglich gilt für die Wahrscheinlichkeit, dass der Zug akzeptiert wird, gemäß dem Metropolis-Algorithmus (Einsetzten in Gleichung (2.7)):

$$\tilde{P}(a \to b) = min\left\{1, \frac{A_{innen} \cdot (1-p)^{n_{diff}}}{A_{innen} \cdot (1-p)^{n_{gleich}}} \cdot \frac{e^{-\beta(E_{innen}+E_{aussen}+n_{gleich}\cdot J - n_{diff}\cdot J)}}{e^{-\beta(E_{innen}+E_{aussen}-n_{gleich}\cdot J + n_{diff}\cdot J)}}\right\} \tag{3.7a}$$

$$= min\left\{1, \frac{(1-p)^{n_{diff}}}{(1-p)^{n_{gleich}}} \cdot \frac{e^{-2\beta n_{gleich} J}}{e^{-2\beta n_{diff} J}}\right\} \tag{3.7b}$$

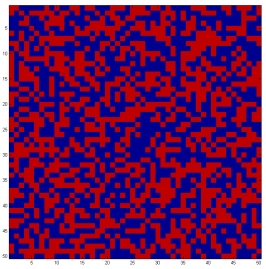
$$= min\left\{1, \left(\frac{(1-p)}{e^{-2\beta J}}\right)^{n_{diff}} \cdot \left(\frac{e^{-2\beta J}}{(1-p)}\right)^{n_{gleich}}\right\} \tag{3.7c}$$

$$= 1 \tag{3.7d}$$

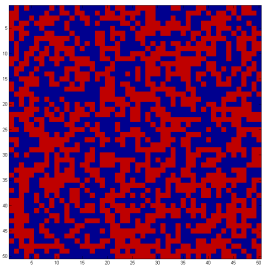
wenn $p = 1 - e^{-2\beta J}$ gewählt wird. Damit wird für dieses p jeder Zug immer akzeptiert!

3.2 Numerik

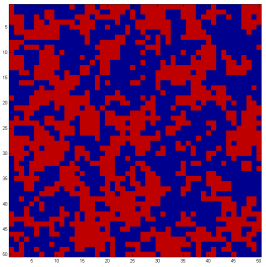
Eine Beispielimplementierung des Algorithmus liefert nun z.B. folgende Ergebnisse für die Gleichgewichstmagnetisierung (alle Simulationen wurden, soweit nicht anders angegeben für ein quatratisches Gitter mit Kantenlänge $N = 50$ gerechnet): Deutlich erkennt man, das aus der Theorie bekannte Ergebnis, dass das zweidimensionale Ising-Modell eine von 0 verschiedene Magnetisierung für



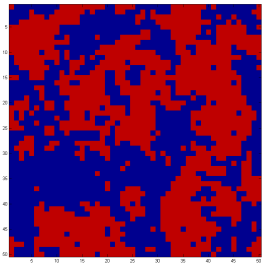
(a) $\frac{J}{k_B T} = 0.$



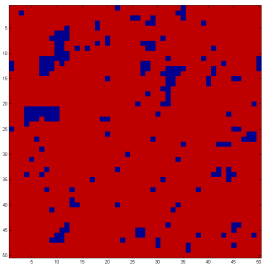
(b) $\frac{J}{k_B T} = 0,1.$



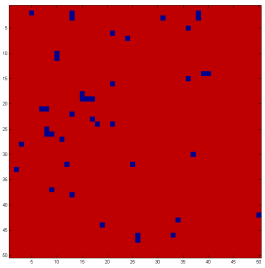
(c) $\frac{J}{k_B T} = 0,3.$



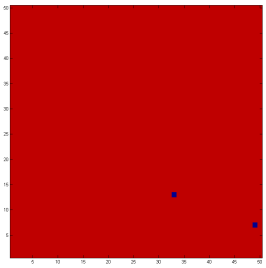
(d) $\frac{J}{k_B T} = 0,4.$



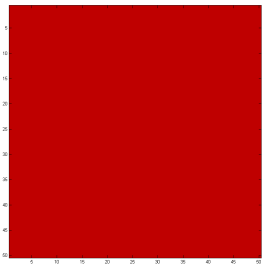
(e) $\frac{J}{k_B T} = 0,47.$



(f) $\frac{J}{k_B T} = 0,6.$



(g) $\frac{J}{k_B T} = 1.$



(h) $\frac{J}{k_B T} = 5.$

Abbildung 3.2: Simulation der Gleichgewichtsmagnetisierung bei verschiedenen $\frac{J}{k_B T}$ (z.B. verschiedenen Temperaturen, hier absteigend von $T = \infty$). Es wurden zwischen 5.000 und 10.000 Schritte des Wolff-Algorithmus berechnet.

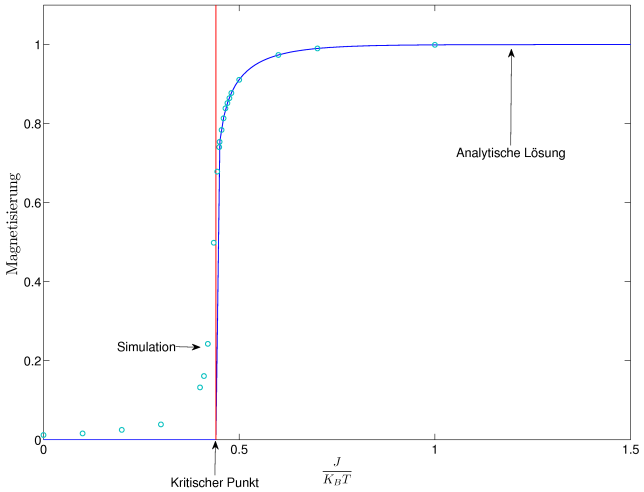


Abbildung 3.3: Verlauf der Magnetisierung über $\frac{J}{k_B T}$ mit gut erkennbarem Phasensprung und analytischer Lösung.

kleine Temperaturen aufweist ($\frac{J}{k_B T}$ groß, z.B. Abbildungen 3.2(f) bis 3.2(h)) und bei ansteigender Temperatur einen Phasenübergang durchläuft. Dabei bilden sich die bekannten langreichweitigen „Cluster“ aus (Abbildungen 3.2(d)). Bei weiter ansteigender Temperatur wird die Ausrichtung des magnetischen Moments der einzelnen Spins beliebig, die Magnetisierung verschwindet ($\frac{J}{k_B T}$ klein, Abbildungen 3.2(c) bis 3.2(a)).

3.2.1 Phasenübergang

Betrachtet man nun die Magnetisierung, so sieht man deutlich den Phasenübergang bei einem „Sprungverhältnis“ von $\frac{J}{k_B T} = -\frac{\ln(\sqrt{2})-1}{2} \approx 0,44$ (Abbildung 3.3). Die Abweichung der Simulation von den analytischen Ergebnissen oberhalb des Sprungs dürfte an den Randbedingungen liegen: Die Simulation wurde für periodische Randbedingungen gerechnet und bei kleinen Gittern ($N = 50$) bekommt man dann gewisse Randeffekte.

3.2.2 Rechenzeit und Konvergenzgeschwindigkeit

Die Hauptmotivation Clusteralgorithmen zu entwickeln bestand darin, dass sie schnellere Konvergenzgeschwindigkeiten in der Nähe von Phasenübergängen erreichen sollten. Dabei ist zu beachten, dass es zwei Faktoren gibt:

1. Clusteralgorithmen verhindern durch Operationen auf ganzen Clustern das sogenannte „critical slowing down“ lokaler Algorithmen und sind daher in Bereichen von Divergenezen der Korrelationszeit schneller als lokale Algorithmen (erzeugen schneller, sprich in weniger Rechenschritten, die Folge $(x_k)_{k\in\mathbb{N}}$).

2. Clusteralgorithmen sind meistens aber deutlich komplexer als lokale Algorithmen und berechnen pro Iterantionsschritt mehr Gleichungen etc.

Das heißt, dass sie pro Iterationsschritt im Allgemeinen mehr Rechenzeit brauchen als lokale Algoritmen.

Die reine Zahl der Iterationsschritte ist also ein ungenügendes Maß, um zu bewerten wie schnell ein Algorithmus ist. In der Realität hängt die Rechenzeit aber auch noch von ganz anderen Faktoren ab, wie z.B. der verwendeten Rechnerarchitektur, dem Systemaufbau und der Systemkonfiguration. Insbesondere müssen beide Algorithmen möglichst effizient programmiert sein (möglichst wenig Rechenoperationen pro Iterationsschritt und gleichzeitig diese so angelegt, dass auf dem spezifizierten Zielsystem die Rechenoperationen etc. möglichst schnell ablaufen).
In der Numerik handhabt man es häufig so, dass man die Effizient eines Algorithmus über die Anzahl der Rechenoperationen abschätzt, hier also die Anzahl der Iterationsschritte, die notwendig sind, und die Rechenoperationen pro Schritt.
Dieser Aufwand wurde hier nicht getrieben, sondern es wurde davon ausgegangen, dass die Algorithmen in etwa „gleich effizient“ (ohne dies exakt definieren zu wollen) implementiert wurden und nicht sonderlich auf das verwendete System optimiert wurden. Daher soll hier grob die Rechenzeit bis zum erreichen des Gleichgewichtzustandes als Vergleichsmaßstab für die Konvergenzgeschwindigkeit dienen.
Betrachtet man nun Abbildung 3.4, so erkennt man, dass der Wolff-Algorithmus schon nach sehr kurzer Zeit konvergiert, während der lokale Algorithmus dafür sehr viel länger braucht. Dies deutet darauf hin, dass die Änderung der Konfiguration klein ist, also der lokale Algorithmus lange Autokorrelationszeiten hat. Diese dürften signifikant länger als beim Wolff-Algorithmus sein, allerdings ist auch ganz klar, dass der Wolff-Algorithmus deutlich mehr Operationen pro Schritt benötigt als der lokale Algorithmus, denn es sind etwa 10.000 Schritte des Wolff-Algorithmus und ca. 700.000 Schritte des lokalen Algorithmus dargestellt.
Der Wolff-Algorithmus ist daher in diesem Fall (nahe des Phasenübergangs, $\frac{J}{k_B T} = 0,5$) schneller als der lokale Algorithmus.
Tatsächlich müsste man allerdings die Rechenzeit und die Autokorrelationszeit zusammen betrachten, um eine auch vom theoretischen Standpunkt aus akzeptable Größe zu erhalten.

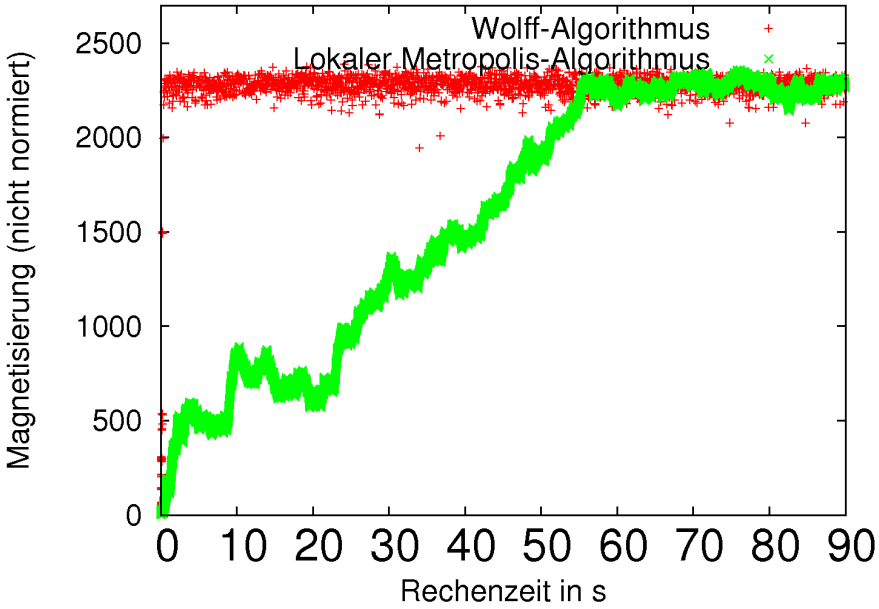


Abbildung 3.4: Vergleich der Rechenzeit zwischen Wolff-Algorithmus und einem lokalen Monte-Carlo-Algorithmus.

Kapitel 4

Andere Modelle und Erweiterungen

In diesem Kapitel soll nun primär vom Wolff-Algorithmus weggegangen werden einmal ein ganz anderes System betrachtet werden. Dafür brauchen wir dann auch einen anderen Algorithmus (auch dieser Abschnitt ist nach dem Buchauszug von Krauth [5] gestaltet).

4.1 Pocket-Algorithmus

Das im Folgenden betrachtete Modell sei eine Menge „harter Scheiben“ (motiviert von harten Kugeln) ohne sonstige Wechselwirkungen, d.h. das Potential um eine Kugel hat folgende Form:

$$U(r) = \begin{cases} 0 & r > \sigma \\ \infty & r \leq \sigma \end{cases} \tag{4.1}$$

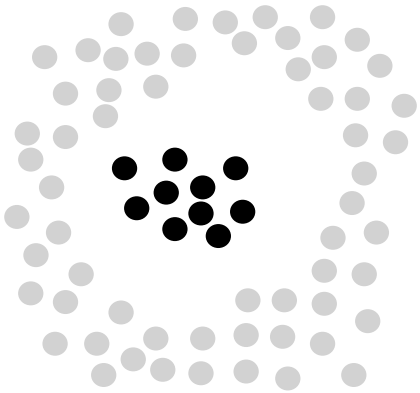
mit dem radialen Abstand r und dem Kugelradius σ. Weiter werden diese im „freien“ Raum (Kontinuum, kein Gitter) mit weiterhin periodischen Randbedingungen betrachtet.

4.1.1 Problemstellung

Man stößt nun auf folgende Schwierigkeit:
In Abbildung 4.1(b) ist recht gut ein potentieller Cluster zu erkennen (die einzige Kategorie, nach der sinnvoll Cluster generiert werden können, hat mit Abständen und Überlappung zu tun). Transformiert man diesen mittels einer Transformation T (hier sei T eine Translation), so kommt man zur Konfiguration b (Abbildung 4.1(c)). In dieser ist auf dieselbe Art nur recht schwer ein Cluster ausfindbar zu machen, es besteht keine so offensichtliche Möglichkeit wie zuvor. Das führt auf folgendes Problem:

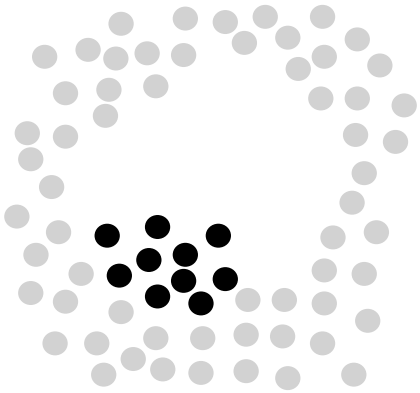
- Cluster in Abbildung 4.1(b) gut identifizierbar.
- Nach Anwendung der Transformation (Abbildung 4.1(c)) ist kein trivial auffindbarer Cluster vorhanden.

(a) Konfiguration a.



(b) Cluster ist in Konfiguration a gut identifizierbar.

(c) Konfiguration b.



(d) Cluster ist in Konfiguration b nicht gut identifizierbar.

Abbildung 4.1: Problem der möglicherweise sehr geringen Akzeptanzraten ($\tilde{P}$ ist klein), wenn b aus a über eine zufällige ausgewählte Transformation T (also $b = T(a)$, hier eine einfache Translation) hervorgeht.

- Cluster gut auffindbar: $A(a \to b)$ ist groß, Zug würde von einem Algorithmus sehr wahrscheinlich vorgeschlagen.
- Cluster nicht gut auffindbar: $A(b \to a)$ ist klein, Zug würde von einem Algorithmus eher weniger wahrscheinlich vorgeschlagen.
- Da sich keine Kugeln überlappen ist $\pi(a) = \pi(b)$ und damit wird gemäß dem Metropolis-Algorithmus nach Gleichung (2.7) $\tilde{P}(a \to b)$ sehr klein, der Zug wird kaum akzeptiert, zufälliges Auswählen und Verschieben ist also sehr ineffektiv.

Ein möglicher Ausweg besteht darin, dass $T = T^{-1}$ gefordert wird. Damit würde nach obiger Argumentation $\tilde{P}(a \to b)$ groß bleiben. Eine solche Möglichkeit bieten zum Beispiel Punktspiegelungen. Ein Algorithmus, der diese benützt, ist der sogenannte „pocket"-Algorithmus.

4.1.2 Algorithmus

Auch dieser Algorithmus soll hier wieder mit Graphiken (Abbildung 4.2) illustriert werden, den Pseudokode findet man wieder bei Krauth ([5]).

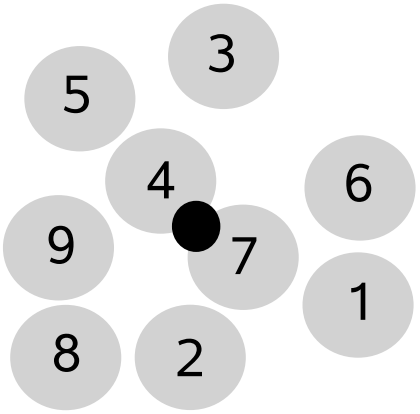
Algorithmus 4.1.1 („Pocket"-Algorithmus)**.**

1. *Abbildung 4.2(a): Wähle Spiegelpunkt (schwarzer Punkt in 4.2) aus (Punktspiegelung ist selbstinvers).*
2. *Abbildung 4.2(b): Wähle erste Scheibe aus und stecke sie in die „Tasche" (engl. „pocket"), wobei das bedeutet, dass einfach die Auswahl gespeichert wird. Alle Scheiben mit dunklem Inneren sind in der Tasche.*
3. *Abbildung 4.2(c): Ziehe aus der Tasche eine Scheibe (erste) und spiegle sie. Alle von der gespiegelten Scheibe Getroffenen kommen in die Tasche.*
4. *Abbildung 4.2(d): Wähle nächste Scheibe aus Tasche und verfahre wie zuvor.*
5. *Abbildung 4.2(e): Wiederhole Schritte 3. bis 4. bis die Tasche leer ist.*

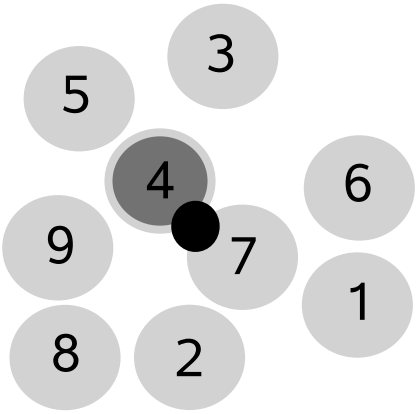
4.1.3 Wo sind die Cluster?

Bisher ist nicht klar, warum der „pocket"-Algorithmus als Clusteralgorithmus bezeichnet werden kann. Dies soll hier nun geklärt werden (Transformation T sei nun die Punktspiegelung). Betrachte hierzu Abbildung 4.3.

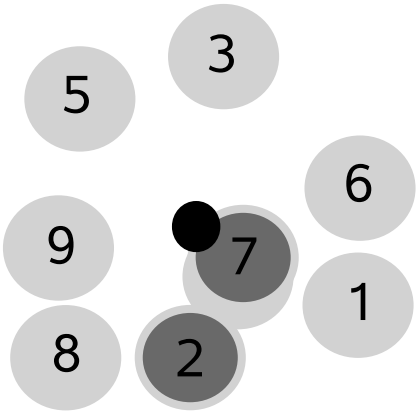
Abbildung 4.3(a) zeigt die Ausgangskonfiguration (Konfiguration a), Abbildung 4.3(b) Konfiguration a komplett punktgespiegelt (Konfiguration b). Überlagert man nun beide Konfigurationen, so bekommt man Abbildung 4.3(c). Dort erkennt man nun, dass es „Subensembles" gibt, die beim gewählten Spiegelungspunkt nur in sich gespiegelt werden. Das heißt, dass diese „Subensembles" unabhängig von allen anderen Punkten (Subensembles) transformiert werden (ein Beispiel wäre das Subensemble $\{2, 3, 4, 7\}$).
Diese Subensembles können als Cluster betrachtet werden und so haben wir wieder einen Clusteralgorithmus, wenn wir den „pocket"-Algorithmus auf die einzelnen Subensembles beschränken (wenn also jeder Cluster einzeln gespiegelt werden kann, oder eben auch nicht).



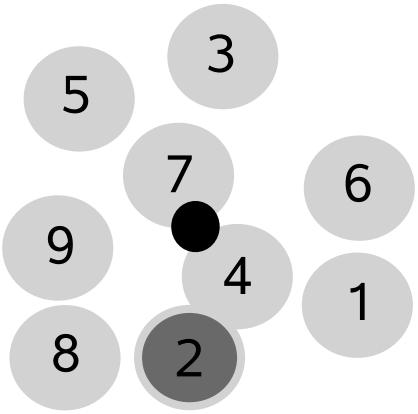
(a) Konfiguration a, mit gewähltem Spiegelungspunkt (schwarz).



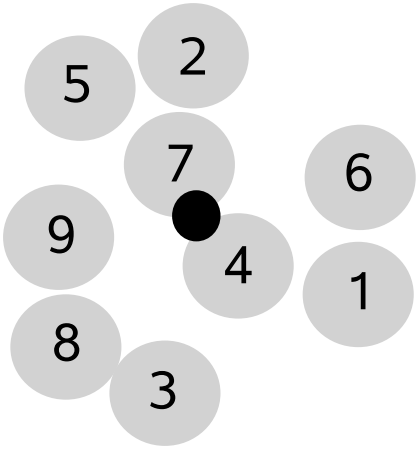
(b) Erste Scheibe (4) ausgewählt.



(c) Erste Scheibe gespiegelt, die Überdeckten (7,2) gehen in die Tasche.

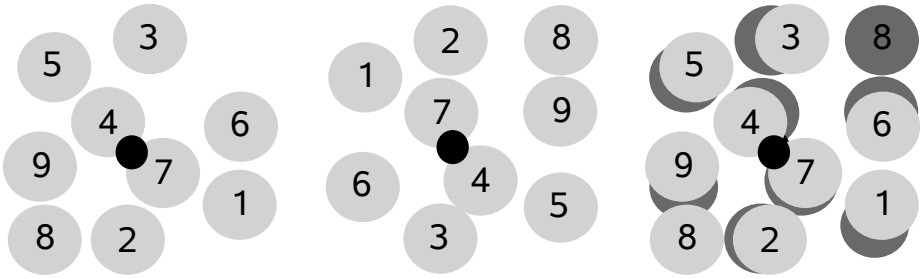


(d) Eine Scheibe aus Tasche „gezogen“ (7) und gespiegelt.



(e) Neue Konfiguration b am Ende des Algorithmus.

Abbildung 4.2: Der „pocket“-Algorithmus graphisch illustriert.



(a) Konfiguration a.

(b) Konfiguration b (Konfiguration a punktgespiegelt).

(c) Überlagerung von Konfiguration a und b.

Abbildung 4.3: Cluster beim „pocket“-Algorithmus.

4.1.4 Anwendung: Phasentrennung binärer Mischungen

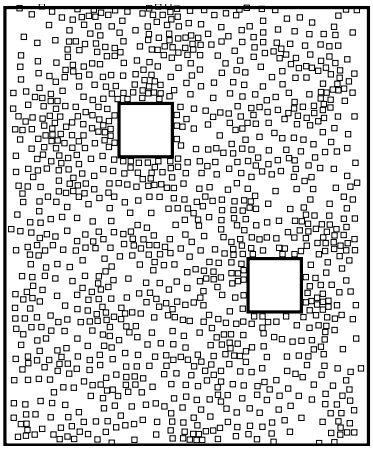
Nach dem alle Theorie ja grau ist, soll auch zum pocket-Algorithmus kurz eine Anwendung geschildert werden. Dazu betrachten wir eine zweikomponentige Mischung (hier aus kleinen und großen „harten“ Quadtraten).

Modell

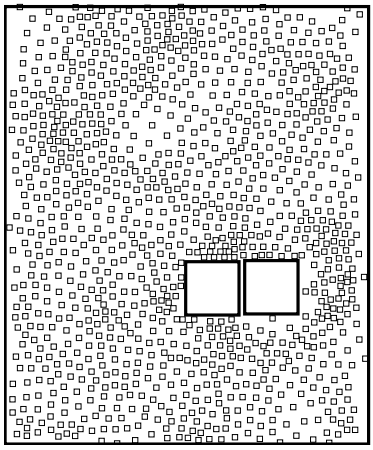
Betrachte folgende Situation harter Quadrate (harte Quadrate ohne Wechselwirkungen, in Analogie zu harten Scheiben) in einem Testbereich („Box“) mit, wie üblich, periodischen Randbedingungen (möglich werden z.B. kolloidale Lösungen mit (projeziert) näherungsweise quadratischen Partikeln):

- In Abbildung 4.4(a) sind die (großen) Quadrate weit von einander entfernt. Es herrscht aufgrund der Bewegung der kleinen Quadrate zwischen den großen eine schwache effektiv repulsive „Wechselwirkung“ bei Annäherung, da die kleinen Quadrate die großen wieder auseinander drängen.
- In Abbildung 4.4(b) jedoch sind die großen Quadrate so dicht bei einander, dass keine kleinen Quadrate mehr dazwischen passen. Daher werden die großen Quadrate von den beiden äußeren Seiten zusammengedrückt, zwischen ihnen befindet sich aber nichts, das sie wieder auseinander drückt, und somit existiert eine effektiv attraktive „Wechselwirkung“. Dies ist ein rein entropischer Effekt (es gibt keine anderen Wechselwirkungen).

Man bezeichnet diese Modell auch als „Ising-Modell für zweikomponentige Flüssigkeiten“. Qualitativ ist es ähnlich zu dem von harten Scheiben (bzw. Kugeln), allerdings ist aufgrund der großen Kontaktfläche die attraktive Wechselwirkung bei kleinen Abständen sehr viel größer. Ein lokaler Algorithmus wird hier sehr langsam sein, insbesondere wenn das Verhältnis der Durchmesser der Quadrate klein ist ($\frac{d_{klein}}{d_{gross}} \to 0$) wird er sogar unerträglich langsam sein, da praktisch keine Bewegung der großen Quadrate mehr möglich ist. Dies liegt daran, dass aufgrund der Zwangsbedingung der Überlappungsfreiheit praktisch kein Zug der großen Quadrate akzeptiert werden wird.



(a) Konfiguration a, die großen Quadrate sind weit entfernt. Bei Annäherung kommt es zu einer effektiv repulsiven Wechselwirkung.



(b) Konfiguration b, die großen Quadrate sind sehr nahe bei einander. Es gibt eine effektiv attraktive Wechselwirkung.

Abbildung 4.4: Entropische Wechselwirkungen bei „harten“ Quadraten.

Die interessante Frage hier ist, ob es einen Phasenübergang gibt, sich also die zwei Komponenten entmischen. Dies kann unter anderem per Simulation mittels eines pocket-Algorithmus untersucht werden. Dabei stellt man fest, dass es tatsächlich einen solchen Phasenübergang gibt und die großen Quadrate mehr oder weniger kristallisieren. Nachfolgend soll kurz der entsprechende pocket-Algorithmus illustriert werden.

pocket-Algorithmus angewandt

Der Algorithmus entspricht weitgehend dem schon in Abschnitt 4.1.2 vorgestellten Algorithmus, Abbildung 4.5 erläutert wiederum graphisch das Vorgehen. Wiederum sind alle Quadrate mit dunkelgrauem Inneren in der Tasche:

1. Abbilundung 4.5(a): Spiegelungspunkt auswählen.
2. Abbilundung 4.5(b): Es wird ein Startquadrat ausgewählt und in die Tasche gesteckt (hier ein großes Quadrat).
3. Abbilundung 4.5(c): Das Quadrat wird gezogen und transfomiert (gespiegelt).
4. Abbilundung 4.5(d): Besonderheit: Alle voll vom (großen) Quadrat Überdeckten können direkt neu plaziert werden, da sie in einen leeren Berich gespiegelt werden. Sie müssen nicht „durch die Tasche“.
5. Abbilundung 4.5(e): Teilüberdeckte Quadrate gehen in die Tasche.
6. Abbilundung 4.5(f): Aus der Tasche wird ein Quadrat gezogen und transformiert (gespiegelt), voriger Schritt wird wiederholt (Abbildung 4.5(g)).
7. Schritte 4 bis 6 werden solange wiederholt, bis die Tasche leer ist und man somit bei der neuen Konfiguration b angekommen ist (Abbildung 4.5(h)).

So wie Abbildung 4.5(h) gezeichnet ist, war die ganze Box ein Cluster in diesem Zug.
Es fehlen noch Transformationen, die die Quadrate um einen beliebigen Winkel drehen. Diese können in Form von Geradenspiegelungen eines Teilsystems hinzugenommen werden (was dann für dieses Teilsystem auch eine selbstinverse Transformation ist). Dies ist im Allgemeinen aber keine Symmetrieoperation mehr für das ganze System.

4.2 Erweiterungen von Wolff- und Pocket-Algorithmus

Nachdem nun die beiden Algorithmen für ganz bestimmte Systeme und Bedingungen vorgestellt worden sind, stellt sich natürlicherweise die Frage, inwiefern sie sich erweitern lassen. Daher sollen hier nun eingie Erweiterungensideen vorgestellt werden bzw. Referenzen dafür angegeben werden. Es soll hier aber nicht darum gehen, die konkrete Erweiterung zu besprechen, da dies den Rahmen des Vortrags gesprengt hätte.

Erweiterungen des Wolff-Algorithmus

Zu erwähnen ist, dass Wolff seinen Algorithmus nicht für ein Ising-Modell vorgestellt hat, sondern für ein sogenanntes $O(n)$-Modell [14]. Das heißt, dass die Spins S_i auf der S^n liegen und

$$H = -J \sum_{<x,y>} S_x \cdot S_y, \tag{4.2}$$

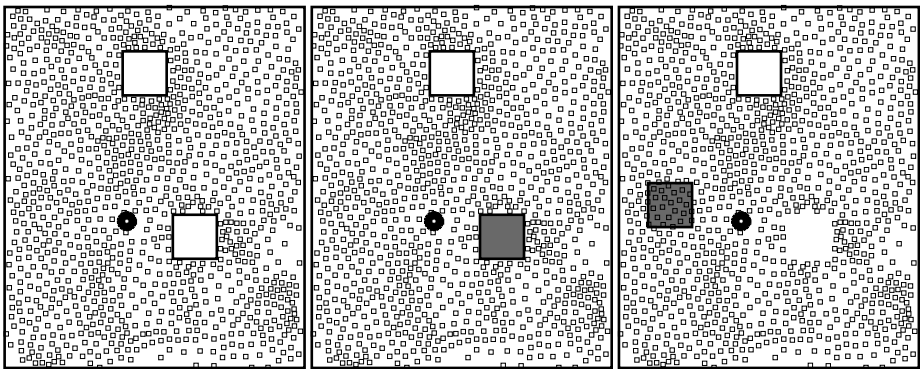
wobei die Summe wieder über nächste Nachbarn geht. Dieses Modell ist also sehr viel allgemeiner als das hier verwendete Ising-Modell. Dabei verwandte Wolff Spiegelhyperebenen anstatt des hier vorgestellten Spiegelpunkts. Trivialerweise können alle auf dieses Modell abbildbaren Modelle damit berechnet werden.
Bei Spinmodellen ist es natürlich auch immer interessant externe Magnetfelder zu berücksichtigen (vor allem da diese schnell analytisch nicht mehr berücksichtigt werden können). Daher kann man versuchen diese und andere externen Wechselwirkungen mit zu betrachten.

Schließlich sollte noch erwähnt werden, dass es einen Versuch gab, den Swendsen-Wang-Algorithmus (was fast dem Wolff-Algorithmus entspricht) weg vom Gitter auf ein kontinuierliches Problem mit Paarwechselwirkungen (harte Kugeln) zu übertragen (siehe hierzu [1]). Dies ist auch geglückt, der Algorithmus war in diesem konkreten Fall aber langsamer als ein gut angepasster lokaler Algorithmus.

Erweiterungen des Pocket-Algorithmus

Die vorgestellte Variante des pocket-Algorithmus wird im Prinzip auch als **G**eometric **C**luster **A**lgorithm bezeichnet. Ein interessante Erweiterung besteht darin, dass nicht nur harte Kugelpotentiale sondern noch zusätzlich Potentiale (Paarwechselwirkungen) mitgenommen werden. Beispiele hierzu sind folgende zwei Artikel:

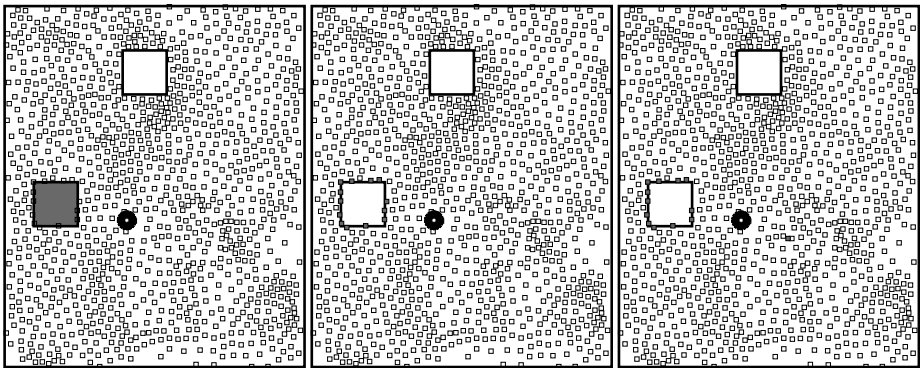
- „Generalized geometric cluster algorithm for fluid simulation“, Erweiterung um Yukawa-artiges Potential [7]



(a) Konfiguration a mit Spiegelungspunkt.

(b) Startquadrat ausgewählt.

(c) Startquadrat gespiegelt.



(d) Vollüberdeckte Quadrate können direkt gespiegelt werden ohne in die Tasche zu müssen.

(e) Quadrat aus Tasche gezogen.

(f) Gezogenes Quadrat wurde gespiegelt.

(g) Überdecktes Quadrat geht in die Tasche.

(h) Tasche leer, Konfiguration b erreicht. Wegen der periodischen Randbedingungen schaut das eine große Quadrat von oben am unteren Ende herein.

Abbildung 4.5: Der pocket-Algorithmus für „harte“ Quadrate.

- „Rejection-Free Geometric Cluster Algorithm for Complex Fluids“, Erweiterung um Gauß-artiges Potential [6]

Man bezeichnet den Algorithmus dann auch als „generalized geometric cluster algorithmus“.
Weiterhin können Vielteilchen-Terme (sofern deren energetische Störung klein ist) als Störung mitgenommen werden.

Andere Modelle

Natürlich kann es auch interessant sein andere Modelle zu betrachten. Ganz konkret könnten das zum Beispiel ionische Flüssigkeiten (Coulombpotential) sein oder Modelle für Flüssigkristalle („Lebwohl-Lasher“-Modell, der Algorithmus ist vom Wolff-Algorithmus abgeleitet, da sich auch um ein Modell mit Gitter handelt), wie in den nachfolgend angegebenen Artikeln.

- „Cluster Monte Carlo simulations of the nematic-isotropic transition“ [11]
- „Free energy and phase equilibria for the restricted primitive model of ionic fluids from Monte Carlo simulations“ [10].

Kapitel 5

Grenzen von Clusteralgorithmen und Fazit

5.1 Grenzen von Clusteralgorithmen

Bisher wurde nur besprochen, was mit Clusteralgorithmen besser geht und gezeigt, welche Vorteile sie haben. Dabei wurde zwar immer erwähnt, dass es sich um ganz bestimmte Bedingungen handelt, für die der Algorithmus konstruiert und betrachtet wurde, aber niemals wurden mögliche Probleme direkt angesprochen. Dies soll mit nachfolgender stichpunktartiger Liste nachgeholt werden:

- Clusteralgorithmen sind häufig schneller als lokale Algorithmen im Bereich von kritischen Punkten (d.h. nahe der Perkolationsgrenze bzw. den Phasenübergängen).
- Entfernt von kritischen Punkten kann die Konvergenz (ins Gleichgewicht) sehr schlecht werden.
- Konkreter Algorithmus kann für bestimmte (Anfangs-) Bedingungen versagen.
- Ein Problem besteht darin, dass der Algorithmus ganze Box in einen Cluster verwandeln kann. Dies kann zum Beispiel beim „harte Kugeln“-Modell mit Paarwechselwirkungen passieren, wenn die Dichte der Kugeln sehr groß ist und die ganze Box (da es sich um periodische Randbedingungen handelt, kann man auch sagen, dass es das ganze System ist) sich in einen großen Cluster verwandelt, wenn man mit dem Algorithmus nicht aufpasst. Dies liefert dann keine sinnvollen Ergebnisse mehr.
- Konkretes Beispiel eines Versagens: Onsagerproblem für Flüssigkristalle. Dabei werden die Moleküle als harte zylinderförmige Stäbe simuliert. Im Prinzip ist das Problem dann analog zu dem der zweikomponentigen Mischungen. Man kann nun aber zeigen, dass, wenn man einen Stab in die Box hineinschiebt, durchschnittlich mehr als ein Stab in der Box berührt wird. Damit würde dieser Algorithmus die ganze Box in einen Cluster verwandeln.

5.2 Fazit

Insgesamt stellen wir damit fest:

- Clusteralgorithmen sind häufig schneller als lokale Algorithmen im Bereich von Phasenübergängen (bzw. nahe der Perkolationsgrenze). Dies liegt daran, dass die Operation mit Clustern das „critical slowing down“ (z.B. im Bereich der Perkolationsgrenze) kompensieren.
- Clusteralgorithmen sind häufig gut parallelisierbar (skalieren gut mit Anzahl der Cores).
- Wegen ihrer in kritischen Bereichen noch guten Geschwindigkeit, konnten mit Clusteralgorithmen manche nicht analytisch lösbaren Systeme erstmals untersucht werden (da lokale Monte-Carlo-Algorithmen viel zu langsam sind, um diese Systeme mit interessanter Größe zu simulieren).
- Allerdings wird das ursprüngliche Problem verlagert: Man sucht einen Algorithmus, der einen oder mehrere für die Fragestellung möglichst gut geeignete(n) Cluster möglichst schnell aufbaut. Das heißt z.B. bei Algorithmen die mit dem Konzept der a-priori-Wahrscheinlichkeit arbeiten, dass sie möglichst hohe Akzeptanzraten der vorgeschlagenen Züge haben sollten („rejection free“).
- Clusteralgorithmen sind stark spezialisiert auf Modell und Fragestellung. Sie sind teils nur schlecht verallgemeinerbar oder übertragbar (Übertrag des Wolff-Algorithmus vom Ising-Modell auf ein kontinuierliches System ist z.B. teils langsamer als ein lokaler Algorithmus).
- Clusteralgorithmen können aufgrund ihrer Spezialisierung auf eine bestimmte Art von Systembedingung (z.B. auf die Nähe von Phasenübergängen) ihre gute Konvergenz gegenüber lokalen MC-Methoden verlieren.

Clusteralgorithmen sind also extrem auf ihren Einsatzzweck hin optimiert und für diese gehören sie zu den schnellsten Algorithmen. Diese Spezialisierung erkauft man sich um den Preis individuell zu kreierender Algorithmen und nicht gut übertragbarer Konzepte.

Literaturverzeichnis

[1] N. G. Almarza and E. Lomba. Cluster algorithm to perform parallel monte carlo simulation of atomistic systems. *The Journal of Chemical Physics*, 127(8):084116, 2007.

[2] A. A. Barker. Monte Carlo calculations of the radial distribution functions for a proton-electron plasma. *Australian Journal of Physics*, 18:119–133, April 1965.

[3] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[4] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, February 1925.

[5] Werner Krauth. *New Optimization Algorithms in Physics*, chapter 2, pages 5–22. Wiley-VCH, 8 2004.

[6] Jiwen Liu and Erik Luijten. Rejection-free geometric cluster algorithm for complex fluids. *Phys. Rev. Lett.*, 92(3):035504, Jan 2004.

[7] Jiwen Liu and Erik Luijten. Generalized geometric cluster algorithm for fluid simulation. *Phys. Rev. E*, 71(6):066701, Jun 2005.

[8] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. V. Chayes. Invaded cluster algorithm for equilibrium critical points. *Phys. Rev. Lett.*, 75(15):2792–2795, Oct 1995.

[9] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092, June 1953.

[10] Gerassimos Orkoulas and Athanassios Z. Panagiotopoulos. Free energy and phase equilibria for the restricted primitive model of ionic fluids from monte carlo simulations. *The Journal of Chemical Physics*, 101(2):1452–1459, 1994.

[11] N. V. Priezjev and Robert A. Pelcovits. Cluster monte carlo simulations of the nematic-isotropic transition. *Phys. Rev. E*, 63(6):062702, May 2001.

[12] Robert H. Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in monte carlo simulations. *Phys. Rev. Lett.*, 58(2):86–88, Jan 1987.

[13] Yusuke Tomita and Yutaka Okabe. Probability-changing cluster algorithm for potts models. *Phys. Rev. Lett.*, 86(4):572–575, Jan 2001.

[14] Ulli Wolff. Collective monte carlo updating for spin systems. *Phys. Rev. Lett.*, 62(4):361–364, Jan 1989.