ＩＳＳＮ 1341—6839

情報処理センター
研　究　報　告

The Bulletin of the Information Processing Center

第　19　号

（１９９８．３）

岡　山　理　科　大　学

Okayama University of Science

岡山市理大町１－１
Tel（086）252—3012（直）

目　　次

JAVAによる遺伝的アルゴリズムの研究

情報処理センター　岩崎彰典

遺伝的アルゴリズムは、組合わせ問題の近似解をヒューリスティクに探索するアルゴリズムのなかでは比較的有効な方法の一つである。しかしながら、問題によってはパラメータを適切に決定する必要がある。そこで、本研究ではJAVA言語を用いて遺伝的アルゴリズムを構成し、結果を視覚化することでパラメータの決定を容易にすることを試みた。

1　はじめに

遺伝的アルゴリズムは、様々な組合わせ問題において比較的良好な解を短時間に見つけるがことできることで知られている。しかしながら、実用的な問題においては、単純な遺伝的アルゴリズムだけで解を見つけることは一般に困難であり、問題の特性を生かしたパラメータの決定が必要となる。本研究では、遺伝的アルゴリズムを適用する組合わせ問題として0-1ナップザック問題をとりあげる。0-1ナップザック問題は、問題の単純さにもかかわらず、組合わせ問題の本質的難しさを備えており、また、探索空間上の解の分布を視覚化しやすい特徴を持つ。JAVA言語によって遺伝的アルゴリズムのプログラムを作成し、解が改善される様子をアニメーションによって示す。

2　0-1ナップザック問題

0-1ナップザック問題は次のように定式化される：

$$\begin{aligned} &\text{maximize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i x_i \\ &\text{subject to} \quad g(\boldsymbol{x}) = \sum_{i=1}^{N} g_i x_i \leq b \\ &x_i = 0 \text{ or } 1, \quad i = 1, 2, \cdots, N \end{aligned} \tag{1}$$

ここで、係数f_i , g_iは、その目的関数,制約関数、bは制約許容量である。

3　遺伝的アルゴリズム

遺伝的アルゴリズム (Genetic Algorithm :GA) は、生物進化の原理に着想を得たアルゴリズムであり、確率的探索、学習、最適化の一手法である。このアルゴリズムとは生物の進化過程をシミュレーションすることによって、数多くの組み合わせの中から最適な解を見つけ出す。ダーウィンの自然淘汰理論を基礎に、生物の進化過程を工学的モデルに応用したものである。それぞれの組み合わせを遺伝子からなる染色体を解に対応させて、その組み合わせごとの目的関数を計算する。そして、その値の高いものを選択し、それを残すように何世代にもわたって行わせれば最終的に、高い目的関数値をもつ解を得ることができる。
まず、0と1の値を持つ遺伝子からなる染色体と考えこの染色体の集まりを集団と言う。これらの集団に対して、交叉、突然変異、選択の操作を施す。

3.1　交叉

交叉 (crossover) は、二つの親の染色体を組み替えて子の染色体を作る操作である。

1. **単純交叉 (simple crossover)**
   最も単純な方法では、交叉する位置を一つ決めてその前と後で、どちらかの親の遺伝子型を受け継ぐかを変える方法である。これを単純交叉, または一点交叉 (one-point crossover) と呼ぶ。図3.1では、4番目と5番目の遺伝子座の間に交叉位置があり、個体Bの5番目から最後までの遺伝子が、新しい個体の遺伝子となる。また、その逆が、他の新しい個体の遺伝子となる。

個体A　1001 | 111 → 1001000
個体B　0011 | 000 → 0011111

＜図3.1＞　1点交叉の例

2. **複数点交叉 (multipoint crossover)**
   複数点交叉は、交叉位置が複数ある方法である。例えば交叉位置が2と5なら、新たな個体の一つは個体Aの先頭から2番目まで個体Bの3番目から5番目まで個体Aの6番目から最後までによって、遺伝子が作られる。同時に、その逆の組み合わせで、もう一つの新たな個体の遺伝子が作られる。(図3-2)

   本実験では、ランダムに選んだ20個の解からランダムに2つを選びさらにランダムに選んだ2ヵ所で切り、中央を入れ替える2点交叉を使用した。

個体A 10|011|11 → 1011011
個体B 00|110|00 → 0001100

＜図3.2＞　複数点交叉の例

3.2　突然変異

突然変異(mutation)は、各染色体において、突然変異(mutation)は、各染色体において、ある確率で、遺伝子の一部が0から1、あるいは1から0に反転させる操作である。大きな変異確率に設定するとランダム探索と同じようなことになってしまうが、ある程度の変異は必要である。突然変異が無い場合は、初期の遺伝子の組合せ以外の空間を探索することはできず、従って、求められる解の質にも限界が出てくる。図3.3では、文字列の1ヵ所をランダムに選択し反転させる1点突然変異を示した。

変異前 1001|0|11 → 変異後 1001|1|00

＜図3.3＞ 突然変異の例

3.3　選択

交叉、突然変異を経た遺伝子を持った染色体のうち、環境へのより高い適応性を持つものを選択する。本研究では、エリート戦略(ここでは集団中で最も高い目的関数値をそのまま次世代に残す）を用いた。

4　計算機実験

JAVA言語を用い、ランダムな値をもつ50変数の0-1ナップザック問題を作成し、遺伝的アルゴリズムを適用した。実験では２０個の解集団を生成し、遺伝的アルゴリズムによってその解集団が最適化される様子を調べた。

実験で用いた遺伝的アルゴリズムのパラメータを次に示す。

個体数：20

交叉方法：2点交叉

突然変異率：mut= $\frac{\text{突然変異数}}{\text{全遺伝子数}} \times 100 \quad (\%)$

$$\text{適応度関数} = \begin{cases} f(x) & g(x) \leq b \\ f(x) - \lambda(g(x) - b) & g(x) > b \end{cases}$$

λ：ペナルティ乗数

選択方法：エリート戦略

5 結果

図5.1はランダムに生成した20個の解の分布を示す。制約条件がないとき、もしくは制約関数の総和が制約許容量と等しいとき、最適解は全ての変数が1の明らかな解をもつ。しかし、単なるランダム探索では全ての変数が1となる確率はほとんど0であり、解の品質もほとんど改善されない。

図5.2は突然変異率を0％、図5.3は突然変異率を5％として、制約条件がない問題に遺伝的アルゴリズムを適用し、解の改善していく軌跡を表している。突然変異率が0％では、解は改善されてはいるが、最適解付近までは到達していない。しかし、突然変異率を5％とすると、解ほとんど最適解付近まで到達している。

図5.4は制約をもつ問題へ適用した結果である。ペナルティ乗数を1としたとき、比較的良好に解が改善されている。

図5.5はペナルティ乗数を1とし、制約許容量の非常に少ない問題へ適用した結果である。実行可能解が得られていないことがわかる。

図5.6はペナルティ乗数を１０とし、制約許容量の非常に少ない問題へ適用した結果である。解ほとんど最適解付近まで到達している。

このように、遺伝的アルゴリズムは問題によってパラメータを適切に定める必要があり、解の挙動の可視化はパラメータの決定を容易にすることがわかる。

6 むすび

本研究では、JAVA言語を用いた遺伝的アルゴリズムを0-1ナップザック問題に適用し、結果を視覚化した。遺伝的アルゴリズムはパラメータの適切な決定が重要であり、解の改善されていく様子の可視化はそれを容易にすることがわかった。

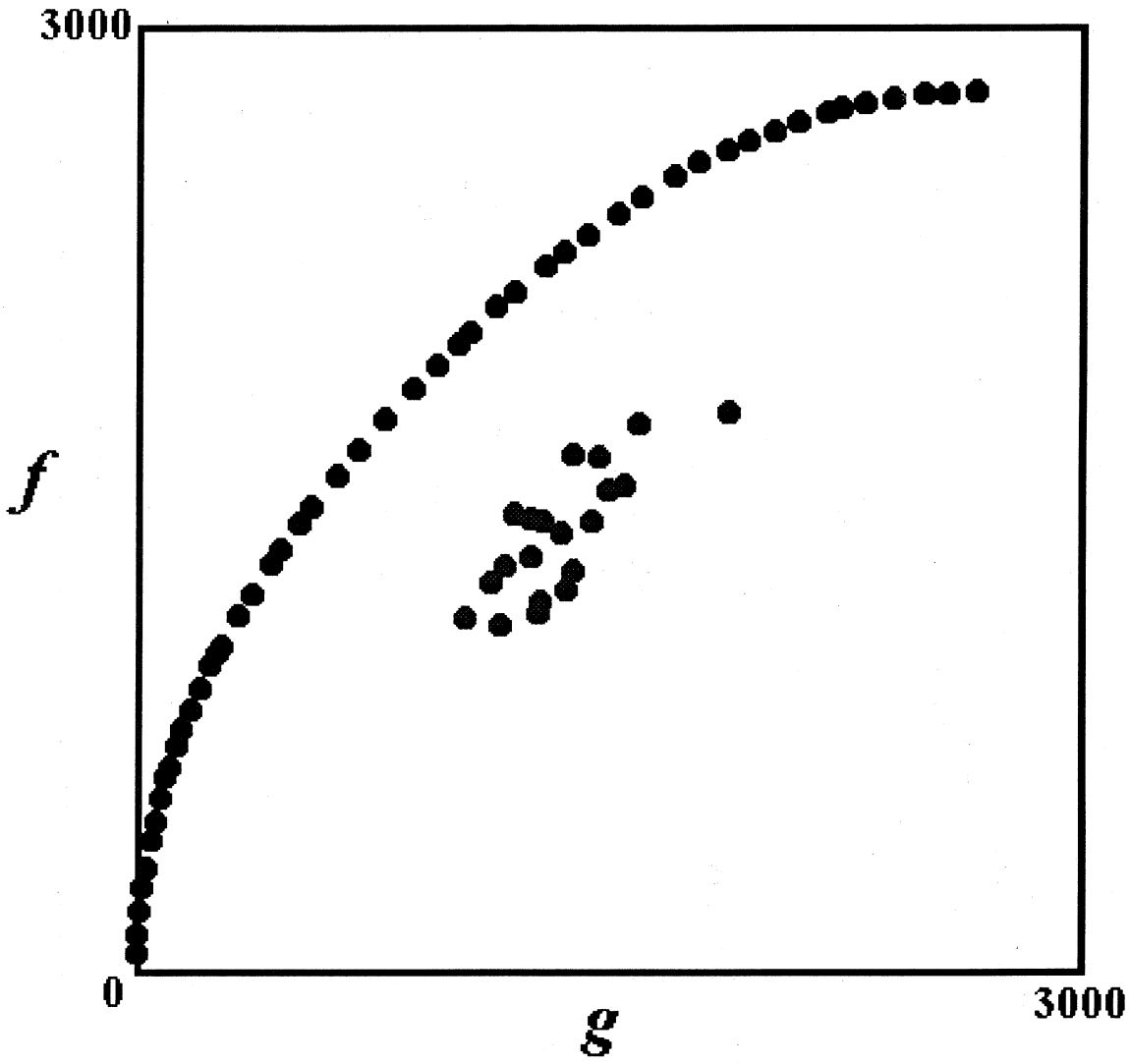


図-5.1 ランダムな20個の解

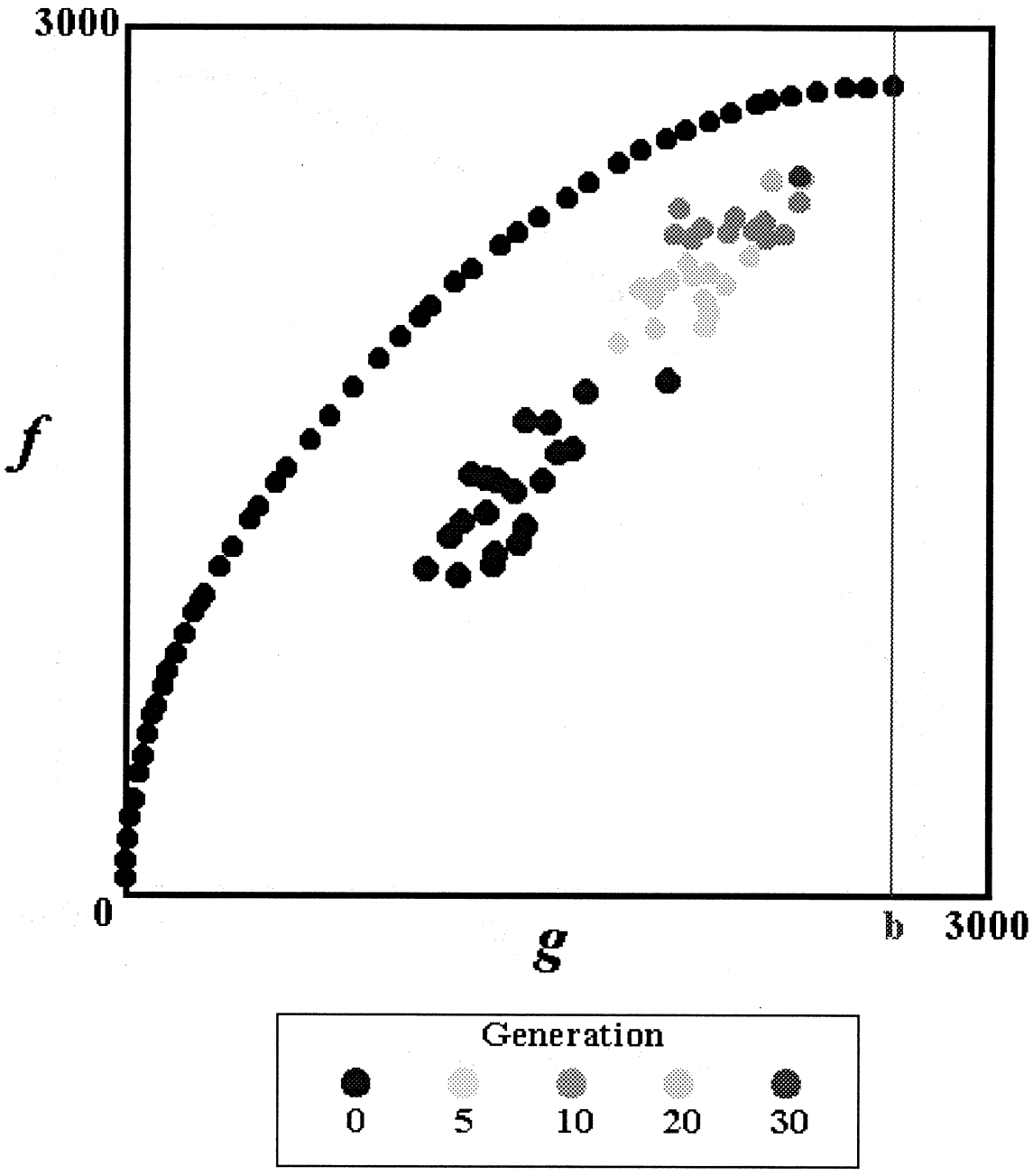


図-5.2 b=2666　mut=0.0(%)　λ=1.0

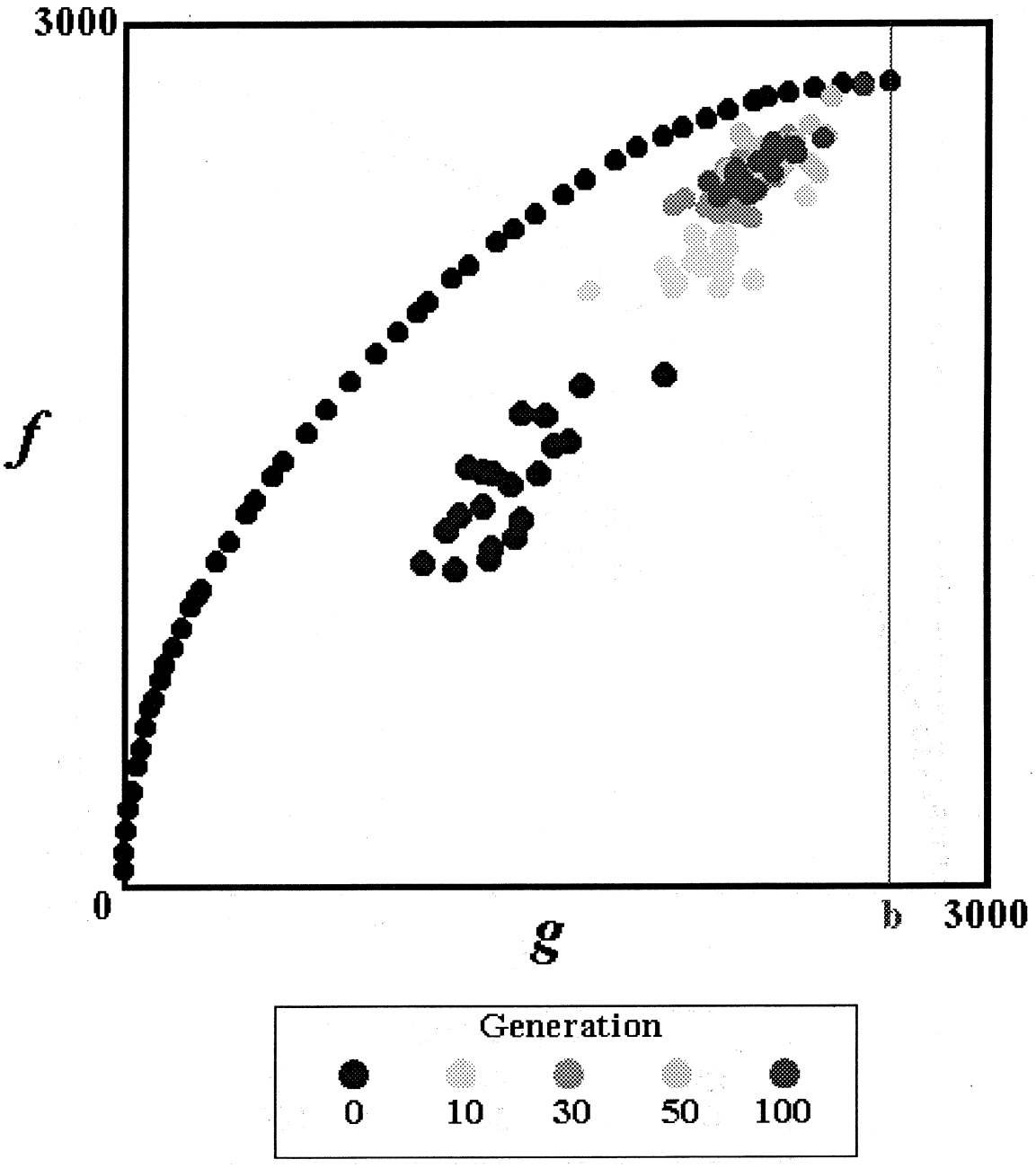


図-5.3 b=2666　mut=5.0(%)　λ=1.0

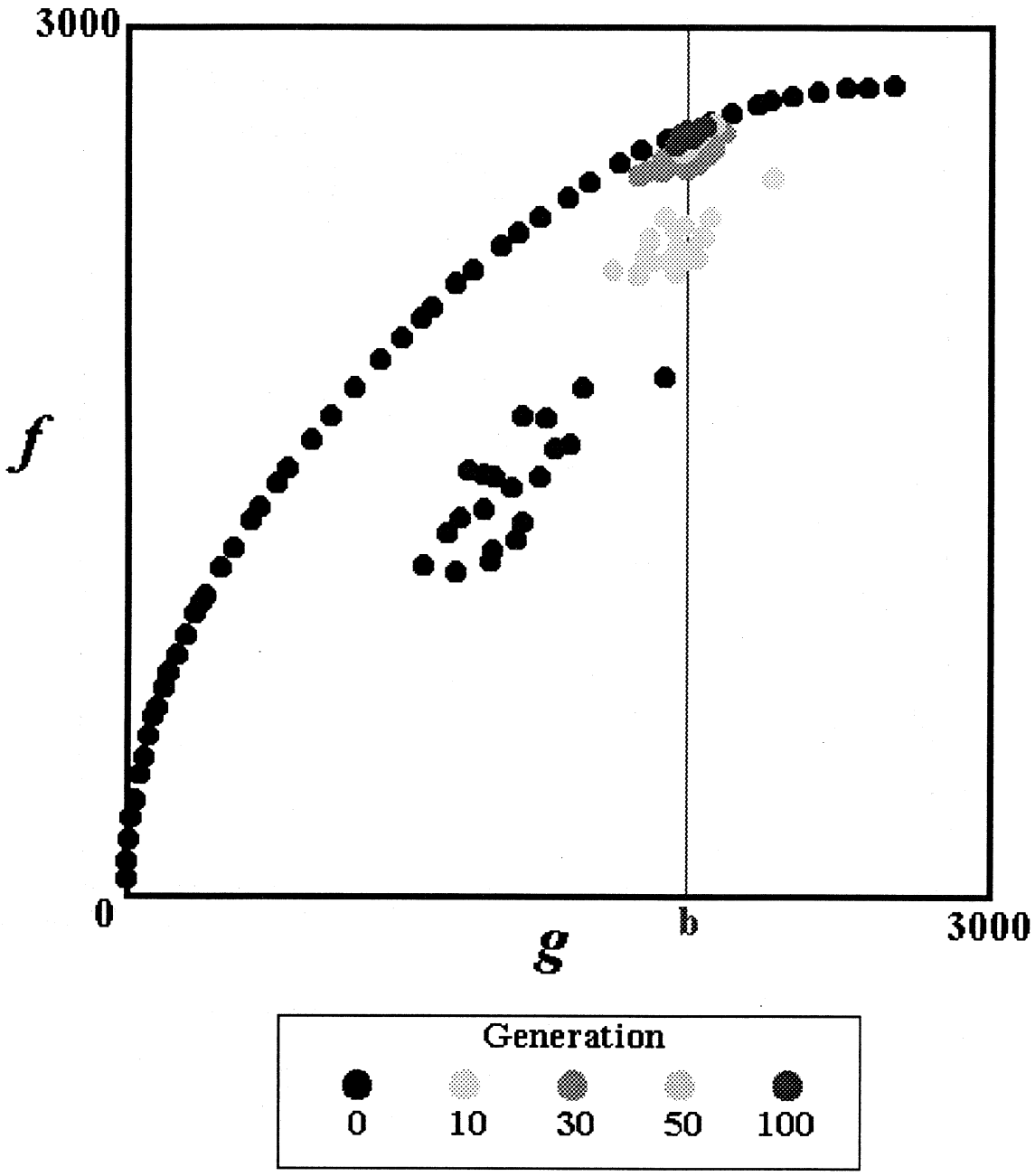


図-5.4 b=1955　mut=1.0(%)　λ=1.0

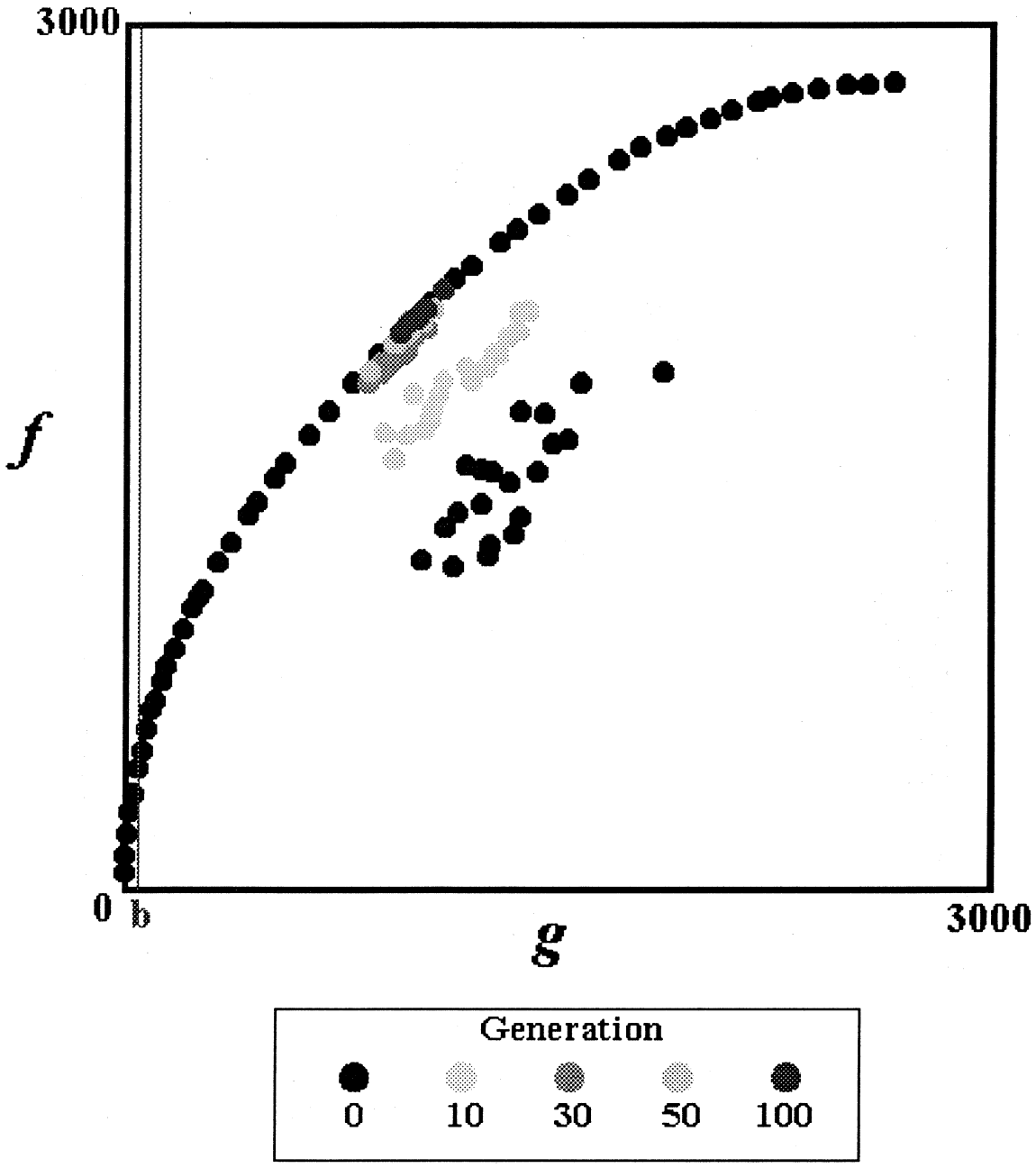


図5.5 b=50　mut=1.0(%)　λ=1.0

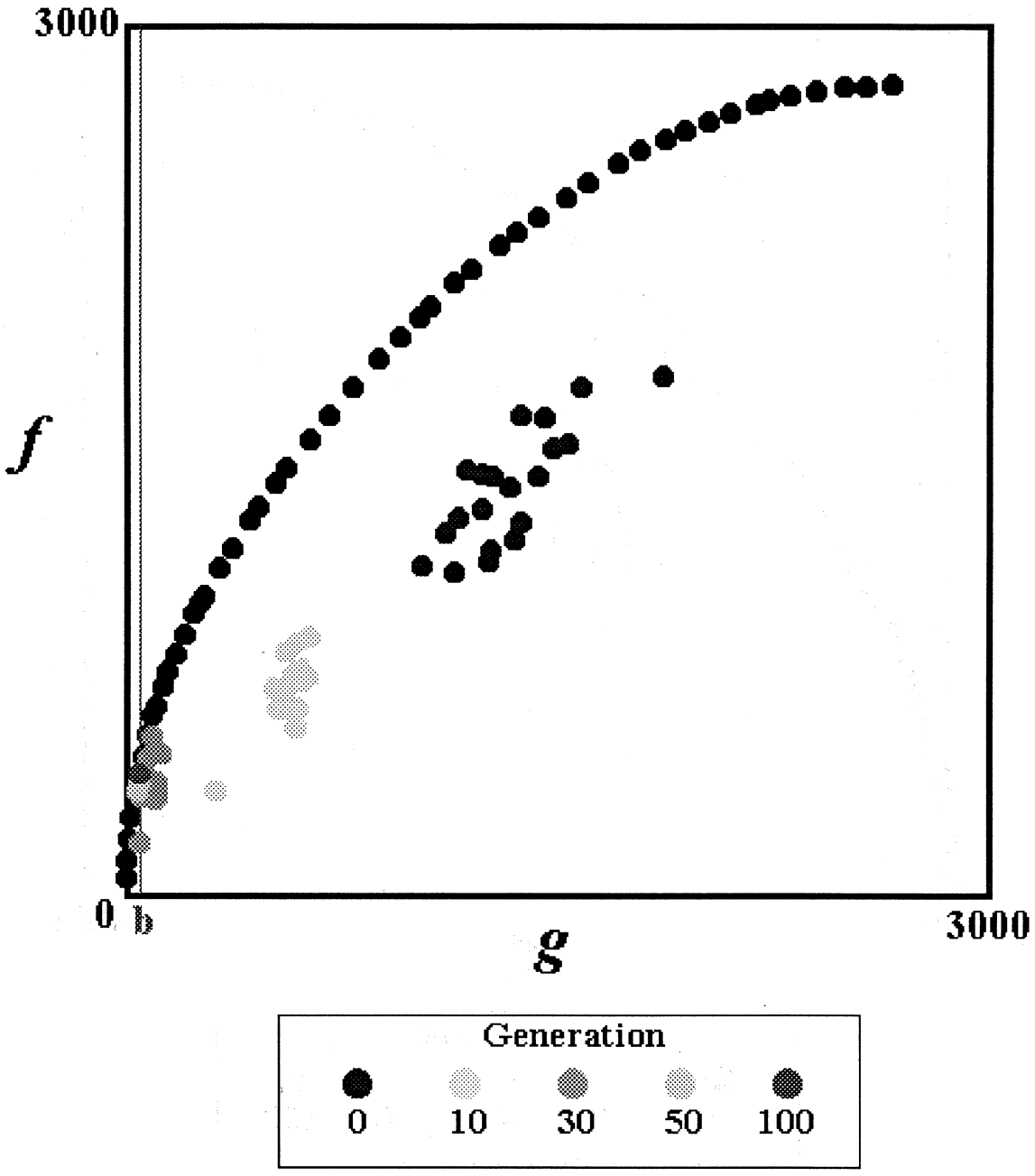


図-5.6 b=50　mut=1.0(%)　λ=10

硫黄架橋モリブデン
クラスター錯体の電子状態

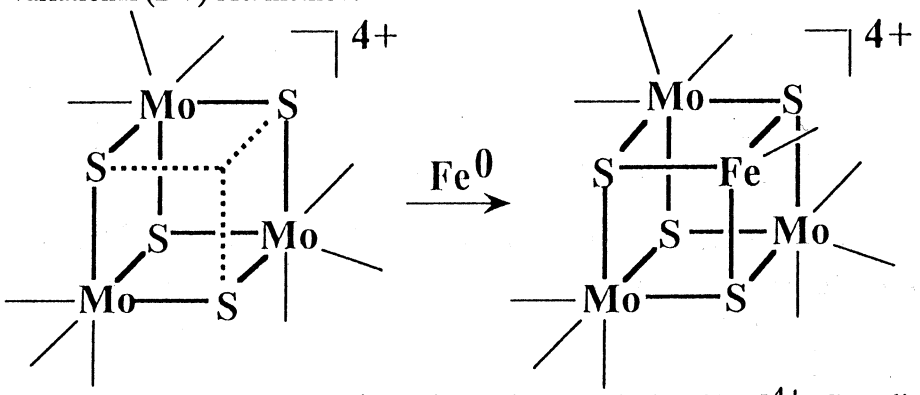
理学部　化学科　柴原隆志

Electronic Structures of Sulfur-Bridged Cubane-Type Mixed-Metal Clusters with Mo_3MS_4 (M=Fe, Co, Ni, Cu, Ga, In, Sn, Sb) Cores

Department of Chemistry, Okayama University of Science
Takashi SHIBAHARA

The incomplete cubane-type sulfur-bridged molybdenum aqua cluster $[Mo_3S_4(H_2O)_9]^{4+}$ reacts with iron metal to give the molybdenum-iron mixed-metal cluster $[Mo_3FeS_4(H_2O)_{10}]^{4+}$(Scheme 1), which caused much research on metal incorporation reaction of this type, where the missing corner of the incomplete cubane-type core is filled with another metal atom to give mixed-metal clusters with Mo_3MS_4 cores (M= Fe, Co, Ni, Cu, Ga, In, Sn, Sb, Hg, ...). There exist three kinds of cubane-type cores as shown in Figure 1.

Electronic structures of Mo_3MS_4 cores (M= Fe, Co, Ni, Cu, Ga, In, Sn, Sb) have been calculated by the discrete-variational (DV)-Xα method.



Scheme 1. Formation of molybdenum-iron cluster $[Mo_3FeS_4(H_2O)_{10}]^{4+}$. Coordinated H_2O's are omitted for clarity.

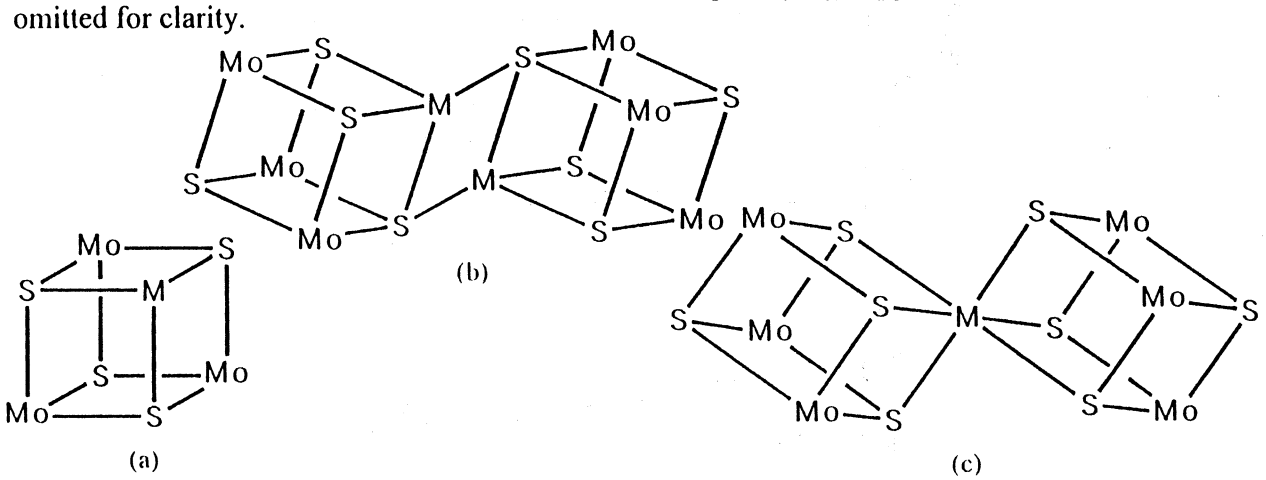


Figure 1. Three kinds of sulfur-bridged cubane-type mixed-metal cores: (a) Single cubane-type; (b) Double cubane-type; (c) Sandwich cubane-type.

【序論】硫黄架橋不完全キュバン型モリブデンアクア錯体$[Mo_3S_4(H_2O)_9]^{4+}$は酸性溶液中で金属 M(M=鉄, コバルト, ニッケル, 銅, ガリウム, インジウム, アンチモン, 水銀など)と反応して混合金属キュバン型Mo_3MS_4骨格をもつ錯体になる。今回は$Mo_3MS_4^{4+}$骨格(M=Cr, Mn, Fe, Co, Ni, Cu, Zn, Ga, Cd, In, Sn, Sb)の電子状態を計算し、比較検討した。

【計算】$Mo_3MS_4^{4+}$の座標は$[Mo_3FeS_4(H_2O)_9](CH_3C_6H_4SO_3)_4\cdot 7H_2O$のX線構造より$C_{3v}$を仮定して求めた。サンプル点の数は10000、使用した原子軌道はMo, 1s～5p; M(Cr, Mn, Fe, Co, Ni, Cu, Zn, Ga), 1s～4p; M(Cd, In, Sn, Sb), 1s～5p; S, 1s～3d である。Self consistent な計算において、計算前後の Mulliken Population Analysis による電荷の差が 0.003 電子以下になるまで計算を繰り返した。HOMO 近傍のエネルギー準位図を Figure 2(M=Cr, Mn, Fe, Co, Ni, Cu, Zn, Ga), Figure 3(M=Cd, In, Sn, Sb)にそれぞれ示す。

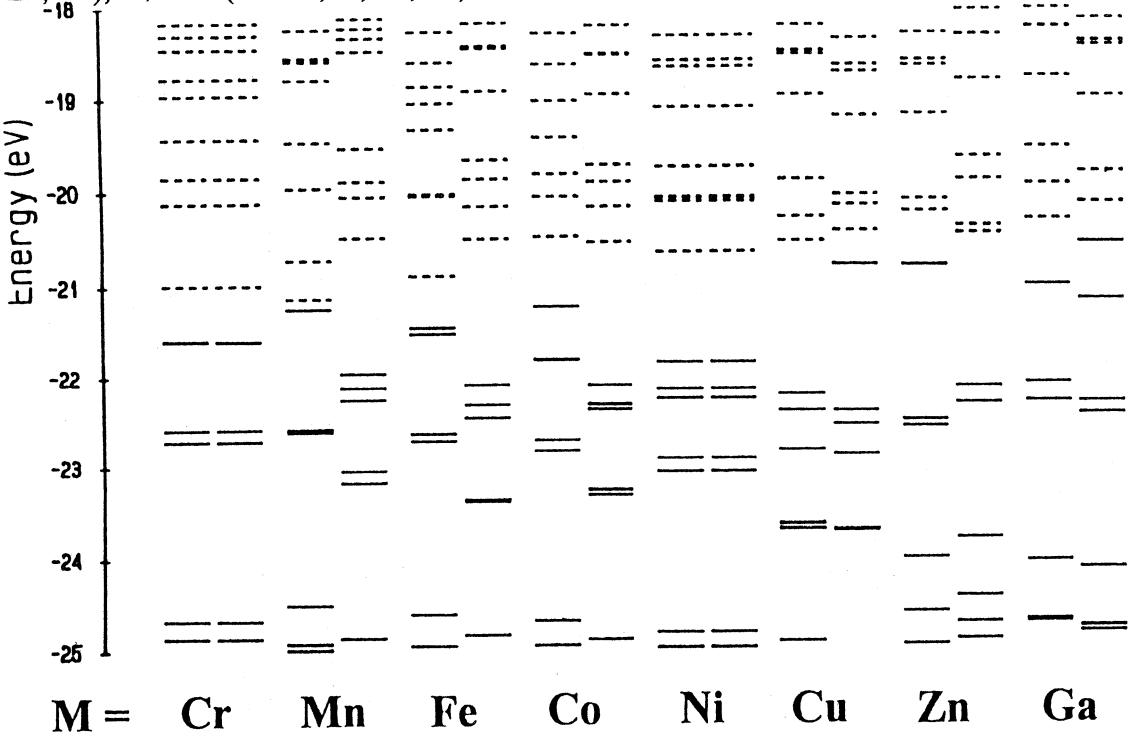


Figure 2. Energy levels for $Mo_3MS_4^{4+}$(M = Cr, Mn, Fe, Co, Ni, Cu, Zn, Ga)

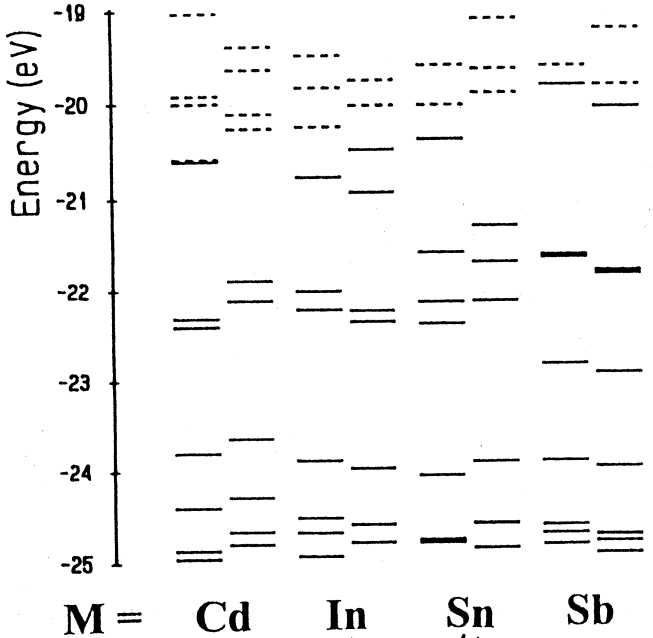


Figure 3. Energy levels for $Mo_3MS_4^{4+}$(M = Cd, In, Sn, Sb)

ナノ細孔内における固液相転移

化学科 森重 國光

概要

モデル多孔体MCM-41と複雑な連結細孔構造をもつVycor glass内における水の凝固・融解（固液相転移）挙動をX線回折により調べた。得られたデータに非線形最小二乗法によるローレンツ曲線の当てはめを行い、正確なピーク位置・幅・高さを求め、ナノ細孔内での水の液固相転移を考察した。水から立方晶氷への相転移は、細孔直径29Åより大きな細孔内では不連続的に生じ、これより小さな細孔内では連続的に生じる。シリンダー状の理想的な細孔をもつMCM-41内における水の凝固点と融解点はほとんど一致するが、Vycor glass内ではそれらの温度間に大きな差が生じてヒステリシスが見られる。

はじめに

シリカゲルを代表とする無機化合物の多孔体内における水の凝固・融解挙動は核磁気共鳴（NMR）分光法[1]、熱量測定法[2]、あるいは中性子回折法[3]などによって幅広く調べられている。これらの研究から、細孔内には自由水と束縛水の2種類の水があり、それらの凝固・融解温度はバルク水に比べて常に低いことが明らかになっている。また、束縛水の凝固・融解は細孔径にほとんど依存せずに連続的にゆっくりと生ずるが、自由水の凝固・融解温度は細孔径が小さくなるほど低下することが知られている。また、自由水の凝固・融解にしばしば大きなヒステリシスが観測されることも知られている。しかしながら、細孔内の水に関する理解がこのように進んでいるにもかかわらず、いまだに多くの問題が未解明のまま残っている。例えば、細孔が次第に小さくなっていったとき、自由水の液固相転移はバルク水のようにいつまでも不連続的（1次の相転移）であるのか。束縛水の凍結によって生じる氷はどのような構造をとるのか。また、自由水の凝固・融解に際してみられるヒステリシスは何故生じるのか、などである。

解析

回折データの解析は情報処理センターのワークステーション IBM59H を使って行った。ローレンツ曲線の当てはめは粟屋の非線形最小二乗法プログラム[4]を用いて行った。

結果と考察

細孔直径（D）が35ÅのMCM-41内の水の冷却時のX線回折パターン変化

を図１に示す。温度が低下すると、225K 付近で回折パターンが急激に変化して、２θ＝11,18,21°の付近に立方晶構造の氷の形成を示す比較的シャープな回折ピークが生じた。温度をさらに下げていっても、回折パターンにあまり大きな変化は見られなかった。NMR 測定[5]から、束縛水の凍結が 220 から 180 Kにかけてゆっくりと生じることが報告されているが、それに対応した大きな変化は見られない。自由水の凍結による構造変化のみが観測されている。図２は、ローレンツ曲線の当てはめによって得られた主ピークの位置と幅を温度に対してプロットしたものである。ピーク幅は225K付近で急激に変化しており、細孔内の水の液固相転移がこの温度で不連続的に生じていることがわかる。ピーク位置も同時に変化しており、密度の不連続的な変化を伴っている。凝固と融解温度はよく一致しており、ヒステリシスの存在しないことが示される。このような測定を細孔径の異なる他の MCM-41 試料についても行った。水の主ピークの幅の温度変化をまとめると、図３のようになる。この図は、いずれの細孔に対しても凝固・融解挙動にヒステリシスがほとんど見られなかったので、冷却側のデータのみを示している。D＝29Åの細孔を境にして、これより大きな細孔内では水の液固相転移は不連続的に生じ、小さな細孔内での水の凝固はゆっくりと連続的に生じることがわかる。したがって、29Åという値は細孔内の自由水の液固相転移がバルク水のように不連続的に生じるのか、あるいは連続的にゆっくりと生じるかを分ける臨界細孔径を示している。細孔径が小さくなると、水の凝固点は次第に低下しており、そして 180Kにおけるピークの幅は増大している。細孔径が小さくなると、凍結によって生じた立方晶氷の結晶子サイズも小さくなることを示している。図４に見られるように、D=29Å以上の細孔内での自由水の液固相転移にともなってピーク位置の急激な変化、すなわち密度の不連続的な減少が見られる。ところが、それ以下の細孔に対しては、細孔内の水の凝固にともなうピーク位置の変化はなだらかであり、水から立方晶氷への転換が連続的に進行していることがわかる。臨界細孔径以下の細孔内の水は凍結する前にすでにほぼ立方晶氷と非常によく似た構造をとるようになっていることが推定される。180Kでのピーク位置は、細孔径が小さくなると高角側に少し移動しており、氷の格子定数の減少、すなわち密度の増大が生じている。

Vycor glass 内に 0.9 の充填率で入っている水の冷却時のX線回折パターン変化を図５に示す。冷却していくと、226K付近から急激に変化し始め、立方晶氷に固有な回折パターンを生じた。ところが、一度低温まで冷却した試料を昇温すると、凝固温度を過ぎても細孔内の氷は融解せず、250K以上でようやく融解し始めた。ピーク位置と幅の温度変化を図６に示す。MCM-41 とは違って、大きなヒステリシスが見られる。Vycor glass の細孔径はかなり分布をもっ

ているが、平均細孔直径は 70Å付近であると思われる。電子顕微鏡観察[6]から、Vycor glass の細孔は途中で膨らんだり縮まったり、また互いにつながったりした連結細孔構造であることが分かっている。これとは別に、結晶核形成のエネルギー障壁はサイズとともに増大し、したがって、ある大きさ以上の微粒子の凝固・融解にヒステリシスが生じてもよいことが理論的に報告されている[7]。Vycor glass 内の水の凝固・融解挙動に見られる大きなヒステリシスがサイズ効果なのか、あるいは複雑な細孔構造の効果なのか今のところ判断できない。

回折ピーク位置から六方晶氷の格子定数を、またピーク幅からシェラーの式を使って結晶子サイズを求めることができる。MCM-41 と Vycor glass 内の水について、凝固・融解温度、格子定数、結晶子サイズを表1にまとめて示す。ここで、f は細孔内での水の充填率を示す。MCM-41 の場合は、冷却により膨張して細孔外で通常の六方晶氷が形成されるのを防ぐため、f=0.9 付近で測定を行った。MCM-41 試料内の氷の密度は、格子定数の規則的な変化から示されるように、細孔径が小さくなるとともに増大することがわかる。立方晶氷のサイズが小さくなるほど、構造欠陥が増して、密度が増すものと思われる。正四面体の頂点方向に形成される水分子のネットワークの秩序が小さな細孔内の方が低いため、それだけ密度の高い氷が形成されるものと思われる。立方晶氷の 80K での格子定数は 6.353Å と報告されている。結晶子サイズが(111)反射からのものより（220）反射からのものが小さくでているのは、細孔内の氷に歪みがあることを強く示唆している。D=24Å の細孔に対する(220)反射からの格子定数が急に小さくなっていることは、おそらくこの細孔内の氷はもはや結晶と呼べる状態にはないことを示しているものと思われる。非晶質状態の解析に用いられる動径分布による方法を適用した研究が必要である。Vycor glass の場合、(111)反射からの結晶子サイズが(220)反射からのものより小さくなっている。歪みの効果では説明できない。(111)反射がサイズ効果や歪みの効果から考えられるよりもブロードになりすぎていることは明らかである。（111）反射の近くには通常の六方晶氷の反射がある。六方晶の氷が混ざっているために、(111)反射が見かけ上で広がっているとも考えることができるが、Vycor glass 内で生じた立方晶氷の(111)反射と(220)反射の相対強度は MCM-41 内のものと比べて逆に小さい。したがって、通常の氷が混ざっているために(111)反射がブロードになっているとは考えられない。Vycor glass 内の立方晶氷が特定の結晶面の方向に発達している可能性もあるが、積層欠陥などの効果も入れた回折パターンのシミュレーションが必要である。凝固・融解温度のヒステリシスとなんらかの関係があるかもしれない。ヒステリシスが水の充填率の低下とともに大きくなっていることは、ヒステリシスの原因が単なるサイズ効果によるもので

ないことを示しているようである。

MCM-41 の細孔内の水の凝固点降下度を細孔半径の逆数に対してプロットすると、図 7 のようになった。一般的に、微小粒子の凝固点降下度（ΔT）と粒子半径（R）の間に次のような Kelvin 式が成立することが知られている。

$$\Delta T = 2\Delta\sigma \cdot V \cdot T_0 / (\Delta h_f \cdot R)$$

ここで、Δσは固液界面張力、Vは液体の分子容、T_0はバルク液体の凝固点、Δh_fは凝固熱である。細孔壁に束縛された不凍水の厚みを０，２Å、６Åとした時のプロットが示してある。不凍水がないとしたとき、および不凍水の厚みが NMR 測定から報告されているほぼ２分子層（６Å）としたときのプロットはいずれも原点を通らない直線となっている。したがって、今回の測定から不凍水の厚みとして２Å（約１分子）という値が求まった。この結果から、自由水の性質が大きく変わる臨界細孔径は約２５Åということになる(図８)。水分子の直径は約３Åであるので、水分子が約８個横に並ぶような大きさのシリンダー状空間が臨界径となる。細孔壁の影響が細孔中心部に向かって４~５分子の層まで及ぶために自由水の凝固・融解挙動がこのような細孔径で変化するのか、あるいは D＝25Åというサイズが水のクラスター的挙動とバルク相的挙動の境界であるのかは現在のところ不明である。

むすび

ナノ細孔内における水の凝固・融解挙動を X 線回折により調べた。回折ピークの解析から、水の凝固・融解挙動に与える細孔サイズや形状の効果を明らかにすることができた。今後、回折パターンのより詳細な解析を行うことによって、細孔内の水の液固相転移に関する未解明な問題を明らかにすることができるものと思われる。

参考文献

（１） K.Overloop and L.Van Gerven, J.Magn.Reson.A101(1993)179.
（２） M.Brun, etal. Thermochim.Acta 21(1977)59.
（３） D.C.Steytler, J.C.Dore, and C.J.Wright, J.Phys.Chem.87(1983)2458.
（４） 粟屋 隆：データ解析，学会出版センター，東京（1994）.
（５） R.Schmidt, E.W.Hansen, M.Stocker, D.Akporiaye, and O.H.Ellestand, J.Am.Chem.Soc.117(1995)4049.
（６） P.Levitz, etal. J.Chem.Phys.95(1991)6151.
（７） R.R.Vanfeet and J.M.Mochel, Surf.Sci.341(1995)40.
（８） K.Morishige and K.Nobuoka, J.Chem.Phys.107(1997)6965.

表１．メソ多孔体内の水の凝固・融解温度，立方晶氷の格子定数・結晶子サイズ

	T_f	T_m	(111)反射		(220)反射	
			a	L	a	L
	(K)	(K)	(Å)	(Å)	(Å)	(Å)
MCM-41 D=42Å	232	232	6.43	26	6.39	22
D=36Å	228	228	6.39	22	6.39	19
D=35Å	225	225			6.36	19
D=29Å	212	212	6.39	19	6.36	15
D=24Å			6.37	13	6.08	9
Vycor glass f=1.1	245	255			6.36	33
f =0.9	244	255	6.40	25	6.36	33
f =0.6	234	250	6.40	20	6.36	27

T_f：凝固温度

T_m：融解温度

a ：格子定数

L：結晶子サイズ

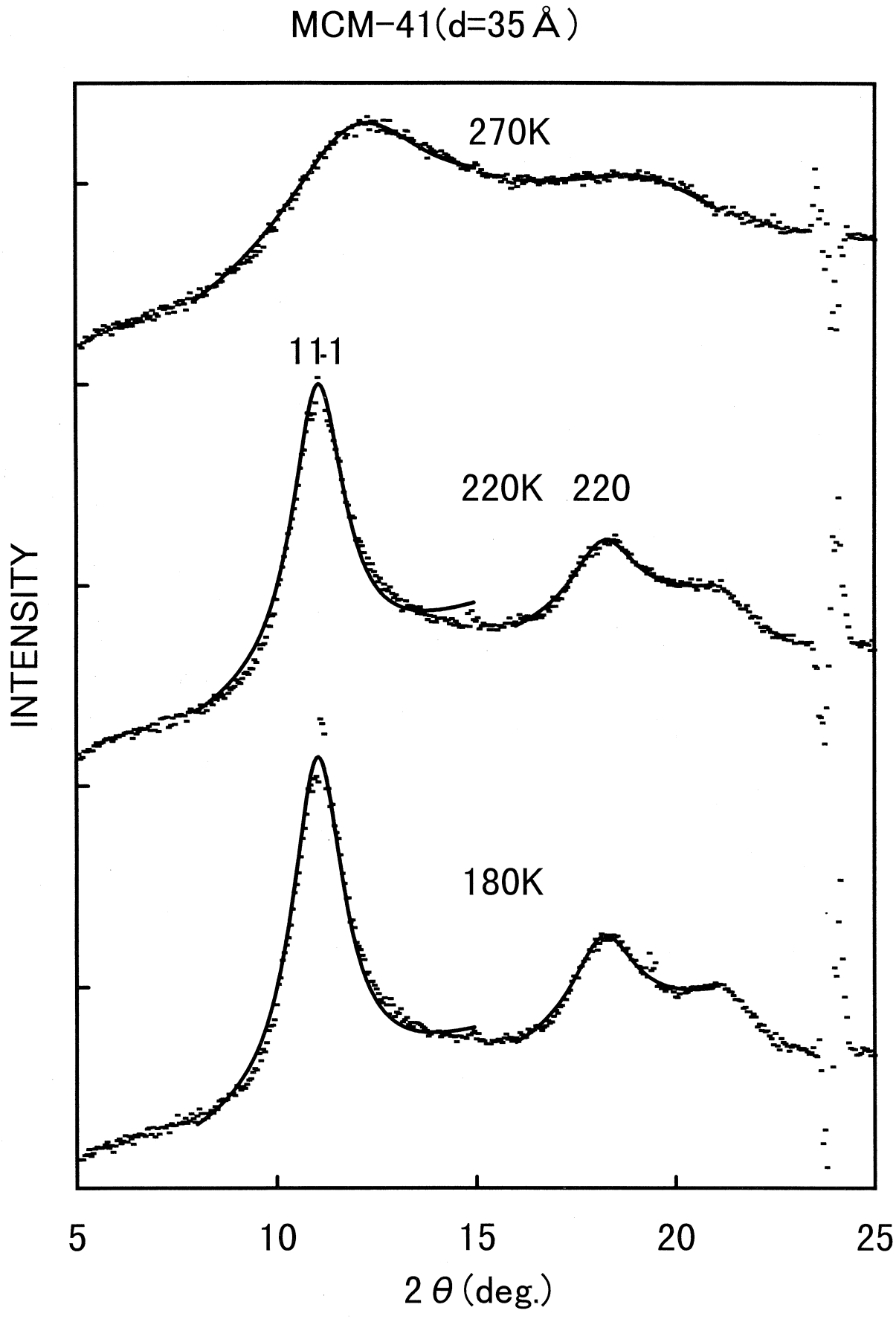


図１．MCM-41(D=35 Å)の細孔内における水のＸ線回折プロファイルの温度変化

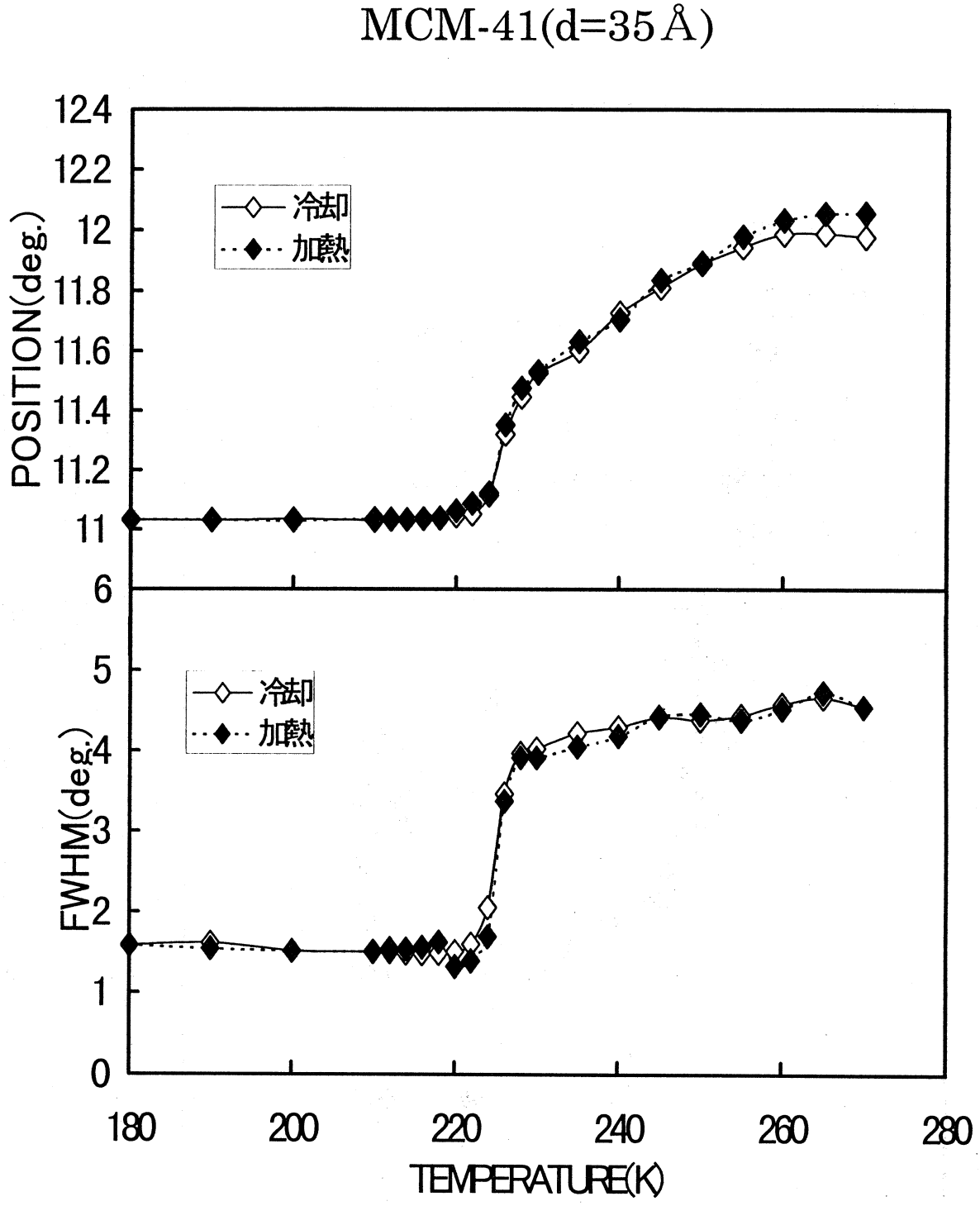


図２．MCM-41(D=35Å)の細孔内の水に対する回折ピーク位置と幅の温度変化

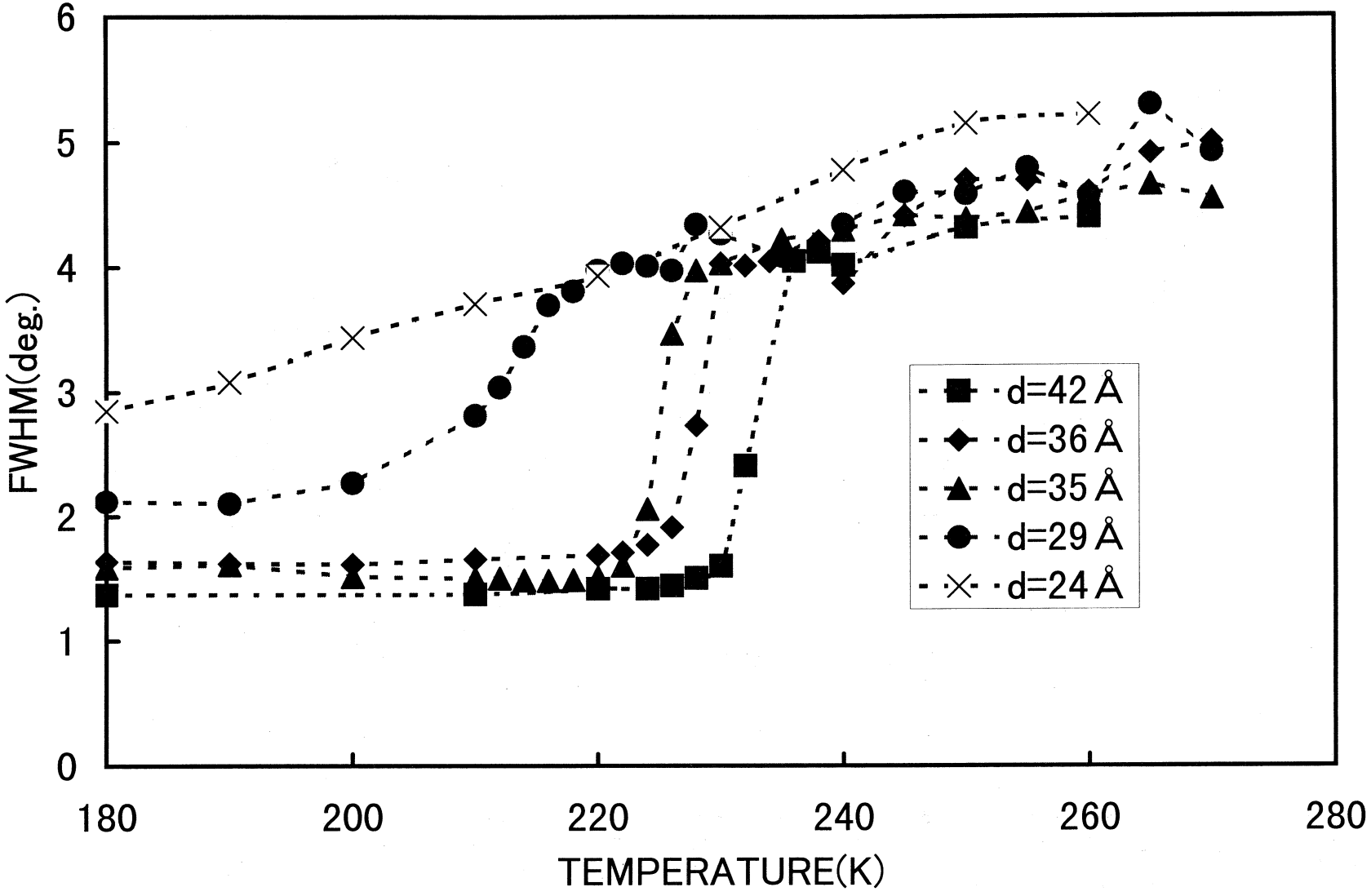


図３．いろいろな細孔径の MCM-41 内の水に対する回折ピーク幅の温度変化

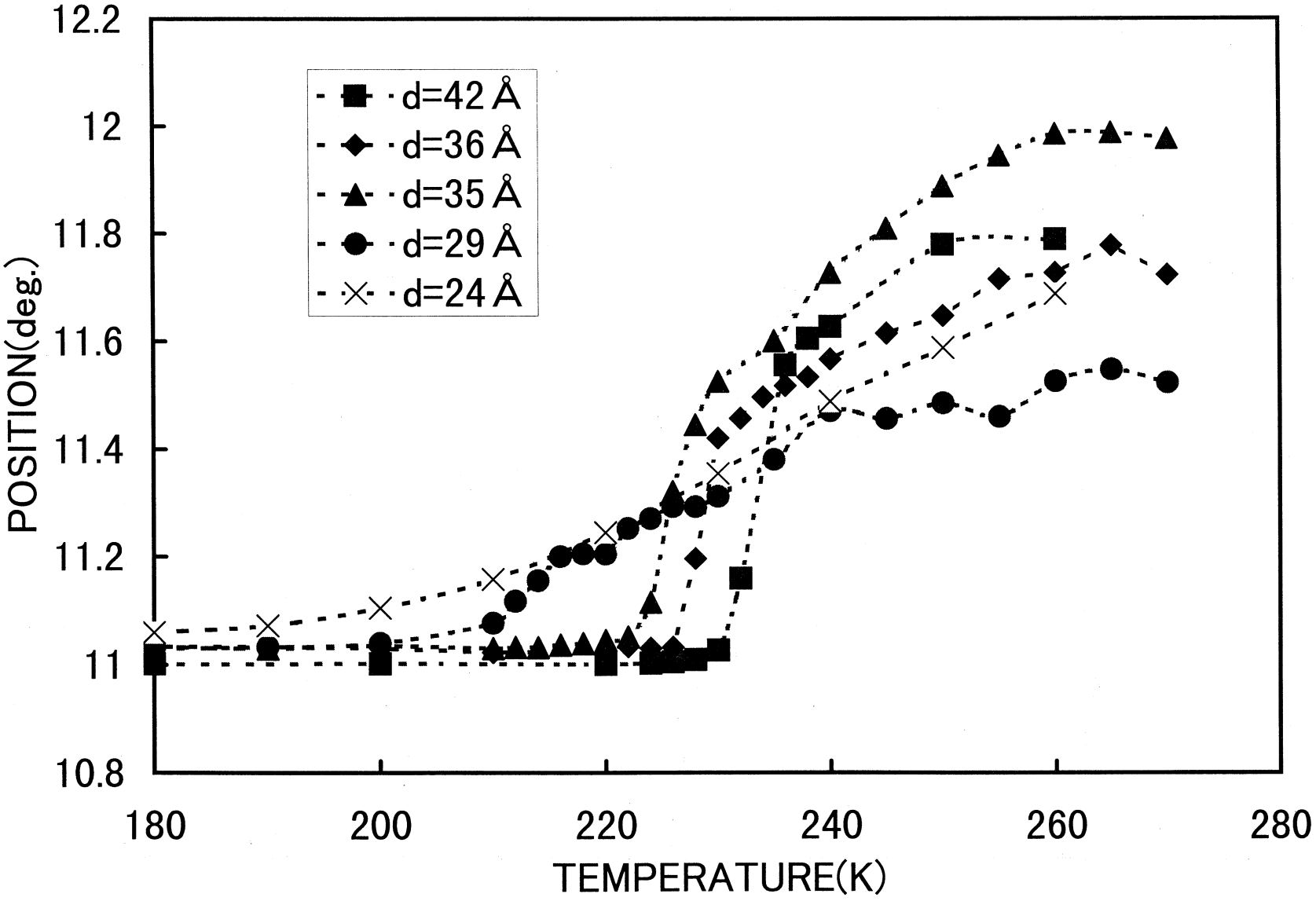


図４．いりいりな細孔径の MCM-41 内の水に対する回折ピーク位置の温度変化

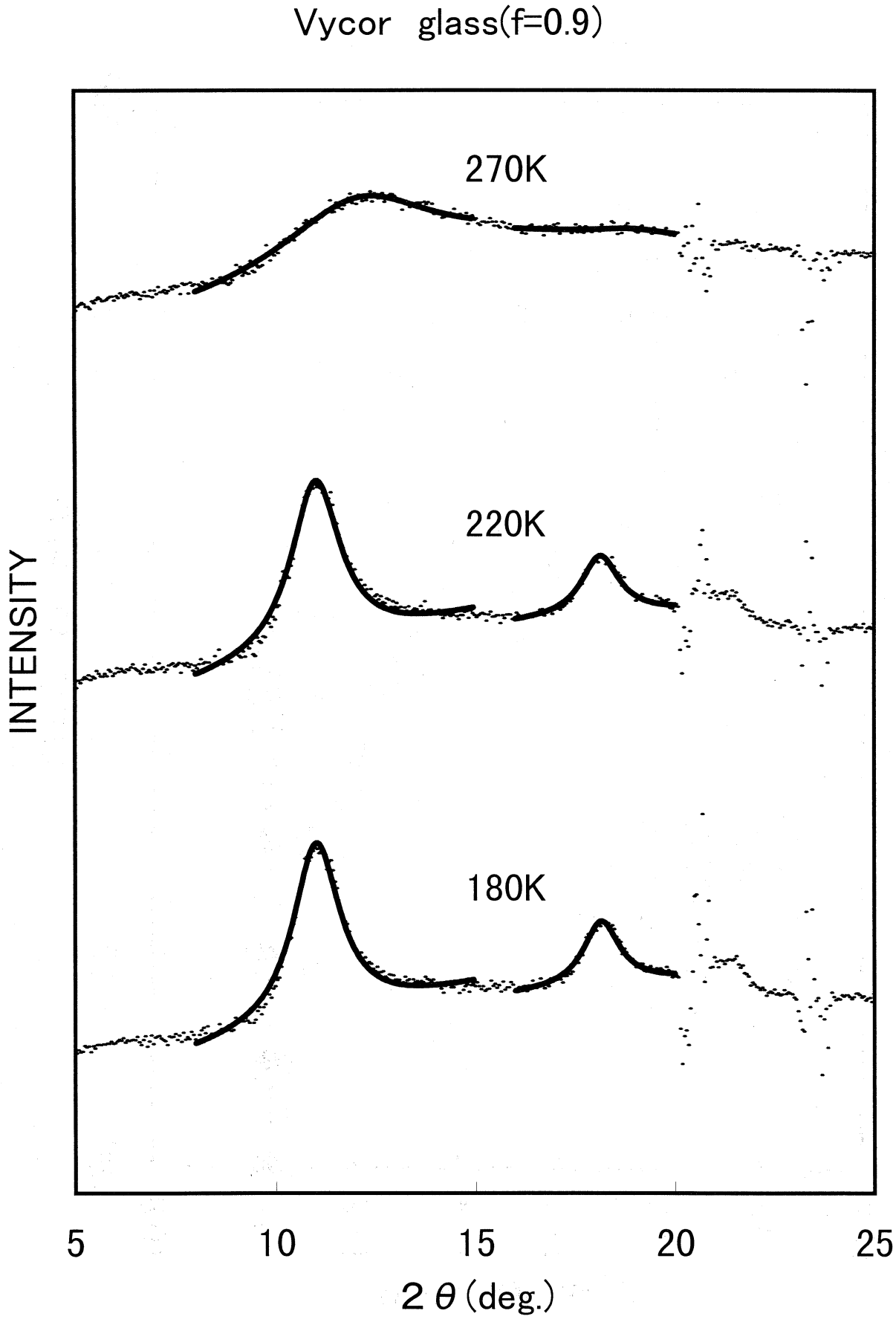


図５．Vycor glass の細孔内の水（f=0.9）に対するＸ線回折プロファイルの温度変化

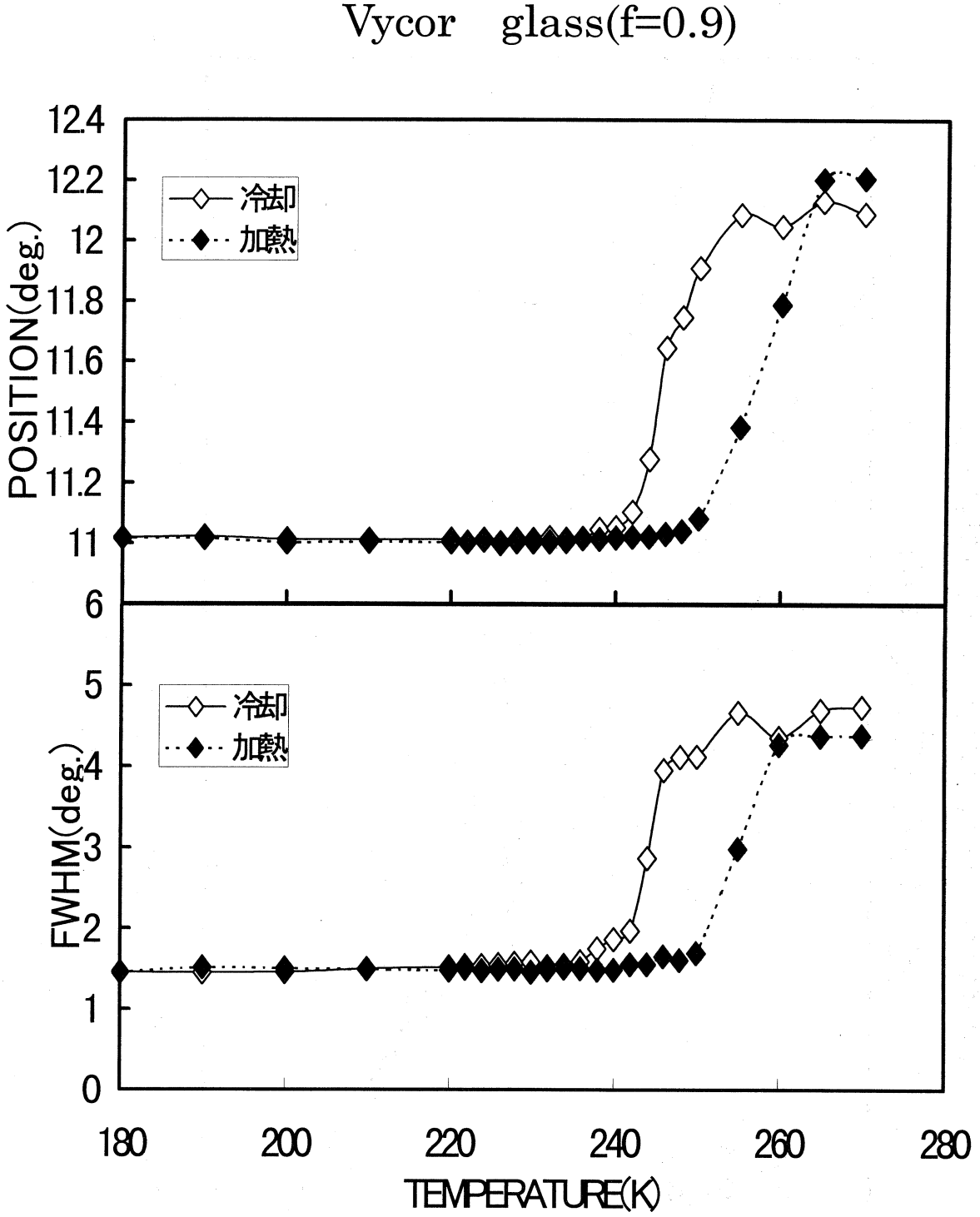


図６．Vycor glass 内の水のＸ線回折ピーク位置と幅の温度変化

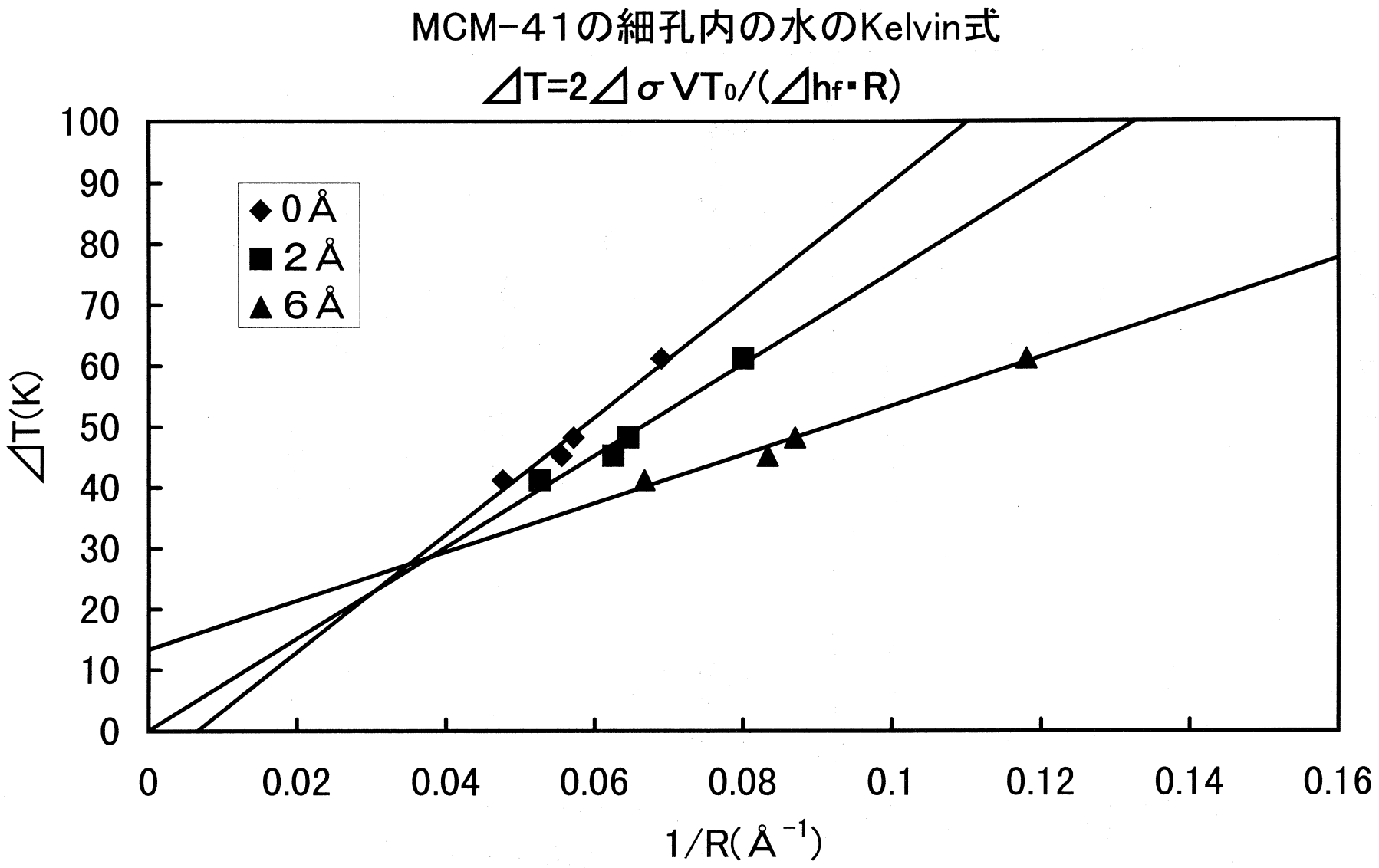


図７．MCM-41 内の水の凝固点降下度と細孔径との関係

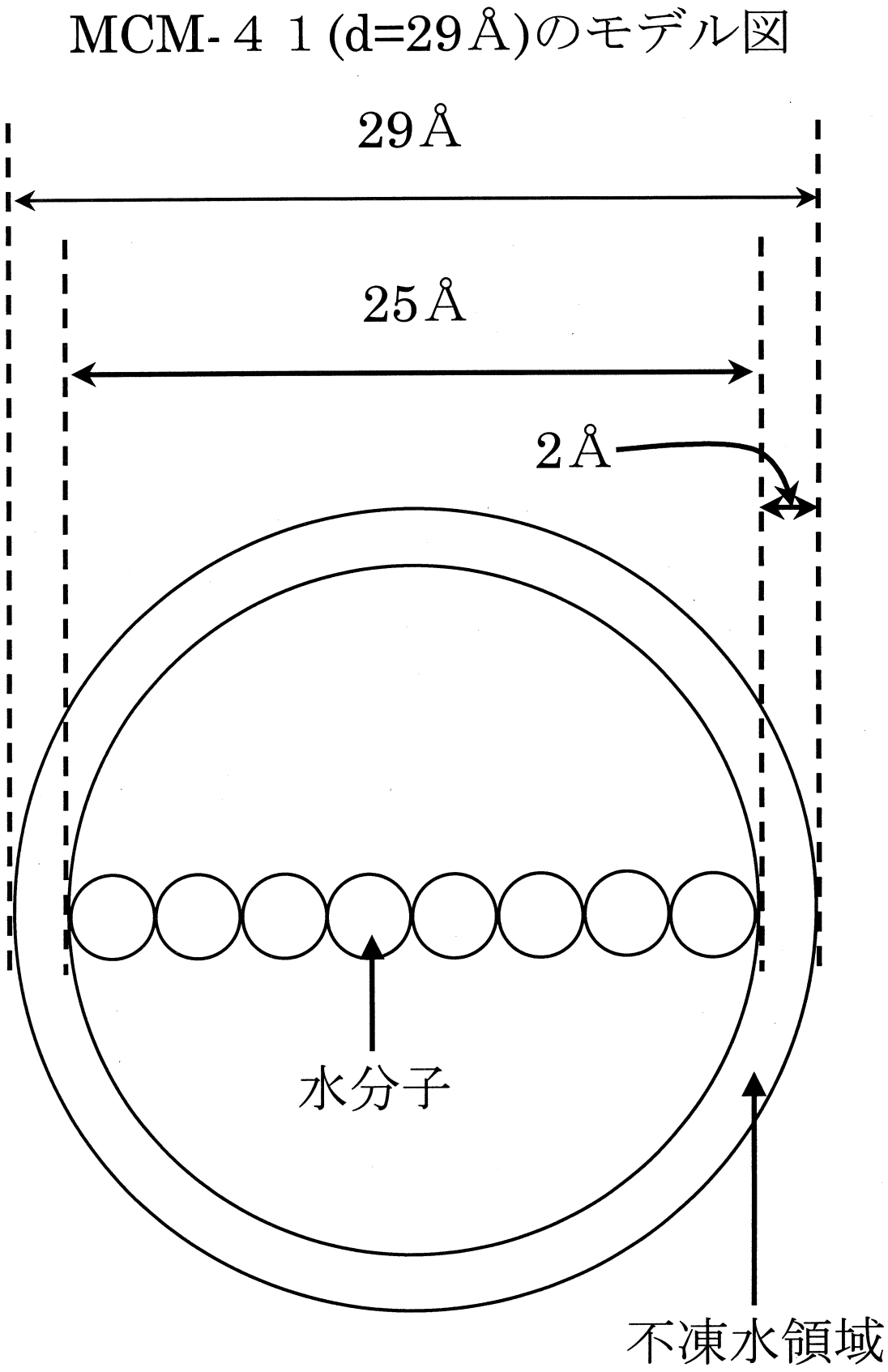


図８．D=29Åのシリンダー状細孔内における水のパッキング

ＤＶ-Ｘα法による$[MoCl_6]^{3-}$の電子状態計算

理学部　化学科　坂根弦太

【序論】　錯体化学で大きく関心がもたれているのは、化合物（錯体）の色である。色を計算により理解し、未知の化合物（錯体）の色を予測することは錯体化学者の夢である。今回はモリブデン単核錯体$[MoCl_6]^{3-}$の電子状態をＤＶ-Ｘα法で計算し、電子スペクトルの実験値との比較検討を行った結果、ＤＶ-Ｘα分子軌道計算プログラムのいくつかの入力パラメーターが予想以上に計算結果に影響を与えていることがわかった[1)]。ＤＶ-Ｘα分子軌道計算プログラムでは入力ファイル０５にMSCF、井戸形ポテンシャル、Roなどいくつかのパラメーターが存在する。これらのパラメーターが実際にどの程度計算結果に影響を与えているのか、またパラメーターはどのような値を選んだら良いのか、検討を行った。またサンプル点作成のパラメーターVECBASについても検討した結果、興味深い結果が得られたのであわせて報告する。

【計算】　座標はＸ線構造解析データ（図１）より求めた原子間距離・角度をもとにOh対称を仮定した。計算には以下の原子軌道を用いた：Mo, 1s～6p; Cl, 1s～5p. 入力データ（表１）は次の通り、MSCF=SCFS, RIDO=7.0, VIDO=-5.0,IDOA=1, Ro(Cl)=2.2, Ro(Mo)=2.8, ALF=0.3, VECBAS=√2, √3, √5, NMAXK=10000, 各原子軌道について出発の電荷と分子軌道計算後のMulliken Population Analysisにより得られた電荷との差が0.003電子になるまで、計算を繰り返した。HOMO近傍エネルギー準位図（図２）、HOMOおよびその一つ下の軌道の波動関数等高線図（図３）、電子密度(up,down)－全電子密度－有効スピン密度各等高線図（図４）、遷移エネルギーおよび遷移確率の計算結果と実測の電子スペクトルとの比較（図５）をそれぞれ示す。

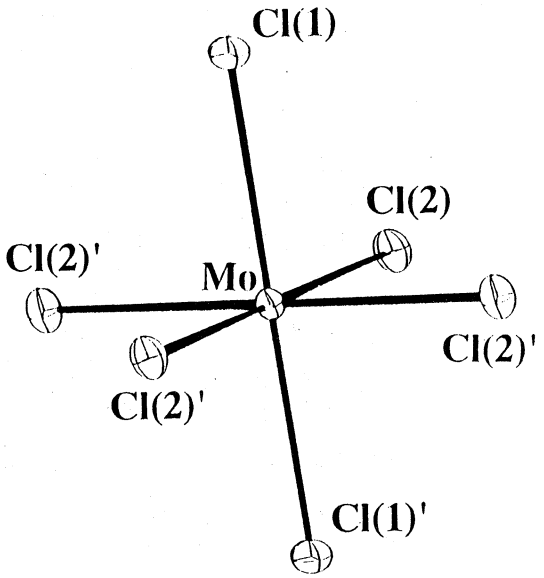


図１．$[MoCl_6]^{3-}$

　その後、パラメーターをいろいろ変えて（表２）、同様の計算を試みた。またＶＥＣＢＡＳについて、実際に発生している数（０～１の間に無秩序に分布する）をより直視的に検定するプログラムを作成し、デフォルト値の√2, √3, √5と共に、その他の無理数の組み合わせについても計算処理を行い、結果としてより均一にサンプル点がばらまかれる無理数の組み合わせを模索した。

表１．ＤＶ-Ｘα分子軌道計算プログラム　$[MoCl_6]^{3-}$入力データ

DV-Xα SCAT input data

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
    6
  (MOCL6)3- SPIN OH7 MF751AAA                7.00000  -5.00000     1
SCFSOGRN     10  30010000     099999     0     0    -4     0     0.00000000000D+00
    7     2    25     7     2     0
    1 17.0     1 17.0     1 17.0     1 17.0     1 17.0     1 17.0     2 42.0
 147.00000   0   0   0    0.00000   0   0   0    0.00000   0   0   0    0.00000
    0
 CL HFS ATOM CALC.
    1   300    11
  17.00000    0.70000  20.00000  32.00000   0.00000   0.00000
  1.0   0.0         -102.00000     1.00007     1.00008
  2.0   0.0           -9.70000     0.99860     0.99861
  2.0   1.0           -7.60000     2.99983     2.99980
  3.0   0.0           -0.90000     1.02884     1.03016
  3.0   1.0           -0.40000     2.93976     2.97831
  3.0   2.0           -0.10000     0.01844     0.03836
  4.0   0.0           -0.10000    -0.03555    -0.03729
  4.0   1.0           -0.10000    -0.01778    -0.00469
  4.0   2.0           -0.10000     0.03102     0.02212
  5.0   0.0           -0.10000     0.00182     0.00237
  5.0   1.0           -0.10000    -0.00292    -0.00334
00000000000
 MO HFS ATOM CALC.
    2   300    14
  42.00000    0.70000  30.00000  32.00000   0.00000   0.00000
  1.0   0.0         -682.00000     0.99994     0.99997
  2.0   0.0          -95.00000     1.00021     1.00013
  2.0   1.0          -88.00000     3.00000     3.00000
  3.0   0.0          -16.00000     1.00106     1.00082
  3.0   1.0          -13.00000     3.00001     3.00012
  3.0   2.0           -8.50000     4.99996     5.00000
  4.0   0.0           -2.30000     0.99900     0.97710
  4.0   1.0           -1.50000     2.96946     2.93670
  4.0   2.0           -1.50000     3.61034     1.12142
  5.0   0.0           -0.20000    -0.24614    -0.41867
  5.0   1.0           -0.10000    -0.15898    -0.37097
  5.0   2.0           -0.10000     0.05773    -0.35216
  6.0   0.0           -0.10000     0.01769     0.00324
  6.0   1.0           -0.10000    -0.02307    -0.04453
00000000000000
   19
    1     1     1     1     1     1     1     1     1     1
    1     1     1     1     1     1     1     1     1
    1     1     2     0     3     0     0     3     0     0
    3     0     0     3     0     0     2     0     1
    0.46240000000D+01     0.00000000000D+00     0.00000000000D+00     1
   -0.46240000000D+01     0.00000000000D+00     0.00000000000D+00     1
    0.00000000000D+00     0.46240000000D+01     0.00000000000D+00     1
    0.00000000000D+00    -0.46240000000D+01     0.00000000000D+00     1
    0.00000000000D+00     0.00000000000D+00     0.46240000000D+01     1
    0.00000000000D+00     0.00000000000D+00    -0.46240000000D+01     1
    0.00000000000D+00     0.00000000000D+00     0.00000000000D+00     2
    0.70000000000D+00
    0.00000000000D+00     0.00000000000D+00     0.00000000000D+00
    0.13535870870D+00     0.30000000000D+00     0.22000000000D+01
    0.27071741740D+00     0.30000000000D+00     0.22000000000D+01
    0.40607612610D+00     0.30000000000D+00     0.22000000000D+01
    0.54143483480D+00     0.30000000000D+00     0.22000000000D+01
    0.67679354350D+00     0.30000000000D+00     0.22000000000D+01
    0.81215225220D+00     0.30000000000D+00     0.22000000000D+01
    0.10000000000D+01     0.30000000000D+00     0.28000000000D+01
    1     1
   11   100
   0.00000    7.00000  -5.00000   0.00000   0.00000
  1.0   0.0         -102.00000     1.00007     1.00008
  2.0   0.0           -9.70000     0.99860     0.99861
  2.0   1.0           -7.60000     2.99983     2.99980
  3.0   0.0           -0.90000     1.02884     1.03016
  3.0   1.0           -0.40000     2.93976     2.97831
  3.0   2.0           -0.10000     0.01844     0.03836
  4.0   0.0           -0.10000    -0.03555    -0.03729
  4.0   1.0           -0.10000    -0.01778    -0.00469
  4.0   2.0           -0.10000     0.03102     0.02212
  5.0   0.0           -0.10000     0.00182     0.00237
  5.0   1.0           -0.10000    -0.00292    -0.00334
   14   100
   0.00000    7.00000  -5.00000   0.00000   0.00000
  1.0   0.0         -682.00000     0.99994     0.99997
  2.0   0.0          -95.00000     1.00021     1.00013
  2.0   1.0          -88.00000     3.00000     3.00000
  3.0   0.0          -16.00000     1.00106     1.00082
  3.0   1.0          -13.00000     3.00001     3.00012
  3.0   2.0           -8.50000     4.99996     5.00000
  4.0   0.0           -2.30000     0.99900     0.97710
  4.0   1.0           -1.50000     2.96946     2.93670
  4.0   2.0           -1.50000     3.61034     1.12142
  5.0   0.0           -0.20000    -0.24614    -0.41867
  5.0   1.0           -0.10000    -0.15898    -0.37097
  5.0   2.0           -0.10000     0.05773    -0.35216
  6.0   0.0           -0.10000     0.01769     0.00324
  6.0   1.0           -0.10000    -0.02307    -0.04453
 WELLSCCS
 10000    0.1000000    0.0000000   0.0000000   0.0000000
    2     2     2     2
 AAAASCFS
 10000    0.1000000    0.0000000   0.0000000   0.0000000
    2     2     2     2
```

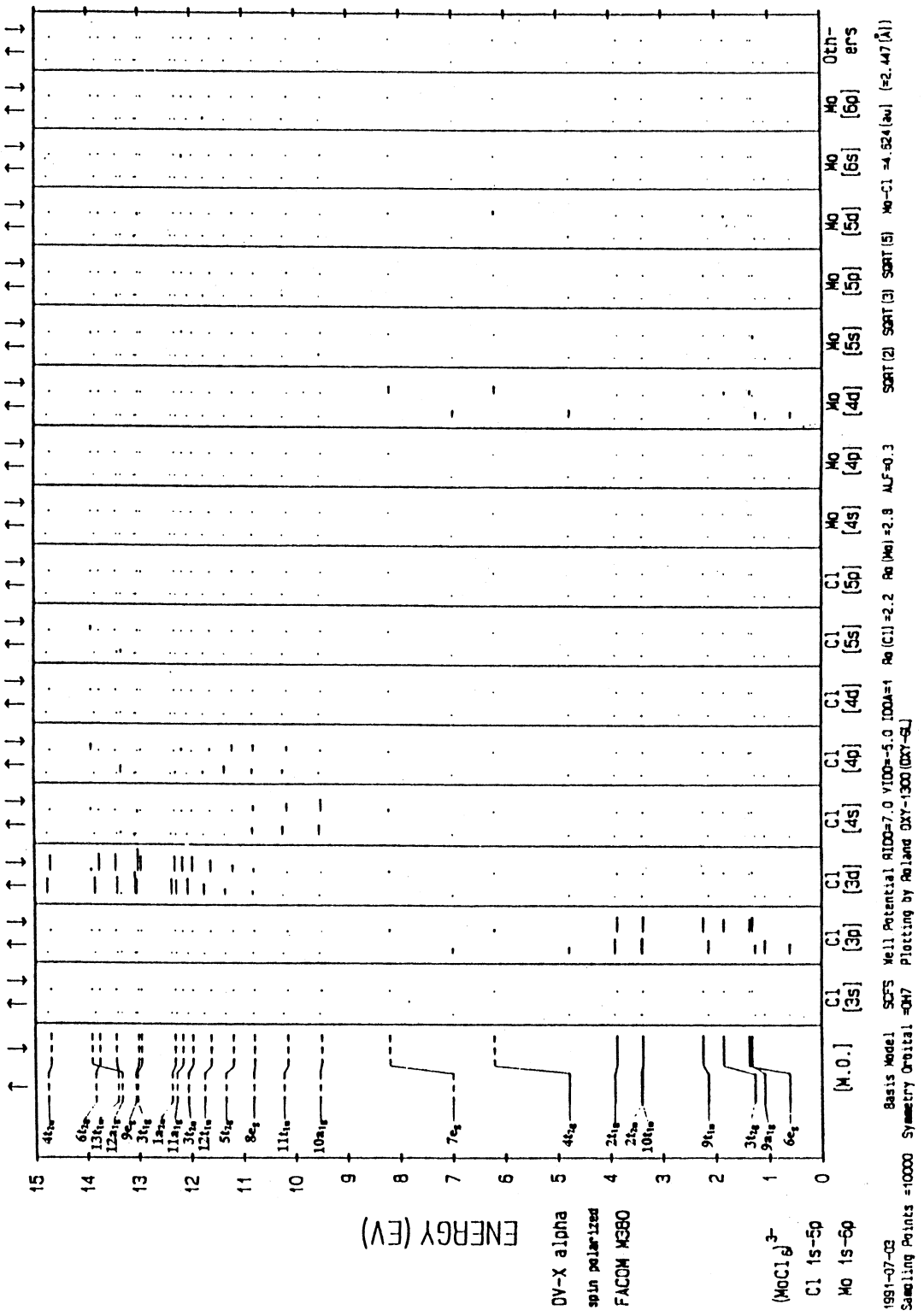
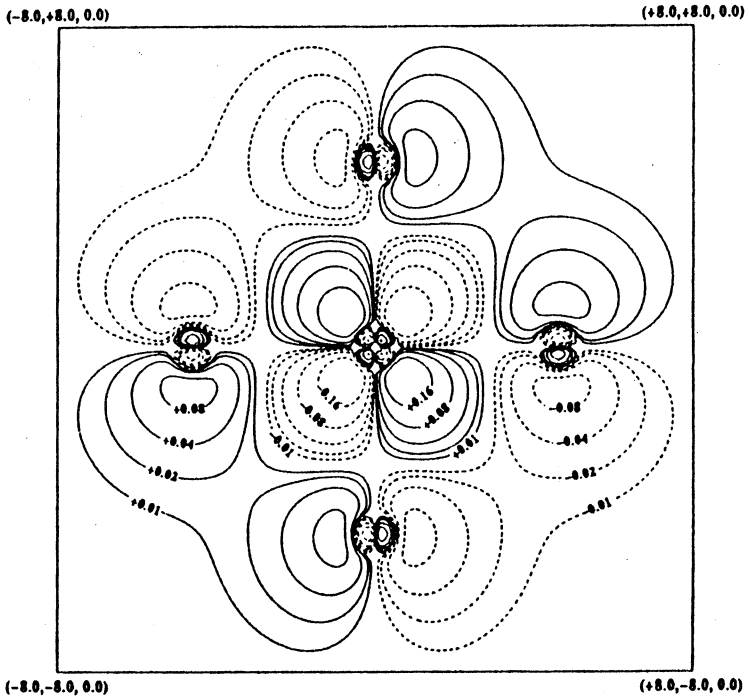
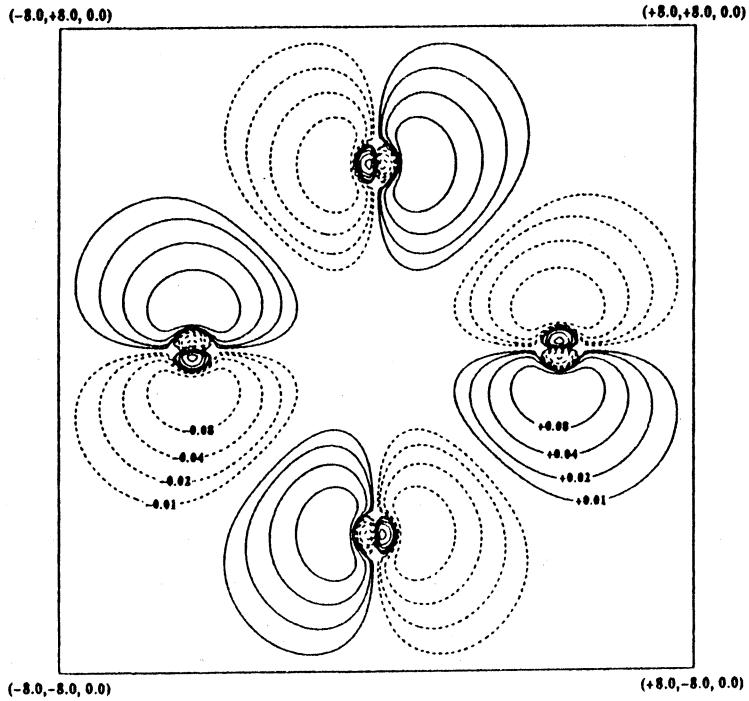


図２．ＨＯＭＯ近傍エネルギー準位図



Contour map of $4t_{2g}/3$(up) MO wave function

—— positive ····· negative



Contour map of $2t_{1g}/3$(up) MO wave function

—— positive ····· negative

図３．HOMO およびその一つ下の軌道の波動関数等高線図

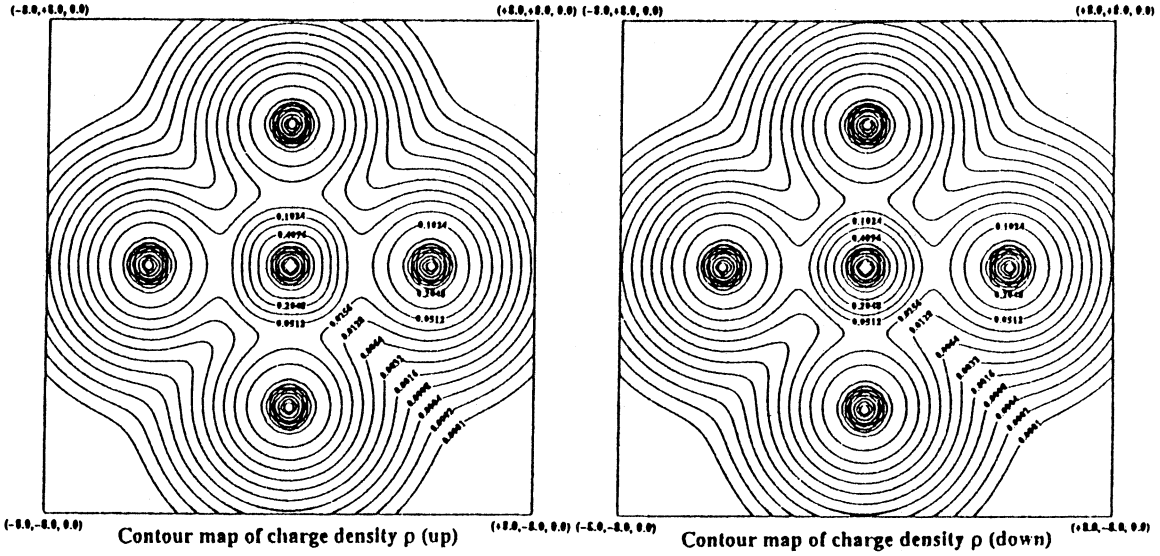


図４．電子密度、全電子密度、有効スピン密度、各等高線図

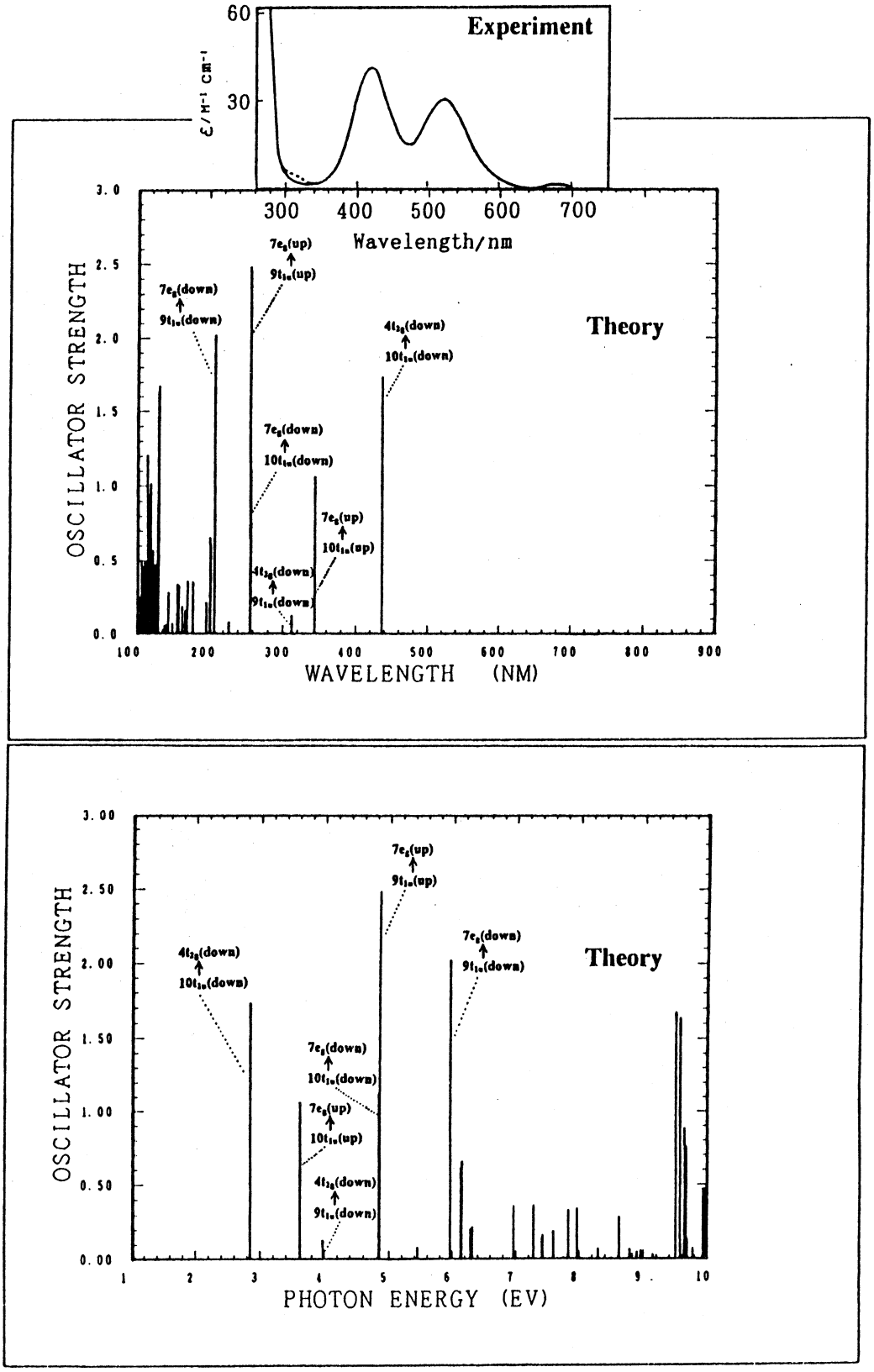


図５．遷移エネルギーと遷移確率

表２．各種パラメーター

DV–Xα file05

·**MSCF** ····· SCFS or SCCS
·**Well Potential**
　RIDO ····· 3.0, 4.0, ···, 9.0, 10.0
　VIDO ····· –0.5, –1.0, ···, –5.0
　IDOA ····· 0 or 1 or 2
·R_0**(Mo)** ····· 1.0, ···, 5.0
·R_0**(Cl)** ····· 1.0, ···, 5.0
·**ALF** ····· 1.0, ···, 0.3
·**Sampling Points**
　NMAXK ····· 1000, 5000, 10000, 30000, 60000
　VECBAS ····· (SQRT(2),SQRT(3),SQRT(5))
　　····· (SQRT(5),SQRT(27),SQRT(33))

·**MSCF** ····· SCFS
·**Well Potential**
　RIDO ····· 7.0
　VIDO ····· –5.0
　IDOA ····· 1
·R_0**(Mo)** ····· 2.8
·R_0**(Cl)** ····· 2.2
·**ALF** ····· 0.3
·**Sampling Points**
　NMAXK ····· 10000
　VECBAS ····· (SQRT(2),SQRT(3),SQRT(5))

【結果と考察】　井戸形ポテンシャルについては、RIDO(半径[au])=3,4,5,6,7,8,9,10、VIDO(深さ[au])=-0.5, -1.0, -1.5、IDOA(ポテンシャルのつなぎ目の形,垂直(0),曲線(1),斜線(2))の組み合わせすべてについて計算した。井戸形ポテンシャルは原子の感ずるポテンシャルに原子核位置から RIDO の距離で VIDO の深さだけ井戸形ポテンシャルを重ね合わせて、波動関数の r →∞での不必要な振幅を減衰させるものである。比較の結果、RIDO は 6,7,8 あたりで変化が少なく、3,4,5 または 9,10 で変化が大きかった。VIDO は-1.0 と-1.5 ではそれほど違いがなかったが、-0.5 とは違いがあった。IDOA は、2 の場合、計算が発散しやすく、0 と 1 では収束がはやかった。また RIDO,VIDO の違いの影響は、IDOA=2 がいちばん大きく、1 がいちばん小さかった。井戸形ポテンシャルは、RIDO が 6〜8、VIDO は深めにとって-1.0〜-5.0、IDOA は 1 が良いと考えている。

MSCF については、SCFS と SCCS で結果が大きく異なった。SCFS は基底関数を作るとき、周りの原子の影響を取り入れる方法であり、SCCS は周りの原子を考慮せずに孤立原子として扱う。SCFS の方がいろいろ調整が効くが、それだけパラメーター（特に Ro）の設定が難しい。SCCS では、Ro の影響はほとんどないが、SCFS では著しい違いが見られる。Ro はサンプル点の動径方向に対する分布関数としての役割をもっており、SCCS ではこの役割のみをもつ。ところが SCFS では、それ以外にもう一つ、基底関数を作るときの各原子の原子球半径という役割がある。この役割が SCFS では計算結果に大きな影響力をもつ。実際に計算に用いられた各原子の動径波動関数を点検してみると、Ro の選び方次第では周りの原子の位置でふくらみをもつようなおかしな形になっている場合があり（図６）、注意が必要である。このように R0 の基底関数の形に対する影響は大きいので、この Ro にどのような値を選べば良いのか、非常に難しい問題であるが、考え方として二つの方法を試みてみた。一つは、核間距離をそれぞれの“原子番号”の重みで分け合う考え方、これを用いた計算結果は、例えば LUMO において塩素の 3p 軌道が主成分となった。もう一つは“イオン半径”の重みで核間距離を分け合う考え方、これによると LUMO はモリブデンの 4d 軌道が主成分となった。このように Ro を変えると計算結果があまりに大きく変わるので、Ro については適当な値を検討中である。

Ro と関連するパラメーターに ALF がある。ALF は値が小さければ積分の精度が上がるが、サンプル点の数を増やしてやらないと逆に精度が落ちる。標準値は 1.0 だが、今回 ALF=0.3 の計算も行ってみたところ、占有軌道ではそれほど差がないが、空軌道では違いが見られた。今回はサンプル点を 10000 と十分とっているので、ALF は 0.3 を基準とした。

サンプル点は一般に多ければ多いほど良いが、NMAXK=1000, 5000, 10000,

30000, 60000 それぞれの計算結果を比較したところ、5000 以上ではそれほど違いはなかった。

VECBAS については、デフォルト値√2, √3, √5 の検定の結果、√2 と√5 で発生する面の分布が均一でないことがわかった（図７）。このデフォルト値を変更して、例えば√5, √27 √33 を用いると、ほぼどの面も均一になった（図８）。VECBAS は三つの無理数で指定する。この三つの無理数に対して１から順にサンプル点数（今回は 10000)になるまでその数を掛けてやり、その少数部分を取り出していく。そしてそれを対応する三つの座標軸で表される長さが１の空間内にプロットしていく。そして最終的にプロットした点が空間内で均一ならば、サンプル点は実際の空間で角度に関して均一に分布する。デフォルト値（√2, √3, √5）のように、長さが１の空間内の分布が均一でない場合、実際にばらまかれるサンプル点は角度に関して重複する点があり。何割かが無駄になっているものと思われる。実際√5, √27 √33 を用いて計算したところ、計算結果に違いが見られた。

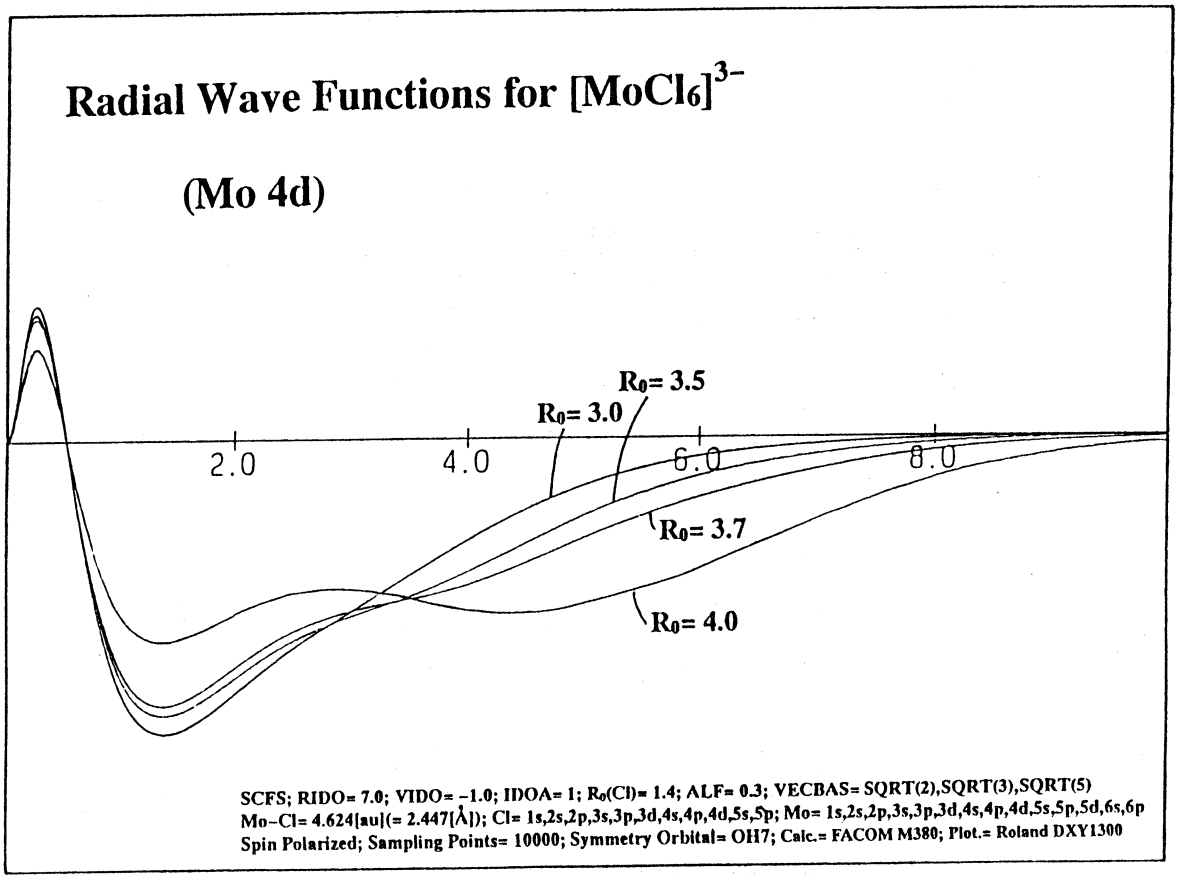


図６．Ro を変えた際、用いられた各原子の動径波動関数

```
****************** DV-Xa Sampling Points ******************

      VECBAS(1)=1.414213538169861    ( SQRT( 2 ) )
      VECBAS(2)=1.732050776481628    ( SQRT( 3 ) )
      VECBAS(3)=2.236068010330200    ( SQRT( 5 ) )

 DV-Sampling Points  = 10000
```

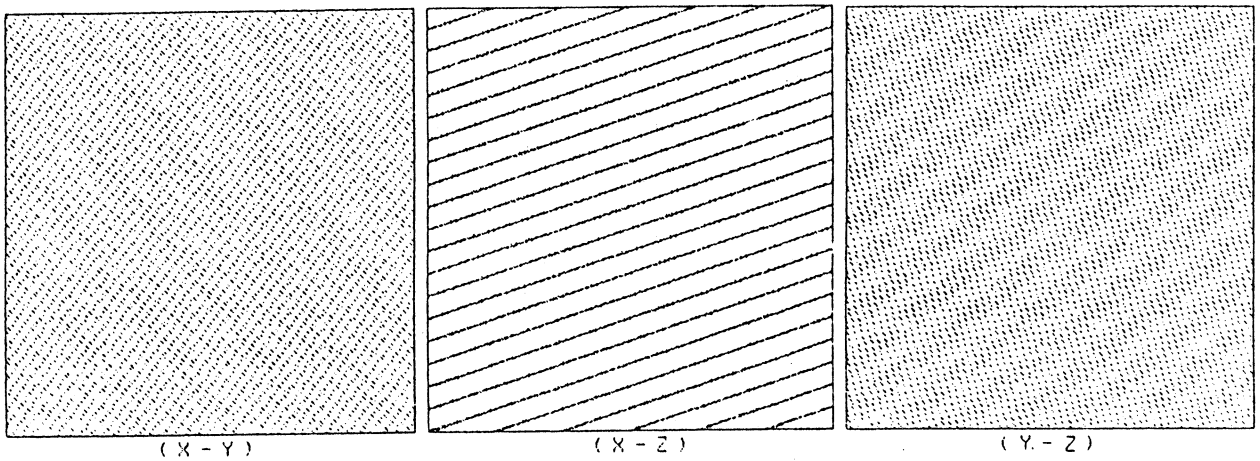


図7. VECBAS = $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}$

```
****************** DV-Xa Sampling Points ***************

      VECBAS(1)=2.236068010330200    ( SQRT( 5 ) )
      VECBAS(2)=5.196152687072754    ( SQRT( 27 ) )
      VECBAS(3)=5.744562625885010    ( SQRT( 33 ) )

 DV-Sampling Points  = 10000
```

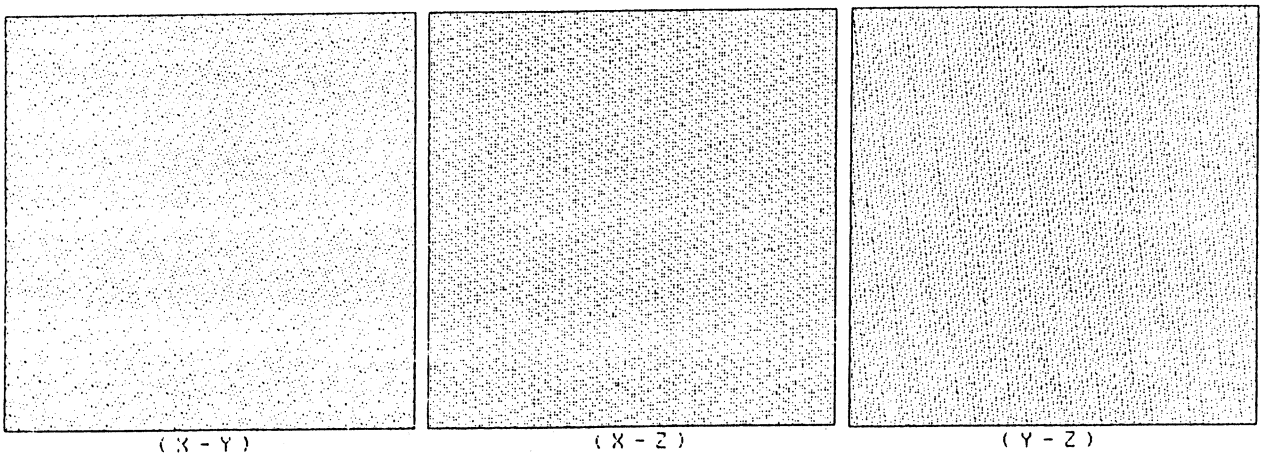


図8. VECBAS = $\sqrt{5}$, $\sqrt{27}$ $\sqrt{33}$

【最後に】　今回計算した錯体、$[MoCl_6]^{3-}$は中性モデルに比べて電子が過剰に存在している錯陰イオンである。一般に錯陰イオンの計算は発散しやすく、パラメーターの値によってはおかしな計算結果に陥ってしまうことがある。

今回検討したＤＶ-Ｘα分子軌道計算の各種パラメーターは飽くまで$[MoCl_6]^{3-}$についてのものであり、他のモデルにそのまま適用できるものではない。むしろＤＶ-Ｘα計算では一般に、座標データのみを入力し、一切パラメーターを修正することなく実験値とよく一致した結果が得られる[2-7]。

【文献】

1) 坂根弦太,柴原隆志,足立裕彦, *Bulletin of the Society for Discrete Variational Xα*, **4**, 9(1991).
2) 足立裕彦著,“量子材料化学入門”，三共出版(1991).
3) 足立裕彦監修,“はじめての電子状態計算”，三共出版(1998).
4) 足立裕彦他著,“量子材料学の初歩”，三共出版(1998).
5) 足立裕彦他著,“ＤＶ-Ｘα法による電子状態計算”，三共出版(1996).
6) Genta Sakane, Takashi Shibahara, and Hirohiko Adachi, *Journal of Cluster Science,* **6** (4), 503-521(1995).
7) Takashi Shibahara, Genta Sakane, Yoshiyuki Naruse, Kazuhisa Taya, Haruo Akashi, Akio Ichimura, and Hirohiko Adachi, *Bulletin of the Chemical Society of Japan,* **68** (10), 2769-2782(1995).

T-matrix Poles of $\Lambda N - \Sigma N$ Interactions

H. Yamamura, S. Imai, K. Miyagawa

Department of Applied Physics, Okayama University of Science,
Ridai-cho Okayama 700, Japan

Analyses of few-baryon systems including strangeness have been recently developed.[1] Those analyses with the $\Lambda N - \Sigma N$ coupling incorporated have an advantage in evaluating hyperon-nucleon (YN) interaction models strictly. One of the differences between YN and NN interaction is that the former has the $\Lambda N - \Sigma N$ coupling, and it shows a almost cusp just at the ΣN threshold in the ΛN elastic total cross section. This is important because it implies the existence of poles near the ΣN threshold. This pole should correlate to the strength of the $\Lambda\Sigma$ conversion. We thus searched for poles of t-matrices on the complex energy plane for meson-theoretical YN interactions such as the Nijmegen softcore and the hard core models D and F. We obtained off-shell t-matrices by solving the Lippmann-Schwinger equation with the $\Lambda N - \Sigma N$ coupling in momentum space, and those are analytically continued to the complex energy plane.

The results are given in Table 1 and 2. We find poles not only around the ΣN but also around the ΛN threshold. The antibound-state poles below the ΛN threshold are responsible for the attraction of the YN interactions in the low-energy region (Those are correlated to the scattering lengths). On the other hand, we find that the positions of the poles close to the ΣN threshold are determined by the shape of the ΛN elastic cross section. The pole located in the third quadrant of $k_{\Sigma N}$ (relative momentum between Σ and N) form cusps just at the threshold, while the poles in the second quadrant cause a round resonance peak below the threshold. The NSC and the model D correspond to the former, while the model F the latter.

model	partial wave	$k_{\Lambda N}$ [fm^{-1}]	$k_{\Sigma N}$ [fm^{-1}]	E [MeV]	a [fm]
NSC	$^1S_0$	(0,-0.28)	(0,1.47)	(2051.5,0)	-2.48
	$^3S_1 - {}^3D_1$	(0,-0.45)	(0,1.51)	(2046.8,0)	-1.38
D	$^1S_0$	(0,-0.35)	(0,1.49)	(2050.0,0)	-1.83
	$^3S_1 - {}^3D_1$	(0,-0.35)	(0,1.49)	(2050.0,0)	-1.89
F	$^1S_0$	(0,-0.31)	(0,1.48)	(2050.8,0)	-2.19
	$^3S_1 - {}^3D_1$	(0,-0.36)	(0,1.49)	(2049.7,0)	-1.83

Table 1: The poles near the ΛN threshold. $k_{\Lambda N}$, $k_{\Sigma N}$ are relative momenta of ΛN and ΣN channels, respectively. E indicates the center of mass energy, and a does scattering length.

model	$k_{\Lambda N}$ [fm^{-1}]	$k_{\Sigma N}$ [fm^{-1}]	E [MeV]
NSC	(1.37,0.01)	(-0.04,-0.39)	(2126.3,1.07)
D	(1.43,0.01)	(-0.18,-0.08)	(2132.8,1.13)
F	(1.44,0.02)	(-0.28,0.12)	(2134.2,-2.49)

Table 2: The poles near the ΣN threshold for the force components $^3S_1 - {}^3D_1$. See the captain to Table 1 for details.

Reference

[1] K.Miyagawa, H.Kamada, W.Glöckle and V.Stokes, Phys.Rev.C51, 2905(1995)

ICCG 法による行列計算の並列化に関する検討

三浦隆志　　橋本禮治

岡山理科大学大学院工学研究科電子工学専攻

1. はじめに

並列処理技術は、従来より気象予報、ＣＦＤ（Computational Fluid Dynamics:流体解析)、ＱＣＤ(Quantum Chromodynamics:量子色力学)、構造解析、電子回路シミュレーション、原子炉安全性解析、核融合シミュレーション、ニューラルネットシミュレーション、画像処理など多くの分野でその利用が注目されている。並列処理技術は、

- スーパーコンピュータのような超高速コンピュータの開発
- 価格性能比の優れたコンピュータシステムの開発
- 信頼性の高いコンピュータシステムの開発

などに対して必要不可欠のものであり、半導体技術の限界にともない、今後ますます重要性を増すものと思われる。

本研究では、不完全コレスキー分解による前処理付き共役傾斜法（以下ＩＣＣＧ法）を並列化することにより連立方程式を高速に解く事を試みたものである。ＩＣＣＧ法は、電磁気学や流体力学などに欠かすことのできない大規模連立１次方程式の解を解く反復法の１つである。現在の半導体技術では、100 万項を越えるような大規模連立方程式を解くのに、最速のベクトル演算機を用いたとしても、大変時間がかかってしまうため並列化に期待が集まっている。ところが ICCG 法は、一般的に並列化が困難とされている。そこで、ICCG 法を並列化するための手法としてブロードキャストの繰り返しによる方法を検討してみた。

研究を行うに当たっては、 本学にある intel 社の分散メモリ型並列コンピュータ paragon XP/S を使用した。これは、ノード数を 296 個持つ並列コンピュータで、その演算性能は最大 22.2GFLOPS である。

2. 実験

2.1 共役傾斜法

主要な連立１次方程式の解法としては消去法と反復法が知られている。消去法は、方程式から変数を１つずつ消去していくことによって解く方法で有限回の演算で厳密解が得られる。これに対し反復法は、適当な第１近似解から出発して「第ｋ近似値をもとに、より精度の良い第ｋ＋１近似値を作る公式」を繰り返して適用することによって、次第に精度

を高めていく方法である。この方法では消去法よりもはるかに早く実用上十分な精度の解が得られることもあるが、非常に多数回の反復をしても十分な精度が得られないこともあり、それが反復法の欠点とも言える。しかしながら共役傾斜法（以下ＣＧ法）は、反復法の一種であるがうまくいかずに早く答えが得られない場合でも、ある一定の回数まで反復すれば確実に厳密解に到達するため非常に便利な反復法といえる。ただし、ＣＧ法は、係数行列が正値対称行列のときしか適用できない。よって正値対称でない場合は、前処理を加えて正値対称に変換する必要がある。

ところで、ＣＧ法は、反復法であるので、係数行列が単位行列に近いほど速く収束すると言える。そこで、ＣＧ法の前処理としてコレスキー分解（後述）を不完全に行った不完全コレスキー分解を適用して係数行列を単位行列に極めて近い形に変換する方法をとった。これが不完全コレスキー分解による前処理付き共役傾斜法（ICCG 法）である。

2.2 コレスキー分解

コレスキー分解は、係数行列 $\boldsymbol{A}$ が正値対称行列であるときの $\boldsymbol{LU}$ 分解である。$\boldsymbol{A}$ が正値対称行列であることから

$$a_{ij} = a_{ji} \tag{1}$$

となり、与えられた方程式は、$\boldsymbol{A}$ を $\boldsymbol{LU}$ 分解すると $\boldsymbol{L} = \boldsymbol{U}^t (l_{ij} = u_{ji})$ より

$$\boldsymbol{U}^t\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b} \tag{2}$$

となる。また $\boldsymbol{A}$ を $\boldsymbol{LU}$ 分解したときの $\boldsymbol{L}$ および $\boldsymbol{U}$ の各要素と $\boldsymbol{A}$ の要素の関係は

$$\sum_{k=1}^{\min(i,j)} l_{ik}u_{ki} = a_{ij} \qquad (1 \le i,j \le n) \tag{3}$$

のような n^2 個の連立方程式として表される。式(3)は、式（1）より未知数 $n(n+1)/2$ に対し方程式が $n(n+1)/2$ 個得られるので $\boldsymbol{U}$ の各要素は

$$\begin{aligned}
u_{11} &= \sqrt{a_{11}} \\
u_{1j} &= \frac{a_{1j}}{u_{11}} \quad (2 \le j \le n) \\
u_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^{\,2}} \quad (2 \le i \le n) \\
u_{ij} &= \left(a_{ij} - \sum_{k=1}^{i-1} u_{ki}u_{kj}\right) \Big/ u_{ii} \quad (2 \le i < j \le n) \\
u_{ij} &= 0 \quad (1 \le j < i \le n)
\end{aligned} \tag{4}$$

と求まる。これより、三角方程式 $\boldsymbol{U}^t\boldsymbol{y} = \boldsymbol{b}$ を解くと

$$y_1 = \frac{b_1}{u_{11}}$$
$$y_i = \frac{b_i - \sum_{k=1}^{i-1} u_{ki} y_k}{u_{ii}} \quad (i = 2,3,\cdots,n) \tag{5}$$

となる。さらに、$Ux=y$ を解くと連立方程式(2) の解が次式のように求まる。

$$x_n = y_n$$
$$x_i = y_i - \sum_{k=i+1}^{n} u_{ik} x_k \quad (i = n-1, n-2, \cdots, 1) \tag{6}$$

2.3 ICCG 法

解くべき方程式 $Ax=b$ の係数行列 A は正値対称行列であるとして、A を次のように不完全にコレスキー分解する。

$$A \cong LDL^T \tag{7}$$

L は左下三角行列、D は対角行列である。式(7)の D の成分の平方根を対角成分に持つ行列を $D^{\frac{1}{2}}$ と書くと、式（7）は、

$$LDL^T = LD^{\frac{1}{2}}(LD^{\frac{1}{2}})^T \tag{8}$$

と書くことができる。A が正値であれば、$LD^{\frac{1}{2}}$ は特異ではない。そこで

$$[(LD^{\frac{1}{2}})^{-1} A((LD^{\frac{1}{2}})^T)^{-1}](LD^{\frac{1}{2}})^T x = (LD^{\frac{1}{2}})^{-1} b \tag{9}$$

という連立方程式を考える。これはもとの方程式 $Ax=b$ と同値である。したがって、$Ax=b$ と式（9）の解は等しい。仮に式（8）が A の正しいコレスキー分解ならば、式（9）の左辺に現れる行列

$$B = (LD^{\frac{1}{2}})^{-1} A((LD^{\frac{1}{2}})^T)^{-1} \tag{10}$$

は完全な単位行列になるが、式（7）が不完全なためにそうはならない。しかし、不完全ながらほぼコレスキー分解に近い操作を行っているのだから、式（10）は単位行列に近い行列になっている。したがって、方程式（9）にＣＧ法を適用すれば、かなり少ないループ回数で収束して厳密解に到達すると期待される。

2.4 計算手順

ＩＣＣＧ法の計算手順をベクトルを用いて表現してみると以下のようになる。

連立１次方程式の係数行列A、定数項bおよび解ベクトル$\mathbf{x}$を

$$A=\begin{bmatrix} a_{11}\ a_{12}\cdots a_{1n} \\ a_{21}\ a_{22}\cdots a_{2n} \\ \cdots\cdots\cdots\cdots \\ a_{n1}\ a_{n2}\cdots a_{nn} \end{bmatrix} \quad b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{11}$$

とすれば、方程式は、

$$Ax=b \tag{12}$$

と表すことができる。また、ＩＣＣＧ法では、Aの左下三角行列L、対角行列Dを

$$L=\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \quad D=\begin{bmatrix} d_1 & & & \\ & d_2 & & 0 \\ & & \ddots & \\ 0 & & & d_n \end{bmatrix} \tag{13}$$

とする。次に、第k近似解x_k、第k回の修正方向ベクトルp_k、第k回の残差ベクトルr_kを

$$x_k=\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix} \quad p_k=\begin{bmatrix} p_1^{(k)} \\ p_2^{(k)} \\ \vdots \\ p_n^{(k)} \end{bmatrix} \quad r_k=\begin{bmatrix} r_1^{(k)} \\ r_2^{(k)} \\ \vdots \\ r_{n(k)} \end{bmatrix} \tag{14}$$

として考える。

1) $A \cong LDL^T$ (15)

2) x_0を適当に選ぶ

3) $r_0=b-Ax_0$ (16)

4) $p_0=(LDL^T)^{-1}r_0$ (17)

5) $k=0$

6) $\alpha_k=\dfrac{(r_k,(LDL^T)^{-1}r_k)}{(p_k,Ap_k)}$ (18)

7) $x_{k+1}=x_k+\alpha_k p_k$ (19)

8) $r_{k+1}=r_k-\alpha_k Ap_k$ (20)

9) $\beta_k=\dfrac{(r_{k+1},(LDL^T)^{-1}r_{k+1})}{(r_k,(LDL^T)^{-1}r_k)}$ (21)

10) $p_{k+1}=(LDL^T)^{-1}r_{k+1}+\beta_k p_k$ (22)

11) r_kの収束判定をする

収束が十分ではないとき、$k=k+1$として6)に戻る

2.5 並列化

ＩＣＣＧ法は、大きく不完全コレスキー分解部（以下ＩＣ部）と共役傾斜法（以下ＣＧ

部）に分かれる。よってそれぞれ別々に説明をする。

2.5.1 IC 部の並列化

図１にＩＣ部のアルゴリズムを示す。

$k = 1,2,\cdots,n$

$i = 1,2,\cdots,k-1$

$$l_{ki} = a_{ki} - \sum_{j=1}^{i-1} l_{kj} l_{ij} d_j$$

continue i

$$d_k = (a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2 d_j)^{-1}$$

continue k

図 1　IC 部のアルゴリズム１

図１のl_{ki}を求める際にkを小さいものから順に計算していかなければならない依存関係が存在するため並列化を困難なものにしている。そこで、図１のアルゴリズムを図２のように変換する。

$j=1,2,\cdots,n$

$i=n,n-1,\cdots,j+1$

$$l_{ji} = a_{ji} - \sum_{k=i}^{n} l_{kj} l_{ij} d_j$$

continue i

$$d_j = (a_{jj} - \sum_{k=j+1}^{n} l_{kj}^2 d_k)^{-1}$$

continue j

図 2　IC 部のアルゴリズム２

このように変換する事により、kの依存関係をなくしてjのループを１回終了させる度にグループ通信のブロードキャストを実行して並列化を可能にした。この時、２次元配列 U の分割方法は、サイクリックに分割した。これは、各ノードの計算量を均等に近くするためである。

例として、n×n 行列 A に対して４ノードのプロセッサーを用いた場合を考えてみる。不完全コレスキー分解には、図 3 において上から順に計算していかなくてはならない依存

関係があるため、並列化が困難である。そこで次のような手法をとる。ここでは、n×n の行列 A に対して 4 個のノードを用いた例を示す。

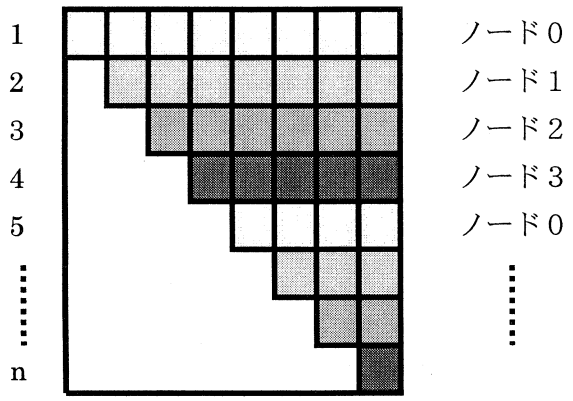


図 3　IC 部の 2 次元配列の分割

1. 図 3 のようにサイクリックにデータを分割する。
2. 1 行目をノード 0 において計算する。
3. 1 行目の計算結果を各ノードに送信する。
4. 各ノードは 2～n 行目の計算を 1 行目の結果を用いて可能な限り計算する。
5. 続いて 2 行目の計算結果をノード 1 から各ノードに送信して同様に可能な限り計算する。
6. 以下 3～n 行目についても同様に送信、計算を繰り返す。

2.5.2 CG 部の並列化

ＣＧ部の並列化に関しては、表 1 の反復部分のアルゴリズムから説明する。 ***A*** は実数の 2 次元配列、***x,p,r*** は実数の 1 次元配列であり、また、 α、β は実数の変数である。また、○、×は、各ノードで独立に計算可能か不可能かを表す。

表 1　ICCG 法におけるデータの並列性

アルゴリズム	A	x	p	r
1) $k = 0$				
2) $\alpha_k = \dfrac{(r_k,(LDL^T)^{-1}r_k)}{(p_k,Ap_k)}$				
3) $x_{k+1} = x_k + \alpha_k p_k$		○	×	
4) $r_{k+1} = r_k - \alpha_k A p_k$	○		○	○
5) $\beta_k = \dfrac{(r_{k+1},(LDL^T)^{-1}r_{k+1})}{(r_k,(LDL^T)^{-1}r_k)}$				
6) $p_{k+1} = (LDL^T)^{-1}r_{k+1} + \beta_k p_k$			○	×
7) r_{k+1}の収束判定をする。収束が十分ではないとき、$k = k+1$ として 2)に戻る				

この表 1 において、×の出現する個所に、全てのデータを全てのノードに集める処理を加えることで並列化は完成する。

2.6 結果

以上の方法で ICCG 法の並列化を実行して性能評価をした結果を図 4 に示す。

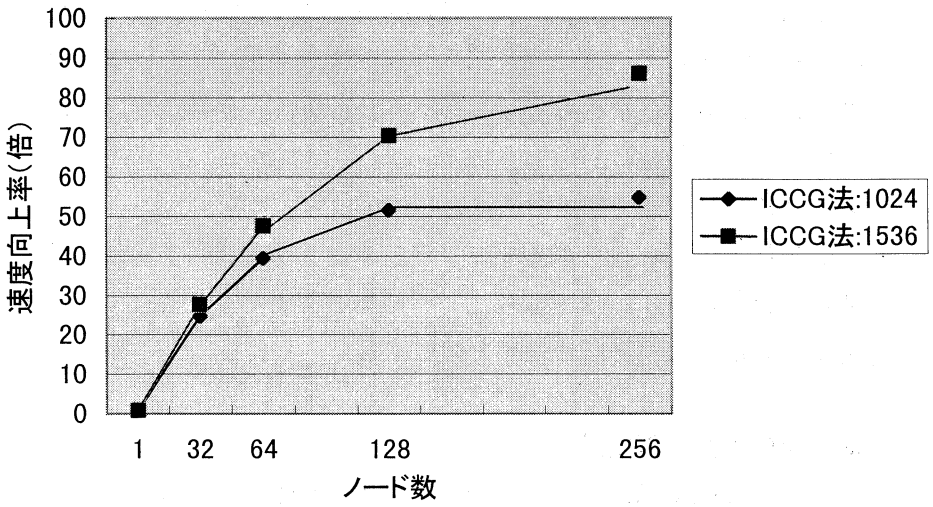
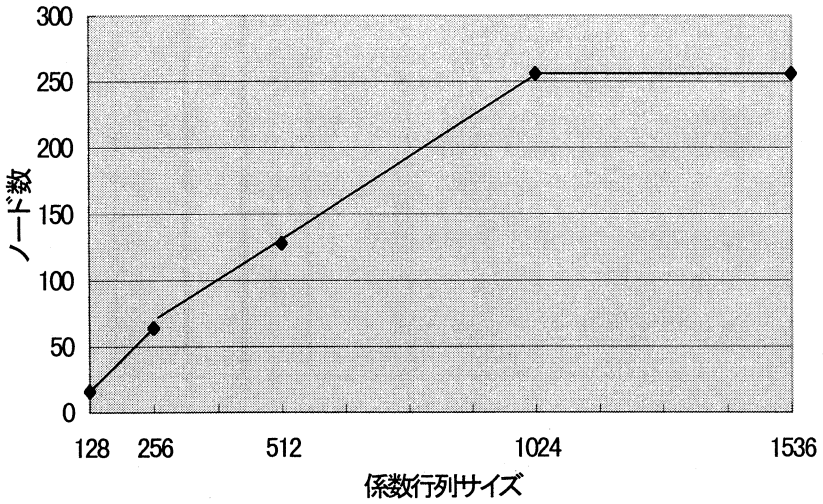


図 4　ノード数増加に伴う速度向上率の変化

図４において 1024 と 1536 は、係数行列のサイズを表している。ノード数の増加にともない理想値から離れていき、やがて飽和している。この飽和する点を図５にまとめてみた。図５より係数行列のサイズが大きくなるほど数多くのノードを有効に使えることが分かる。また逆に係数行列のサイズが小さければノード数を多くとっても逆効果であるといえるだ



ろう。

図 5　係数行列の増加による速度向上率の飽和点

3.むすび

以上の結果より ICCG 法のブロードキャストによる並列化は、ある程度の効果を示すことが分った。 しかしながら、 飽和点がみられるのは致し方のないことだといえる。 このように飽和していくのは、通信時間が主な原因である。通信時間は、ゼロにする事はできないが、少なくする方法はあるかもしれない。それには、次のような方法が考えられる。

- アルゴリズムを通信方法に応じて再検討する.

並列化不可能と見られるプログラムでも、アルゴリズムを変更したり、数式のモデル化を再検討したりする事で並列化できるものである。そこでもっと効率の良いプログラムを構築する事により通信の処理を減らす。

- 通信手段の再検討

今回の実験では、グループ通信をメインに使っている。それを１対１通信（特に非同期通信）に変更してみる。１対１通信の方が通信している間の処理や同期をとって待っている間の処理を細かく設定できる。

● データの分割方法の再検討

データの分割は、各ノードにデータを均等に分ける他にデータの連続性をなるべく維持するという目的をもっている。データを分割することは、データの連続性を維持することと相反した事柄である。この矛盾の妥協点を探すのも重要である。

4.参考文献

[1]笠原博徳「並列処理技術」　株式会社コロナ社 1991,6,20

[2]水上孝一「コンピュータによる数値計算」　株式会社朝倉書店 1985,4,15

[3]戸川隼人「共役勾配法」　教育出版株式会社 1977,7,20

並列ハイブリッド遺伝的アルゴリズム

成久洋之　　平林永行*　　片山謙吾**

岡山理科大学工学部情報工学科

*岡山理科大学大学院工学研究科修士課程情報工学専攻

**岡山理科大学大学院工学研究科博士課程システム科学専攻

(1998年03月27日　受理)

概　要

本研究は，巡回セールスマン問題（TSP）に対する並列ハイブリッド遺伝的アルゴリズム（並列ハイブリッドGA）の計算時間の短縮による加速率および効率性について検討する．具体的な効率性評価のために三種類の優れた交叉法を取り上げ，各ハイブリッドGAの場合を検討する．これらの逐次処理と複数のプロセッサによる並列化を考慮する場合に，どの程度の効率化が可能か，また交叉法の違いにより並列化の効率性にどのような影響を与えるかなどについて検討する．

1　はじめに

組合せ最適化問題の多くはNP-完全なクラスに属し，大規模な問題に対して多項式時間で最適解を得ることは不可能であるとされている．しかしながら，理論的に最適解を多項式時間で求める解法が存在しなくても，経験的に比較的良好な解を求めるアルゴリズムは数多く提案されている．その一つとして，生物の進化過程にヒントを得た遺伝的アルゴリズム（genetic algorithm, GA）[1, 2, 3]が近年注目を集めている．GAは確率的探索原理により，さまざまな組合せ最適化問題や応用問題[4]に適用され，その成果を上げつつある探索アルゴリズムの一つである．一般的にGAは，遺伝的操作（選択，交叉，突然変異）を持ち，解空間に対して大域的な探索が非常に優れ，良好な解を算出できる解法である．

強力なロバスト性を誇るGAは，扱う問題に対して合致した構造を取りやすく，アルゴリズムの詳細な部分にわたり，いろいろな工夫を施すことで更に高性能なGAを開発することが可能である．そのようなGAアプローチの一つとして，ハイブリッドGA（Hybrid GA, HGA）の研究が数多く行われている[5, 6, 8]．HGAは探索効率の向上を図り，高精度な解を得る目的で，局所的な探索に優れるLocal Search（局所探索法）を組み込んだ，GAの拡張版である．

並列処理は逐次処理の操作手順を分割することによって効率よく処理を行い，逐次処理で得られる解を劣化させることなく計算時間の短縮を計るものである．一般的にGAは複

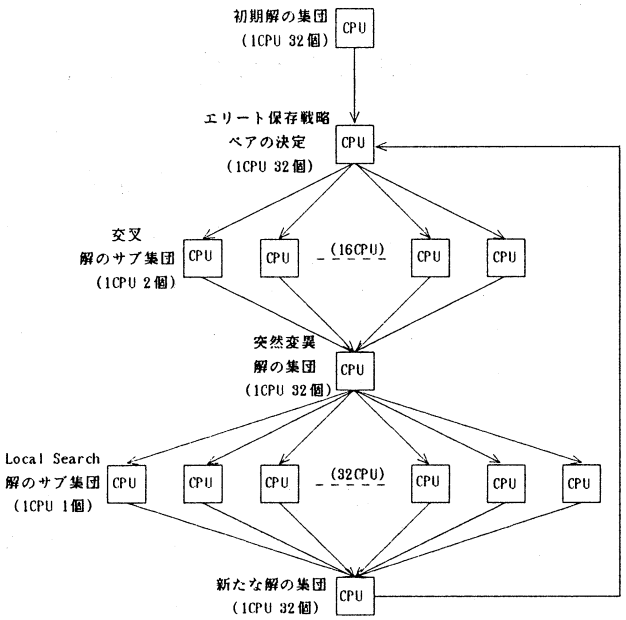


図 1: 32CPUの処理手順

数の候補解を常に保持しているので，逐次処理においてはかなりの計算時間を必要とし，更にLocal Search をハイブリッドすることにより膨大な計算時間を必要とする場合が少なくない．よって本研究の検討事項として，HGA で保持される候補解の集団を並列化することにより，どの程度の計算時間の短縮が期待できるか，並列計算機を使用して，その加速率及び効率性を示す．

その効率性評価の具体例として，巡回セールスマン問題（traveling salesman problem, TSP）に対して現在有望とされるMühlenbein らが提案したMaximal Preservative Crossover (MPX)[5]，Starkweather らの分析によって優れた交叉法とされる，改良されたEdge Recombination Crossover(Improved ERX, IERX) [7] と，片山らの提案した新しい交叉法，Complete Subtour Exchange Crossover(CSEX) [9] をHGA に施した場合を検討する．片山らは，交叉性能を評価するHGA の枠組みを提案している[10]．従って本研究ではその枠組みを利用し，HGA で保持される候補解の集団をサブ集団に分割した並列化による，各交叉法のアプローチの違いによって生じる効率性の相違を検討する．

2 並列ハイブリッド遺伝的アルゴリズム

並列ハイブリッド遺伝的アルゴリズム（Parallel Hybrid GA, PHGA）について記述する．並列化に際しては，GA で用いるすべての候補解（個体）の集団をいくつかのサブ集団で構成する方法などさまざま考えられる[2, 3, 5, 6]．本研究のPHGA で並列化の対象とするのは，主として交叉とハイブリッドされたLocal Search の処理部分であり，分割されたサブ集団に対して各世代ごと行われる．つまり，対象となる個体集団に対して，複数個のCPU で並列化する場合，各CPU は，$\frac{\text{個体集団}}{CPU\text{数}}$個の個体で基本的に構成される．

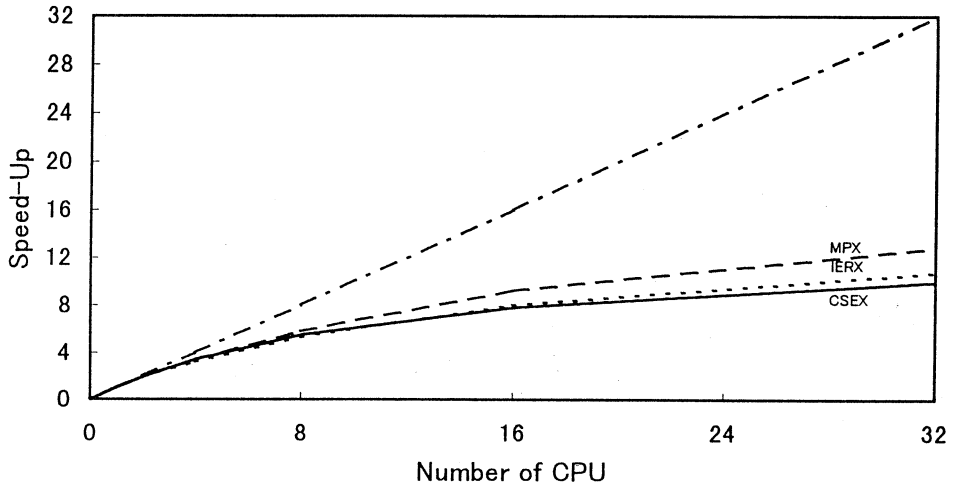


図 2: CPU 数と加速率の関係（kroA100）

以下，並列化に関する実験の詳細について記述する．実験では並列計算機を使用し，CPU 数が最大32までの場合を検討する（なお，処理中に保持されるGAの全個体数は32）．PHGA で並列化の対象となるのは，交叉とハイブリッドされたLocal Searchの処理部分であるので，1CPUに割り当てられる個体数は，CPU数 1，2，4，8，16，32 に対して，それぞれ 32(16)，16(8)，8(4)，4(2)，2(1)，1(1) 個の個体がLocal Searchに適用される．( )内の数値は交叉を適用する際に1CPU に割り当てられるペアー数を示す．つまり，交叉を適用する際には最低でも二つの個体が必要となるので，GA で保持される全個体数に対しては，$\frac{個体数}{2}$ 個のペアーで構成される．よって，CPU数16，32 で，そのペアー数は1になることに注意されたい．なお，図1は，32個のCPUを用いて並列化を行った場合の処理手順を表す．本 PHGA では，交叉の部分，Local Searchの部分を並列化しており，それぞれで最後に処理が終了したCPUに合わせて同期を取っている．

3 むすび

本論文は，巡回セールスマン問題（TSP）に対する並列ハイブリッド遺伝的アルゴリズム（PHGA）について検討した．TSPLIBからkroA100（100都市問題）を用いたPHGAの実験では，逐次処理で算出された解の精度を劣化することなく並列化したことを確認し，並列化による計算時間の短縮が逐次処理に比べ，32個の CPU を使用した我々の並列化実装形態においては，最悪でも約 1/10 の時間で算出可能であることを示した．更に，基本的な枠組みを用いて，TSP に対して三種類の交叉法を用いた並列化における効率性の相違を比較した．そこでは，図2，3ように各交叉法のアプローチの相違によって，並列化に対して加速率や計算時間に大きな違いが表れることなどを示した．

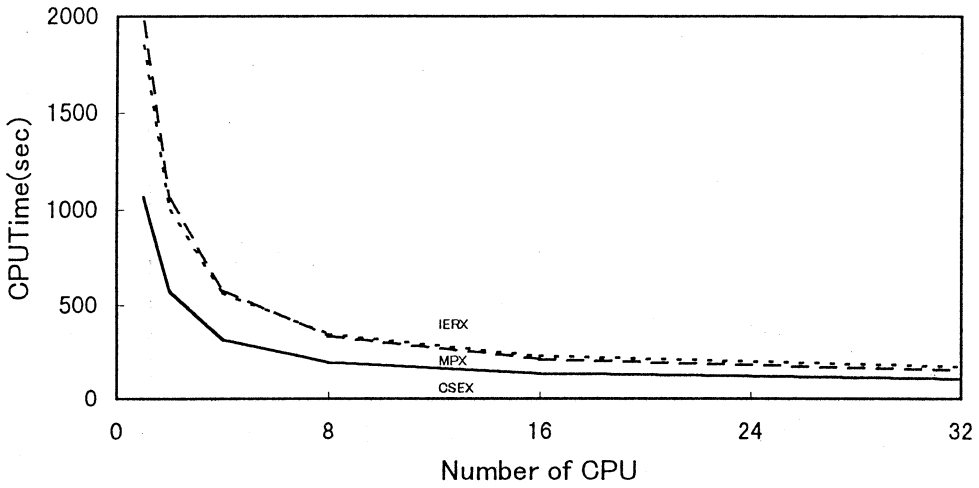


図 3: CPU 数と計算時間の関係（kroA100）

参考文献

[1] D.E. Goldberg, "Genetic algorithms in search Optimization and machine learning," Addison-Wesley Publishing Company Inc., 1989.

[2] 北野宏明 編, "遺伝的アルゴリズム," 産業図書, pp.3-60, 1993.

[3] 樋口哲也, 北野宏明, "遺伝的アルゴリズムとその応用," 情処学論, vol.34, no.7, pp.871-883, 1993.

[4] L. Chambers, "Practical handbook of genetic algorithms applications," CRC Press Inc., vol.1, pp.143-172, 1995.

[5] H. Mühlenbein, M. Gorges-Schleuter and O. Krämer, "Evolution algorithms in combinatorial optimization," Parallel Computing 7, North-Holland, pp.65-85, 1988.

[6] H. Mühlenbein, "Parallel genetic algorithms in combinatorial optimization," Computer Science and Operations Research, Pergamon Press, pp.441-453, 1992.

[7] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley and C. Whitley, "A comparison of genetic sequencing operators," Proc. 4th ICGA, pp.69–76, 1991.

[8] K. Katayama, H. Sakamoto, and H. Narihisa, "An efficiency of hybrid mutation genetic algorithm for traveling salesman problem," Proc. 2nd Australia-Japan Workshop on Stochastic Models in Engineering, Technique & Management, Gold Coast, Australia, pp.294-301, 1996.

[9] 片山謙吾, 成久洋之, "完全サブツアー交換交叉の共有サブツアー高速列挙アルゴリズムと動作特性," 信学論(D-I), vol.J81-D-I, no.2, pp.213–218, 1998.

[10] 片山謙吾, 平林永行, 成久洋之, "TSP に対する並列ハイブリッド遺伝的アルゴリズムの一検討," 信学技報, COMP97-31, pp.17-24, July, 1997.

Island型並列遺伝的アルゴリズム

片山謙吾　　平林永行*　　池田早人**　　成久洋之**

岡山理科大学大学院工学研究科博士課程システム科学専攻

*岡山理科大学大学院工学研究科修士課程情報工学専攻

**岡山理科大学工学部情報工学科

(1998年03月27日　受理)

概　要

本研究は，巡回セールスマン問題（TSP）に対し遺伝的アルゴリズム（GA）の並列化を考慮することにより計算時間の短縮，加速率および効率性についてIsland型並列遺伝的アルゴリズム（IGA）を用いて検討する．また，逐次処理と複数のプロセッサによる並列化を考慮する場合に，どの程度の効率化が可能か，片山らの提案した新しい交叉法，Complete Subtour Exchange Crossover(CSEX)を用いたIGAについて検討する．

1　はじめに

組合せ最適化問題の多くは$\mathcal{NP}$-完全なクラスに属し，大規模な問題に対して多項式時間で最適解を得ることは不可能とされている．そのような問題を解く場合の解法として，あくまでも最適解を算出する厳密解法と，比較的短時間に実用上許容される近似解を算出する解法に大別される．特に後者に属するものには，さまざまな組合せ最適化問題に適用可能な多くの優れた解法が提案されている．その一つに，生物の進化過程をヒントに考案された遺伝的アルゴリズム（genetic algorithm, GA）[1]が近年注目を集めている．一般にGAは，遺伝的操作（選択，交叉，突然変異）を持ち，解空間に対して大域的な探索に優れ，良好な解を算出できる近似解法である．特にGAの大きな特徴の一つである交叉は，組合せ最適化問題を代表する巡回セールスマン問題（traveling salesman problem, TSP）に対して数々の方法が提案され，その有効性について多くの報告がある[2, 3, 4, 5, 7, 8]．

我々が以前提案した完全サブツアー交換交叉（complete subtour exchange crossover, CSEX）[5]は，二つの親に共通して含まれる全く同じ方向性を有した共通区間（またはサブツアーと呼ぶ）を列挙し，そのサブツアーを利用して子孫を生成する交叉法である．我々は，TSPに対して有望とされる強力な交叉法として，国際的に有名なMühlenbeinらの提案したMaximal Preservative Crossover（MPX）[4]，Starkweatherらの提案したEdge Recombination Crossover [7]の改良版（IERX）[8]，および我々のCSEXとの比較検討を，

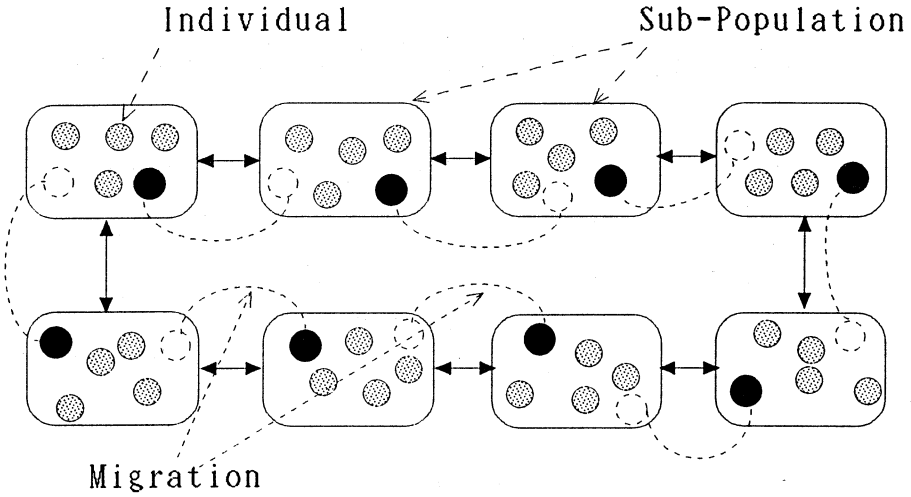


図 1: リング型の近接構造を持つIsland GA

2-Opt近傍の局所探索法を有したハイブリッドGAの枠組みのもとで行い，CSEXが比較的優れた交叉法であることを報告した[6].

GAの研究においては，これら遺伝的操作の交叉法等の開発のみならず，並列処理への実装の容易さから並列GAの開発も活発に行われており，さまざまな並列モデルが提案されている．その代表的なモデルとしては，生物学における棲み分けと分化（niche and speciation）の現象を土台に考案されたIsland GAや，空間性の概念を取り入れたCellular GA等があり，これらの改良版などさまざまである[9, 10]．本論文では，最も基本的な近接構造を有するIsland GA（IGA）を取り上げ，TSPに対するCSEXを組み込んだIGAの有効性および効率性を，並列計算機 Paragonを用いて検討する．

2 Island型並列遺伝的アルゴリズム

Island GA [9, 10]（IGA）は，生物学における棲み分けと分化のアイデアから考案された．つまり，Island（島）は，地理的に独立した環境が与えられた典型的な一例であり，GAにおいては，そのような複数のサブ集団を構成することで個々の集団の独立的な進化をねらったモデルである．一般にそれらのサブ集団は，ある近接構造（topology）で定義され，同一の適応度関数を有している．また各サブ集団で進化した最良の個体を定期的に他のサブ集団へ移住（migration）を行い，GA探索における集団の多様化を維持する戦略がとられる．

一般にIslandモデルの近接構造にはさまざまあるが，本研究では，与えられた集団を図1のように分割する簡潔なモデルを検討する．これは，複数個の個体で構成されたサブ集団がリング型の近接構造を有し，各サブ集団で遺伝的操作を繰り返し，ある定められた時間

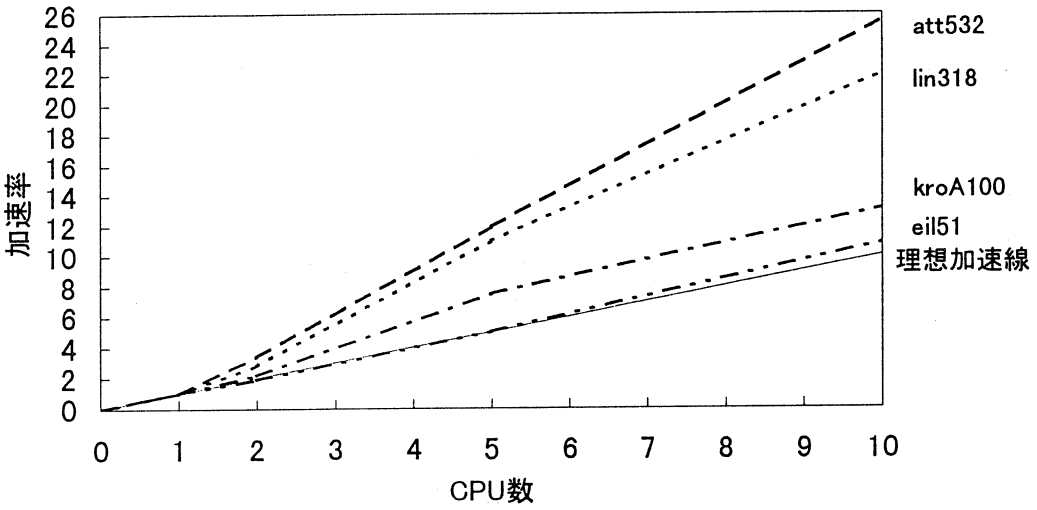


図 2: CPU 数と加速率の関係

(世代) に達したら一つの優れた個体を強制的に隣のサブ集団へ移住させるモデルである. なお, このモデルはGA における並列化の最も基本的なものと考えられており, 交叉法による効率性などが顕著に現れると考え, このモデルを取り上げた. 以下, その手順を示す.

Step1 与えられたPSをSub個に等分割し, Sub個のサブ集団をリング型に構成する.

Step2 各サブ集団で$MigInterval$世代の間それぞれ進化させる.

Step3 $MigInterval$に達したら, 各サブ集団の最良個体を隣のサブ集団へ移住させる.

Step4 任意の世代に到達したら処理を終了. 到達してない場合はStep2 からを繰り返す.

なお, PSは与えられたすべての個体数, Subは並列処理するサブ集団の数を指す. 従って, 各サブ集団の個体数$SubPS$はPS/Sub個で構成される. また$MigInterval$は個体を移住させる世代数の間隔を指す.

3 むすび

本論文は, 巡回セールスマン問題 (TSP) に対する並列遺伝的アルゴリズムの検討をIsland モデルを用いて行った. 以前我々の提案した完全サブツアー交換交叉 (CSEX) をIsland モデルに用いた場合, 図2のように加速率は理想値を上まわる極めて優れた効率化および高速化が実現可能であることを示した. 更に図3より, 単純GA (Simple GA, SGA) の算出結果よりもIGA で算出された解の質の方が優れていることを示している. これは, CSEX が二つの親に共通して含まれるサブツアーを利用することで, そのサブツアー数に対応した, 最良子孫の抽出時間とサブ集団内における個体の類似性との関係が多大に影響するものと考えられる.

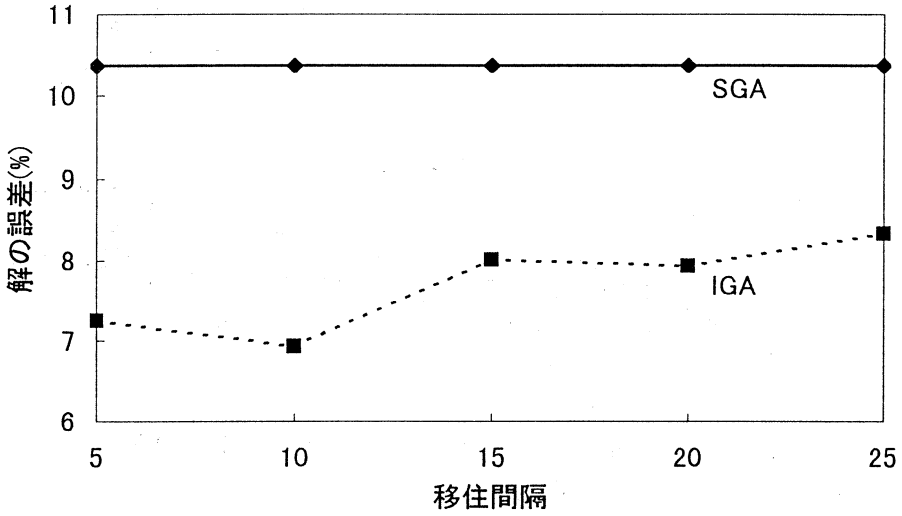


図 3: 移住間隔と解の誤差との関係（att532）

参考文献

[1] D.E. Goldberg, "Genetic algorithms in search, optimization and machine learning," Addison-Wesley Publishing Company Inc., 1989.

[2] R.M. Brady, "Optimization strategies gleaned from biological evolution," Nature, vol.317, pp.804–806, 1985.

[3] 山村雅幸, 小野貴久, 小林重信, "形質の遺伝を重視した遺伝的アルゴリズムに基づく巡回セールスマン問題の解法," 人工知能誌, vol.7, no.6, pp.1049–1059, 1992.

[4] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolution algorithms in combinatorial optimization," Parallel Computing 7, North-Holland, pp.65–85, 1988.

[5] 片山謙吾, 成久洋之, "完全サブツアー交換交叉の共有サブツアー高速列挙アルゴリズムと動作特性," 信学論（D-I）, vol.J81-D-I, no.2, pp.213-218, 1998.

[6] K. Katayama, H. Hirabayashi, and H. Narihisa, "Performance analysis of a new genetic crossover for the traveling salesman problem," IEICE Trans. Fundamentals, 1998.（5月号, 掲載予定）

[7] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesman: The genetic edge recombination operator," Proc. 3rd ICGA, pp.133–140, 1989.

[8] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A comparison of genetic sequencing operators," Proc. 4th ICGA, pp.69–76, 1991.

[9] J.P. Cohoon, S.U. Hegde, W.N. Martin, and D. Richards, "Punctuated equilibria: A parallel genetic algorithm," Proc. 2nd ICGA, pp.148–154, 1987.

[10] R. Tanese, "Parallel genetic algorithm for a hypercube," Proc. 2nd ICGA, pp.177–183, 1987.

計算幾何学における大量分類問題(II)

佐藤吉将　　岡 倫弘　　島田英之　　宮垣嘉也

1 概要

有限な2次元正方格子上の三点形の合同類の数に関して誘導された上界式の近似度を調べるために，計算機により真値を知る必要があるが，通常の総当たり方式では40×40の格子サイズが時間的限界であった。以前の研究では並列計算機Paragonにより並列化を行い，合わせて，通常の方式とは異なるアルゴリズムを使用することで200×200の格子サイズまで計算することができた[2]。今回は後に述べるミンコウスキの距離において，ある正の整数p（以下次数と呼ぶ）が3および4の場合における真値を求めることが目的である。

2 予備知識

図1 に示すような有限な2次元正方格子(m,n) $(m \geq n \geq 1)$上の拡張された三角形（以下，三点形と呼ぶ）を合同類別する。

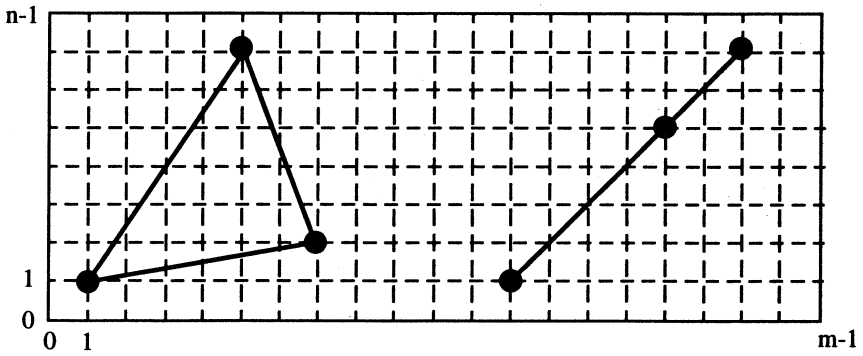


図1：有限な2次元正方格子と三点形

2点$(x_1,y_1),(x_2,y_2)$間の距離を，ミンコウスキの距離と呼ばれる以下の式(1)により定義する。

$$(|x_1-x_2|^p+|y_1-y_2|^p)^{1/p} \quad (p=1,2,3,\cdots) \tag{1}$$

ここで，対応する3辺の長さの相等によって合同を定義する時，相異なる三点形が幾つあるかを考える。その数を$N_t(m,n \mid p)$とすると，これに対するpによらない理論的な上界値（以下理論値と呼ぶ）$N(m,n)$が誘導されている[1]。

3 研究内容

本研究の目的は，先の理論値の信頼性を確かめるために，数値計算により真値を求め，比較することにある。前回では次数$p=2$の場合のみ計算を行ったため，今回は次数$p=3,4$の場合の計算を行う。そのためのアルゴリズムは次のようになる。

1. まず，長さのp乗値$L(=1,2,3,\cdots)$の辺が格子系の内部に存在できるか調べる。具体的には以下の式を満たすことのできる格子系の座標x,yがあるかどうかを調べる。

$$L=x^p+y^p \tag{2}$$

式(2)を満足した時，Lとそれを構成できる(x,y)の組み合わせを全て配列に保存する。

2. Lの配列から辺を3つ取り出し，これをa,b,cとする。ただし，$a \leq b \leq c$とする。

3. 3つの辺により三点形が構成できるかどうか調べる。具体的には3つの内2つの辺の座標より2点間の距離のp乗値を求め，それが残る1辺の長さのp乗値と等しければ，その3点は三点形を構成できると判断する。

なお，辺の長さのp乗値を使う理由としては，通常の辺の長さでは長さが小数で表されるため誤差が生じる危険性があり，比較の際に誤った計算を行ってしまう可能性があるからである。

4 実行結果

本研究での計算結果と計算時間を表1 に示す。なお，サイズ100 , $p=3$ の計算はParagonにおいて150ノードを用いて計算し，それ以外についてはPentium 166MHzの計算機1台により計算したものである。

表1：$p=3$ならびに$p=4$における計算結果と所要時間

サイズ	$p=3$		$p=4$	
	計算結果	所要時間（秒）	計算結果	所要時間（秒）
10	1,581	0.068	1,581	0.058
20	27,347	3.260	27,361	2.019
30	142,561	34.652	142,591	21.690
40	457,453	210.613	457,521	118.481
50	1,127,288	708.257	1,127,401	523.935
100	18,384,352	10,124	-	-

また，各格子サイズにおける$N_t(m,n)$と$N(m,n)$，そしてその相対誤差εを表2 に示す。ここで，誤差εは次式で計算されるものとする。

$$\varepsilon = \frac{N(m,n) - N_t(m,n \mid p)}{N_t(m,n \mid p)} \times 100 \quad (\%) \tag{3}$$

表2：真値と上界値との誤差

サイズ	上界値 $N(m,n)$	$p=2$		$p=3$		$p=4$	
		真値	ε(%)	真値	ε(%)	真値	ε(%)
10	1,581	1,544	2.40	1,581	0.0	1,581	0.0
20	27,361	26,439	3.49	27,347	0.05	27,361	0.0
30	142,591	137,380	3.79	142,561	0.02	142,591	0.0
40	457,521	440,285	3.91	457,453	0.01	457,521	0.0
50	1,127,401	1,084,444	3.96	1,127,288	0.01	1,127,401	0.0
100	18,384,801	17,666,719	4.06	18,384,352	0.002	-	-
200	297,039,601	285,362,372	4.09	-	-	-	-

5 まとめ

本研究では三点形の合同類別の計算において，ミンコウスキの距離の次数pが3,4の場合の計算を行った。理論値と実際に計算した結果を比較してみると，次数pが大きくなるほど，誤差が小さくなっていくことが確かめられた。

計算時間は$p=3$と$p=4$で幾らかの差が見られた。アルゴリズムの理論から計算時間は変わらないと考えていたので，なぜこのような結果になったか考察する必要がある。現段階ではアルゴリズムの実装方法によるものか，それとも数学的な理由によるものかは判断できていない。

より大きなサイズでの計算を行おうとしたが，アルゴリズムの実装方法に問題があり，これ以上の計算が時間的に不可能であった。この問題点を解消できれば計算可能であると思われる。

6 課題

今回のアルゴリズムはまだ問題点を抱えているため，それを修正し，並列計算機上で実行させることが当面の課題である。また，次数の違いによる計算時間の違いの原因についても追究したいと考えている。

本来，真値が必要とされている格子サイズは10000×10000であり，本アルゴリズムを用いてもメモリ容量，速度の観点からまだ実現はできそうにない。今後の課題としては，より高速であり，かつメモリ消費量の少ないアルゴリズムを考案する必要があると考えられる。

参考文献

[1] 宮垣,島田：“格子上の三点形の合同類別に関する一考察”, 岡山理科大学紀要, 32-A, pp.143–151(1997-03).

[2] 宮垣,島田,佐藤,岡：“計算幾何学における大量分類問題”, 岡山理科大学情報処理センター研究報告 18号, pp.39–43(1997).

自動並列化コンパイラの研究

工学部 情報工学科　橘井 邦夫, 小畑 正貴

大規模な数値計算では, 並列化による高速化が期待されている. しかし, 並列計算機向きのプログラムの記述は, 並列可能部分の認識や通信関数の導入などの困難があるため, 現状ではユーザには使いにくい.

このため, 逐次言語で記述されたプログラムを自動的に並列化するコンパイラの実現が求められている.

本研究では, 並列化を自動で行なうコンパイラのプロトタイプを開発し, 一般的な数値解析のプログラムで評価を行ない問題点を洗い出し, それに対するコンパイラの設計の変更をおこない, 現状では新規の開発を行ないつつある.

コンパイルの対象となるプログラムはC言語で逐次形式で書かれたプログラムであり, 並列化コンパイラに対しての並列化に関する特殊な命令を含んでいないものである. また, 出力ファイルは, プログラムの組変えと並列計算機でのメッセージ通信の標準ライブラリであるMPI(Message-Passing Interface) を用いた専用ライブラリを加えて並列化を行なったものである.

1　はじめに

次世代の高性能計算機として分散メモリ型の並列計算機に大きな期待が寄せられているが, 一般に分散メモリ型の並列計算機に対して, 従来のプログラム言語と通信ライブラリを用いたプログラミングを行なうと, 各ノードにプログラムの仕事を割り当てるための処理, プログラムの実行されているノードに, そのプログラムが必要としているデータが存在しない場合のデータの転送処理, 等の逐次プログラムには必要の無い処理を書き足す必要が生じるためにプログラムがどうしても繁雑になる. これらの理由によって, 現在の手法では高速なプログラムを書くためには, ある程度の技術が必要になってしまうので, 逐次計算機向きに記述されたプログラムに対して, できるだけ変更を加えずに並列計算機上で効率の良い実行結果を得られるようにする環境の実現が求められている.

本研究では, 逐次計算機向きに記述されたプログラムを自動で並列化するコンパイラのプロトタイプの開発とその評価[(1)] を, 一般的な数値解析のプログラムで行なう. また新規開発中であるコンパイラについての概要を述べる.

この並列化コンパイラはC言語で書かれたプログラムを対象とし, 出力ファイルは専用の通信ライブラリを使用するC言語のプログラムの形である. これによって, ユーザーは今までの逐次のプログラムをそのまま並列計算機上で最大限の効率を発揮できる形式に変換でき, 対象となる並列計算機上の通常のコンパイラで再コンパイルすることで最大限の実行効率を得ることができるよう

になる可能性がある. また専用ライブラリ内では標準的なメッセージ通信ライブラリである MPI (Message-Passing Interface)[2] [3] を使用している.

MPI を用いることによって現在の評価に用いている Paragon のみならず, 他の様々な並列計算機でも効率良く実行できる可搬性のある並列プログラムを生成することを目標としている.

本研究では並列性が一番わかりやすい形で現れることが経験的にわかっているループ部分の解析をし, 並列化を自動で行なうC言語を対象とした並列化コンパイラを作成し Paragon(Intel 社. 分散メモリ型並列計算機) 上で評価を行なう.

本稿では, まず 2 章では本研究で開発中の並列化コンパイラの概要についてで, これはプロトタイプの結果を踏まえて修正したものである, 3 章ではプロトタイプの並列化コンパイラによる一般的な数値解析のプログラムの並列化の効果と有効性の評価, 4 章でまとめと今後の課題について述べる.

2 並列化コンパイラの概要

2.1 並列化コンパイラの目的

本研究で開発中の並列化コンパイラは, 以下のような戦略に基づいている.

(1) 並列化が容易で効率的に実行できることがわかっているループ部分を並列化する.
(2) 並列化が容易ではなく, 並列化を行なっても効率の向上が得られにくい場合には並列化を断念する.
(3) 並列化コンパイラはユーザーから並列化に関する情報をコンパイルオプションを除き全く求めない.
(4) 出力結果を実行環境に依存しないようにする.

以上よりユーザーが気をつけることは

(1) プログラムを書く上で並列化に適したアルゴリズムを採用する.
(2) ユーザーは並列化しやすいようにプログラムを書く.

つまりユーザは現在のベクトル化機能のあるコンパイラと同じようにコンパイラを使用すればよいようにすることを目的にしている.

2.2 並列化コンパイラの構造

本研究で開発された並列化コンパイラは, 複数の構成要素に分解できる.

(1) C言語から中間言語へのコンパイラ
(2) 中間言語の並列化トランスレーター

(3) 並列化された中間言語の通信命令のオプティマイザ

(4) 中間言語からC言語への逆コンパイラ

以下にそれぞれの機能の説明と, 用語の説明を行なう.

- 中間言語

  アセンブリ言語に近い形式で, 解析の簡略化のために使われる. レジスタは無くスタックを中心とした形式で実装されている. C言語の要素を全て分解できるできるようにしている. 並列化処理後のC言語への再変換もできるように構成している.

- 中間言語の並列化トランスレーター

  依存関係の調査による定型を持つ命令の抽出, 出力の依存関係の調査とそれを用いたループの分割, 転送命令の前後関係から生成される依存関係を解決する命令の挿入, の三段階に分かれ, それぞれが独立したプログラムである.

  中間言語を読み込み, 並列化された中間言語を生成する. ループの並列化を主な処理内容にしている.

- 並列化された中間言語の通信命令のオプティマイザ

  並列化された中間言語を読み込み, 挿入された通信命令の内で余分な命令の削除と, データの分割の最適化とそれに伴う命令の挿入を行なう. これによって最終的なコンパイル結果が得られる.

- 並列化ライブラリ

  並列化のための情報を得るための関数と, 幾つかの入力出力関数を, 無駄な動作をしないように並列化, もしくは, 選択動作できるようにしたもの. 並列化コンパイラで自動的に使用される. また並列化コンパイラ内での変換の簡略化のために用いる. 出力ファイルがC言語であることを利用して, 主にマクロ命令として定義された命令で構成され, 所有ノードや実行ノードの判断を行なっている.

2.3 並列化コンパイラの仕様要求

本研究で開発した並列化コンパイラは以下のルールにしたがっている.

(1) 並列化部分はループ部分だけである

(2) 並列化のネストはしない

(3) ループの構造をできるだけそのまま利用する

(4) ソースに加える変更をできるだけ少なくする

(5) 関数は main 関数内に inline 展開される.

2.4 並列化コンパイラの動作

本研究で開発した並列化コンパイラでは中間言語に対して並列化に関する処理を行なう．具体的には以下の順序 (図１参照) で処理を行なう．

- ループの反復回数の検出
  C言語では文法上, ループの反復回数は明示されないし, その終了条件は自由に決めることができる．このため, 並列化コンパイラではループ分割するためにループの反復回数を検出しなくてはならない．

- ループが並列化可能かどうかの判定
  ループの中には並列化して効果を得ることができないものが多い．このために依存解析を行なって, 並列化効果の無いであろうループは並列化しないようにしている．

- データ分割から見たループ分割の最適化
  並列化可能なループが複数ある時, それぞれの並列化の組合せで生じるデータ転送の負荷とその性格について調査して最善の構成を選択する．

- 転送命令の付加
  並列化ループ内で生成されるデータを他のループへ転送するための命令を挿入する．またこのための前処理と後処理の命令もそれぞれ挿入する．並列化ライブラリでは多数の小さな通信データをまとめて一つの長い通信データとして扱うことによって, 通信命令の起動回数を減らして効率化を行なっている．このためにまとめるためのメモリを確保する必要がある．また転送時に同時に演算を行なう必要がある場合には特別な命令を付加して効率を向上している．

- 転送命令の最適化
  転送命令の付加の際に用いている手法は単純であり, この時点でプログラムの前後関係を解析し最適な処理しているわけではない．このため無駄な転送が生じるので, この削除を行なう．

- 並列化に関する後処理
  本研究で開発した並列化コンパイラでは前もってinline展開されている関数しか並列化しない．またこのinline展開によって関数自体の大きさが膨れ上がるので, 得られたプログラム列からの関数の生成によるプログラムの圧縮が必要になる．

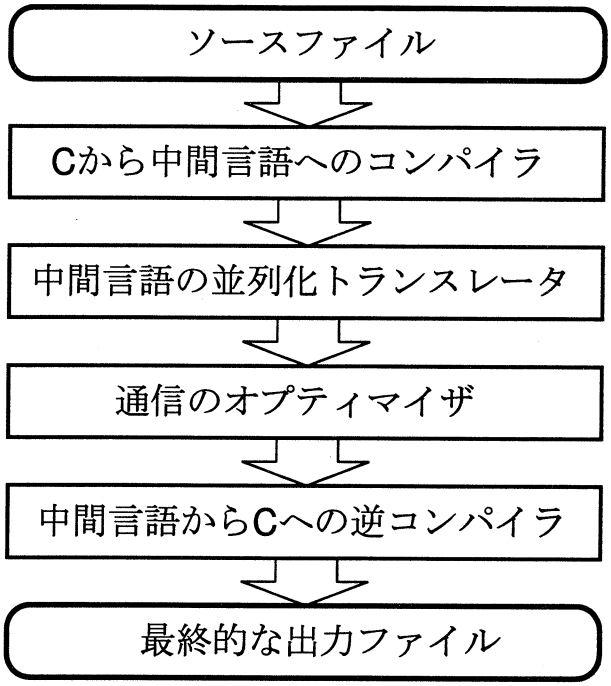


図 1: 並列化コンパイラの動作順序

3 数値処理プログラムによる評価

3.1 評価目的

本研究で開発した並列化コンパイラを用いて一般的な数値解析のプログラム[4] [5] [6] を実際に自動並列化したプログラムと並列化を行なわないプログラムをそれぞれ実行し, 並列化によって得られた効率を算出した.

分散メモリ型並列計算機である Paragon(Intel 社) 上で評価を行なっているが最終的には MPI の動作する全ての計算機で効率良く実行できるようにすることを目標としている.

一般的な数値解析のプログラムとして, 行列積, Gauss-Seidel 法, 共役傾斜法, クラウト法のプログラムを選んだ. ただし, これらのプログラムには並列化コンパイラがまだ対応していない命令 (プリプロセッサ命令,typedef,struct,union 他) を置き換える処理を行ない, 時間計測のために MPI の命令を付加している.

3.2 結果

逐次プログラムを自動で並列化するコンパイラのプロトタイプを開発し, 数値計算プログラムを用いて評価した.

行列積	: 16 台で 15.9 倍の台数効果を得た
共役傾斜法	: 16 台で 8.96 倍の台数効果を得た
Gauss-Seidel 法	: 7 台で 4.29 倍まで台数効果を得たが以降は低下
SOR 法	: 16 台で 3.22 倍の台数効果を得た
クラウト法	: 台数効果は得られなかった

4 むすび

主にループ部分を検出し解析し自動で MPI を用いた並列化をし, 分散メモリ型並列計算機を対象とする, 自動並列化コンパイラのプロトタイプを開発した.

数値計算の評価によって, 並列化に向いているような計算ではそれなりの実行効率を得ることができたがデータの転送が演算に比べて多いようなプログラムの並列化においては, ノード数を増やせば増やすほど実行効率が低下することがわかった.

データ通信の最適化を施すだけでは, データの転送の大小が大きくプログラムの実行速度に影響することがわかったので, データの転送量を少なくするための新たな機構として, 次の開発ではループブロック間のデータ転送を考慮して並列化を行なう機構をコンパイラに搭載することを決定し開発中である.

MPI を用いることによって様々な並列計算機で効率良く実行できる可搬性のある並列プログラムを生成することを目標としているので他の計算機上での実行効率の計測を行いたい.

5 参考文献

(1) 橋井邦夫, 小畑正貴,MPI を対象とする自動並列化コンパイラ, 情報処理学会 ハイパフォーマンス・コンピューティング研究会 68-5(1997.10.17) ,1997

(2) 青山 幸也,RS6000 SP 並列プログラミング虎の巻 MPI版, 日本アイ・ビー・エム株式会社,1996

(3) MPI フォーラム MPI 日本語訳プロジェクト訳,MPI：メッセージ通信インターフェース標準（日本語訳ドラフト）,MPI フォーラム,1996

(4) 奥村 晴彦, C言語による最新アルゴリズム事典, 技術評論社,1991

(5) 戸川 隼人,UNIX ワークステーションによる科学技術計算ハンドブック 基礎編 C言語版, サイエンス社,1992

(6) 水上 孝一, 市山 寿夫, 野田 松太郎, 南原 英生, 渡辺 敏正 共著, コンピュータによる数値計算, 朝倉出版,1985

自動並列化コンパイラの研究

工学研究科・情報工学専攻　　橋井 邦夫

大規模な数値計算では, 並列化による高速化が期待されている. しかし, 並列計算機向きのプログラムの記述は, 並列可能部分の認識や通信関数の導入などの困難があるため, 現状ではユーザには使いにくい.

このため, 逐次言語で記述されたプログラムを自動的に並列化するコンパイラの実現が求められている.

本研究では, 並列化を自動で行なうコンパイラのプロトタイプを開発し, 一般的な数値解析のプログラムで評価を行ない問題点を洗い出し, それに対するコンパイラの設計の変更をおこない, 現状では新規の開発を行ないつつある.

コンパイルの対象となるプログラムはC言語で逐次形式で書かれたプログラムであり, 並列化コンパイラに対しての並列化に関する特殊な命令を含んでいないものである. また, 出力ファイルは, プログラムの組変えと並列計算機でのメッセージ通信の標準ライブラリである MPI(Message-Passing Interface) を用いた専用ライブラリを加えて並列化を行なったものである.

1　はじめに

次世代の高性能計算機として分散メモリ型の並列計算機に大きな期待が寄せられているが, 一般に分散メモリ型の並列計算機に対して, 従来のプログラム言語と通信ライブラリを用いたプログラミングを行なうと, 各ノードにプログラムの仕事を割り当てるための処理, プログラムの実行されているノードに, そのプログラムが必要としているデータが存在しない場合のデータの転送処理, 等の逐次プログラムには必要の無い処理を書き足す必要が生じるためにプログラムがどうしても繁雑になる. これらの理由によって, 現在の手法では高速なプログラムを書くためには, ある程度の技術が必要になってしまうので, 逐次計算機向きに記述されたプログラムに対して, できるだけ変更を加えずに並列計算機上で効率の良い実行結果を得られるようにする環境の実現が求められている.

本研究では, 逐次計算機向きに記述されたプログラムを自動で並列化するコンパイラのプロトタイプの開発とその評価[(1)] を, 一般的な数値解析のプログラムで行なう. また新規開発中であるコンパイラについての概要を述べる.

この並列化コンパイラはC言語で書かれたプログラムを対象とし, 出力ファイルは専用の通信ライブラリを使用するC言語のプログラムの形である. これによって, ユーザーは今までの逐次のプログラムをそのまま並列計算機上で最大限の効率を発揮できる形式に変換でき, 対象となる並列計算機上の通常のコンパイラで再コンパイルすることで最大限の実行効率を得ることができるよう

になる可能性がある. また専用ライブラリ内では標準的なメッセージ通信ライブラリであるMPI (Message-Passing Interface)[2] [3] を使用している.

MPIを用いることによって現在の評価に用いているParagonのみならず, 他の様々な並列計算機でも効率良く実行できる可搬性のある並列プログラムを生成することを目標としている.

本研究では並列性が一番わかりやすい形で現れることが経験的にわかっているループ部分の解析をし, 並列化を自動で行なうC言語を対象とした並列化コンパイラを作成しParagon(Intel社. 分散メモリ型並列計算機) 上で評価を行なう.

本稿では, まず2章では本研究で開発中の並列化コンパイラの概要についてで, これはプロトタイプの結果を踏まえて修正したものである, 3章ではプロトタイプの並列化コンパイラによる一般的な数値解析のプログラムの並列化の効果と有効性の評価, 4章でまとめと今後の課題について述べる.

2 並列化コンパイラの概要

2.1 並列化コンパイラの目的

本研究で開発中の並列化コンパイラは, 以下のような戦略に基づいている.

(1) 並列化が容易で効率的に実行できることがわかっているループ部分を並列化する.
(2) 並列化が容易ではなく, 並列化を行なっても効率の向上が得られにくい場合には並列化を断念する.
(3) 並列化コンパイラはユーザーから並列化に関する情報をコンパイルオプションを除き全く求めない.
(4) 出力結果を実行環境に依存しないようにする.

以上よりユーザーが気をつけることは

(1) プログラムを書く上で並列化に適したアルゴリズムを採用する.
(2) ユーザーは並列化しやすいようにプログラムを書く.

つまりユーザは現在のベクトル化機能のあるコンパイラと同じようにコンパイラを使用すればよいようにすることを目的にしている.

2.2 並列化コンパイラの構造

本研究で開発された並列化コンパイラは, 複数の構成要素に分解できる.

(1) C言語から中間言語へのコンパイラ
(2) 中間言語の並列化トランスレーター

(3) 並列化された中間言語の通信命令のオプティマイザ
(4) 中間言語からＣ言語への逆コンパイラ

以下にそれぞれの機能の説明と, 用語の説明を行なう.

- 中間言語
  アセンブリ言語に近い形式で, 解析の簡略化のために使われる. レジスタは無くスタックを中心とした形式で実装されている. Ｃ言語の要素を全て分解できるできるようにしている. 並列化処理後のＣ言語への再変換もできるように構成している.

- 中間言語の並列化トランスレーター
  依存関係の調査による定型を持つ命令の抽出, 出力の依存関係の調査とそれを用いたループの分割, 転送命令の前後関係から生成される依存関係を解決する命令の挿入, の三段階に分かれ, それぞれが独立したプログラムである.

  中間言語を読み込み, 並列化された中間言語を生成する. ループの並列化を主な処理内容にしている.

- 並列化された中間言語の通信命令のオプティマイザ
  並列化された中間言語を読み込み, 挿入された通信命令の内で余分な命令の削除と, データの分割の最適化とそれに伴う命令の挿入を行なう. これによって最終的なコンパイル結果が得られる.

- 並列化ライブラリ
  並列化のための情報を得るための関数と, 幾つかの入力出力関数を, 無駄な動作をしないように並列化, もしくは, 選択動作できるようにしたもの. 並列化コンパイラで自動的に使用される. また並列化コンパイラ内での変換の簡略化のために用いる. 出力ファイルがＣ言語であることを利用して, 主にマクロ命令として定義された命令で構成され, 所有ノードや実行ノードの判断を行なっている.

2.3 並列化コンパイラの仕様要求

本研究で開発した並列化コンパイラは以下のルールにしたがっている.

(1) 並列化部分はループ部分だけである
(2) 並列化のネストはしない
(3) ループの構造をできるだけそのまま利用する
(4) ソースに加える変更をできるだけ少なくする
(5) 関数は main 関数内に inline 展開される.

2.4 並列化コンパイラの動作

本研究で開発した並列化コンパイラでは中間言語に対して並列化に関する処理を行なう．具体的には以下の順序 (図 1 参照) で処理を行なう．

- ループの反復回数の検出
  C言語では文法上, ループの反復回数は明示されないし, その終了条件は自由に決めることができる．このため, 並列化コンパイラではループ分割するためにループの反復回数を検出しなくてはならない．

- ループが並列化可能かどうかの判定
  ループの中には並列化して効果を得ることができないものが多い．このために依存解析を行なって, 並列化効果の無いであろうループは並列化しないようにしている．

- データ分割から見たループ分割の最適化
  並列化可能なループが複数ある時, それぞれの並列化の組合せで生じるデータ転送の負荷とその性格について調査して最善の構成を選択する．

- 転送命令の付加
  並列化ループ内で生成されるデータを他のループへ転送するための命令を挿入する．またこのための前処理と後処理の命令もそれぞれ挿入する．並列化ライブラリでは多数の小さな通信データをまとめて一つの長い通信データとして扱うことによって, 通信命令の起動回数を減らして効率化を行なっている．このためにまとめるためのメモリを確保する必要がある．また転送時に同時に演算を行なう必要がある場合には特別な命令を付加して効率を向上している．

- 転送命令の最適化
  転送命令の付加の際に用いている手法は単純であり, この時点でプログラムの前後関係を解析し最適な処理しているわけではない．このため無駄な転送が生じるので, この削除を行なう．

- 並列化に関する後処理
  本研究で開発した並列化コンパイラでは前もって inline 展開されている関数しか並列化しない．またこの inline 展開によって関数自体の大きさが膨れ上がるので, 得られたプログラム列からの関数の生成によるプログラムの圧縮が必要になる．

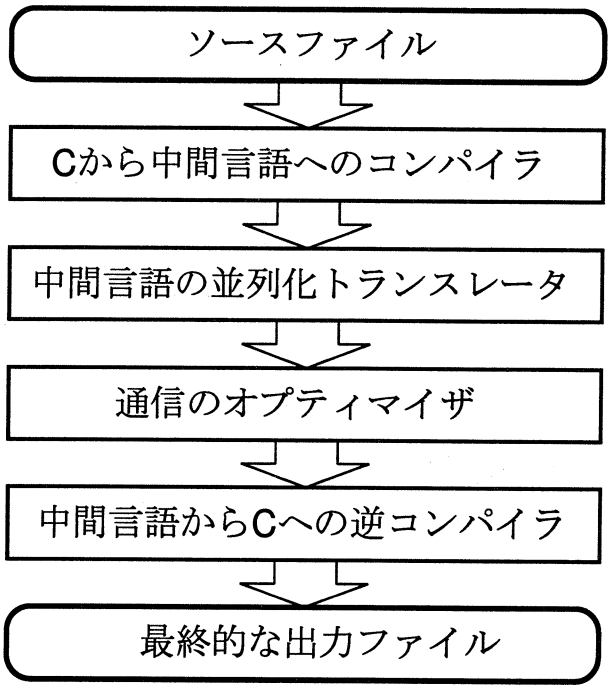


図 1: 並列化コンパイラの動作順序

3 数値処理プログラムによる評価

3.1 評価目的

本研究で開発した並列化コンパイラを用いて一般的な数値解析のプログラム(4) (5) (6) を実際に自動並列化したプログラムと並列化を行なわないプログラムをそれぞれ実行し, 並列化によって得られた効率を算出した.

分散メモリ型並列計算機である Paragon(Intel 社) 上で評価を行なっているが最終的には MPI の動作する全ての計算機で効率良く実行できるようにすることを目標としている.

一般的な数値解析のプログラムとして, 行列積, Gauss-Seidel 法, 共役傾斜法, クラウト法のプログラムを選んだ. ただし, これらのプログラムには並列化コンパイラがまだ対応していない命令 (プリプロセッサ命令,typedef,struct,union 他) を置き換える処理を行ない, 時間計測のために MPI の命令を付加している.

3.2 結果

逐次プログラムを自動で並列化するコンパイラのプロトタイプを開発し, 数値計算プログラムを用いて評価した.

行列積	: 16 台で 15.9 倍の台数効果を得た
共役傾斜法	: 16 台で 8.96 倍の台数効果を得た
Gauss-Seidel 法	: 7 台で 4.29 倍まで台数効果を得たが以降は低下
SOR 法	: 16 台で 3.22 倍の台数効果を得た
クラウト法	: 台数効果は得られなかった

4 むすび

主にループ部分を検出し解析し自動で MPI を用いた並列化をし, 分散メモリ型並列計算機を対象とする, 自動並列化コンパイラのプロトタイプを開発した.

数値計算の評価によって, 並列化に向いているような計算ではそれなりの実行効率を得ることができたがデータの転送が演算に比べて多いようなプログラムの並列化においては, ノード数を増やせば増やすほど実行効率が低下することがわかった.

データ通信の最適化を施すだけでは, データの転送の大小が大きくプログラムの実行速度に影響することがわかったので, データの転送量を少なくするための新たな機構として, 次の開発ではループブロック間のデータ転送を考慮して並列化を行なう機構をコンパイラに搭載することを決定し開発中である.

MPI を用いることによって様々な並列計算機で効率良く実行できる可搬性のある並列プログラムを生成することを目標としているので他の計算機上での実行効率の計測を行いたい.

5 参考文献

(1) 橋井邦夫, 小畑正貴,MPI を対象とする自動並列化コンパイラ, 情報処理学会 ハイパフォーマンス・コンピューティング研究会 68-5(1997.10.17) ,1997

(2) 青山 幸也,RS6000 SP 並列プログラミング虎の巻 MPI版, 日本アイ・ビー・エム株式会社,1996

(3) MPI フォーラム MPI 日本語訳プロジェクト訳,MPI：メッセージ通信インターフェース標準（日本語訳ドラフト）,MPI フォーラム,1996

(4) 奥村 晴彦, C言語による最新アルゴリズム事典, 技術評論社,1991

(5) 戸川 隼人,UNIX ワークステーションによる科学技術計算ハンドブック 基礎編 C言語版, サイエンス社,1992

(6) 水上 孝一, 市山 寿夫, 野田 松太郎, 南原 英生, 渡辺 敏正 共著, コンピュータによる数値計算, 朝倉出版,1985