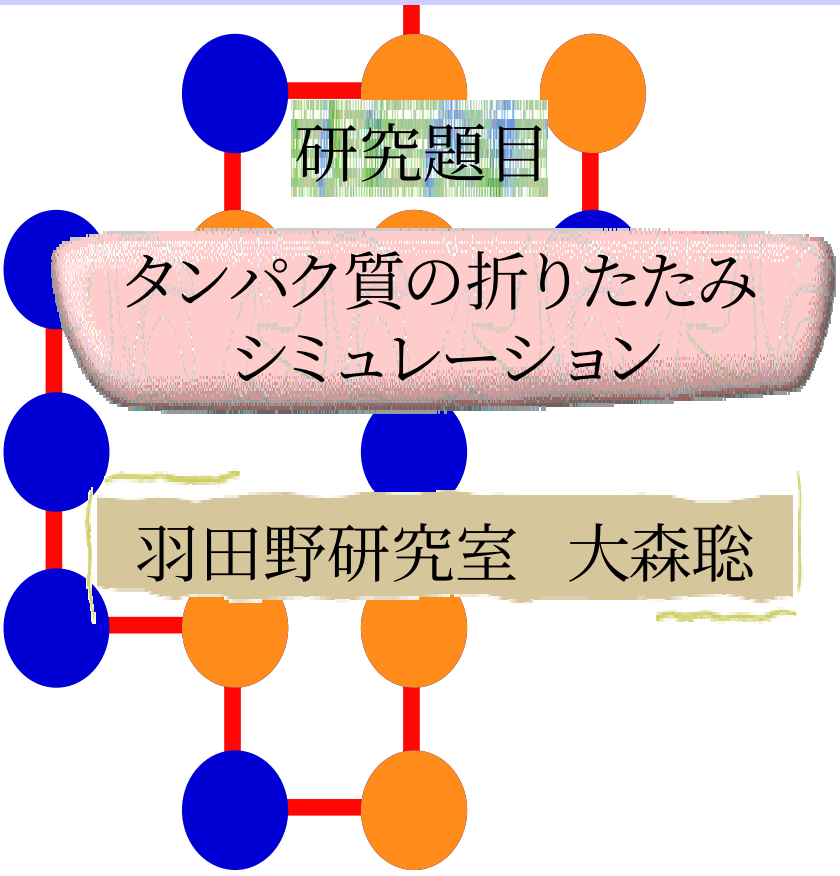
2001年度
青山学院大学理工学部物理学科
卒業論文



研究題目

タンパク質の折りたたみシミュレーション

羽田野研究室　大森聡

タンパク質の折りたたみ シミュレーション

大森聡
羽田野研究室

平成 14 年 3 月 12 日

概 要

タンパク質を 2 次元正方格子上に簡単化してモデル化し、その安定な折りたたみ状態をシミュレーションによって求めた。タンパク質はアミノ酸のチェーンが折りたたまれた立体構造を持つ。しかしその多くはアミノ酸配列と立体構造や機能との因果関係がわかっていない。本研究では疎水性相互作用がタンパク質の最も主要な駆動力であるという立場をとり、H-P 模型というタンパク質模型の折りたたみ問題を統計力学的立場から解明した。低エネルギー状態を確実に求めるために、自己回避規則を緩和してマルチカノニカル・モンテカルロ・シミュレーションを行った。その結果、通常の自己回避規則に則ったモンテカルロ・シミュレーションでは不可能だった $N = 64$ の長さのアミノ酸チェーンの基底状態探索に成功した。

目 次

1　タンパク質の折りたたみとは？

1.1　タンパク質の折りたたみ問題

「アミノ酸配列の情報からそのタンパク質の立体構造を予測すること」。これが本研究の最終的な目標である。タンパク質はアミノ酸が一次元的に連なって出来たチェーンが３次元的に折りたたまれて形成された高分子である（図 1)。チェーンのアミノ酸の数は通常 10〜1000 個程度である。

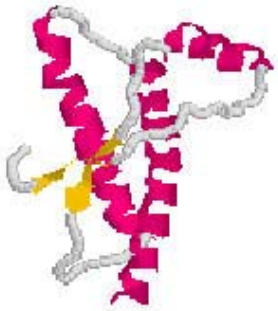


図 1: アミノ酸が一次元的に連なったチェーンが３次元的に折りたたまれる。

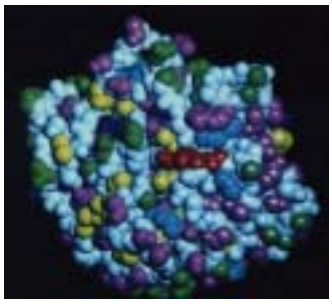


図 2: タンパク質は固有の立体構造を持つ。

このチェーンのアミノ酸配列によってタンパク質は特徴付けられ、１つのアミノ酸配列に対して１つの固有の立体構造が対応する (図 2)。タンパク質はその固有の立体構造を持って初めて生体としての機能を果たす。

タンパク質が固有の立体構造に折りたたまれるのは、タンパク質分子内、水とタンパク質の間の相互作用による。しかし、アミノ酸配列とタンパク質の立体構造との因果関係はまだよくわかっていない。従って「アミノ酸配列の情報からそのタンパク質の立体構造を予測すること」は未解決な問題である。

1.2　どのように取り組むか

このタンパク質の折りたたみ問題を理論的に解明するために、本研究ではモンテカルロ・シミュレーションを使っている。モンテカルロ・シミュレーションは、分子の配置をある確率法則の下に乱数を用いて作成

していく確率論的な方法である。この方法は熱力学的平衡状態にある系に対するシミュレーション方法である。

2　タンパク質の格子モデル化

2.1　格子モデル化の概念

本研究ではタンパク質を２次元格子モデルに簡単化してタンパク質の折りたたみのシミュレーションを行う。タンパク質は化学式で書くと図3の上のように表される。「１つのアミノ酸」と書かれた区切りがアミノ酸チェーンのポリマーを形成している１つ１つのアミノ酸を表している。実際にはこれが１次元的に10〜1000個くらい連なっている。アミノ酸の種類は、化学式に「R_1、R_2、R_3...」として示された「側鎖」によって決まる。側鎖が異なれば違うアミノ酸である。自然界には二十数種類のアミノ酸が存在する。

チェーンを形成するアミノ酸はペプチド結合と呼ばれる結合によってつながれている。図3でペプチド結合と書かれた区切りの中の部分は常に固定された平面上にある。それが側鎖の繋がっているC原子を要として回転する自由度を持つため、全体としてはチェーンのように動くことができるのである。

本研究ではペプチド結合の部分をボンド、その間の部分をセグメントとしてアミノ酸チェーンを格子モデル化する。図3がその概念図である。セグメントの区別だけでアミノ酸の種類わけを表している。

このように本研究では非常に簡単化されたモデルによってシミュレーションを行う。しかし、シミュレーションする際の相互作用を適切に選べば、実際のタンパク質のように、全状態数に対してごく少数の基底状態を持つようにできる。例え２次元正方格子上のモデルであっても、チェーンを完全に自由に動かせば、その状態数はアミノ酸数をNとして4^Nとなる。チェーンが重ね合わさることを禁止した自己回避規則を適用しても、全状態数は$N^{\gamma-1}\mu^N$のオーダーに及ぶことがわかっている（γは二次元正方格子上でおよそ$\frac{4}{3}$、μは2から4の間の数） [1]。よってNが実際のタンパク質を構成するアミノ酸の数程度でも、その折りたたみの状態数は天文学的数字となる。

以上のことから、このような非常に簡単化されたモデルでも、与えられたアミノ酸配列の折りたたみ方を完全に解くことができる方法が開発

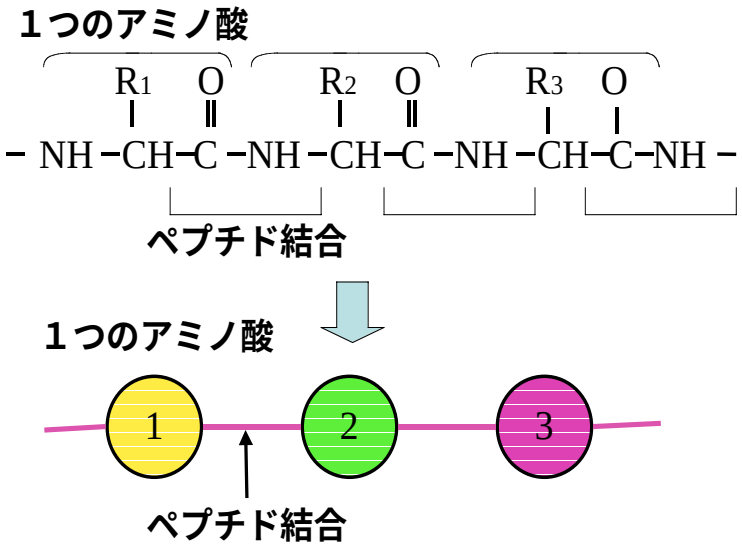


図 3: タンパク質の格子モデル化の概念図。

されることは意義がある。それが実際のタンパク質で「アミノ酸配列の情報からそのタンパク質の立体構造を予測すること」という問題を解決する手助けとなることが期待されるのである。

2.2 疎水性相互作用

本研究では「タンパク質の折りたたみの駆動力は疎水性相互作用である」という立場をとってシミュレーションを行う。疎水性相互作用とは非極性分子が水を避ける相互作用である。アミノ酸は大きくわけて、電気的極性を持っていて水に解けやすい親水性アミノ酸と、電気的極性を持っていないために水に解けにくい疎水性アミノ酸に分類される。親水性アミノ酸には、それ自身が電荷を持っているイオンのアミノ酸の他に、電子軌道の局在によって極性を持つものも含まれる。

本研究で取り扱われるのは水溶性の球状タンパク質である。球状タンパク質はいわば水の中に浸かった状態にある。そのため、疎水性アミノ酸は水を避けてタンパク質の内側に、親水性アミノ酸は水に解けやすいためにタンパク質の外側に位置しやすいと考えられる。これは、既に生理条件下での立体構造が解明されているタンパク質でも実際に見られる

傾向である。したがって、タンパク質が固有の立体構造を形成するにあたって、疎水性相互作用が重要な駆動力となっていることがうかがえる。疎水性相互作用を２次元格子上のモデルで再現すれば、実際に近いタンパク質の折りたたみのシミュレーションが行えるだろうというのが本研究のコンセプトである。

2.3 H-P 模型

本研究では**H-P 模型**と呼ばれる２次元正方格子モデルを用いた。これは**Lau** と **Dill** によって 1993 年に提出されたモデルである [2]。このモデルではアミノ酸は疎水性 (Hydrophobic) と親水性 (Polar) の２種類だけに簡単化される。タンパク質のアミノ酸配列は H と P の２種類のアミノ酸を表すセグメントをボンドでつなぎ合わせたチェーンとして表現される (図 4)。

チェーンは正方格子上で形状の自由度を持つ。そして H モノマー同士が最近接格子上に配置された時に系のエネルギーが１だけ下がるものとして、タンパク質の折りたたみでの疎水性相互作用を再現する。図 4 に例示するように、モデル全体は１本のチェーンが折りたたまれたものとなっている。H-P 模型の全エネルギーは

$$E(r) = \sum_{i<j+1} u(S_i, S_j)\Delta(r_i, r_j) \tag{1}$$

で与えられる。ここで r はチェーンの構造を表し、r_i は i 番目のアミノ酸の位置を示している。$\Delta(r_i, r_j)$ はもし r_i と r_j が最近接格子点なら -1 で、その他は 0 をとる記号である。S_i は i 番目のアミノ酸が親水性 (P) か疎水性 (H) かを指定し、記号 u は $u(H,H) = -1, u(H,P) = u(P,H) = u(P,P) = 0$ で定義される。

H-H contact が増える程エネルギーは低くなり、熱力学的に安定な状態となる。ある H-P 配列が与えられた時、その最もエネルギーの低い折りたたみ構造を求めるのが本研究の目的である。

3 自己回避規則とその欠点

現実のタンパク質の折りたたみでは、当然ながら同じ位置に２つ以上のアミノ酸は存在しない。また折りたたみの過程でアミノ酸のチェーンが

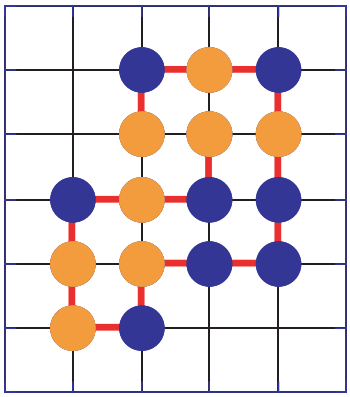


図 4: H-P model: オレンジ色のセグメントが疎水性 (Hydrophobic) アミノ酸、ブルーのセグメントが親水性 (Polar) アミノ酸、赤い線がペプチド結合を表す。

一旦切れるということもない。従って H-P 模型においては、２つ以上のセグメントが同一格子点上に配置されないようにチェーンを形作らなくてはならない。この規則を自己回避規則または排除体積規則と呼ぶ [3]。

また、本研究のシミュレーションの対象となる水溶性タンパク質（もしくは球状タンパク質）は溶液に浸かった熱平衡状態にあり、また粒子の出入りのない閉じた系である。そのため、状態のエネルギー分布はカノニカル分布に従う。カノニカル分布の状態の出現確率はエネルギーの関数として以下のように表される。

$$W(E_i) \propto \exp\left(-\frac{E_i}{T}\right) \tag{2}$$

従ってカノニカル分布ではエネルギーの低い状態ほど高い確率で出現する。

しかし、この自己回避規則とカノニカル分布に基づく状態出現確率の重みにそのまま従ってモンテカルロ・シミュレーションを行うと、多くの場合真の基底状態を求めることは困難である。以下で、カノニカル分布に基づくモンテカルロ・シミュレーションの手法とその欠点について述べる。

3.1 単純サンプリングと重み付きサンプリング

ここでは熱浴法によるモンテカルロ・シミュレーションの手法について説明する。カノニカル分布する系で、その系のエネルギーのみに依存する物理量 A の期待値 $\langle A \rangle$ は、

$$\langle A \rangle = \frac{\sum_n A_n \exp\left(-\frac{E_n}{T}\right)}{\sum_n \exp\left(-\frac{E_n}{T}\right)} \tag{3}$$

で与えられる。ここで $\sum_n$ は全状態に関する和である。この和をモンテカルロ・サンプリングで置き換える手法を単純サンプリングという。

しかし、低温においてはこのサンプリングは著しく不正確になる。重み $\exp\left(-\frac{E_n}{T}\right)$ のため、低温 $T \sim 0$ ではごく一部の低エネルギー状態しか和 $\sum_n$ に寄与しない。しかし単純サンプリングでは、一部の重要な状態をサンプリングできる確率は非常に小さい。

そこで考え出されたのは、全ての状態を平等にサンプリングするのではなく、期待値に大きく関わってくる状態、つまり出現確率の高い状態を重点的にサンプリングする方法である。出現確率に比例した頻度でサンプリングを行うと、その期待値は

$$\langle A \rangle = \frac{\sum_{n=1}^{M} A_n}{M} \tag{4}$$

として表される。ここで和 $\sum_{n=1}^{M}$ はサンプルに関する和、M はサンプリングの総数である。このサンプリング方法を重み付きサンプリングという。

3.2 カノニカル分布に基づくモンテカルロ・シミュレーション

ではどのようにサンプリングを行えば系の状態出現率に従った重みでサンプリングができるのだろうか？　その方法を以下で説明する。

系の状態を次々と生成するマルコフ過程をシミュレーションするというのが、メトロポリスらのアイディアである [4]。例えば熱浴法では、カノニカル分布する系がある状態 i からある状態 j に遷移する確率 P_{ij} を

$$P_{ij} = \frac{\exp\left(-\frac{E_j}{T}\right)}{\exp\left(-\frac{E_i}{T}\right) + \exp\left(-\frac{E_j}{T}\right)} \tag{5}$$

と定義する。この遷移確率によって系の状態を次々に生成する。すると、出現する状態の分布はカノニカル分布 $W(E_i) \propto \exp\left(-\frac{E_i}{T}\right)$ に近付くことが証明できる。こうして重み付きサンプリングが行われる。

実際にシミュレーションを行う上では、式 (5) の代わりに、状態 i と状態 j のエネルギー差を ΔE_{ij} として

$$P_{ij} = \frac{\exp\left(-\frac{\Delta E_{ij}}{T}\right)}{1 + \exp\left(-\frac{\Delta E_{ij}}{T}\right)} \tag{6}$$

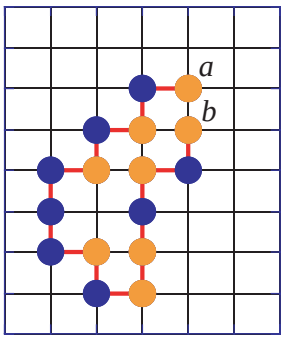
として計算を行う。いちいち系全体のエネルギーを求めるより、遷移前と遷移後のエネルギー差によってのみ遷移確率を算出した方がはるかに計算が速いからである。

カノニカル分布に基づくモンテカルロ・シミュレーションでは、より効率良く基底状態を実現するために、シミュレーションの初期では温度 T を高く設定する。その後ステップを重ねるに従って徐々に T を下げていく。初期では系がより広範囲の状態を取り得るようにして、少ないステップ数で基底状態に行き着こうというものである。

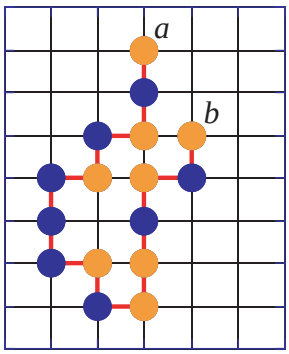
3.3 自己回避規則とその欠点

自己回避規則のある系を上のようにシミュレーションすると、以下のような欠点がある。例として、ある程度温度 T が下がった段階で系が図 5(a) のような状態にあるとする。これはこの H-P 配列での基底状態ではない。図 5(a) のアミノ酸チェーンは図 4 のアミノ酸チェーンと同じ H-P 配列を持っているが、図 4 では H-H contact の数が 7 つあるのに図 5(a) では 6 つしかない。よって図 5(a) の状態の方が図 4 の状態よりもエネルギーが高い。

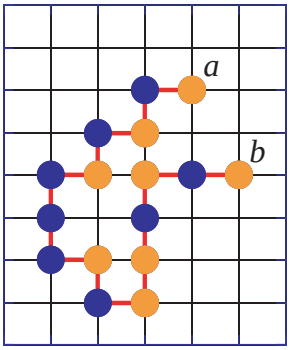
図 5(a) の状態から自己回避規則に従ったままシミュレーションを行なってより低いエネルギー状態に行き着くには、折りたたみを一旦ほどいた状態を経由しなければならない。しかし図 5(a) の状態からアミノ酸 a が図 5(b) のように遷移すると、H-H contact の数が少なくなるためエネルギーが上がる。またチェーンのもう一つの端にあるアミノ酸 b が図 5(c) のように遷移しても、やはりエネルギーがあがる。したがって自己回避規則に従っている場合、図 5(a) の状態は図 6 で赤い丸で示したようなエネルギー極小状態であるといえる。カノニカル分布ではエネルギーの高い状態ほどサンプリングされにくいので、特にある程度温度 T が低くなった



(a)



(b)



(c)

図 5: (a):アミノ酸チェーンの準安定状態。(b):アミノ酸 a が遷移したところ。(c):アミノ酸 b が遷移したところ。ここでオレンジのセグメントは H（疎水性）のアミノ酸を、ブルーのセグメントは P（親水性）のアミノ酸を、赤い線はボンドを表す。

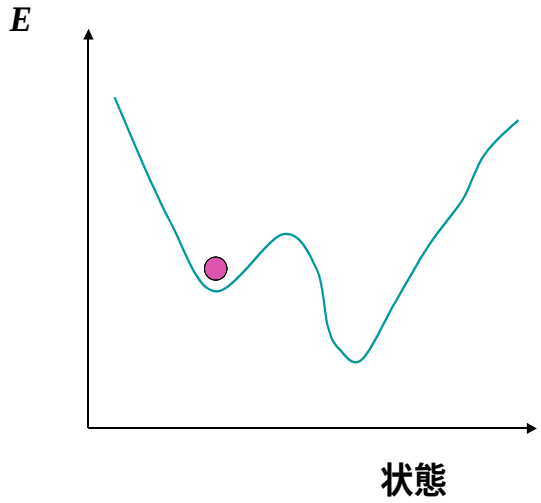


図 6: 系がエネルギーのローカルな極小状態にトラップされている。

段階で図 5(a) のような状態になってしまった場合、この状態から脱出して真の基底状態をサンプルするのは非常に困難である。これは長いチェーンであるほど顕著である。

4 マルチカノニカル・モンテカルロ・シミュレーション

以上のようなカノニカル分布に基づくモンテカルロ・シミュレーションの欠点から脱却するために、本研究ではマルチカノニカル・モンテカルロ・シミュレーションという手法を用いた。マルチカノニカル・モンテカルロ・シミュレーションとは、ある変数に対して等確率で状態が出現するようにマルコフ過程の遷移確率の重みを定義するシミュレーションである。

4.1 オーバーラップの緩和

ここでは、状態遷移の過程で複数のセグメントが同一格子点上に位置することを許す [5, 6]。遷移確率の重みを定義する変数としてエネルギー E と共に、セグメント同士の同一格子点上での重なり（オーバーラップ）として q という変数を使った。シミュレーションの途中では折りたたみの状態を E と q について等確率で出現させる。最終的には $q = 0$ での最もエネルギーの低い状態を抽出し、自己回避規則に従った基底状態を求めようというものである。

このシミュレーションでは重みを定義する変数について等確率で出現するため、ローカルなエネルギー極小状態のようなものが存在しない。従って、系がある特定の状態にトラップされて抜け出せなくなることがない。よって、より確実に真の基底状態を求めることができると期待される。以下ではマルチカノニカル・モンテカルロ・シミュレーションの手法を説明する。

4.2 マルチカノニカル・モンテカルロ・シミュレーションの重み

マルチカノニカル・モンテカルロ・シミュレーションも従来のモンテカルロ・シミュレーションと同様に重み付きのサンプリング法である。しかし、従来の重み

$$W(E_i) \propto \exp\left(-\frac{E_i}{T}\right) \tag{7}$$

ではエネルギーについて等確率で状態が出現しないので新たな重みを用いる。ある系でエネルギー E とオーバーラップ q による状態密度が $D(E, q)$ で表されたとすれば、状態 i の出現する確率を

$$W(E_i, q_i) \propto \frac{1}{D(E_i, q_i)} \tag{8}$$

としてシミュレーションを行えば、その結果出現する状態は E, q について等確率なはずである。

しかし、実際には H-P 配列を入力しただけでは、どの状態がどのくらい存在するか予測するのは不可能である。つまりシミュレーションの開始段階では $D(E, q)$ は未知である。そこで、シミュレーションではまず $D(E, q)$ を推測し、そこから E, q について等確率で出現させる $W(E, q)$ を算出していく、という道筋をたどる。

4.3 重みの算出

まず、シミュレーションの開始段階ではあらゆる状態 i に対し

$$W_0(E_i, q_i) = 1 \tag{9}$$

という重みを与えてチェーンの折りたたみの遷移を行う。ある程度のモンテカルロ・ステップ数の遷移を行えば、出現した状態の分布は実際の状態密度 $D(E,q)$ に近い分布になると期待される。つまり、この段階のサンプリングで出現した状態のヒストグラムを $H_0(E,q)$ とすると、

$$H_0(E,q) \simeq W_0(E,q)D(E,q) \tag{10}$$

となる。そこで、この時点では状態密度を

$$D_0(E,q) \equiv \frac{H_0(E,q)}{W_0(E,q)} \tag{11}$$

と推測する。次に、ここで推測した状態密度 $D_0(E,q)$ を基にして新たなシミュレーションの重み $W_1(E,q)$ を

$$W_1(E,q) \propto \frac{1}{D_0(E,q)} = \frac{W_0(E,q)}{H_0(E,q)} \tag{12}$$

と与えてシミュレーションを行う。以下同様にして

$$W_n(E,q) \propto \frac{1}{D_{n-1}(E,q)} = \frac{W_{n-1}(E,q)}{H_{n-1}(E,q)} \tag{13}$$

として重み関数 $W_n(E,q)$ をアップデートする。この試行を繰り返すことによって徐々に E、q について等確率で系の状態を出現させようというものである。

5 格子モデルの動き

ここでは本研究のシミュレーションでのマルコフ過程の各 1 ステップでの H-P 模型の動きについて説明する。このシミュレーションでは 1 ステップで 1 つのアミノ酸だけを動かす。系のエネルギーを求める際に、動かすアミノ酸 1 つについてだけ遷移前と遷移後の H-H contact の数を比較するだけでいいからである。1 ステップで複数のモノマーを動かすより

もエネルギーの計算が速く、したがってより短時間で多くの状態をサンプリングすることができ、より速く状態密度 $D(E,q)$ を推定することができると期待される。

また、各ステップで動かすアミノ酸はランダムに決定される。また、アミノ酸をつなぐボンドは遷移過程で切断されたり伸びたり縮んだりしないものとする。

5.1 Pivot move

図 7(a) は pivot move と呼ばれる動きである。ピンク色のセグメントが動くアミノ酸を示す。動くアミノ酸がチェーンの両端のどちらかだった場合、その隣のアミノ酸を要 (pivot) として $\pm$ 90 °または 180 °回転する動きである。

5.2 One-bead flip

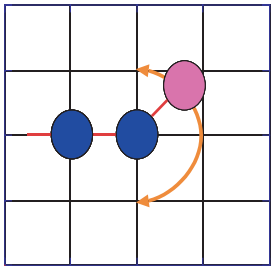
図 7(b) は one-bead flip と呼ばれる動きである。これは、動くアミノ酸がチェーンの折れ曲がったコーナーにあるときにのみ適用される動きである。その条件を数式で表すと、動くアミノ酸の座標を x_i, y_i、その両隣りにあるアミノ酸の座標をそれぞれ x_{i-1}, y_{i-1}、x_{i+1}, y_{i+1} として、

$$\begin{cases} x_i = x_{i-1} \text{かつ } y_i = y_{i+1} \\ y_i = y_{i-1} \text{かつ } x_i = x_{i+1} \end{cases} \tag{14}$$

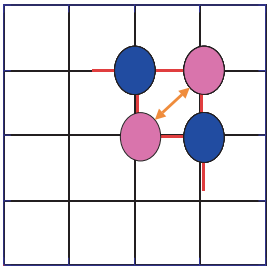
である。動きとしてはコーナーが逆に折れ曲がる動きである。

5.3 Jackknife move

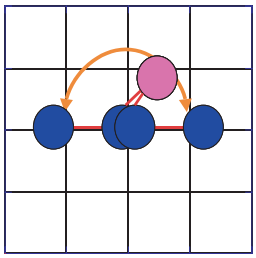
図 7(c) は jackknife move と呼ばれる動きである。これは、動くアミノ酸 i に対して $i-1$ 番目と $i+1$ 番目のアミノ酸が同じ格子点上にある場合にのみ適用される動きである。自己回避規則を緩和して初めてできる動きであると言える。動きとしては pivot move と同様に、i 番目のアミノ酸がその隣のアミノ酸を要として $\pm$ 90 °または 180 回転する動きである [6]。以上で述べた動きが、ボンドを切断したり変形させたりしないで可能なアミノ酸 1 個の動きの全てである。



(a) pivot move;



(b) one-bead flip;



(c) jackknife move;

図 7: マルコフ過程の各ステップでのアミノ酸の動かし方。

5.4　遷移確率

マルチカノニカル・モンテカルロ・シミュレーションではどのように遷移確率が求められるのかを説明する。重み関数 $W(E,q)$ を n 回アップデートした後を考える。つまり重みは関数は $W_n(E,q)$ と書き表せる。今、系が状態 i にあり、ランダムに選ばれたアミノ酸の１ステップの遷移で到達可能な状態が状態 i、状態 j、状態 k、状態 l だったとする。それぞれの状態のエネルギーとオーバーラップ数を E_i、E_j、E_k、E_l、q_i、q_j、q_k、q_l とする。この時、状態 i から状態 j に遷移する確率 P_{ij} は

$$P_{ij} = \frac{W_n(E_j, q_j)}{W_n(E_i, q_i) + W_n(E_j, q_j) + W_n(E_k, q_k) + W_n(E_l, q_l)} \tag{15}$$

として表される。重み関数のアップデートを重ねると、エネルギーとオーバーラップによって定義される状態の出現数は等しくなることが期待される。

6　オーバーラップ数の制限規則

6.1　オーバーラップ数のカウント方法

まず初めに、このシミュレーションでのオーバーラップ数のカウントの仕方を説明する。例として、系が図8の状態にあるとする。この時、赤い矢印で示した格子点でアミノ酸が２つ重なっているのでオーバーラップ数は $q = 1$ となる。ここで強調したいのはオーバーラップ数とはアミノ酸同士が現在重なっている回数であって複数のアミノ酸が重なっている格子点の数ではないということである。もし後者のカウントの仕方をしてしまうと、１つの格子点の上に３つ以上のアミノ酸が存在しても q の数が変わらないことになる。すると q の数があまり違わないのに実際のチェーンの形状は大きく異なってしまい、正しくマルチカノニカル・モンテカルロ・シミュレーションが行われなくなる。

6.2　状態数の重要性

第4.3節で、「ある程度のモンテカルロ・ステップ数の遷移を行えば、出現した状態の分布は実際の状態密度に近い分布になると考えられる。」と述べた。これは状態密度 $D(E,q)$ を正確に推定するには、その系のとり得

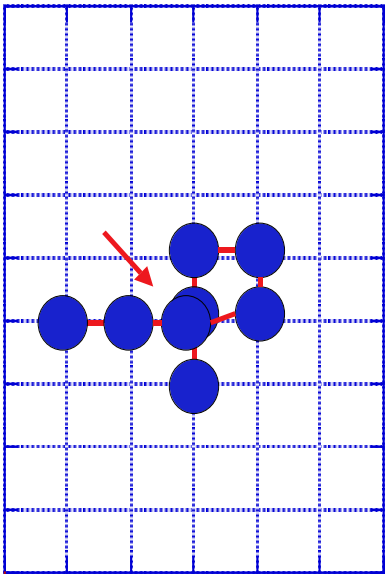


図 8: $q = 1$ の状態

る状態数に対し十分多くの回数のモンテカルロ・ステップを行わなければならないことを意味する。状態密度 $D(E, q)$ が正確に求まらなければ、シミュレーションの重み関数 $W(E, q)$ も不適当なものとなり、正しいマルチカノニカル・モンテカルロ・シミュレーションとならない。

しかし、オーバーラップを無制限に許してしまうと、状態数は 4^N となる。例えば $N = 64$ のチェーンの場合、状態数は $4^{64} \simeq 3.4 \times 10^{38}$ にも及んでしまう。この状態数に対して十分なモンテカルロ・ステップ数を行って $D(E, q)$ を求めるのは非常に困難である。そこで考えられるのは、オーバーラップにある程度制限を設け状態数を減らし、$D(E, q)$ を算出しやすくするという方法である。

6.3 制限規則

本研究のシミュレーションではオーバーラップに次のような制限を設けた:

- 1 つの格子点上に存在できるアミノ酸は 2 つまで。
- $q \leq \frac{N}{8}$.

１つめの制限規則は１つの格子点上に 3 つ以上のアミノ酸が存在することを禁止するため、系の状態数を減らすのに有効であると考えられる。2 つめの制限規則は、最終的に求めたいのは $q = 0$ での低エネルギー状態なので、それと懸け離れた状態を禁止することで能率的なサンプリングが行われることが期待される。

6.4 ボンドの規則の重要性

以上のように、本研究のシミュレーションでは、実際にはありえないアミノ酸同士のオーバーラップを遷移過程である程度緩和している。しかし、ボンドについては、自己回避規則に則ったモンテカルロ・シミュレーションと同様に一切の緩和を行っていない。ボンドについてもある条件下でフリーにしたり伸びたり縮んだりすることを許せば、遷移過程でモデルがより形状を自由に変えて、少ないステップ数で幅広い状態をとることによって更に能率的に基底状態を探すことが出来るのではないだろうか？

残念ながらこの方法は、特に粒子数 N の多いモデルについては有効ではない。ボンドの規則を緩和することによって取り得る状態数があまりにも増大してしまい、$D(E, q)$ を推測できないからだ。例えば、64 個のセグメントをつなぐ 63 本のボンドの内 1 本だけでもフリーにすると、その時点で系の取り得る状態数は無限大になってしまう。フリーにまではしなくても、１つの状態につき 1 本のボンドだけ 0〜2 の範囲で（通常のボンドの長さを１とする）伸び縮みを許すシミュレーションを実際に行ったが、モンテカルロ・ステップ数をかなり多く行っても、E と q の 2 次元ヒストグラム上に状態密度 $D(E, q)$ の概形をサンプリングすることが出来なかった。したがって、ボンドを緩和してしまうとマルチカノニカル・モンテカルロ・シミュレーションを行うのは現実的に不可能であると考えられる。

7　シミュレーション結果

7.1　2 次元 H-P 模型（$N = 64$）

本研究では以下のような 64 個の H-P 配列を 2 次元正方格子上で折りたたむことに取り組んだ。

$$H_{12}PHPHP_2H_2P_2H_2P_2HP_2H_2P_2H_2P_2HP_2H_2P_2H_2P_2HPHPH_{12} \quad (16)$$

この H-P 配列は Unger と Moult によって 1993 年に提出された [7]。H-P 模型を使ったタンパク質の折りたたみのアルゴリズムの性能評価に使われている。

7.2　重みの学習

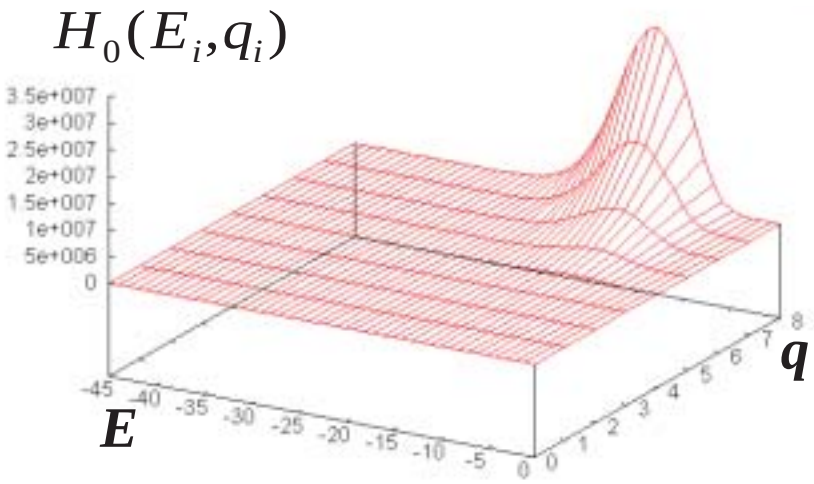
この H-P 配列について、第 5 節と第 6 節の動きとルールに基づいてマルチカノニカル・モンテカルロ・シミュレーションを行った。重み関数 $W_n(E, q)$ の学習にあたっては、状態密度 $D_n(E, q)$ のサンプリング 1 回につき 5×10^8 モンテカルロ・ステップだけ行った。5×10^8 モンテカルロ・ステップ毎に $W_n(E, q)$ をアップデートしてシミュレーションを行った。

1 回目のサンプリングでは全ての状態 i に対して重みを $W_0(E_i, q_i) = 1$ としてサンプリングを行った。

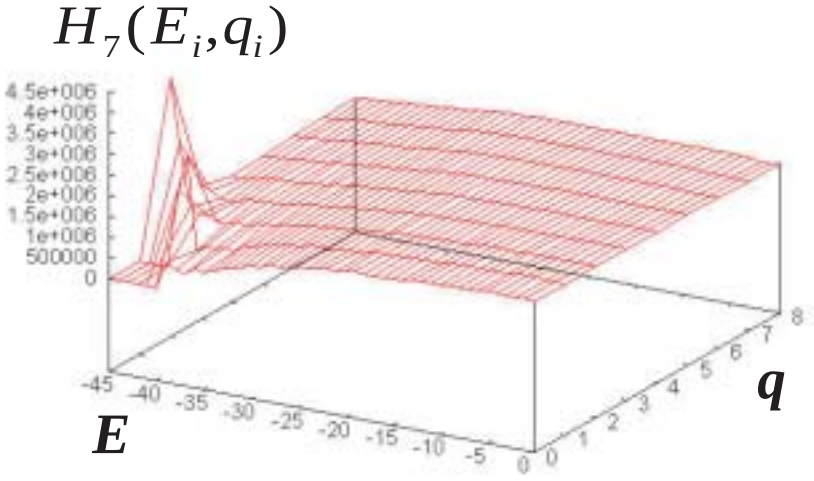
得られたヒストグラム $H_0(E, q)$ が図 7.2(a) で示された 2 次元ヒストグラムである。遷移に用いた重みがフラットであり、またこれ以上モンテカルロ・ステップ数を増やしても概形は変わらない。よってこのヒストグラムはこの系の状態密度 $D(E, q)$ の概形を表していると考えられる。

以降、第 4 節に述べた方法で重みをアップデートしてシミュレーションを続けた結果、8 回目のサンプリングでは図 7.2(b) のようなヒストグラムが得られた。1 回目のヒストグラムよりは E と q について、よりフラットな確率で出現していることがわかるだろう。つまり、この時点でほぼ正確に、E と q について等確率で状態を出現させる重みが計算されていると考えられる。

30 回目のサンプリングでは図 7.2(c) のようなヒストグラムが得られた。10 回目のサンプリングあたりからほぼこのようにフラットなヒストグラムが得られる。その後 30 回目のサンプリングまで安定してフラットなので、重みが正確に計算されたと考えられる。また同時に、1 回のサンプリ

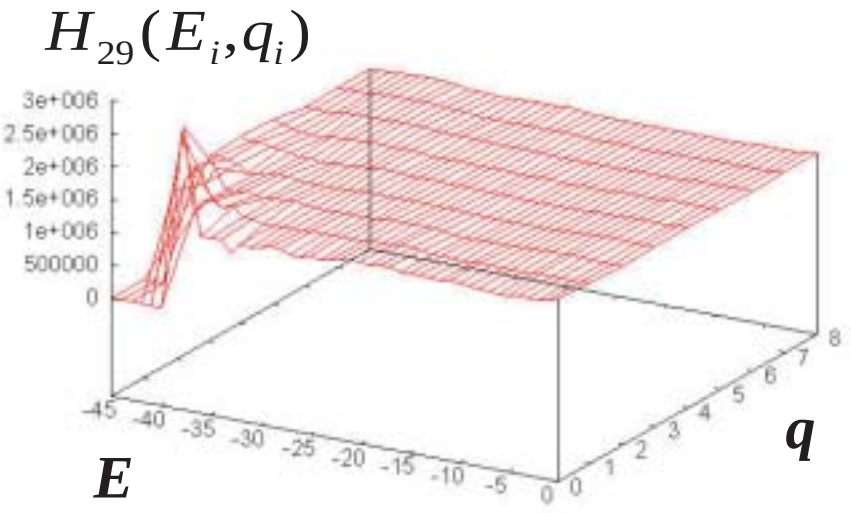


(a)



(b)

図 9: アミノ酸配列 16 に対して得られたヒストグラム: (a)1 回目; (b) 8 回目; (c)30 回目.


$H_{29}(E_i, q_i)$
3e+006
2.5e+006
2e+006
1.5e+006
1e+006
500000
0
-45
-40
-35
-30
-25
-20
-15
-10
-5
0
E
0
1
2
3
4
5
6
7
8
q

(c)

ングについて 5×10^8 というモンテカルロ・ステップ数が十分な回数であることを示している。

もしサンプリング 1 回当たりのモンテカルロ・ステップ数が足りなかった場合、以下のようなことが起こる。まず、サンプリングの回数が余りにも不足していた場合、1 回目のサンプリングでの $H_0(E, q)$ のヒストグラムが図 7.2(a) の概形にならない。それよりは多くの回数のモンテカルロ・ステップを行い 1 回目のサンプリングによる $H_0(E, q)$ のヒストグラムの概形がほぼ同じになっても、2 回目のサンプリングでどこかの状態に鋭いピークができてしまうことがある。更にそれよりもモンテカルロ・ステップ数を増やし、もし 1 回目、2 回目、3 回目とサンプリングを重ねていく毎にヒストグラムがフラットになっていったとしても、モンテカルロ・ステップ数が不足している場合はあるサンプリング回数を境にヒストグラムが大きく波打つような形になり、結局どこかの状態にピークができてしまう。以上より、フラットなヒストグラムが安定して得られた 5×10^8 というモンテカルロ・ステップ数が十分であることがわかる。

7.3 折りたたまれたチェーン

以上のように、重みをアップデートしながらマルチカノニカル・モンテカルロ・シミュレーションを行った結果、$q = 0$ でのエネルギーが最も低い状態は図 10 のようになった。オレンジ色のセグメントで表された疎水性アミノ酸がコンパクトに集合していることがわかる。このアミノ酸チェーンでは、アミノ酸を 1 直線上に配置するだけで 28 個の H-H contact を持つ。これがエネルギー最大の状態である。エネルギー最大の状態を $E = 0$ とすると、求めた折りたたみ状態では $E = -42$ となる。このアミノ酸チェーンはそもそも $E = -42$ が基底状態になるように人為的にデザインされたものである。$E = -42$ が本当に基底状態なのか厳密に数学的に証明することはできないが、現在までのところこれより低いエネルギーの折りたたみ方は発見されておらず、非常に高い信頼を持って $E = -42$ が基底状態であると思われている。よって本研究のシミュレーションで、この H-P 配列の基底状態のサンプリングに成功したと考えられる。

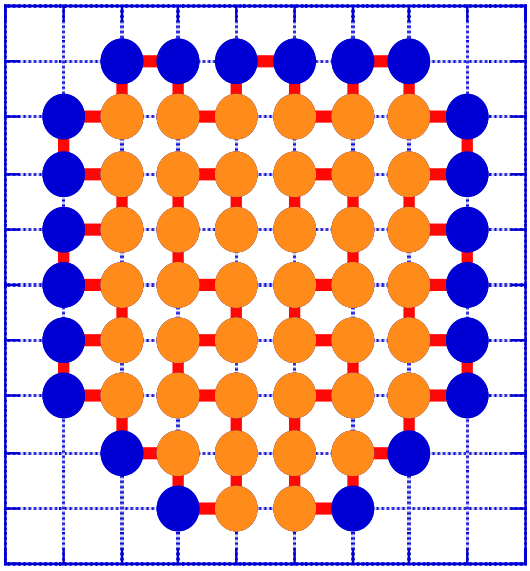


図 10: 本研究のマルチカノニカル・モンテカルロ・シミュレーションで得られた $N = 64$ のアミノ酸チェーンの折りたたみの基底状態。

7.4 考察

本研究のシミュレーションでは、重みを求めるための 1 回のサンプリング当たり 5×10^8 のモンテカルロ・ステップを行うことによって基底状態のサンプリングに成功した。1 回のサンプリングには $369MHz$ の PC で約 1 時間 20 分程を要した。モンテカルロ・ステップ 5×10^8 によるシミュレーションは 3 回行い、3 回とも基底状態のサンプリングに成功したが、基底状態をサンプリングするまでの重みのアップデート回数はその時によって異なった。最も早くて 6 回目のサンプリング、遅くて 9 回目のサンプリングで基底状態を達成した。しかし、いずれのサンプリングでもヒストグラム $H(E, q)$ がフラットに落ち着くのは 10 回目あたりからだった。

図 10 をよく見ると、オレンジ色のセグメントの疎水性アミノ酸の配置の仕方がこの図と少し違っても、つまり折りたたまれたチェーンの中の部分は他の折りたたみ方をしても同じエネルギー状態を実現できることがわかる。従って、この H-P 配列のアミノ酸チェーンはいくつかの状態が基底状態に縮退していることになる。よって本研究で扱った問題は実際のタンパク質のように基底状態が一意的なアミノ酸の配列よりも基底

状態の探索が簡単なものだったと言える。しかし、本研究のシミュレーションでは $N = 64$ の配列の E と q による状態を等確率で出現させる重みの算出に成功したので、例え基底状態が 1 つしかない H-P 配列が与えられたとしても、$N = 64$ 程度なら十分基底状態を探索できる性能があると考えられる。

よってタンパク質の折りたたみのシミュレーションを行うアルゴリズムの性能を評価する上で、重みの算出に要した時間が重要な要素になってくる。本研究のアルゴリズムでは大体 5×10^8 モンテカルロ・ステップのサンプリングを 10 回、つまり合計で 5×10^9 モンテカルロ・ステップの遷移を行ったが、$N = 64$ のアミノ酸チェーンの状態数の多さを考えれば十分能率的なシミュレーションと言えるだろう。

以上のように、本研究のシミュレーションのアルゴリズムは $N = 64$ のアミノ酸チェーンの基底状態探索には十分な確実性を持っており、なおかつ重みの算出に要した時間も今まで報告されているものより数倍速いものである。しかし、$N = 64$ よりも大きなアミノ酸チェーンの基底状態探索はまだ不十分である。$N = 100$ のチェーンの折りたたみに関しては、同じ H-P 配列を用いたシミュレーションで現在報告されている最も低いエネルギー状態に到達していない。本研究のアルゴリズムが重みを短時間で計算できた要因としては、

- 1 ステップで動かすアミノ酸が 1 個だけなので、エネルギーの算出にかかる時間が短く、同じ時間でより多くのステップを行うことができる。

- 第 6.3 節で示した制限規則により、チェーンの折りたたみの全状態数が少なく、状態密度 $D(E, q)$ の探索が容易である。

ことが考えられる。しかし、1 つめの条件のために 1 ステップは素早く行えるものの、動きが小さいために N が大きいチェーンでは十分多くの状態をサンプリングするのに非常に大きなモンテカルロ・ステップ数を要する。また、2 つめの条件のために形状がより複雑になる N の大きなチェーンでは遷移のしかたが制限されてしまい、状態のサンプリングに限界がある。これらの要因によって N の大きなアミノ酸チェーンでの基底状態探索の性能が不十分なものになってしまったと考えられる。とくに、遷移の制限規則にはまだ多くの改良の余地が残されていると思われる。

8　結論

本研究では$N = 64$のアミノ酸チェーンの折りたたみの基底状態探索に成功した。基底状態に行き着く過程では現実に存在しないオーバーラップの状態を許し、またシミュレーションの状態の重みに関しても現実の状態の出現の重みと異なるものを用いた。しかし、算出した結果は現実に即したアミノ酸チェーンが自己回避している状態である。

全ての自己回避の状態は互いに自己回避規則に従ったまま到達できるパスを持っている。タンパク質の折りたたみ問題を解明する上で、それを形成するアミノ酸チェーンの基底状態を求めるのがまず第一歩だと考えられる。その点で本研究は問題解決に意義のあるものである。

9　謝辞

本研究を行うにあたり、羽田野先生には大変お世話になりました。特に、テーマ選びとマルチカノニカル・モンテカルロ・シミュレーションの方法について親切にご指導いただきました。また、先輩方、同級生の方々にも幾度となく助けていただきました。この場をお借りして御礼申し上げます。

参考文献

[1] ド・ジャン『高分子の物理学』(吉岡書店、1984)

[2] K.F. Lau, K.A. Dill, *Macromolecules* **22**, 3986-3997 (1989).

[3] 土井正男・小貫明『高分子物理・相転移ダイナミクス』(岩波書店、1992)

[4] 神山新一・佐藤明『モンテカルロ・シミュレーション』(朝倉書店、1997)

[5] 千見寺浄慈『計算統計力学的手法による格子タンパク質模型の研究』物性研究　73-6 (2000-3).

[6] Y. Iba, G.Chikenji, and M. Kikuchi, *J. Phys. Soc. Jpn.* **67**, 3327 (1998).

[7] R. Unger and J. Moult, *J. Mol. Biol.* **231**, 75-81 (1993).

A　アミノ酸チェーンの折りたたみのプログラム

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#define _N 64
#define SIMPLE_SPRNG
#include "sprng.h"

  int overlap(int f,int g,int first,int last);
  int energy(int a,int b,int c,int first,int last);

main(){
  int j,k,l,s,s_max,m,m_max;
  static int p[_N],x[_N],y[_N],x_min[_N],y_min[_N],h[442][_N],sum_h;
  int x_plus,y_plus,x_minus,y_minus,x_flip,y_flip,x_reverse,y_reverse;
  int v,v_plus,v_minus,v_flip,v_reverse,sum_v,v_max;
  static double R,w_plus,w_minus,w_flip,w_reverse;
  int e,e_plus,e_minus,e_flip,e_reverse;
  int sum_e,sum_e_min,sum_e_max,sum_e_min_min;
  int index_e_plus,index_e_minus,index_e_flip,index_e_reverse,index_e;

  int index_v_plus,index_v_minus,index_v_flip,index_v_reverse;
  static double sum_r_h,w[442][_N],sum_w,r_w;
  int i;
  int seed=2734534;

  int overlap(int f,int g, int first,int last){ /*overlap count*/
    int ans;
    ans=0;
       for(j=first;j<last;j++){
        if(f==x[j] && g==y[j]){
          ans=ans+1;
}
        if(ans>=2){
```

```
          ans=v_max;
          break;
}
       }
       return ans;
  } /*overlap count end*/

  int energy(int a,int b,int c,int first,int last){ /*energy count*/
    int ans;
    ans=0;
   if(c==1){
     for(j=first;j<last;j++){
        if(((a+1==x[j] || a-1==x[j]) && b==y[j]) && p[j]==1)
          {
            ans=ans-1.0;
          }
        if(((b+1==y[j] || b-1==y[j]) && a==x[j]) && p[j]==1)
          {
            ans=ans-1.0;
          }
     }
   }
   return ans;
  }/*energy count end*/

  FILE *datafile;
  init_sprng(DEFAULT_RNG_TYPE,seed,SPRNG_DEFAULT);

  s=m=0;
  sum_e_min=sum_e_max=0;

  s_max=300000000;
  m_max=100;

  v_max=9;        /*plot area*/
```

```
  sum_e_min_min=46;

  for(l=0;l<442;l++){
   for(j=0;j<_N;j++){
    w[l][j]=0.0;
   }
  }
  for(l=0;l<sum_e_min_min;l++){
    for(j=0;j<v_max;j++){
    w[l][j]=1.0;
    }
  }
  for(l=0;l<442;l++){
   for(j=0;j<_N;j++){
    h[l][j]=0;
   }
  }
 p[0]=1;p[1]=1;p[2]=1;p[3]=1;p[4]=1;p[5]=1;p[6]=1;p[7]=1;p[8]=1;p[9]=1;
  p[10]=1;p[11]=1;p[12]=0;p[13]=1;p[14]=0;p[15]=1;p[16]=0;p[17]=0;p[18]=1;
  p[19]=1;p[20]=0;p[21]=0;p[22]=1;p[23]=1;p[24]=0;p[25]=0;p[26]=1;p[27]=0;
  p[28]=0;p[29]=1;p[30]=1;p[31]=0;p[32]=0;p[33]=1;p[34]=1;p[35]=0;p[36]=0;
  p[37]=1;p[38]=0;p[39]=0;p[40]=1;p[41]=1;p[42]=0;p[43]=0;p[44]=1;p[45]=1;
  p[46]=0;p[47]=0;p[48]=1;p[49]=0;p[50]=1;p[51]=0;p[52]=1;p[53]=1;p[54]=1;
  p[55]=1;p[56]=1;p[57]=1;p[58]=1;p[59]=1;p[60]=1;p[61]=1;p[62]=1;p[63]=1;

  for(j=0;j<_N-1;j++){ /*initial energy*/
   sum_e_max=sum_e_max-(p[j]*p[j+1]);
  }
  for(j=0;j<_N;j++){ /*sequence input*/
   x_min[j]=j;
   y_min[j]=0;
  }
for(m=1;m<=m_max;m++){  /* "m" roop*/

  for(j=0;j<_N;j++){ /*sequence input*/
```

```
  x[j]=j;
  y[j]=0;
 }
 sum_e=sum_e_max;
 sum_v=0;

 printf("present sum_e_min=%2d m=%d\n",sum_e_min,m);

for(l=0;l<sum_e_min_min;l++){ /*hystgram initialize*/
  for(j=0;j<v_max;j++){
   h[l][j]=0;
  }
} /*hystogramu initialize end*/
sum_h=0;

for(s=1;s<=s_max;s++){ /*"s" roop*/

 k=sprng()*_N;
 e=e_plus=e_minus=e_flip=e_reverse=0;
 index_e_plus=index_e_minus=index_e_flip=index_e_reverse=index_e=0;
 v=v_plus=v_minus=v_flip=v_reverse=0;
 index_v_plus=index_v_minus=index_v_flip=index_v_reverse=0;
 x_plus=x_minus=x_flip=x_reverse=x[k];
 y_plus=y_minus=y_flip=y_reverse=y[k];
 w_plus=w_minus=w_flip=w_reverse=r_w=0.0;
 sum_h=0;

 e=energy(x[k],y[k],p[k],0,_N);
 for(j=0;j<_N;j++){ /*overlap count*/
     if(k!=j && (x[k]==x[j] && y[k]==y[j]))
       {
        v=v+1;
        break;
       }
 }/*overlap count end*/
```

```
if(k==0){ /*[0]monomer pivot*/
     x_plus=x[1]-(y[0]-y[1]); /*+90 pivot*/
     y_plus=y[1]+(x[0]-x[1]);
     v_plus=overlap(x_plus,y_plus,1,_N);
     e_plus=energy(x_plus,y_plus,p[0],1,_N);

     x_minus=x[1]+(y[0]-y[1]); /*-90 pivot*/
     y_minus=y[1]-(x[0]-x[1]);
     v_minus=overlap(x_minus,y_minus,1,_N);
     e_minus=energy(x_minus,y_minus,p[0],1,_N);

     x_reverse=x[1]-(x[0]-x[1]);/*180 pivot*/
     y_reverse=y[1]-(y[0]-y[1]);
     v_reverse=overlap(x_reverse,y_reverse,1,_N);
     e_reverse=energy(x_reverse,y_reverse,p[0],1,_N);

} /*[0]monomer pivot move end*/


else if(k==_N-1){ /* [_N-1]monomer pivot move*/
     x_plus=x[_N-2]-(y[_N-1]-y[_N-2]); /*+90 pivot*/
     y_plus=y[_N-2]+(x[_N-1]-x[_N-2]);
     v_plus=overlap(x_plus,y_plus,0,_N-1);
     e_plus=energy(x_plus,y_plus,p[_N-1],0,_N-1);

     x_minus=x[_N-2]+(y[_N-1]-y[_N-2]); /*-90 pivot*/
     y_minus=y[_N-2]-(x[_N-1]-x[_N-2]);
     v_minus=overlap(x_minus,y_minus,0,_N-1);
     e_minus=energy(x_minus,y_minus,p[_N-1],0,_N-1);

     x_reverse=x[_N-2]-(x[_N-1]-x[_N-2]); /*180 pivot*/
     y_reverse=y[_N-2]-(y[_N-1]-y[_N-2]);
     v_reverse=overlap(x_reverse,y_reverse,0,_N-1);
     e_reverse=energy(x_reverse,y_reverse,p[_N-1],0,_N-1);
```

```
} /*[_N-1] pivot move end*/


else{
 if(x[k]==x[k-1] && y[k]==y[k+1]){ /* "plus flip" */
     x_flip=x[k+1];
     y_flip=y[k-1];
     v_flip=overlap(x_flip,y_flip,0,_N);
     e_flip=energy(x_flip,y_flip,p[k],0,_N);
 }/* "plus flip" end*/

 if(x[k]==x[k+1] && y[k]==y[k-1]){ /* "minus flip" */
     x_flip=x[k-1];
     y_flip=y[k+1];
     v_flip=overlap(x_flip,y_flip,0,_N);
     e_flip=energy(x_flip,y_flip,p[k],0,_N);
 } /* "minus flip" end*/

 if(x[k-1]==x[k+1] && y[k-1]==y[k+1]){/* jackknife move*/
     x_plus=x[k-1]-(y[k]-y[k-1]); /*+90 pivot*/
     y_plus=y[k-1]+(x[k]-x[k-1]);
     v_plus=overlap(x_plus,y_plus,0,_N);
     e_plus=energy(x_plus,y_plus,p[k],0,_N);

     x_minus=x[k-1]+(y[k]-y[k-1]);/*-90 pivot*/
     y_minus=y[k-1]-(x[k]-x[k-1]);
     v_minus=overlap(x_minus,y_minus,0,_N);
     e_minus=energy(x_minus,y_minus,p[k],0,_N);

     x_reverse=x[k-1]-(x[k]-x[k-1]); /*180 pivot*/
     y_reverse=y[k-1]-(y[k]-y[k-1]);
     v_reverse=overlap(x_reverse,y_reverse,0,_N);
     e_reverse=energy(x_reverse,y_reverse,p[k],0,_N);
 }
```

```
  }/*"else" end*/


  /*multicanonical method*/
  R=sprng();

   index_e=-1*(sum_e-sum_e_max);

  if(k==0 || k==_N-1){
   index_e_plus=-1*(sum_e+e_plus-e-sum_e_max);
   index_e_minus=-1*(sum_e+e_minus-e-sum_e_max);
   index_e_reverse=-1*(sum_e+e_reverse-e-sum_e_max);
   index_v_plus=sum_v+v_plus-v;
   index_v_minus=sum_v+v_minus-v;
   index_v_reverse=sum_v+v_reverse-v;
   r_w=w[index_e_plus][index_v_plus]+w[index_e_minus][index_v_minus]+
w[index_e_reverse][index_v_reverse]+w[index_e][sum_v];
   w_plus=w[index_e_plus][index_v_plus]/r_w;
   w_minus=w[index_e_minus][index_v_minus]/r_w;
   w_reverse=w[index_e_reverse][index_v_reverse]/r_w;

   if(w_plus>R){
    x[k]=x_plus;
    y[k]=y_plus;
    sum_e=sum_e+e_plus-e;
    sum_v=sum_v+v_plus-v;
   }
   else if(w_plus+w_minus>R){
    x[k]=x_minus;
    y[k]=y_minus;
    sum_e=sum_e+e_minus-e;
    sum_v=sum_v+v_minus-v;
   }
   else if(w_plus+w_minus+w_reverse>R){
    x[k]=x_reverse;
```

```
    y[k]=y_reverse;
    sum_e=sum_e+e_reverse-e;
    sum_v=sum_v+v_reverse-v;
   }
  }

  else if((x[k]==x[k-1] && y[k]==y[k+1]) || (x[k]==x[k+1] && y[k]==y[k-1])){
   index_e_flip=-1*(sum_e+e_flip-e-sum_e_max);
   index_v_flip=sum_v+v_flip-v;
   r_w=w[index_e_flip][index_v_flip]+w[index_e][sum_v];
   w_flip=w[index_e_flip][index_v_flip]/r_w;
   if(w_flip>R){
    x[k]=x_flip;
    y[k]=y_flip;
    sum_e=sum_e+e_flip-e;
    sum_v=sum_v+v_flip-v;
   }
  }

  else if(x[k-1]==x[k+1] && y[k-1]==y[k+1]){
   index_e_plus=-1*(sum_e+e_plus-e-sum_e_max);
   index_e_minus=-1*(sum_e+e_minus-e-sum_e_max);
   index_e_reverse=-1*(sum_e+e_reverse-e-sum_e_max);
   index_v_plus=sum_v+v_plus-v;
   index_v_minus=sum_v+v_minus-v;
   index_v_reverse=sum_v+v_reverse-v;
   r_w=w[index_e_plus][index_v_plus]+w[index_e_minus][index_v_minus]+
w[index_e_reverse][index_v_reverse]+w[index_e][sum_v];
   w_plus=w[index_e_plus][index_v_plus]/r_w;
   w_minus=w[index_e_minus][index_v_minus]/r_w;
   w_reverse=w[index_e_reverse][index_v_reverse]/r_w;

   if(w_plus>R){
    x[k]=x_plus;
    y[k]=y_plus;
```

```
   sum_e=sum_e+e_plus-e;
   sum_v=sum_v+v_plus-v;
  }
  else if(w_plus+w_minus>R){
   x[k]=x_minus;
   y[k]=y_minus;
   sum_e=sum_e+e_minus-e;
   sum_v=sum_v+v_minus-v;
  }
  else if(w_plus+w_minus+w_reverse>R){
   x[k]=x_reverse;
   y[k]=y_reverse;
   sum_e=sum_e+e_reverse-e;
   sum_v=sum_v+v_reverse-v;
  }
 }

  /*multicanonical end*/

 if(sum_e<=sum_e_min && sum_v==0){ /*ground state*/
  sum_e_min=sum_e;
  for(j=0;j<_N;j++){
   x_min[j]=x[j];
   y_min[j]=y[j];
  }
 }/*ground state end*/

 h[-1*(sum_e-sum_e_max)][sum_v]=h[-1*(sum_e-sum_e_max)][sum_v]+1;

} /* "s"roop end */

 sum_r_h=0.0;

 for(l=0;l<sum_e_min_min;l++){
   for(j=0;j<v_max;j++){
```

```
    sum_h=sum_h+h[l][j];
  }
}

for(l=0;l<sum_e_min_min;l++){ /*calculate sum_r_h*/
  for(j=0;j<v_max;j++){
   sum_r_h=sum_r_h+w[l][j]/(h[l][j]+1.0);
  }
}/*calculate sum_r_h end*/

for(l=0;l<sum_e_min_min;l++){ /* w[l][j] calculate */
  for(j=0;j<v_max;j++){
   w[l][j]=(w[l][j]/(h[l][j]+1.0))/sum_r_h;
  }
}/* w[l][j] calculate end */

sum_w=0.0;
for(l=0;l<sum_e_min_min;l++){
  for(j=0;j<v_max;j++){
    sum_w=sum_w+w[l][j];
  }
}

/*exe indicate*/
datafile=fopen("shape_biv.dat","w");
for(j=0;j<v_max;j++){
 for(l=0;l<sum_e_min_min;l++){
  fprintf(datafile,"%2d %2d %d\n",-1*l,j,h[l][j]);
 }
 fprintf(datafile,"\n");
}
fclose(datafile);

datafile=fopen("shape_w.dat","w");
for(j=0;j<v_max;j++){
```

```
   for(l=0;l<sum_e_min_min;l++){
    fprintf(datafile,"%2d %2d %f\n",-1*l,j,log(w[l][j]));
   }
  fprintf(datafile,"\n");
  }
  fclose(datafile);

  datafile=fopen("shape_64_mod.dat","w");
  for(j=0;j<_N;j++){
   if(p[j]==0){
    fprintf(datafile,"%d %d\n",x_min[j],y_min[j]);
   }
   if(p[j]==1){
    fprintf(datafile,"%d %d %d %d\n",x_min[j],y_min[j],x_min[j],y_min[j]);
   }
  }
  fclose(datafile);

  printf("minimum sum_e=%2.0d\n",sum_e_min);
  printf("%2f %d\n",sum_w,sum_h);

} /*"m" roop end */

  printf("\n""minimum sum_e=%2.0d\n",sum_e_min);

}
```