ワークステーションクラスタを用いたホモロジー解析

坂田聡子[†1] 日向寺祥子[†2] 長嶋雲兵[†3]
関口智嗣[†4] 佐藤三久[†4] 細矢治夫[†3]

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．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）において，必要となる主記憶容量と計算時間は配列の長さ N の2乗に比例するため，現実的に計算できる配列の大きさの限界は主記憶容量によって決まる．本研究ではその計算可能次数の拡大および計算時間の縮小を図るため，この方法の並列化を行った．Toshiba AS4040(SUN4 ipc) 30台によるワークステーションクラスタを用いた実験では，データ列の計算可能次数上限を大幅に拡大することが可能となった．さらに，1台による実行時間から期待される台数倍の性能が並列化により得られた．

Parallel Homology Analysis Using Workstation Cluster

SATOKO SAKATA,[†1] SACHIKO HYUGAJI,[†2] UMPEI NAGASHIMA,[†3]
SATOSHI SEKIGUCHI,[†4] MITSUHISA SATO [†4] and
HARUO HOSOYA [†3]

Homology analysis of protein has an important role in biology. In the homology analysis, required memory size and computing time proportionally increase with the square of the number of amino acids in protein. Therefore the size of protein for homology analysis is limited by the available main memory size. Namely, the upper limit is usually less than about 10^4 on an available scalar computer. In order to expand the limit of size and to carry out the analysis effectively, we parallelized the program for homology analysis using a message passing library: TCGMSG, on workstation cluster where 30 Toshiba AS4040s are connected by Ethernet. By parallelization, the possible size became several hundred times larger than that by the sequential version of program and linear speed-up has been observed.

1. はじめに

ホモロジー (homology) とは，類似性，相同性という意味の言葉である．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盛んに行われてきた．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[4),5),9)]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 と呼んでいる．この NWS アルゴリズムはホモロジー解析の方法として広く使われており，これがホモロジー解析の1つの指標として定着している．

この DP においては，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 の2乗に比例する主記憶および CPU 時間が必要となる．このように 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それは現実的ではない．

本論文では，DP の現実的な計算可能サイズの拡大

†1 お茶の水女子大学大学院理学研究科
Ochanomizu University
†2 東海大学電子計算センター
Tokai University
†3 お茶の水女子大学理学部情報科学科
Ochanomizu University
†4 電子技術総合研究所
Electrotechnical Laboratory

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．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った．さらに並列化したDPを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ので報告する．

2. ダイナミック・プログラミング法

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．

DPでは，ホモロジー・スコア行列 F 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．

（例）文字列数が9である二つの文字列を
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

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

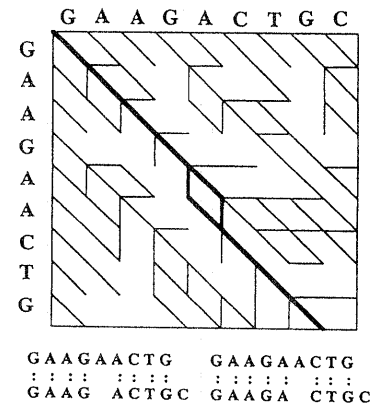


図1 ホモロジー・スコア行列 F とその経路図
Fig. 1 Homology score matrix F and path graph.

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，その和を取る．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，上端の文字列を一文字ずつ比較していくわけであるが，その時の計算方法は以下のようになる．

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

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

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が最良のホモロジー・スコアとなる．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・スコアの逆数を取ったもので表すことができる．従って，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. ダイナミック・プログラミング法の並列化

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ワークステーションクラスタ上での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2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伴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

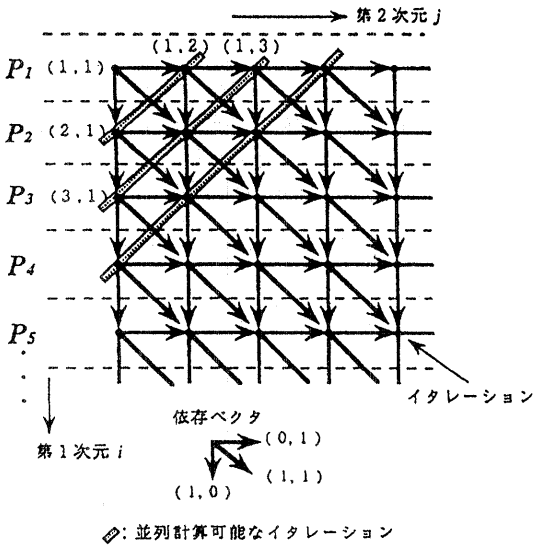


図2 ホモロジー・スコア計算におけるイタレーション空間
Fig. 2 Iteration space in dynamic programming method.

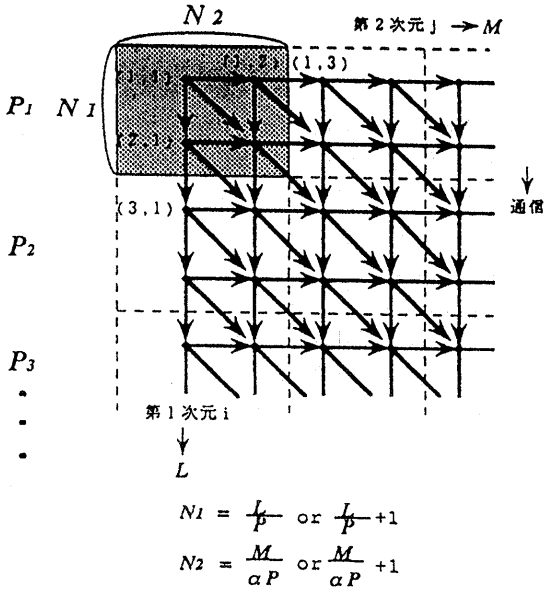


図3 ホモロジー・スコア計算におけるタイリング
Fig. 3 Tiling and iteration space in dynamic programming method.

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[10]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．

(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[6]．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[3),6),7)]．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タを $d=(d_1, d_2)$ とする．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タの第1成分は0でないものとする．なお，依存距離ベクタの一般的性質として，第1成分は0でなければ正である．依存距離ベクタが複数ある場合は，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，このベクタ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．従って，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時には，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一行と左のタイルの一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タは $d=(1,0)$ 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．従って，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，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ということを考慮すると，1タイルの大きさはできるだけ小さい方が望ましいので，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回の通信当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押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．

3.1 タイルサイズと実行時間の関係

ここで，DP の並列化プログラムにおける最適タイルサイズを決定するため，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3に示すように，$P_1, P_2, \ldots$ の各プロセッサに割り当てる．$mod(L, P) \neq 0$ 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．$mod(M, \alpha P) \neq 0$ のときも同様である．ここでタイルサイズを L/P，$M/(\alpha P)$ とすると，1タイル当たりの実行時間は，

$$T_{tile} \cong \frac{L}{P} \times \frac{M}{\alpha P} \times T_e + \frac{M}{\alpha P} \times T_t + T_s \qquad (2)$$

と表せる．第1項は，1タイル当たりのプロセッサ内演算時間である．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) 当たりのプロセッサ内演算時間を表す．第2項の T_t は1ワード当たりの転送時間を表し，第3項の T_s はプロセッサ間通信の起動時間を表す．つまり，第2，3項は1タイル当たりの通信時間である．

1番目のプロセッサ起動から最終プロセッサ起動までに実行されるタイル数は $P-1$ 個となる．また，最終プロセッサが実行するタイル数は αP 個である．結局，全実行時間 T は，P 台のプロセッサによる並列環境生成および終了にかかる時間を T_{inp} とすると，

$$\begin{aligned} T &\cong T_{inp} + T_{tile} \times ((P-1) + \alpha P) \\ &\cong T_{inp} + \left((P-1) \cdot \frac{LM}{\alpha P^2} + \frac{LM}{P}\right) T_e \\ &\quad + \left(\frac{(P-1)M}{\alpha P} + M\right) T_t \\ &\quad + ((P-1) + \alpha P) T_s \end{aligned} \qquad (3)$$

となる．ただし，このモデルでは，プロセッサ間通信の起動が完全に並列実行されることを仮定している．実際には，プロセッサ間通信の起動の並列実行は保証されないので，プロセッサ間通信の起動時間 T_s にかかる係数 C_{Ts} は，一般に次のような式で表すべきである．

$$C_{Ts} = \delta \cdot ((P-1) + \alpha P) + (1-\delta) \cdot \alpha P(P-1) \qquad (4)$$

ここで，δ はプロセッサ間通信の起動が並列実行される(重複される)割合を示し，プロセッサ間通信の起動が重なりあって完全に並列実行されると仮定すると，

$$C_{Ts} = ((P-1) + \alpha P) \qquad (5)$$

であり，プロセッサ間通信の起動が並列実行されないと仮定すると，

$$C_{Ts} = \alpha P(P-1) \qquad (6)$$

と見積もることができる．従って，式(3)は，一般に次のような式で表される．

$$\begin{aligned} T &\cong T_{inp} + \left((P-1) \cdot \frac{LM}{\alpha P^2} + \frac{LM}{P}\right) T_e \\ &\quad + \left(\frac{(P-1)M}{\alpha P} + M\right) T_t \\ &\quad + C_{Ts} \cdot T_s \end{aligned} \qquad (7)$$

次に粒度制御パラメータ α を変化させたときの T の最小値を求める．式(7)を α で微分し，$dT/d\alpha = 0$ を解くことにより，T を最小にする α は，

$$\alpha \cong \sqrt{\frac{(P-1)(LT_e + PT_t)M}{(\delta P + (1-\delta)P(P-1))T_s P^2}} \qquad (8)$$

であり，この値を式(7)に代入することにより各々の場合の最小の実行時間 T_{opt} が得られる．

4. 並列化の効果

DP について並列化を行った結果を以下に示す．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．用いたメッセージパッシングライブラリは TCGMSG[2] である．TCGMSG は化学の分野で非常によく使われており，その移植，インストールおよびインプリメントが非常にしやすい[8]．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)/22 MB，30台である．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．ここでは，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．

図4 はプロセッサ数 P を増加させた時の，ホモロジー・スコア行列の計算可能な問題のサイズの上限の変化を表している．ここでは，$\alpha \geq 10{,}000$ とした．

我々のプログラムにおける各 α に対する計算可能次数 N の式は，

$$N=\frac{\alpha P^2}{2}\left(-A+\sqrt{A^2+\frac{-4(2P+3)+M_{lim}}{\alpha P^2}}\right)$$

$$where \quad A=\frac{5\alpha+\alpha P+8}{4\alpha P} \qquad (9)$$

である．この式は，我々の作成したDPプログラムで必要な配列のメモリ領域の総和から導出したものである．ここで，M_{lim} は1台のマシンのメモリ上限 (byte単位) を示す．図4より，各プロセッサの負担する行列の領域をタイリングにより縮小したので，求めるべき行列全体の計算可能次数は並列化により大幅に上昇し，しかも α の値を大きくするほどその計算可能次数上限が大きくなっていることがわかる．本研究で用いた30台のプロセッサによる計算可能な行列の次数は約 10^7 であり，これは通常の大規模文字列および生物学における塩基配列のホモロジー解析に十分対応できる大きさであるといえる．巨大な塩基配列のホモロジー解析は一般に専用並列計算機を用いて行われることが多いが，このように少ない記憶容量のワークステーションクラスタでも実行可能である．つまり，1台でのみ実行することを想定したDPのプログラムに比べ，並列化に対応したDPのプログラムはホモロジー・スコア行列の次数を大幅に拡大できるということである．しかし，数百字数程度の比較的短いデータ文字列を扱う場合，タイリングを行わず1台でのみ実行することを想定したプログラムのほうが実行時間は短い．例えば，$L=M=500$ のとき，タイリングを行わない逐次プログラムの実行時間は5.5(sec) ほどであるが，タイリングを行う並列化プログラム ($\alpha=30$, $P=10$) の場合，実行時間は約7.7(sec)と遅くなっている．従って，タイリングの手法はデータ文字列のサイズが大きいときにより有効である．

次に，タイルサイズの変化に伴う全実行時間 T の変化について，実測値と，前章で求めた理論値との比較を**図5** に示す．図5(A)は $L=M=4000$，図5(B)は $L=M=10{,}000$ であり，それぞれ $P=9$ とした．理論値の計算において，$P=9$ の並列環境生成および終了のた

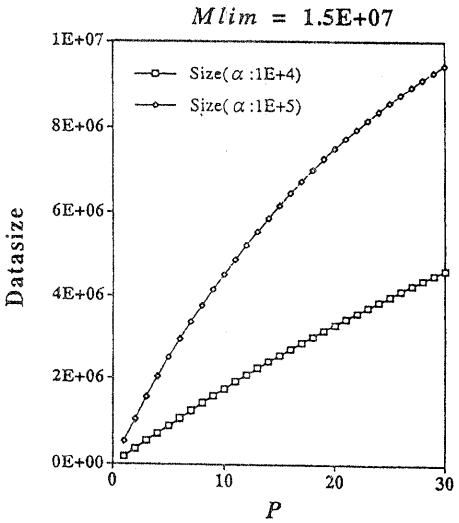


図4 並列化による行列計算可能次数の拡大
Fig. 4 Upper limit of executable matrix size.

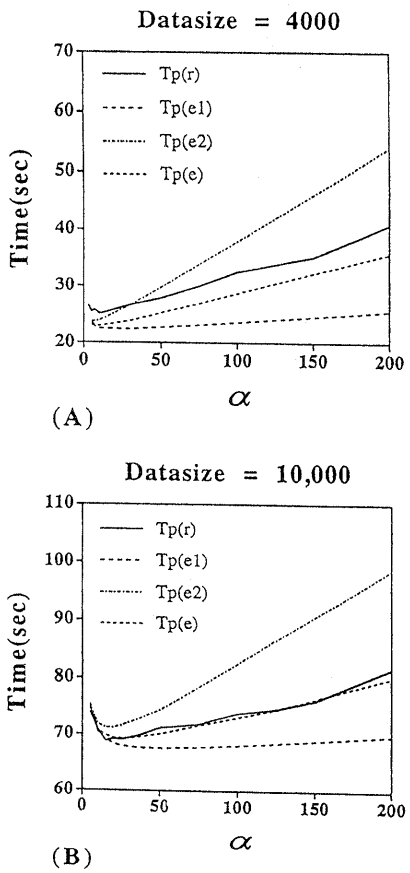


図5 タイルサイズと実行時間の関係
Fig. 5 Relationship of elapsed time and α.

めの時間 T_{in9} は本実験とは別に実測した実行時間の平均値 $T_{in9}=13.0$(sec) を用いた．また，1イタレーション（1要素）当たりの演算時間 T_e は，プロセッサ1台での全実行時間から，1台での並列環境生成および終了に要する時間 T_{in1} を除いたものをイタレーション数で割った値 $T_e=4.73\times10^{-6}$(sec) を用いた．また，通信起動時間 T_s および1ワード当たりの転送時間 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から算出した．ここでは $T_s=2.28\times10^{-3}$(sec)，$T_t=3.76\times10^{-6}$(sec) としている．本研究の環境では，T_e, T_s, T_t の値は，T_e, T_t が同じ 10^{-6} のオーダーであるのに対し，T_s は，その千倍である 10^{-3} のオーダーであることがわかる．

図5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(秒)を表す．$T_p(r)$ は T の実測値を，$T_p(e1)$ は式(5)の場合，つまりプロセッサ間通信の起動が完全に並列実行されると仮定したときの T の理論値を示し，$T_p(e2)$ は式(6)の場合でプロセッサ間通信の起動が並列実行されないと仮定したときの T の理論値を示す．また，$T_p(e)$ は，式(4)の $\delta=0.64$ の場合，つまりプロセッサ間通信の起動が約9/14の割合で並列実行されると仮定したときの T の理論値である．このように δ をおくと，理論値と実測値の傾きがほぼ一致するので，プロセッサ間通信の起動（反対角線上のタイル計算の並列実行）が，本研究の環境では約64%の割合で同時に実行されているということが予測される．さらに，図5(A)，図5(B)のどちらにおいても理論値 $T_p(e)$ と実測値 $T_p(r)$ は，ほぼ一致しており，$L=M=4000$(図5(A)) のとき $\alpha=10$ で，$L=M=10{,}000$(図5(B)) のとき $\alpha=26$ で，それぞれ最小の T 値をとっている．$L=M=4000$ のとき，理論値と比較して実測値の T の方が大きくなるのは，通信のオーバヘッドの見積りが悪いことによるものと考えられる．実行時間 T を最小にする α の見積りがほぼ実測に一致するので，T_{inp}, T_e, T_s, T_t を別の方法で測定しておけば，L, M, P を固定し，α の値を1，2点動かしたときの実行時間を求めるだけで，式(7)より δ の値を求めることができ，実行時間の予測が可能になる．δ の値をより正確に求める場合は，L, M や α の値を幾つかとって実行時間を測定し，理論式に代入することで可能となる．従って，式(8)により実行時間を最小にする最適の α を決定することができる．この式は，本論文で用いたワークステーションクラスタのように，シンプルなネットワーク構造をもつ並列化環境に特に適している．

図5に示した例の場合，配列の長さ L, M は10,000以下と小規模であり，メモリに十分余裕があるため T を最小にする $\alpha\leq26$ でも計算可能であるが，実際ホモロジー解析の対象となるのは1Mワード以上の大規模なデータ文字列である．大規模なデータ文字列のホモロジー解析を限られたプロセッサ数で行うことを考慮すれば，α の値が必要以上に小さいとメモリ不足となり，実行不可能となる．つまり，データのサイズによって α の値の下限が決定される．本研究で作成したプログラムにおいては，α は式(9)より，下記のような範囲の数値をとる必要がある．

$$\frac{4N(N+2P)}{P(-8P^2+(M_{lim}-12-N)P-5N)}\leq\alpha\leq\frac{N}{P} \tag{10}$$

従って，サイズの大きい文字列をデータとする場合は，α の値を十分大きくして，各プロセッサの負担する1タイルの大きさをメモリに入るように小さくする必要がある．

このような手法において通信コストの削減により計算効率を最大限に向上させることは，データ文字列のサイズにより α の値が制限されるので難しいが，並列化のためのタイリングの際，各タイルの列数が小さくなると効率はシステムの持つ通信の性能に依存して低下するので，α の値を可能な範囲で調節して，タイル幅が必要以上に小さくならないよう考慮する必要がある．つまり，α の下限の制限下で，最適の α を用いることで「最大並列度の向上」と「通信コストの削減」のトレードオフをはかることが重要である．ここで，最適の α は次のような式で決定される．

$$\alpha=max\left(\sqrt{\frac{(P-1)(LT_e+PT_t)M}{(\delta P+(1-\delta)P(P-1))T_sP^2}},\ \frac{4N(N+2P)}{P(-8P^2+(M_{lim}-12-N)P-5N)}\right) \tag{11}$$

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 $S(P)$ の関係グラフを**図6**に示す．このとき，データサイズは $L=M=10{,}000$ および $50{,}000$ 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 $S(P)=P$ という直線を示す．$L=M=10{,}000$ のときは期待される線形の性能向上をほぼ実現しているが，20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

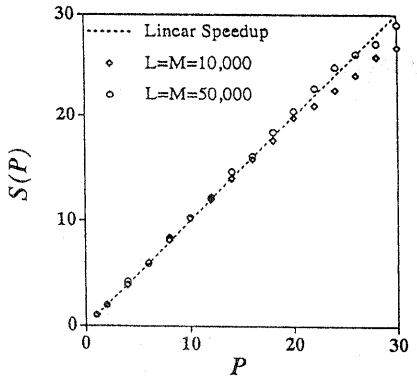


図6 並列化によるプロセッサ数 P と速度向上率 $S(P)$ の関係
Fig. 6 Speedup effect of parallel computation.

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 $L=M=50{,}000$ のときは，線形の性能向上をほぼ実現している．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る．さらに，$L=M=10{,}000$ では $P=8\sim12$ で，$L=M=50{,}000$ では $P=15\sim25$ でスーパリニアスピードアップが実現されている．$L=M=10{,}000$, $50{,}000$ の各サイズについて，実行するプロセッサ数の増加に従い各プロセッサの扱う問題のサイズが縮小されるため，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，計算効率が向上するが，一方でプロセッサ数の増加に伴う通信のオーバヘッドも増加する．従って，この計算効率と通信のオーバヘッドの釣合が非常によくとれた結果がスーパリニアスピードアップであり，それがデータサイズの小さい $L=M=10{,}000$ と大きい $L=M=50{,}000$ のときに，上記の数字のような結果として現われていると考えられる．

プロセッサ数に比例させて問題のサイズ(文字列長)を大きくした時の実行時間への影響についても測定を行ったので，**表1**に結果を示す．ここで，$\alpha=10$ とした．

ここで，T は式(7)より算出し，プロセッサ間の並列起動が先の観測での結果，すなわち64%の割合で行われると仮定したときの理論値である．$L(M)$ を n 倍にし，P もそれに合わせて n 倍にすると1台の処理するタイルサイズは変わらないが，タイル数が n 倍となり，1台当たりの実行時間は1台の時のそれの n 倍の時間となる．表1に示したように問題のサイズに比例させてプロセッサ数を大きくすると，全体の処理のWall clock time値は台数倍以上になっているが，これは $(P-1)$ 個のタイルの処理時間が加算されるためである．この表により，データサイズの変化に対しても，式(7)が良い見積りを与えることがわかる．

表1 問題のサイズの影響
Table 1 Matrix size dependency of wall clock time (sec).

Size (L,M の長さ)	P	Wall clock time	$T_{\delta=0.64}$
1000	1	5.1	5.3
2000	2	11.6	12.1
3000	3	18.4	19.1
4000	4	26.5	27.0
5000	5	33.2	33.9

5. 結　論

DPの並列化の2つの目的，ホモロジー解析可能なデータ文字列数の拡大およびその実行時間の縮小が実現された．さらに，DPのウェーブフロント法へのタイリングの適用において，実行時間を最小にする最適タイルサイズを決定する方法も示し，実行時間の実測値との比較によりその妥当性を示した．これにより，一般の大規模文字列およびもともとホモロジー解析のデータであった生物学での大規模塩基配列の解析にも，コストの低いワークステーションクラスタ上でDPが適用でき，巨大な文字列のホモロジー解析が，最小の実行時間で実行可能であることを示せた．また，このDPをワークステーションクラスタ上に実現することで，少ない記憶容量のワークステーションから構成されるクラスタでも，台数の増加およびそれらのメモリの有効利用により大きなサイズの問題を扱うことができることが示せた．なおかつ台数に比例した速度向上をはかることも可能であることが示せた．

このように並列計算は性能向上が期待されるばかりでなく，CPUが同時に複数台利用できることの速度向上に加え，並列化アルゴリズムにより利用できる実メモリが増加し，巨大なメモリを必要とする大規模計算を実際に実行することができる．

謝辞 本研究を行うに当たり，様々な御討論および御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日本電気(株)の妹尾義樹博士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) Harrison, R. J.: Portable Tools and Applica-

tions for Parallel Computers, *Int. J. Quantum Chem.*, Vol. 40, pp. 847-869 (1991). (TCGMSG is available via electronic mail from ftp.tcg.anl.gov.)
3) 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).
4) 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).
5) 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).
6) 太田 寛，斉藤靖彦，海永正博，小野裕幸：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，*JPSJ SIG Notes*, Vol. 94, No. 68, pp. 13-20 (1994).
7) Ramanujam, J. and Sadayappan, P.: Tiling Multidimensional Iteration Space for Multicomputers, *J. Parallel and Distributed Computing*, Vol. 16, pp. 108-120 (1992).
8) 関口智嗣，長嶋雲兵，日向寺祥子：ワークステーションクラスタとメッセージパッシングライブラリ，情報研報，HPC-47-3 (1993).
9) Sellers, P. H.: On The Theory and Computation of Evolutionary Distances, *SIAM J. Appl. Math.*, Vol. 26, pp. 787-793 (1974).
10) 渡辺勝正：並列処理概説，pp. 139-141，コロナ社，東京 (1991).

(平成6年12月5日受付)
(平成7年5月12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．現在，同大学大学院理学研究科情報科学専攻修士課程2年．並列分散処理，計算機の性能評価の研究に興味をもつ．

日向寺祥子（正会員）

昭和45年生．平成5年お茶の水女子大学理学部化学科卒業．平成7年同大学大学院理学研究科修士課程修了．同年東海大学電子計算センター勤務，現在に至る．理論化学，特に生体分子に対する計算化学，およびその並列分散処理に興味を持つ．

長嶋 雲兵（正会員）

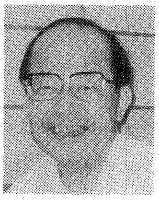
昭和30年生．昭和58年北海道大学大学院博士課程後期修了．理学博士．同年岡崎国立共同研究機構分子科学研究所助手．平成4年お茶の水女子大学理学部情報科学科助教授，現在に至る．理論化学，並列分散処理，性能評価の研究に従事．日本化学会，日本応用数理学会，IEEE各会員．

関口 智嗣（正会員）

1982年東京大学理学部情報科学科卒業．1984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．

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．日本応用数理学会会員．



細矢 治夫（正会員）

1936年生．1959年東京大学理学部化学科卒業．1964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．1967-68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)．1969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－」．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，国際数理化学会ほか会員．