ホモロジー解析プログラムを用いたワークステーションクラスタ, TMC CM-5, Intel Paragonの性能評価

坂田聡子[†,☆] 長嶋雲兵[†] 佐藤三久[††]
関口智嗣[††] 細矢治夫[†]

ホモロジー解析とはデータ配列間の類似性の判断を行うもので，これまで主に，生物学の分野でアミノ酸の塩基配列の類似性の判定を行うのに用いられてきた．ホモロジー解析のための計算時間と主記憶容量は，アミノ酸の塩基の数 N の2乗に比例して増加するので，このホモロジー解析を定量的に行うダイナミック・プログラミング法（Dynamic Programming method: DP）をタイリングによるウェーブフロント法を用いて並列化し，その高速化を行った．さらに本プログラムをワークステーションクラスタ Toshiba AS4040 (SUN4 ipc, 28台) と TMC CM-5 および Intel Paragon を用いて実行し，それらの上での性能を実測した．これらの並列システムの評価を行うにあたり，このプログラムにおける各マシンの特徴を把握するため DP の並列化プログラムの実行時間予測モデルを作った．性能予測モデルから予測される実行時間は実測と非常によく一致し，計算の粒度制御パラメータの変化に対し実測とほぼ同様の振舞いをみせた．ワークステーションクラスタ，CM5，Paragon の各マシンにおける見積もりと実測の相対誤差はそれぞれ，0.1%，5%，2%となった．また，それぞれの計算機で，1台から20台までほぼ線形の性能向上がみられた．Paragon，CM5，ワークステーションクラスタの性能比は問題のサイズやプロセッサ数によらず，ほぼ1：0.38(1/2.6)：0.22(1/4.6)になった．

Performance Evaluation on a Workstation Cluster, TMC CM-5 and Intel Paragon Using a Parallel Homology Analysis Program

SATOKO SAKATA,[†,☆] UMPEI NAGASHIMA,[†] MITSUHISA SATO,[††]
SATOSHI SEKIGUTI[††] and HARUO HOSOYA[†]

Homology analysis of protein has an important role in biology. In the homology analysis, required memory size and computing time increase proportionally with the square of the number of amino acids in protein. In order to expand the limit of size and to carry out the analysis effectively, we parallelized the program for homology analysis adopting the wavefront method with tiling. The performance of a workstation cluster, CM5 and Paragon was compared by using our program. For performance evaluation of these machines, we have made a performance model of our parallel homology analysis computation. Upon the model, we can understand the characteristics of these machines with respect to our program. Execution time estimated by the execution model is in good agreement with the observation. The averaged error of the estimation to the observation on a workstation cluster, CM5, and Paragon are 0.1%, 5%, and 2%, respectively. Almost linear speed-up has been observed from 1PU to 20PUs on them. The ratio of the performance of Paragon, CM5 and a workstation cluster is 1:0.38:0.22. This ratio is constant to the variety of the size of problem and the number of processors.

1. はじめに

ホモロジー（homology）とは，類似性，相同性という意味である．ホモロジー解析とはデータ間の類似性の判断を行うもので，これまで主に，生物学の分野でタンパク質の性質や構造を予測する手段としてさかんに行われてきた．このアミノ酸の塩基配列の類似性を判断する手法として用いられてきたのが，**ダイナ**

† お茶の水女子大学理学部情報科学科
Department of Information Science, Ochanomizu University
†† 電子技術総合研究所情報アーキテクチャ部
Electrotechnical Laboratory
☆ 現在，東京電力株式会社
Presently with Tokyo Electric Power Company

ミック・プログラミング法（Dynamic Programming method: DP）である．DPはアミノ酸の塩基配列のホモロジー解析を定量的に行うのに使われている方法[5),6),11)]で，その特色は類似度を示すホモロジー・スコアの計算により類似性の定量化が可能になることにある．DPは，情報科学の分野における動的計画法に基づきNeedleman, Wunschが提唱しSellersにより修正されたのでNeedleman-Wunsch-Sellers（NWS）アルゴリズムと呼ばれることもあるが，一般的にはこれをDPと呼ぶ．

DPでは，類似度を定量化するホモロジー・スコア計算に必要な文字列の長さ N に対して，N^2 に比例する主記憶およびCPU時間が必要となる．N の上限がメモリにより制限されるため，1台の計算機で実行することを仮定すると巨大なデータ列のホモロジー解析は不可能となる．もちろん仮想記憶などディスクを用いて実行することも可能であるが，ディスクI/Oの遅さを考慮すると，現実的ではない．

すでに我々は，DPの現実的な計算可能サイズの拡大および計算の高速化を図るため，この方法の並列化を行った[10)]．ここでは並列化により，メモリの有効利用ができ，逐次プログラムに比べ台数以上の大きな問題を解くことが可能となることを示し，さらに並列化したDPを単純にイーサネットで結合したワークステーションクラスタで並列実行したところ，期待される線形の性能向上が観測された．

本論文では，ワークステーションクラスタ上で開発されたプログラムをネットワークトポロジーが異なるIntel ParagonとCM5に移植し，各々の計算機上で，計算の粒度制御パラメータの変化に対する実行時間の振舞いを比較した．ここで，ワークステーションクラスタのメッセージパッシングライブラリとしてPVM3.1[2)]を用いた．

各並列システムの性能評価にあたり，実行時間を見積もる性能予測モデルを作成し，粒度制御パラメータの変化にともなう実行時間の振舞いを予測し，効率の良い計算を可能とする最適の粒度の決定を試みた．

我々のモデルにより，実行時間を最小にする最適のタイルサイズの決定，および実行時間の予測が可能となり，このモデルを用いることにより，我々はこれらのマシンの特徴を理解することができた．

本論文では，我々のプログラムにおけるこれらのマシンの性能評価の結果について報告する．

2. ダイナミック・プログラミング法（DP）

ホモロジー解析とは，より具体的には，配列間の類似性を表すホモロジー・スコアを計算し，スコアの良いものを選び出す操作のことをいう．この操作を行うのによく使われるのが以下に示すダイナミック・プログラミング法（DP）である．

（例）文字列数が9である2つの文字列を

H1: G,A,A,G,A,A,C,T,G
H2: G,A,A,G,A,C,T,G,C とするとき

$$F = \begin{array}{c} \\ G \\ A \\ A \\ G \\ A \\ A \\ C \\ T \\ G \end{array} \begin{array}{c} \begin{array}{cccccccccc} G & A & A & G & A & C & T & G & C \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & -2 & 1 & -2 & -2 & -2 & 1 & -2 \\ 0 & -2 & 2 & -1 & -2 & 2 & -1 & -4 & -2 & -1 \\ 0 & -2 & -1 & 3 & 0 & -1 & 0 & -3 & -5 & -4 \\ 0 & 1 & -2 & 0 & 4 & 1 & -2 & -2 & -2 & -5 \\ 0 & -2 & 2 & -1 & 1 & 5 & 2 & -1 & -4 & -4 \\ 0 & -2 & -1 & 3 & 0 & 2 & 3 & 0 & -3 & -6 \\ 0 & -2 & -4 & 0 & 1 & -1 & 3 & 1 & -2 & -2 \\ 0 & -2 & -4 & -3 & -2 & -1 & 0 & 4 & 1 & -2 \\ 0 & 1 & -2 & -5 & -2 & -4 & -3 & 1 & 5 & 2 \end{pmatrix} \end{array}$$

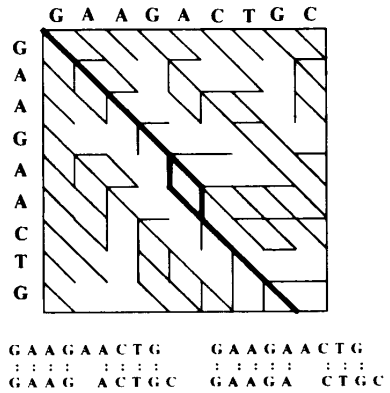


図1 ホモロジー・スコア行列 F とその経路図
Fig. 1 Homology score matrix F and path graph.

DPでは，ホモロジー・スコア行列 F（後述）と経路図を考える．**図1**に示す経路図に書き込まれた経路は，それぞれが特定の並置に対応している．太線で示した経路は図の下部に示した並置を示しており，これが最良の並置になっている．経路図で対角線方向の経路は対応する文字が対になっていることを示し，横方向と縦方向の経路は文字の挿入および欠失を示している．

DPでは，ある並置に対して類似性の程度を定量化するために，文字の一致，不一致，欠失（あるいは挿入）に適当な重みを与えて，その和をとる．2つの配列のホモロジー・スコアは，この和の最大値と定義され，これに最良の並置が対応することになる．(データ文字列数 + 1) 次元の行列 F において，行列の左端の文字列と，上端の文字列を1文字ずつ比較していく．行列要素の計算方法は以下のようになる．

$$F(i,j) = \max\{F(i-1,j-1) + \beta, F(i-1,j) + \gamma, F(i,j-1) + \gamma\} \quad (1)$$

$$\text{where} \begin{cases} \beta: \text{一致の重み } 1, \quad \text{不一致の重み } -2 \\ \gamma: \text{挿入（欠失）の重み } -3 \end{cases}$$

初期値を $F(i,0) = F(0,j) = 0$ として行列全体を計算し，それらのうちで最大値をとるものを最良のホモロジー・スコアとする（この例では5）．ホモロジー・スコアの値は，比較するデータ文字列の相似性の度合に比例し，また，データ文字列間の距離はホモロジー・スコアの逆数をとったもので表すことができる．したがって，ホモロジー・スコアはデータ文字列間の相似性の指標となる．

3. DPの並列化

3.1 ウェーブフロント法とタイリング

DPの計算可能データ数の拡大および計算時間の縮小を図るため，先に説明したDPの並列化を行った．この2つの目的を実現させるためには，プロセス間の通信回数を極力少なく，また計算の最大並列度が長時間保たれるように，なおかつ有限サイズのメモリを有効に利用することが必要となる．これまでDPの並列化においては，共有メモリ型マルチプロセッサ上における上三角行列としてのホモロジー・スコア行列計算の並列化の例（同期型，無通信）がある[1]が，本研究では分散メモリ型並列システム上でのDPの並列化（メッセージパッシング方式）を試みた．

ホモロジー・スコア計算では，逐次ループで記述すると次のような二重ループ構造をとる．

```
for (i = 1; i ≤ L; i++)
    for (j = 1; j ≤ M; j++)
        F[i][j] = max{F[i-1][j-1] + β,
                      F[i-1][j] + γ, F[i][j-1] + γ};
```

このループのイタレーションの集合は，**図2**に示すような2次元のイタレーション空間を構成する．

個々のイタレーション（つまり $F(i,j)$）は黒い点で示されている．ホモロジー・スコア計算における行列要素参照にともなうデータの流れにより，イタレーション間には矢印で示されるような依存関係（実行順序の制約）がある．すなわちホモロジー・スコア行列 F において，各成分 $F(i,j)$ の値は3つの値 $F(i-1,j)$, $F(i,j-1)$, $F(i-1,j-1)$ に依存し，この依存関係は，3種類の依存距離ベクタ $(1,0)$, $(0,1)$ および $(1,1)$ で表される．i ループ，j ループのいずれのループについても，ループ運搬依存があるのでDO ALL型の並列実行はできない．つまり，この2重ループ構造は i 次元に関するDO ACROSS型ループ[12]であり，

(a) 並列実行したい次元についてループ運搬依存がある．

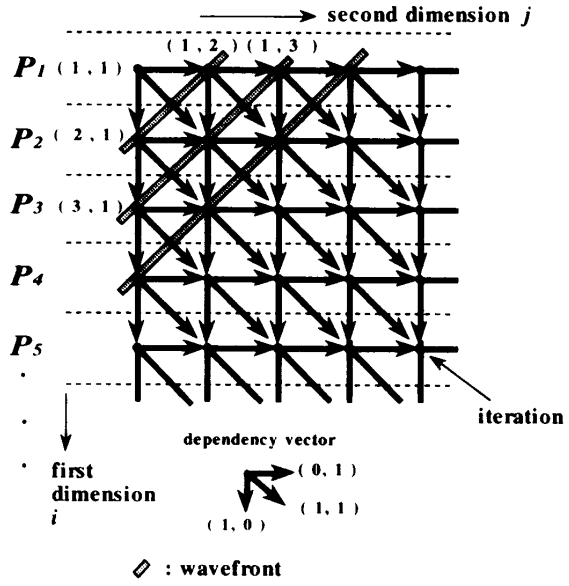


図2 DPにおけるイタレーション空間
Fig. 2 Iteration space in DP.

(b) 並列実行したい次元の内側に別の次元を含む．

という2つの条件を満たすので，**ウェーブフロント型ループ**といえる[7]．この形のループに対しては，以下に述べるウェーブフロント法を適用することができる．

図2に示すように，イタレーション空間を i 次元で分割して1行ずつプロセッサ $P_1, P_2, \ldots$ に割り付ける．最初のステージでプロセッサ P_1 がイタレーション $(1,1)$ を実行し計算結果 $F(1,1)$ をプロセッサ P_2 に転送する．次のステージで，イタレーション $(1,2)$ と $(2,1)$ をプロセッサ P_1 と P_2 が並列に実行し，計算結果をそれぞれ隣のプロセッサに転送する．以下同様に図2に示したウェーブフロント上のイタレーションを各ステージで並列に実行する．しかしこの場合，プロセッサ間通信の起動時間がプロセッサ内演算時間に比べて非常に大きく，またプロセッサ数が有限であることを考慮すると，上記の方法は実用的でない．この問題の解決のため，複数のイタレーションをまとめた「タイル」と呼ばれる単位ごとに演算と通信を行う**タイリング**という技法が提案されている[4),7),9)]．

タイリングされたウェーブフロント法では，イタレーション空間の矩形で表される複数のイタレーションをまとめてタイルを構成する．**図3**では i 次元のイタレーションの最大値 L をプロセッサ数 P で分割し，j 次元のイタレーションの最大値 M を P に粒度制御パラメータ α を乗じた数で分割してタイルを形成している．

このとき依存距離ベクトルを $d = (d_1, d_2)$ とする．

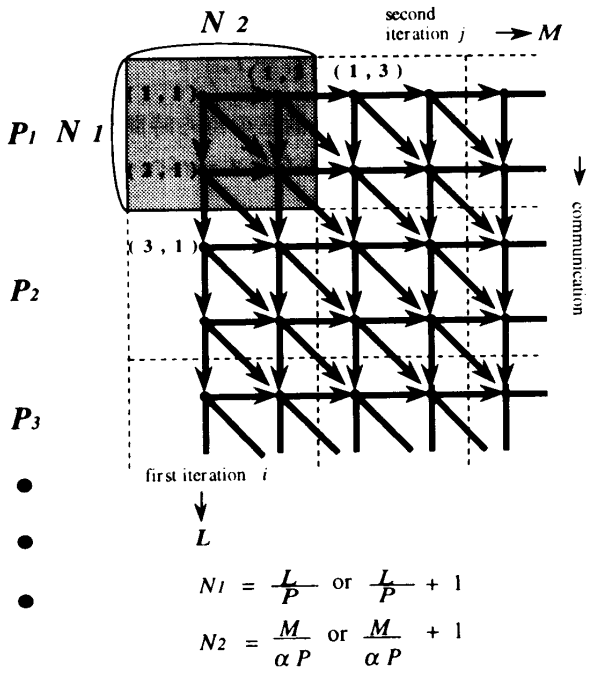


図3 DPにおけるタイリング
Fig. 3 Tiling and iteration space in DP.

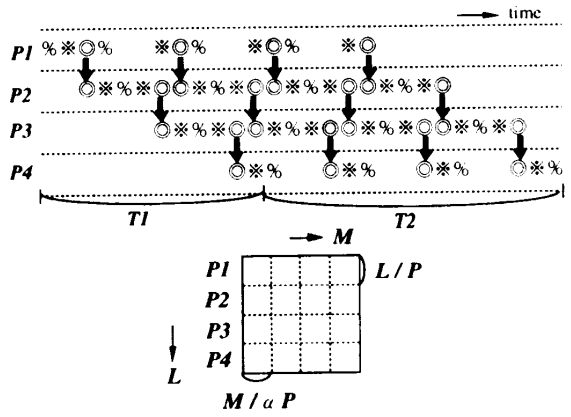


図4 タイリングされたウェーブフロント法の実行時間の流れ図
Fig. 4 Diagram of execution of the tiled wave front method; where $P = 4$, the meanings of the characters such as ※, ◎ and %↓ are as follows:
※: S_s, R_s: communication startup latency at the sender and receiver, $S_s = R_s$ is assumed.
◎: S_t, R_t: the time for transmitting data at the sender and receiver, assumed $S_t = R_t$.
%: T_{tile}: operation time/tile.
↓: communication (sender → receiver).

第1（つまり i）次元に関してループ運搬依存がある，すなわち依存距離ベクトルの第1成分は0でないものとする．なお，依存距離ベクトルの一般的性質として，第1成分は0でなければ正である．依存距離ベクトルが複数ある場合は，それらの中で負方向の傾斜が最も急なもの，すなわち $-d_2/d_1$ が最大であるものを d とする．明らかに，このベクトルで表される依存関係を守れば，他の依存関係も守られる．

ホモロジー・スコア行列 F において，各成分 $F(i,j)$ の値は3つの値 $F(i-1,j)$，$F(i,j-1)$，$F(i-1,j-1)$ に依存する．したがって，タイリングされた各タイル（もとの行列の部分行列）についても同じ関係が成り立つので，各タイルの値を求めるときには，上のタイルの下の隅と左のタイルの右の隅，そして左上のタイルの右下の隅の値があれば良い．つまり，各タイルは上のタイルの1行と左のタイルの1列のデータのみがあれば計算ができるので，このときの依存距離ベクトルは $d = (1,0)$ となり，1つの反対角線方向のタイルは全部独立に並列して計算できる．したがって，図3で示す最大並列度は P であり，αP と P の差に対応する処理ステップ数だけ実行される．

このように，タイルを単位としたウェーブフロント型の並列実行を行う方法では，タイリングしない場合に比べて通信回数が少なく，通信起動オーバヘッドを低減できる．また，各マシンはホモロジー・スコア計算においてタイルの大きさに相当するメモリ領域のみを保持すればよいので，本来保持するべき計算領域を縮小することが可能となる．

このとき，並列計算の粒度を制御するのは，各タイルの大きさに影響を与える粒度制御パラメータ α である．最大の並列性の保持を考慮すると，1タイルの大きさはできるだけ小さい方がよいので，α を大きくする必要がある．一方プロセス間の通信回数対計算回数の比を小さくするには，α を小さくして1回の通信あたりできるだけたくさんの計算をするよう1タイルの大きさを大きくし，通信コストを減らすのが望ましい．前者の場合，最大並列度が何ステップにもわたって保たれるが，通信回数が α に比例して増大する．一方後者の場合，通信コストは少なくなるが，最大並列度でのステップ数も減少する．これらのことを考慮して，使用可能なすべてのプロセッサを利用できるような高さを持ち，なおかつ計算初期および終期の非効率さをなるべくおさえることのできるような幅を持った最適サイズのタイルを α によって決定する必要がある．そこで，次にタイルサイズと実行時間の関係を考察する．

3.2 タイルサイズと実行時間の関係

DPの並列化プログラムにおけるタイルサイズと実行時間の関係をみるため，実行時間をデータサイズ L および M とプロセッサ数 P，粒度制御パラメータ α の関数で表す．タイルは図4に示すように，$P_1, P_2, \ldots$ の各プロセッサに割り当てる．ここでは $P = 4$ とし

ている．

$\mathrm{mod}(L,P) \neq 0$ のときは剰余と同数のタイルについて，タイル幅を1だけ大きくする．$\mathrm{mod}(M,\alpha P) \neq 0$ のときも同様である．各プロセッサ，P_1，P_2，P_3，P_4 はそれぞれ αP 個のタイルを実行する．ここでタイルサイズを L/P，$M/(\alpha P)$ とすると，通信時間を除いた1タイルあたりの実行時間 T_{tile} は，

$$T_{tile} \cong \frac{L \times M}{\alpha P^2} \times T_e + \left(\frac{L}{P} + \frac{M}{\alpha P}\right) \times T_c \quad (2)$$

と表せる．ここで，T_e は1イタレーション（つまりホモロジー・スコア行列の1要素）あたりのプロセッサ内演算時間を表し，T_c は1要素あたりのバッファへのコピー時間を表す．つまり第1項は，1タイルあたりのプロセッサ内演算時間であり，第2項は1タイルあたりのバッファへの要素のコピー時間を表す．

タイリングされたウェーブフロント法では，タイルの下端および右端の値は次のプロセスに渡す必要がある．したがって $M/(\alpha P) \times T_c$ は異なるプロセッサ $P_i \rightarrow P_{i+1}$ に向けた，通信バッファへのタイルの下端の値のコピー時間を表し，$L/P \times T_c$ は P_i 自身へのタイルの右端の値をコピーする時間である．

ここで，我々はプロセッサ間通信のオーバヘッドを以下のように仮定する．

$$T_s = S_s + R_s \quad (3)$$

ここで，S_s と R_s は各々プロセッサの送信，受信のセットアップ時間を示す．また，S_t，R_t を，送信，受信を行うプロセッサの1要素あたりのデータ転送時間とするとき，1要素あたりのデータ転送時間 T_t は S_t および R_t に等しいものとする．

1番目のプロセッサ P_1 の起動から最終プロセッサ P_p の起動までに実行されるタイル数は $P-1$ 個となるため，P_1 の起動から P_p が最初の通信を終えるまでの時間を T_1 とすると，T_1 は以下のような式で表すことができる．

$$T_1 \cong (P-1) \times \left(T_{tile} + \frac{M}{\alpha P} \times T_t + T_s\right) \quad (4)$$

式 (4) の第2括弧の第2，3項はタイリングされたウェーブフロント法におけるプロセッサ間のデータ通信時間を示している．つまり，これらの項の和は1タイルあたりの通信時間を示す．

また，最終プロセッサ P_p が実行するタイル数は αP 個である．よって P_p が計算を始めてから最後のタイルを処理するまでの時間，T_2 は以下の式で表すことができる．

$$T_2 \cong \alpha P \times T_{tile} + (\alpha P - 1) \times \left(2 \times \frac{M}{\alpha P} \times T_t + T_s\right) \quad (5)$$

式 (5) の第2項はデータ通信の待ち時間を示す．さらに，DPの並列化にともなうオーバヘッド T_3 を加える．

$$T_3 \cong \left(\frac{\alpha P \times (\alpha P + 1)}{2}\right) \times T_o \quad (6)$$

ここで，T_o は各タイルの1成分とホモロジー・スコア行列全体での成分を対応させるのに必要となる計算時間である．

結局，T_1，T_2，T_3 を加えることにより，全実行時間 T は，パラメータ L，M，P，α，T_e，T_c，T_s，T_t と T_o を用いて，以下のように表すことができる．

$$T \cong T_1 + T_2 + T_3 \quad (7)$$

ここで，T_e，T_o はDPの並列化の方法に依存する．

このとき，実行時間 T を α の関数として以下のように表せる．

$$\begin{aligned} T \cong\ & \alpha^2 \left(\frac{P^2}{2} \cdot T_o\right) \\ & + \alpha \left(L \cdot T_c + P \cdot T_s + \frac{P}{2} \cdot T_o\right) \\ & + \frac{1}{\alpha} \left(M\left(1 - \frac{1}{P}\right) \cdot \left(\frac{L}{P} \cdot T_e + T_c\right)\right. \\ & \left. + M\left(1 - \frac{3}{P}\right) \cdot T_t\right) + C \end{aligned} \quad (8)$$

where C = constant term on α

このように，T の値は α に関する3つの項の和より求められる．これらの関係を**図5**に示した．α が大きなところのグラフの傾きを決めるのは T_c や T_o などの，CPU性能を反映するパラメータであることが分かる．マシンの性能が高いとき，つまり T_c，T_o の値が小さいとき T は α に強く依存しない．他方，α が小さな領域では，T_e，T_c に加え通信性能を反映する T_t の依存性があり，α が小さくなるにつれて急速に性能が低下する．

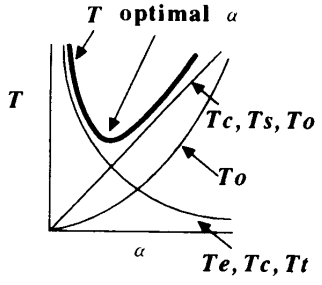


図5 実行時間のグラフ
Fig. 5 The graph of execution time.

4. 並列化の効果

ここで，並列化の結果について述べる．

4.1 測定環境

データはいずれもランダムにアルファベット26文字を並べたものを利用した．DPプログラムのコーディングの際，各データの特徴をより顕著にするような特別な演算を加えなかったので，結果は入力データのパターンには依存しない．用いたシステムは次のとおりである．

SUN4ワークステーションクラスタ—ワークステーションクラスタの機器構成はお茶の水女子大学理学部情報科学科の情報教育用計算システム，Toshiba AS4040 (SUN4 ipc, SPARC 25 MHz)/22 MB，28台まで用いた．これらはイーサネットで相互結合され，NFSでファイル共用がなされている．以下SUN4WSと表記する．

Thinking Machine Corp. CM-5—32プロセッサ（プロセッサはSPARC 32 MHz，キャッシュ64 KB．ネットワークはfat tree．速度は5 MBから20 MB.)．RWCつくば研究センターのものを利用した．以下CM5と表記する．

Intel Paragon/XP—32プロセッサシステム（プロセッサはi860 50 MHz，ネットワークは2次元メッシュ，200 MB/sec（カタログ値，実行速度は30 MB/sec程度))．RWCつくば研究センターのものを利用した．以下Paragonと表記する．

いずれも，メッセージ通信ライブラリはCM5ではCMMD，ParagonではNXライブラリ，ワークステーションクラスタではPVM3.1[2]を用いた．各システムのプログラムは，ほぼ同じものである．ただし，ワークステーションクラスタでは非同期通信，CM5とParagonではある回数以上の非同期通信が不可能であったので同期通信を行った．また，プログラムはシステムのデフォルトの最適化オプションでコンパイルした．なお，各システムにおいて1プロセッサ上で1プロセスのみを実行させた．つまり，プロセス数P = プロセッサ数である．

4.2 実行時間推定のためのパラメータ

理論式を用いた実行時間の推定の計算において，パラメータ値は**表1〜3**に示した．T_sの単位はμs，T_t, T_e, T_c, T_oの単位はμs/wordである．ここで，1 word = 8 byteである．

測定値のT_s, T_tは本実験とは別に実測したメッセージ通信性能の実測値から算出した．

表1 通信における各システムのT_s (μs)
Table 1 Communication startup latency, T_s on SUN4WS, CM5, and Paragon (in μs).

System	T_s (ref.)	T_s (obs.)
SUN4WS		3970
TMC CM-5	13*, 126+, 88++	82
Intel Paragon	164*	88

*：文献13)
+, ++：文献8)
where +: l (number of bytes) mod16 $\neq$ 0;
++: l mod16 = 0;

表2 通信における各システムのT_t (μs/word)
Table 2 Communication time/word, T_t on SUN4WS, CM5 and Paragon (in μs/word where 1 word = 8 byte).

System	T_t (ref.)	T_t (obs.)
SUN4WS	3.2**	17
TMC CM-5	0.4*, 1.8+, 0.5++	0.6
Intel Paragon	0.105*	0.5

*：文献13)
**: peak performance of Ethernet 10 Mbyte/sec.
+, ++：文献8)
where +: l (number of bytes) mod16 $\neq$ 0;
++: l mod16 = 0;

表3 各システムのT_e, T_c, T_o (μs/word)
Table 3 Computation time per element T_e, buffer copy time T_c, overhead time to index translation T_o on SUN4WS, CM5 and Paragon (in μs/word, where 1 word = 8 byte).

System	T_e (obs.)	T_c (obs.)	T_o (obs.)
SUN4WS	19.8	1.4	2.0
TMC CM-5	10.7	1.5	1.4
Intel Paragon	4.2	0.3	0.2

表1に示されるように，SUN4WSのT_sの測定値は，ネットワークの性能を反映してCM5，Paragonの50倍ほど大きい値をとっている．CM5とParagonのT_sの測定値はほぼ同じである．

表2に示すように，SUN4WSのT_tの実測値は，イーサネットのピークパフォーマンス（10 Mbyte/sec）から予期される値より6倍ほど大きな値となっている．ParagonのT_tの実測値は文献値[13]よりも5倍ほど大きい．これより，何らかの通信のコンフリクトが起こっているのではないかと思われる一方，CM5のT_tの実測値は文献値[8]とほぼ同じである．

1イタレーション（1要素）あたりの演算時間T_eは，プロセッサ1台での全実行時間をイタレーション数で割った値を用いた．Paragon（50 MHz），CM5（32 MHz）とSUN4WS（22 MHz）のカタログ上のCPU cycleの比は約2:3:4.5であるが，T_eの測定値の比は，約2:5.1:9.4である．これより，CM5と

SUN4WSの性能を示す T_e の測定値が，Paragonとこれらのマシンの CPU cycle との比から予想される値よりも2倍ほど遅いことが分かる．

T_c，T_o の値はDPのプログラムを実行することにより個々に求めた．Paragon，CM5，とSUN4WSの T_c と T_o の測定値の比はそれぞれ1:5:4.7と1:7:10である．これよりParagonのCPU性能はCM5とSUN4WSのCPU cycleとの比から予期されるものより良いことが分かる．これらの，CPU性能における実測値と予測値との誤差の原因に関しては，現在解析中である．

4.3 測定結果

4.3.1 タイルサイズと実行時間

タイルサイズの変化にともなう全実行時間 T の変化について，実測値と，前節で求めた理論値との比較を図6～8に示す．測定データは各々 $L = M = 4000$，$L = M = 10,000$ であり，プロセッサ数 $P = 9$ あるいは $P = 20$ とした．図6，7，8において，横軸はタイルサイズを決定する α，縦軸は実行時間 T (sec) を表す．**observed** は T の実測値を，**estimated** は T の予測値を示す．

粒度制御パラメータ α の増大にともないタイルサイズが縮小され，データ通信のためのオーバヘッドが増えると，実行時間 T は増加する．

T の値における実測と予測の誤差平均と相対誤差を**表4**に示した．以下，SUN4WS，CM5とParagonの実行時間の振舞いについて考察する．

- **SUN4WS**

図6に示すように，$L = M = 4000$ と $L = M = 10,000$ のどちらのケースにおいても，予測値のグラフの曲線は実測値のグラフに非常によく一致している．したがって，我々の性能予測モデルは α に対する実行時間 T の振舞いを非常によ

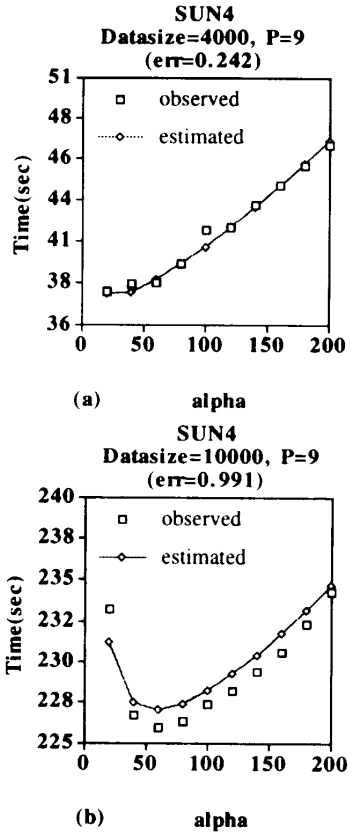


図6 タイルサイズと実行時間の関係（SUN4WS）
Fig. 6 Relationship of elapsed time and α (SUN4WS).

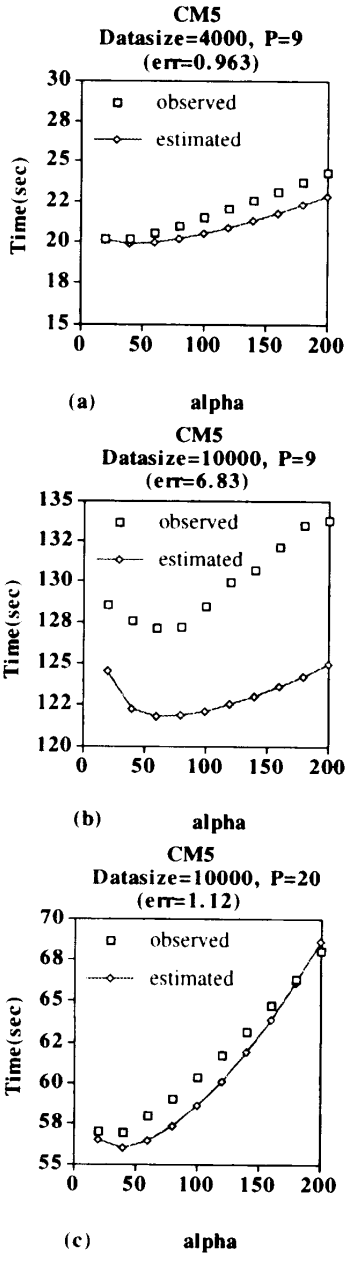


図7 タイルサイズと実行時間の関係（CM5）
Fig. 7 Relationship of elapsed time and α (CM5).

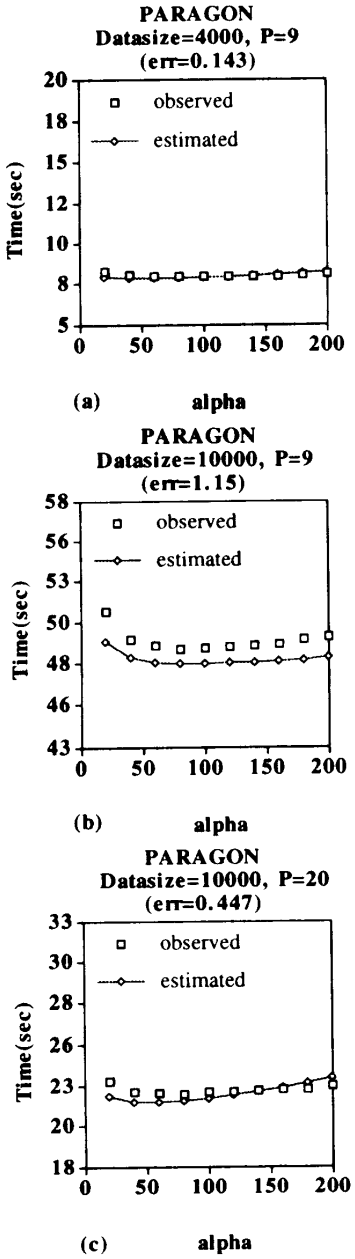


図8 タイルサイズと実行時間の関係 (Paragon)
Fig. 8 Relationship of elapsed time and α (Paragon).

く説明しているといえる．さらに，実行時間を最小にする最適の α の予測値も実測と一致している．$L = M = 4000$ と $L = M = 10,000$ のそれぞれの誤差平均（相対誤差）は，各々わずか 0.24 sec (0.1%)，0.99 sec (0.4%) である．実測値が予測値よりやや下回るので，我々のモデルに用いたパラメータ値はSUN4WSの性能をやや低く見積もっているといえる．我々のモデルから期待されるように，データサイズが大きいとき，すなわち $L = M = 10,000$ のとき，実行時間を最小にする最適の α は，データサイズの小さいときと比較して大きい値をとっている．

表4 T の実測値と予測値の誤差平均と相対誤差
Table 4 Averaged and relative errors between observed and estimated T.

System	error $L = M = 4$ K $P = 9$	error $L = M = 10$ K $P = 9$	error $L = M = 10$ K $P = 20$
SUN4WS	0.24 (0.1%)	0.99 (0.4%)	
CM-5	0.96 (5%)	6.8 (5%)	1.1 (2%)
Paragon	0.14 (2%)	1.2 (2%)	0.45 (2%)

- **CM5**

図7の(a)と(b)は $P = 9$，データサイズは $L = M = 4000$ と $L = M = 10,000$ である．これらはSUN4WS（図6(a)，(b)）に対応している．(c)は，$P = 20$ でデータサイズは $L = M = 10,000$ である．(c)は，1プロセッサあたりのタイルサイズが(a)の $L = M = 4000$，$P = 9$ のときとほぼ同じである．

CM5の3つのグラフに関しても，我々のモデルは，実行時間の振舞いを非常によく表現している．さらに，実行時間を最小にする α についてもよく一致している．各々のグラフにおける誤差平均（相対誤差）は0.96 sec (5%)，6.83 sec (5%)，1.12 sec (2%) である．誤差は微小範囲であるが，SUN4WSと比較して大きめの比率をとっている．これについては，表1，2からも分かるように，CM5の T_s，T_t の値は条件や測定によって非常にばらつきがあり，実際のプログラム実行の際にも，測定したパラメータ値とは異なる値をとっていた可能性が考えられるが，さらに詳しい解析はまだ行っていない．

- **Paragon**

図8にParagonに関する3つのグラフを示した．用いたデータサイズとプロセッサ数は図7のCM5のグラフと同じである．Paragonの3つのグラフに関しても，CM5，SUN4WS同様我々のモデルは，実行時間の振舞いを非常によく説明している．さらに，実行時間を最小にする α についてもよく一致している．各々のグラフにおける誤差平均（相対誤差）は0.14 sec (2%)，1.15 sec (2%)，0.45 sec (2%) である．

Paragonの3つのグラフに関しては，実行時間は粒度制御パラメータ α にあまり強く依存しないことが分かる．この原因は，式(8)より分かるように，ParagonのCPU性能が非常に高く，T_e お

よび T_c と T_o の値がSUN4WSやCM5に比べ小さいためと思われる．

以上の結果より，タイリングによるウェーブフロント法のプログラムに関する我々のモデルは，並列分散処理システム上の実行時間の振舞いを非常によく説明できることが示せた．SUN4WS，CM5とParagonどのマシンについても5%以内の相対誤差で実行時間 T の実測値と予測値は一致した．つまり，パラメータ T_e，T_s，T_t，T_c と T_o を別の方法であらかじめ測定しておけば，L，M と P を用いた関数 T より実行時間の振舞いが予測できる．さらに最適の α をプログラム実行前に決定することができ，効率の良い計算が可能となる．

以下の節ではSUN4WS，CM5とParagonの性能を比較する．

4.3.2 サイズ固定速度向上率の比較

次に一定サイズの行列に対するプロセッサ数 P と並列化されたプログラムの速度向上率 $S(P)$ の関係を図9に示す．このとき，データサイズは $L = M = 5000$ および $10,000$ であり，$\alpha = 15$ とした．

グラフの横軸はプロセッサ数 P を表し，縦軸は速度向上率 $S(P)$ を表す．ただし，

$S(P) = 1$ 台での実行時間/P台での実行時間

である．破線は $S(P) = P$ という線形の性能向上を示す．$L = M = 5000$ のとき，3種類の計算機は期待される線形の性能向上をほぼ実現しているが，10台を境に台数倍の性能より下回っている．この原因としては，プロセッサ数の増加に従い通信回数が計算回数に対して大幅に増加するため，通信の起動コストが実行時間に多大な影響を与え，サイズの小さいデータに対しては，その性能低下が比較的少数のプロセッサの段階で現れるためだと考えられる．それに対し，データのサイズの大きい場合，つまり $L = M = 10,000$ のときは，$P = 20$ のときまで線形の性能向上をほぼ実現している．つまり性能向上のプロセッサ数の範囲が，データサイズの小さい $L = M = 5000$ のときに比べプロセッサ数の大きい方にシフトしている．これは，データのサイズが大きくなるにつれその計算回数が大幅に増加するので多数のプロセッサによる並列計算がより有効になるためであり，プロセッサ数の大きなところでキャッシュミス等のメモリアーキテクチャ上のネックが解消されていくことを示している．

4.3.3 Paragon, CM5, SUN4 ワークステーションクラスタの相対性能

次にParagonの性能を1として各マシンの P を変化させたときの実行時間の比を図10に示す．

図10に示すとおり $L = M = 5000$ および $L = M = 10,000$ の両方の場合においてParagonはCM5に比べて約2.6倍速く，この比は問題サイズには大きく依存しないことが分かる．また P の変化にも大きく依存しない．しかし両者のカタログ性能値の比である1.56よりもかなり開きのあるものとなっている．またワークステーションクラスタとParagonについて

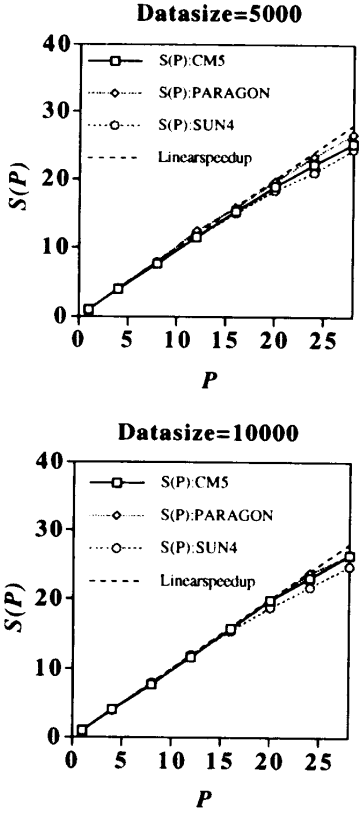


図9 並列化によるプロセッサ数 P と速度向上率 $S(P)$ の関係
Fig. 9 Fixed size Speedup ($S(P)$).

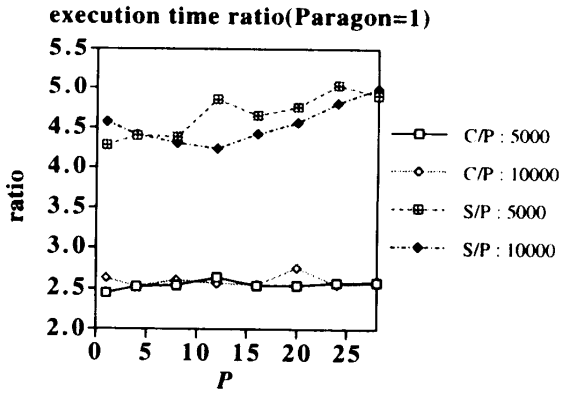


図10 各マシンの実行時間の比
Fig. 10 Ratio of the performance of SUN4WS, CM5, and Paragon (The performance of SUN4WS and CM5 is normalized by that of Paragon).

も同じようにその比はほぼ4.3～5倍であり，問題のサイズと P の変化には強く依存しない．ただし各データサイズにつき，P が増大するに従ってその比はゆるやかに増大する．これはSUN4WSの通信性能の悪さが早い段階で見えてくるためである．またCM5と同様にカタログ性能値の比2.0よりもかなり大きなものとなっていることが分かる．

Paragon，CM5およびワークステーションクラスタの性能比は1：0.38(1/2.6)：0.22(1/4.6)となっている．また問題の大きさやプロセッサ数に対し，性能比がほぼ一定であることは，観測された条件下においては，本プログラムがほぼスケーラブルであることを示している．

5. 結　　論

巨大次元のアミノ酸塩基配列のホモロジー解析を行うためDPのプログラムをタイリングによるウェーブフロント法を用いて並列化し，Paragon，CM5，ワークステーションクラスタの性能を比較した．

並列化にあたり，効率の良い計算を可能とする最適の粒度制御パラメータ α を決定するため，プログラムの性能評価モデルを作成し，実行時間の実測値と比較した．

性能評価モデルの各パラメータの実測値が文献値と多少異なるものの，我々のモデルは粒度制御パラメータ α の変化に対する実行時間の振舞いを非常によく説明し，実行時間を最小にする最適の α の値も実測とよく一致した．

1～20台までのプロセッサ数の範囲でParagon，CM5においてもワークステーションクラスタ同様ほぼリニアスピードアップが観測された．Paragon，CM5およびワークステーションクラスタの性能は問題のサイズおよび P の変化によらずほぼ1:0.38:0.22であることが観測された．これは，3つの並列計算システムにおいて，本プログラムがスケーラブルであることを示している．

Paragon，CM5およびワークステーションクラスタはメッセージパッシングライブラリの違いを除けばほぼ同様のプログラムを有効に走らせることができる．これは，安価なワークステーションクラスタが，ParagonやCM5などの並列専用計算機上のプログラム開発に用いることができ，また，適当なモデルを用いると，それらのおおよその性能予測も可能であることを示している．あるアルゴリズムに基づくモデルによる計算機の性能評価は，実行時間など様々なコストを予測したり，各々のプログラムに関する計算機の特徴を理解しやすくする上できわめて重要である．

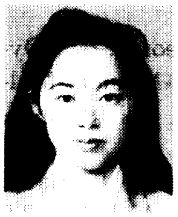
謝辞　本研究を行うにあたり，様々なご討論およびご助言をいただきました基礎生物学研究所の中井謙太博士，およびCM5とParagonを利用させていただいたRWCつくば研究センターに深く感謝いたします．

参 考 文 献

1) Edmonds, P., Chu, E. and George, A.: Dynamic Programming on a Shared-Memory Multiprocessor, *Parallel Computing*, Vol.19, pp.9–22 (1993).
2) Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V.: PVM 3.0 User's Guide and Reference Manual, *Technical Report ORNL/TM-12187*, Oak Ridge National Laboratry (1993).
3) Harrison, R.J.: Portable Tools and Applications for Parallel Computers, *Int. J. Quantum Chem.*, Vol.40, pp.847–869 (1991). TCGMSG is available via electronic mail from ftp. tcg. anl. gov.
4) Hiranandani, S., Kennedy, K. and Tseng, C.-W.: Evaluating Compiler Optimizations for Fortran D, *J. Parallel and Distributed Computing*, Vol.21, pp.27–45 (1994).
5) 中村春木，中井謙太：バイオテクノロジーのためのコンピュータ入門，pp.79–86，コロナ社，東京 (1995).
6) Needleman, S.B. and Wunsch, C.D.: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins, *J. Mol. Biol.*, Vol.48, pp.443–453 (1970).
7) 太田，斉藤，海永，小野：ウェーブフロント型ループの超並列計算機向けコンパイル技法，IPSJ SIG Notes, Vol.94, No.68, pp.13–20 (1994).
8) Ponnusamy, R., Choudhary, A. and Fox, G.: *Communication Overhead on CM5: An Experimental Performance*, IEEE (1992).
9) Ramanujam, J. and Sadayappan, P.: Tiling Multidimensional Iteration Space for Multicomputers, *J. Parallel and Distributed Computing*, Vol.16, pp.108–120 (1992).
10) 坂田，日向寺，長嶋，佐藤，関口，細矢：ワークステーションクラスタを用いたホモロジー解析，情報処理学会論文誌，Vol.36, No.8, pp.1987–1994 (1995).
11) Sellers, P.H.: On the Theory and Computation of Evolutionary Distances, *SIAM J. Appl. Math.*, Vol.26, pp.787–793 (1974).
12) 渡辺勝正：並列処理概説，pp.139–141，コロナ社，東京 (1991).
13) Xu, Z.: Overheads of Parallel Systems (Sum-

mary), Article 6121 of comp.benchmarks, Xref: etl.go.jp comp.arch:39820 comp.benchmarks: 6121 comp.sys.super: 3407

(平成7年 9 月1日受付)
(平成7年12月8日採録)



坂田 聡子

昭和46年生．平成6年お茶の水女子大学理学部情報科学科卒業．平成8年同大学院理学研究科情報科学専攻修士課程修了．並列分散処理，計算機の性能評価の研究に興味を持つ．

長嶋 雲兵（正会員）

昭和30年生．昭和58年北海道大学大学院博士課程後期修了．理学博士．同年岡崎国立共同研究機構分子科学研究所助手．平成4年お茶の水女子大学理学部情報科学科助教授，現在に至る．理論化学，並列分散処理，性能評価の研究に従事．日本化学会，日本応用数理学会，IEEE 各会員．

佐藤 三久（正会員）

昭和34年生．昭和57年東京大学理学部情報科学科卒業．昭和61年同大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成3年より，通産省電子技術総合研究所勤務．現在，同所情報アーキテクチャ部計算機方式研究室主任研究官．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術等の研究に従事．情報処理学会，日本応用数理学会会員．

関口 智嗣（正会員）

昭和57年東京大学理学部情報科学科卒業，昭和59年筑波大学大学院修了．同年電子技術総合研究所入所．以来，データ駆動型スーパーコンピュータ SIGMA-1 の開発等の研究に従事．現在，言語システム研究室主任研究官．科学技術計算用並列数値アルゴリズムと次世代スーパーコンピュータ性能評価技術に興味を持つ．市村賞受賞．日本応用数理学会，ソフトウエア科学会，SIAM 各会員．

細矢 治夫（正会員）

昭和10年生．昭和34年東京大学理学部化学科卒業．昭和39年同大学院博士課程修了．理学博士．同年理化学研究所研究員．当時の研究テーマ：分子の電子構造と反応機構の理論的研究．昭和42～43年米国ミシガン大学博士研究員（ロドプシンの光化学反応）．昭和44年お茶の水女子大学理学部化学科助教授．現在同学部情報科学科教授．現在の研究テーマ：数理化学・情報化学・化学教育（グラフ理論の化学への応用，巨大分子の電子構造の理論的研究，多面体の数理的解析）．著書：「化学反応の機構」，「構造と物性」，「量子化学」，「化学をつかむ」，「絵解き量子化学入門」，「光と物質—そのミクロな世界」．加入学会：本会，日本化学会，化学ソフトウェア学会，高次元科学会，米国物理学会，国際数理化学会，他．