計算機アーキテクチャ　109－10
（１９９４．１２．１３）

# 超並列計算機による星型ポリマー成長模型のシミュレーション

志田 和人　　堀口 進
北陸先端科学技術大学院大学
情報科学科

近年、各種の物理現象の電子計算機での重要性が増し、物理学の分野においては、これが理論と実験につぐ第３の分野「計算機物理学」として認知されるに到っている。現実の現象のよりよいシミュレーションのためには膨大な計算量が必要であり、ベクトル型スーパーコンピュータに代わる並列計算機上でのシミュレーションの応用が不可避のものになってきている。

本報告では、溶液中での高分子化合物の形態をあつかう統計物理学での特異なモデルである「星型ポリマー成長模型の配位数計算」について超並列計算機ＣＭ５でのシミュレーションを行った結果について報告する。

# Large Scale Monte Carlo Simulation of Many-Arm Star Polymers on a Massively-parallel Computer

Kazuhito shida　　Susumu Horiguchi
School of information science ,
Japan Advanced Institute of Science and Technology, Hokuriku

Recently, computer simulation makes the third new field of computational physics, following theoretical physics and experimental physics.

Huge computational times , however , required to simulate real physical phenomena on the most advanced supercomputer. To realize a large scale computer simulation , a massively parallel computer becomes very important.

This paper addresses on Monte Carlo simulation of many-arm star polymers on a massively parallel computer.

# 1 はじめに

星型ポリマーといわれる高分子化合物の一種についての統計力学的コンピュータシミュレーションを行なった。

高分子化合物 (ポリマー) は小さな単位分子 (モノマー) が線状に多数結合してできた化合物の総称である。特に星型ポリマーの場合、図1に示すように、中心核から多数の柔軟な分子鎖が分岐した構造となっている。それぞれの分岐は枝と呼ぶことにする。最近になってこの枝の数や重合度 (つまり、枝の長さ) を自由に制御しながらこのような分子を合成できるようになったため、実用と理論の両面から興味をもたれている。

このような高分子を計算機上でモデリングする場合、最も簡単な手法は、それぞれのモノマーを一つの点や線分と見なしてしまうことである。実際のポリマー鎖を構成するモノマーは、連続的な座標と方向を持ち、モノマー間に遠隔的に働く力が存在していることも考えられるが、全てのモノマーが格子上で離散的な座標と方向を持っている、という近似を置く [1]。これによってより現実に近いモデルでは計算量が膨大すぎて不可能な、比較的大きな数のモノマー群をあつかうことができ、かついくつかの重要な物理的な性質はそこなわれない。

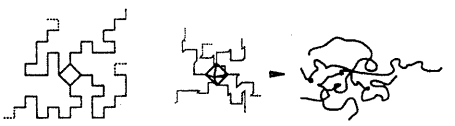


図 1: 格子上の星型ポリマー模型

このようなモデリングによってシミュレーションを行なうことにより得られる基本的な結果は、枝全体のパターンの集合であり、個々のパターンは「配位」と呼ばれる。例えばある一つの配位を図 2に示す。

図中の点線で囲まれた領域が他の星型ポリマーと重複するとモノマー同士が接触する可能性が高いため、溶液中ではこの領域自体が一種の大きな粒子のようにふるまう。現実のポリマー溶液の性質からこのような粒子の平均半径と枝の数、分子量の間の関係を知ることができるが、その関係は格子上のモデルにおいてあらわれるものと同じである。このことから、格子上での離散的あつかいが柔軟な分子鎖の性質をよく表現できることが分かる。

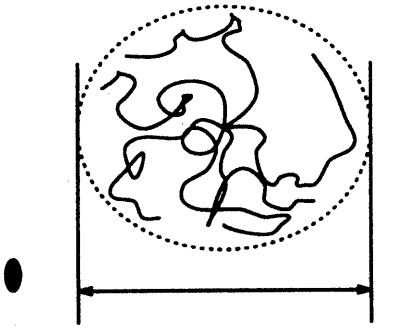

図 2: 星型ポリマーのある配位

より理論的な観点からは、この問題は統計物理学の分野においての興味ある問題の一つとなっている。興味の焦点の一つは「そもそも、配位は全部でいくつくらいあるのか」である。これを全配位数と呼ぶ [2]。

簡単な例として、2 次元空間で、枝が一つしかないとし、かつ、大幅な近似として枝は自分自身と重なっても問題がないとする。この場合は枝が格子一つ分伸びるたびに、必ず上下左右のうち一つの可能性を選択することになるので、明らかに配位数は $4^l$(l は枝の長さ) になる。同様に 3 次元では簡単に $6^l$ である。

しかし、現実の高分子鎖の性質に近づけるために、各格子点についてモノマーの存在数が 1 と 0 しか許されない、つまり、枝が自分自身と重ならないという制限を加えると、その配位は制限がない場合の部分集合になるだけだが、全配位数の問題は急に複雑化する。この制限を「自己回避条件」と呼ぶ。

一例として、2 次元空間で、枝が一つしかないとし、その長さが 12 だという限定条件のもとに、全配位を列挙することを考える。枝が最初は上に向かって出発している全配位の $\frac{1}{4}$ を列挙すれば、他の配位についても回転するだけで得られることが明らかである。この様子は図 3のようになる。

枝が格子一つ分伸びるたびにどの方向に曲ったのかを示す文字を記録し、リストとして表せば、辞書式順序でソートして、どんな配位もこの方法で表現できることが示せる。

ただし、

(右、右、右、.....)

のような簡単な物を除いて、自己回避条件によって禁止される配位を構成的に生成するようなことはできないので、全配位数の問題を簡単に解くことはできそうにない。

星型ポリマーではこれに加えて、自己回避条件は

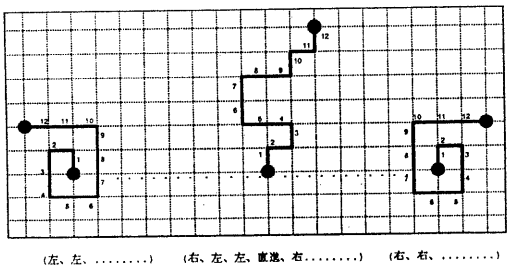


図 3: 線型ポリマー鎖の配位の列挙

どの枝も自分自身と他のどの枝とも重ならないという形に拡張されなければならない。これは非常に困難な問題であるので理論物理的にも多くの研究がなされてきており、それらのアプローチから全配位数 $N_G$ は、枝の長さ $l$ が長くなっていくとき、漸近的に、

$$N_G \sim L^{\gamma_G - 1} \mu^L \qquad (1)$$

のようにふるまうことが知られている。ここに表れる係数についても理論物理的な知見があるが、コンピュータシミュレーションによっても予想を立てることができる。

もしも星型ポリマーについて全配位数、およびその他のデータが得られると、これをさらに複雑な問題について応用することができる。

例えば、「星型」と言っているのは一つの点に全ての辺が集合しているトポロジカルな構造を指しているわけであるが、論文 [3] に示されているように、数理物理学的手法によって多数の鎖と核から成る一般のトポロジーを持つポリマー (ポリマーネットワーク、一例を図 4に示す) の統計力学的特性もまた星型ポリマーのデータを基礎の一つとしてみちびくことができる。

このように、星型ポリマー模型の全配位数を格子モデルの多数の配位を生成することによって計算することは物理学的にも興味深い課題と言える。

## 2 星型ポリマーのシミュレーション

前述のようなシミュレーションを実際に行なうためには、モンテカルロ法の考え方を導入する。すなわち、枝の長さ $l$ の配位を多数用意する。これに対して可能な枝の伸ばしかたの一部について実際に延長を行ない、ここで生じた枝の長さ $l+1$ の配位のうち、どの程度が自己回避条件によって禁止されるかという統計情報を集める。上述のように自己回避条件が無い時の配位数は全く簡単に与えることができるので、禁止されるのがどのくらいの割合いであるかが分れば、自己回避条件のもとでの全配位数を見つもることができるはずである。

もちろん、正確な結果のためには長さ $l$ の配位は多い方が良いし、延長を行なう方法も多い方が良い。効率の良い実行を考えると、上記の操作を $l=1$ から始めて目標の $l$ に到達するまでくり返していくのが良い。

この過程を図 5に示す。

実行のためには、要求される膨大な計算量など種々の問題がある。あつかう全配位について自己回避条件を満足するかどうか確認していかなければならないが、延長する場所に既にモノマーがないことを確認することは計算機上では整数比較演算となり、この回数が全計算複雑度の主要項となる。

ある一ステップでの計算量は、
その時点での既存の枝の長さ、 $l$
枝の総数、 $M$
候補の数、 $P$
その時点であつかっている配位の数 、$S$
を用いれば、

$$lM^2S \qquad (2)$$

で表される。

あつかう配位の数は、延長の初期に発生する多様な配位の内容を反映するために、初期に急激に増加させ、その後もわずかに増加させていく必要があるが、これによる計算量への影響は定数項とみなす。鎖が目標の長さに到達した時点での配位数をシミュレーションで発生させた配位数と定義する。

ただし、「候補」というのは一つの配位についてためしてみる延長の方法の数である。上述のアルゴリ

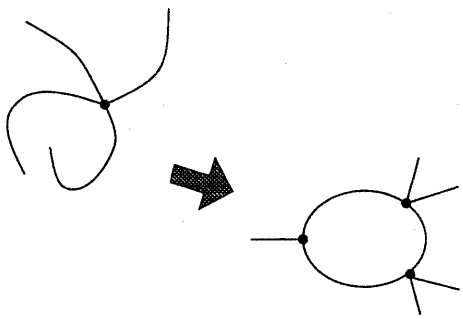

図 4: 一般のトポロジーのポリマーへの展開

ズムを連続して行なっていくには、明らかに自己回避条件を満足する候補が十分たくさんなければならない。このことから枝の数などの初期条件によって適当な候補の数を設定しなければならないが、例えば3次元で枝の数が6の場合、候補は12あれば十分である。つまり、ある配位については、考えうる$5^6$個の枝ののばしかたのうち12個を乱数で選択し、このうち自己回避条件を満足したものから数個を次のステップでの配位としてえらぶ。

ある目標長さまで延長するための合計計算量は、(2)より、

$$\frac{1}{2}L^2MP\overline{S} \qquad (3)$$

となり、枝の数は比較的小さな数なので、枝の長さの二乗のオーダが支配的となる。

このため、従来の類似のモデルをあつかった研究[4]などでは2次元空間でのモデルをあつかっていたが、最近は計算機の性能向上によって3次元での計算が注目されているので、今回は3次元での計算を行なうこととする。最終的な目標として、

- 枝の数が6、12、18、24など。
- 6程度のものが複数重なっているもの。
- 枝の長さは200から300モノマー
- 以上のようなサンプルを十万個のオーダー

を生成することを考えており、これまでの研究に比べてかなり大規模な計算となる。これに必要な演算の数は、枝の数が18、長さ200の場合、従来の40倍、主要部の比較演算だけで、約30テラオペレーションとなる。

かつ、自己回避条件の検証のためにはそれまでに配位につけくわえた全てのモノマーの位置の情報を

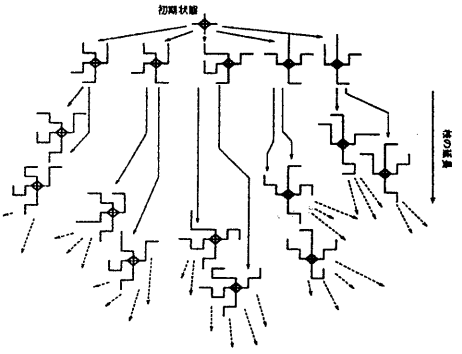


図 5: シミュレーションの進行過程

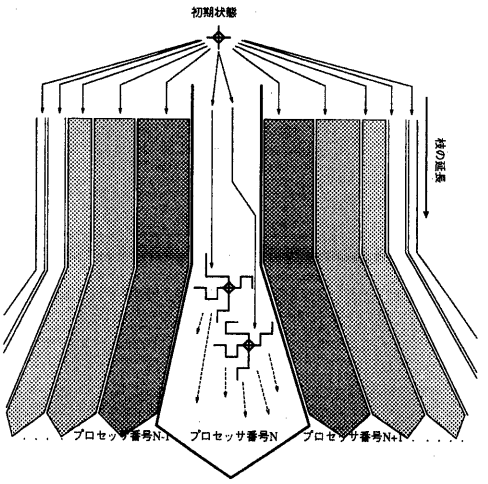


図 6: 並列化されたシミュレーションの進行過程

記憶していく必要があるので、大きなメモリが要求されるという問題もある。

## 3 超並列計算機上でのシミュレーション

この問題には、多数の配位について膨大な計算と大量のメモリが必要である。しかし、それらの処理は基本的に独立に行っても良いという特性もある。この点に注目すると、これらの配位は互いの内部情報には依存しないタスク群とみなしこれらをプロセッサ間に等分配すれば基本的に超並列計算で非常に効率良く解ける問題であると考えられる。

例として、図5の木構造の各枝を分割してプロセッサ群に分配し、それぞれの枝から派生する配位を個々のプロセッサに実行させれば良い。これによって、処理の過程は図6のようになる。

これによって非常に効率の良い並列化が可能であり、計算機の理論性能に極めて近い性能が期待できる。

このようなシミュレーションを実行するため、我々が利用可能であった計算機のなかで最も高速であった64PEのCM5を使用することにした。この計算機はMIMD型疎結合並列計算機であり、理論最大値で、毎秒40ギガ回の整数比較演算が可能であるので、今回の最終目標程度の計算量を手のとどく範囲の時間で処理可能と考えられるが、問題点としては理論最大性能が各PEに装備されたベクトル演算機構の使用を前提としているため、プログラムを並列ベク

トル化しなければならないことが挙げられる。

なお、各ノードの記憶容量は32MByteであり、通信性能は実用的な条件下でどのプロセッサの組み合わせにおいても5から6MByte/secが可能である。もし結合ネットワーク上で最近接の位置にあるなどの有利な条件が得られるなら、通信性能は10MByte/sec程度に達する。

まず、以前に同様の研究[5]に使用されたCM5以外のSIMD並列計算機用のコードを入手した、これはFortran90で書かれており、CM5にもFortran90に近いCM Fortran自動ベクトルおよびパラレル化コンパイラが装備されているため、これを移植すれば予備的なシミュレーションが可能と考えられた。

結果、移植は成功したが、そのパフォーマンスは十分なものでは無かった。これまでの利用経験によると、CM Fortran自動ベクトルパラレル化コンパイラは演算やデータの移動が規則的である場合にはかなり高い実行性能を獲得することができる。この問題においてパフォーマンスが上がらなかったのは、配位を配列にマッピングすることによって自動並列化が行なわれる結果、物理プロッセッサ配置上で好ましくないリダクション演算が発生すること、自己回避条件を満足しなかった配位だけのために消去や回避条件を満足した配位との交換を行なうといった不規則な処理が多いことなどが考えられる。

以上のことから、アルゴリズムから見直すとともに、疎結合MIMD環境上で不規則な処理に対応した高い効率の実行をすることを目標に、自動パラレル化に頼らずCM5上のもう一つのプログラミング環境であるメッセージパッシング方式によって新たなプログラムを作成することとした。

# 4 並列化シミュレーションの問題点

## 4.1 要求メモリ量

基本的な問題として、自己回避条件確認のためのチェックの計算量が膨大なこと、そのためには以前においたものの位置を全て記憶しておく必要があるということがある。特に後者は記憶容量の問題もあって深刻であるため、上記の最終目標の場合を例にして所要メモリ量を見つもってみる。

一つの方法として、モノマーの置ける場所全てに1ビットずつ割り当てて、実際にモノマーが存在する場所のみ真にしておく方法の場合、チェックの際には置きたい場所に対応するビットの中身を一回だけ調

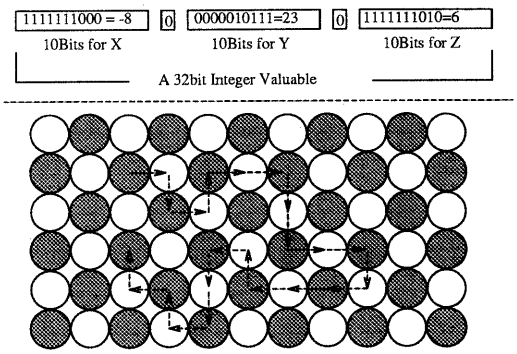


図7: 自己回避判定の高速化

べれば良いので著しく高速だが、このためにメモリが約750Gバイト必要である。

これは現時点では現実的でない大きさであるし、我々が使用するCM5の主記憶は2Gバイトしかない。通常の単精度整数で三次元座標を代入すると、約4Gバイト必要であり、これでも多すぎる。

従来の研究[5]では、要求メモリ量圧縮のため、図形処理で言う方向線素圧縮の三次元版を使用し、単精度整数三つの領域にモノマー40個程度分の情報を圧縮していた。しかし、比較演算のたびに元の単精度整数のフォーマットにもどす手間がかかっていた。

そこで今回、図7の上側のように1ワード32ビットを分割し、3つのフィールドに三軸の座標を2の補数表現で代入する。完全性をたもつため、区切りビットはこのフォーマットのデータを操作する時にだけ、1にする。このようなフォーマットでは、シミュレーション空間が$1024^3$程度の大きさに制限されるが、この程度あれば実用的に十分である。この場合、このフォーマットが空間内で全ての座標に対して一対一の関係を持つので、比較演算の時でも変換の必要が無い。記憶容量とともに比較演算の回数は三分の一になる。といった利点があり、従来の方法よりも優れている。

また、図7の下側はもう一つの効率化手法を示している。簡単だが、従来のものでは行なわれていなかったものである。鎖の端が○の上にあるとき、次は必ず●の上に延長することが分かる、従って、モノマーを二つの記憶域に交互にふり分けて記憶しておけば、比較回数を二分の一にできるのである。一般化して、三方向の座標値の合計から$(x+y+z) mod \alpha$の値で振り分ければこれを$\frac{1}{\alpha}$の回数にすることもできるが効率は$\alpha$が大になるにつれて低下する。

### 4.2 乱数発生

この種のシミュレーションでは高速に発生できる品質の良い乱数が不可欠である。今回は合同乗算法を各プロセッサごとに行ない、かつ各プロセッサ内でも複数の系列を同時に発生させることによってベクトル化を可能にしている。

合同乗算法は長周期秩序の問題があるが、今回のモデルは配位の数が終始変動しながら序々に増加していくという特殊な場合であるため、長周期秩序が表に出るような影響を残すようなことは無いと考えられた。実際に、得られた結果の物理学的なテストによって、そのような影響を受けていないことを確認している。

### 4.3 負荷分散

それぞれの配位に対する処理が基本的に独立にできるため、この処理は多数のＣＰＵを持つ分散記憶型の超並列計算機に最適なのだが、実は完全に独立に処理できるわけではない。

基本的にはシミュレーションのさいにはプロセッサ間通信は得られたデータの集計時以外発生しないが、実際には生成された候補のうち何個が自己回避条件を満たすかは全く確率的なので、シミュレーションの進行にともないそれぞれのプロセッサが所有している配位の数に差が出てきてしまう。

また、全てのプロセッサで発生する配位を対等の確率で選抜しなくてはならず、かつ全配位数は増加していかなければならないので、プロセッサ間に生じた配位の数の差は増幅する傾向がある。

これに対して、配位の自己回避条件を満足する確率は本来確率的なので、たまたま配位数の増減があっても、放置したままシミュレーションを進行すれば逆方向へのゆらぎによって自然消滅するとも考えられる。

いずれにせよ、もし配位の数が大きく不均等になれば処理能力がいちじるしく低下することが考えられる。

## 5 シミュレーション結果の検討

### 5.1 実験結果

上述のような高速化手法をもりこみ、ＣＭ５のベクトル演算機構を最大限に利用することに留意してプログラムを作成した。

この時点では負荷分散については、初期状態において均等に近くしておく以外には何の対策も行なっていなかった。

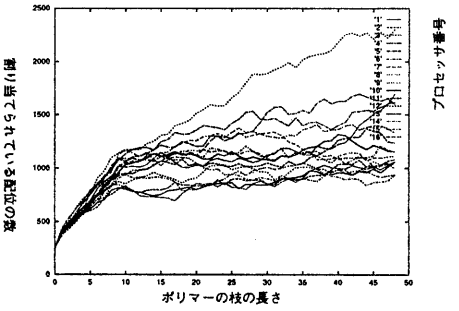


図 8: シミュレーションの進行にともなう各プロセッサでの配位の数の変化

以前に研究されている枝の数が6、長さが100の配位64000個作成するという条件においてこのプログラムを動作させると、その所要時間が4分程度であり、メインフレーム、また従来型スーパーコンピュータ、他機種の並列機などを使用した以前の結果よりはるかに高い性能であった。

また、物理的解析の結果、(1) 式において$\gamma_G = 0.195, \mu = 4.683$ が得られ、従来の研究による結果$\gamma_G = 0.20 \pm 0.05, \mu = 4.6838$ [6][7] と比較してシミュレーションの正確さが確認された。

これらにより、基本的な並列化の方針の正しさを確認することができた。

一方、この実験で、負荷分散は自然には保持できないことがわかった。

各プロセッサの負荷の推移を図8に示す。横軸は鎖の長さを示しており、よってシミュレーションは左から右へと進行している。なお、このグラフではあるプロセッサで偶発的に生じた大きな負荷がその後どうなるかというような個々の挙動に興味があるので縦軸は絶対負荷つまり配位の数を表示しているが、64プロセッサ全部について描くと見にくいのでプロセッサ0から15までに限定してある。ここで見るようにロードバランシングは放置すれば悪化していく傾向があり、最も過負荷なプロセッサが処理時間を決定してしまうことから、理想的なバランスが保持できた場合の50％程度に処理性能が低下しうること、などが確認された。

## 5.2 動的負荷分散

前項での結果から、さらに大規模なシミュレーションでは動的負荷分散機構は不可欠であることが分った。

今回のシミュレーションではある瞬間には全ての配位で鎖の長さもその数も同じなので、全く同様の計算負荷を要するタスクが非常にたくさん存在することに相当する。ただし、通常の意味で言うタスクとは異なり、次のステップに進行する前に全配位についての処理を順に済ましておくしかない。おのおののプロセッサの負荷は処理時間そのものであって、その担当するタスクの数にほぼ比例するよう定義することができ、前述のようにこの負荷が次のステップにどう変化するかは前もって予想することができない。これらから、動的負荷分散機構について、

> 延長と自己回避条件の確認が終了し、次ステップでの負荷の様子が判明した後に配位の一部を移動させてその数がプロセッサ間でなるべく均等になるようにする。
>
> 負荷分散がどれだけ成功しているかは、どれだけ負荷が均等であるかとともに、それにどれだけの時間を要したかによって評価される。

という基本方針が立てられる。また、並列処理において常に問題になる通信時間は、この負荷分散機構の実行に要した時間として評価することとする。なぜなら物理的結果の通信や同期などの作業は実際には計算に対して無視できるほど短い時間しか必要でないため、負荷分散機構が唯一マクロな量の通信時間を消費するからである。

どのようにタスクを移動させるのか決定する機構についても、負荷の均等さを偏重するあまり複雑なアルゴリズムを使用するのは好ましくない。なぜなら、64個のタスク群の内容を入れ換えて各タスク群の大きさが完全に均等になるようにするには、一度ある一ヶ所(例えばCM5のフロントエンド)に収集してから均等に分配しなおすか、一種のナップザック問題を解かなければならない。分配による方法はギガバイト級のデータが一ヶ所を通過するため通信時間の点から論外であり、ナップザック問題は非常に複雑な計算を必要とする。また、図8に示すように、非常に負荷が高くなるのは一部のプロセッサのみであり、他のプロセッサでは実際にランダムな負荷変動に近い状態になっているので、わざわざ複雑な方法を利用して完全な均等分散を実現しても意味がない。

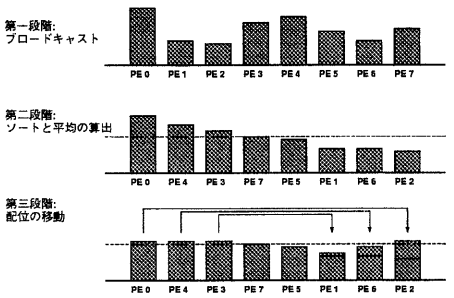


図 9: 自動負荷分散機構の動作

これらを考慮すると、負荷分散に数%の幅を許し、この幅を逸脱すると均等な分散に向かって強力に引きもどされるが、幅の内側ではなにもしないで自然にもどるのを待つような方式が理想的である。

ここで、CM5の通信ネットワークが、比較的全プロセッサ対全プロセッサの通信に対し高性能であることを利用し、図9に示すような単純なアルゴリズムを考案した。

> 第一段階:全プロセッサ対全プロセッサのブロードキャスト通信を行ない、各プロセッサで他の全てのプロセッサが持っている配位の数がわかるようにする。
>
> 第二段階:各プロセッサでこれをソートし、各プロセッサで他の全てのプロセッサが負荷について第何位にあるかわかるようにする。同時に理想的な分配の目標として、各プロセッサの配位数の平均を求める。
>
> 第三段階:負荷が高い側のプロセッサからメッセージ通信によって負荷の低いプロセッサに配位を移動し、その配位数を平均値から上に数%の地点まで引きもどす。余分になった配位を引きうけてもらう相手は、負荷についての順位によって、例えば64プロセッサの場合、
>
> $$\begin{aligned} \text{第1位} &\Longrightarrow \text{第64位} \\ \text{第2位} &\Longrightarrow \text{第63位} \\ &\vdots \end{aligned}$$
>
> のように決定する。

このアルゴリズムは単純なのでこれ自体はほとんど計算時間を消費せず、一回のブロードキャストの後

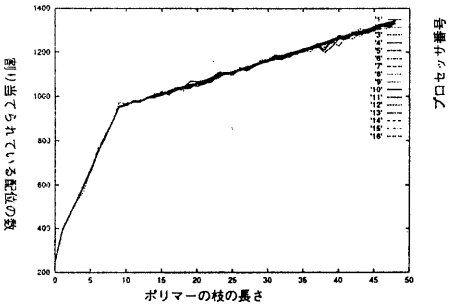


図 10: 自動負荷分散機構によって均等化された負荷の推移

はメッセージパッシングも一度に一つの相手に対してしか行なわない。メッセージパッシングの回数が小さいことは、CM5ではメッセージのセットアップにかかる時間が無視できないので[8]、これは重要な事である。

問題としては、明らかにこれだけでは完全に均等な負荷分散は実現できないということがある。例えば、1プロセッサのみが他に対して非常に高い突出した負荷を持つ場合、他のプロセッサが均等だとすると、完全均等分散に近い点にまでもどるのに非常に多いステップ数が必要である。なぜなら、突出した負荷の大部分がそのまま他のある一つのプロセッサに移動するため、処理時間を決定するプロセッサの最大負荷は一度にわずかしか減少しないのである。

実際にこの方式を前述のプログラムに対して適用してみると、同様の条件において負荷は図10のように高度に均等化された。負荷が高い側は、平均値から上側に10%の地点まで引きもどすようにしている。さらに上で定義した全通信時間も、全処理時間の数%におさまっている。

## 6 おわりに

現在は基本的な初期条件に対して計算結果が物理学的に不自然でないものであることが分ったため、初期条件を今までに研究されたことのない物理的条件に変更してシミュレーションを行ない、えられた結果を解析中である。

本研究によって、多数の配位を並行に発生させて追跡していくような、膨大な計算量をともなう統計物理学のシミュレーションが疎結合ベクトル型の超並列計算機によって効率良く処理できることが明らかとなった。

これからの課題として非常に多数の高分子鎖が存在する環境を想定した計算を行なっているが、このような場合、さらに計算量が増加し、かつ負荷分散が困難になることがわかっている。これらを解決するために、本論文で提案した動的負荷分散を複数回行なうなどの検討を行なっている。

## 参考文献

[1] Pierre-Gilles de Gennes ;" 高分子の物理学 -スケーリングを中心として- ," 吉岡書店 (1984)

[2] M.K.Wilkinson,D.S.Grant,JEG Lipson, and SG Whittington ;"Lattice models of btranched polymers: statistics of uniform stars," J.Phys, A:Math.Gen. pp L469-L473 18 (1985)

[3] Kaoru ohno and Kurt Binder ; " Scaling theory of star polymers and general polymer networks in bulk and semi-infinite good solvents," , J Phys. France. 49 pp 1329-1351 (1988)

[4] Kaoru ohno and Kurt Binder ; " Monte Carlo Simulation of Many-Arm Star Polymers in Two-Dimensional Good Solvents in the Bulk and at a Surface," Journal of statistical physics. Vol.64 Nos.3/4 pp 781-806 (1991)

[5] Kaoru Ohno,Xiao Hu and Yoshiyuki Kawazoe ; " Monte Carlo Simulation of Many-Arm Star Polymers in Three-Dimensional Good Solvents. Proceeding of the Second International Conference and Exhibition on Computer Application to Material and Molecular Science and Engineering," pp 315-318 (1993)

[6] Wilkinson et al ; J. Phys. A19 pp 797 (1986)

[7] Batoulis and Kremer ; Macromolecules 22 pp 427 (1988)

[8] Mengjou Lin, Rose Tsang, David H.C.Du, Alan E. Klietz, Stephen Saroff ; "Performance Evaluation of the CM-5 Interconnection Network" CompCon Spring (1993) .