アミノ酸配列のマルチプルアライメントにおける反復改善過程の並列化とA*アルゴリズムの適用

十時 泰[†,☆] 秋山 泰[††] 鬼塚 健太郎[††]
野口 保[††] 斎藤 稔[†††] 安藤 誠[††]

タンパク質のアミノ酸配列のマルチプルアライメントの問題は，アミノ酸残基の保存や置換，欠失や挿入に一定の指標を与え，その総和（sum-of-pairs）が最大となるものが最も確からしいアライメントであるというモデル化が現在主流になっている．このモデルは，総和の最大化に関する組み合わせ最適化問題を解くこととなる．大規模の問題を計算機で高精度に解くには，組み合わせの数が爆発するため，ヒューリスティクスの導入が必要となり，アライメントの精度と計算時間との間にトレードオフの関係が存在している．そのため実用的には従来から近似的な解法がとられてきた．代表的な近似解法は，ツリーベース法であるが，解の精度は必ずしも十分ではなかった．我々は新しい戦略として反復改善法を拡張し，最良優先探索の効率的な近似化を図った上で，最良優先探索における近傍探索を並列実装した．さらに，A*アルゴリズムを適用して探索空間の効率的な刈り込みを実現した．これらの改良の結果，大規模のマルチプルアライメントの問題を高精度に，現実的な計算時間内で得ることを可能とした．

Multiple Protein Sequence Alignment Using Parallel Iterative Algorithm and A* Algorithm

Yasushi Totoki,[†,☆] Yutaka Akiyama,[††] Kentaro Onizuka,[††]
Tamotsu Noguchi,[††] Minoru Saito[†††] and Makoto Ando[††]

Since the multiple sequence alignment problem requires enormous calculation time, one is faced with a trade-off between computation time and the quality of alignment. To date, although several approximation methods have been proposed, the quality of alignments produced by the previous methods is limited. As a new strategy, we employed an iterative scheme with best-first search, and parallelized its search step. Furthermore we implemented the A* pruning algorithm instead of dynamic programming, to drastically reduce the search space. As a result, our new parallel system enables biologically accurate multiple sequence alignment to be performed within reasonable calculation time.

1. はじめに

マルチプルアライメントは，DNA／タンパク質の複数の配列の類似する部分を縦に揃えて並べ合わせる操作で，DNA／タンパク質配列の相同性解析の基本的手法として，分子生物学の様々な分野で利用される重要な処理である[1)]．マルチプルアライメントの結果からDNA／タンパク質配列の“保存領域”が同定され，DNA／タンパク質の未知の機能や構造を既知情報から予測することが可能となる．また，進化系統樹もマルチプルアライメントに基づいて作成される．DNA／タンパク質の機能・構造の研究や進化系統樹の研究には，精度の良いマルチプルアライメントが必要とされている．

従来，マルチプルアライメントの作業は，生物学者がエディタソフトなどを用いて手作業で行なうことが多く，その編集作業には，数週間から1カ月かかると言われており，非常に手間と時間のかかる作業である．そのため，その手間と時間を節約するための自動化が望まれているが，いまだに高精度かつ高速なシステムは存在せず，近似的な解法で解かれたマルチプルアラ

† 株式会社情報数理研究所
Information and Mathematical Science Laboratory, Inc.
☆ 現在，理化学研究所ゲノム科学総合研究センター
Presently with RIKEN Genomic Sciences Center
†† 技術研究組合 新情報処理開発機構
Real World Computing Partnership
††† 弘前大学理工学部
Faculty of Science and Engineering, Hirosaki University

イメントから, 生物学者がさらに手間と時間をかけて編集を行なっているのが現状である.

タンパク質のアミノ酸配列のマルチプルアライメントの問題を計算機で高精度に解くために, 従来からいろいろなモデルが提唱されてきた. 現在はアミノ酸残基の保存や置換, 欠失や挿入に一定の指標を与え, その総和（sum-of-pairs）が最大となるものが最も確からしいアライメントであるというモデル化が主流になっている[2]. このモデルは, 総和（sum-of-pairs）の最大化に関する組み合わせ最適化問題を解くこととなる. 大規模の問題を計算機で高精度に解くには, 組み合わせの数が爆発するため, アライメントの精度と計算時間との間にトレードオフの関係が存在している. そのため実用的には従来から近似的な解法がとられてきた. 代表的な近似解法は, マルチプルアライメントをペアワイズアライメントの組み合わせとして近似する「ツリーベース法」[3)~5)]（progressive method:累進法）であるが, この方法は比較される配列の類似性が低いと, 初期段階の誤りが増幅される傾向があり, 解の精度は必ずしも十分ではなかった[1].

広沢と十時ら[6]は, グループ間でのペアワイズアライメントを繰り返して行ない,「ツリーベース法」の誤りを反復的に改善する「反復改善法」[7),8)]に注目し, 上記の「ツリーベース法」と「反復改善法」を組み合わせた.「反復改善法」は, Bergerら[7]が提案し, その後, 後藤[8]が, ギャップコスト計算を忠実に行なう方法で再提案したものである. また「隠れマルコフモデル」を用いた方法[9]も, 広い意味では「反復改善法」の一種とみなせることが知られている[10]. 広沢と十時ら[6]は,「反復改善法」のギャップコストの計算に後藤のAlgorithm C[8]を適用し, 後藤の一般化プロファイル法[11]も取り入れて, ギャップコスト計算の高精度化と高速化を実現した.

我々は, 上記の反復改善法に解空間の近傍解の中から最も良い解を選ぶ最良優先探索[1]を適用し, この近傍探索を並列に行なわせることとして, PVMライブラリとMPIライブラリを用いて, それぞれで汎用の並列計算機（Sun SPARCcenter 2000E, SGI Power-Challenge, Hitachi SR2201）上に並列実装した. その結果, 大規模のマルチプルアライメントを高精度に, 現実的な計算時間内で得ることを可能とした. このアルゴリズムは石川と十時ら[1]によって過去に発表されているが, その時の実装は,（財）新世代コンピュータ技術開発機構で開発された専用の並列計算機（分散メモリ型並列計算機PIM/m）上に, 専用の言語（並列論理型プログラミング言語KL1）を用いて行なわれ, プログラムの移植性が悪いものであった. またその計算速度は, 残基数80の9本のアミノ酸配列のマルチプルアライメント計算に3分以上もかかり, 大規模（残基数：100 ~ 1000, 本数：10 ~ 100）のマルチプルアライメントには対応できなかった. PVMやMPIなどの汎用の通信ライブラリを用いて汎用の並列計算機上に並列実装され, 移植性が良くなり, また大規模の問題に対応可能になったのは今回が初めてである. さらに我々は, グループ間でのペアワイズアライメントを求める方法として, 従来の全探索のダイナミックプログラミングに代わり, A*アルゴリズム[12]を適用して探索空間の効率的な刈り込みを行なった. その結果, 解の精度を落とすことなく, さらなる計算時間の短縮に成功した.

現在, 我々の開発した並列マルチプルアライメントシステムは, 64台のパソコンボードを接続したパソコンクラスタ上にも並列実装されて, WWW経由で無料公開されている.

（http://www.rwcp.or.jp/papia/）

2. マルチプルアライメントのモデル化

2.1 マルチプルアライメント

タンパク質の構成要素であるアミノ酸には通常20種類あり, それらを記号で表現するためにそれぞれ異なるアルファベットが割り当てられている. アミノ酸には, 大きさ, 水との親和性, 酸性／塩基性, 極性などの性質があり, どのような性質のアミノ酸がどのような順番に連なっているかで, タンパク質の構造と機能が決まる.[1]

マルチプルアライメントは, 複数の配列の類似する部分を縦に揃えて並べ合わせる操作である. たとえば以下のようなアミノ酸配列が4本あったとする.

```
CCHU    DVEKGKIFIMKCSQCHTVEKGGKHKTGPNL
CCFS    TTGAKIFKTKCAQCHTVKGHKQG
CCZP    DEKKGASLFKTAQCHTVEKGGANKVGPNL
CCCRCF  DAARGEKLRAAQCHTANQGGANGVG
```

ここでは, 左側の見出しが配列の名前で, 右側の文字列がアミノ酸配列を表現する. 最上段の配列の左端からDVEKは, それぞれアスパラギン酸, バリン, グルタミン酸, リシンのアミノ酸残基を意味する. これら4本の配列をアライメントすると, 次のようになる.

```
CCHU    DVEKG-KIFIMKCSQCHTVEKGGKHKTGPNL
CCFS    --TTGAKIFKTKCAQCHTV-KG--HKQG---
CCZP    DEKKGASLFKT--AQCHTVEKGGANKVGPNL
CCCRCF  DAARGEKL---RAAQCHTANQGGANGVG---
        ....G.........QCHT...G.....G...
```

配列のところどころにギャップ "-" を挿入することで，QCHT などの共通文字が同じ列に並べられた．QCHT のように複数の配列で共通な部分文字列を "配列モチーフ" と呼び，タンパク質の重要な部分を指し示していると考えられる．この背景には，タンパク質のアミノ酸配列のうち生理的に重要な部分には遺伝的変異が蓄積しにくいという進化論的考え方[13]がある．

2.2 モデル化

実際のマルチプルアライメントでは，ひとつの列に同一の文字が揃うことは少なく，異なる文字でもそれらが表すアミノ酸の性質が似ていれば同じ列に置くことを許容して処理を行なう．しかしアミノ酸の性質には親水性／疎水性，極性，酸性／塩基性，大きさなど複数あり，その類似性評価も多数の方法がある．現在最も広く使われている類似性評価尺度は，BLOSUM[14] や PAM[15] と呼ばれるマトリクスで，当時知られていたアライメントをもれなく調べ，同じ列に該当アミノ酸残基ペアが並んでいることが偶然に対していかに多いかを数値化したものである．数値は確率の対数値になっているため，それらの足し算は複合事象の共起確率を算出したことに相当する[1]．

本論文のマルチプルアライメントシステムも，この評価尺度を使用している．この評価尺度を基に，マルチプルアライメントのスコアは，すべての組み合わせの配列ペアのすべての位置の残基ペア（または残基とギャップのペア）の評価値 $M(a,b)$ の総和（sum-of-pairs）として与えられる．具体的には，長さ L，本数 N のアライメントの場合は，配列 a_n $(n=1,2,...,N)$ の残基（またはギャップ）を a_{nl} $(l=1,2,...,L)$ とすると，アライメント $A=\{a_n\}$ のスコア $SP(A)$ は以下の式で与えられる．

$$SP(A)=\sum_{i<j}^{N}\sum_{k}^{L}M(a_{ik},a_{jk})$$

アミノ酸残基同士のペアの評価値は類似性評価尺度のマトリクスから算出されるが，残基とギャップとのペアの評価値はギャップコスト関数から算出される．普通，「連続した欠失が一度の出来事で生じうる」という進化論的考え方に基づいて，k 個連続したギャップのコストは $uk+v$ で与えられる（$v\neq 0$：affine gap cost）[16),17]．u,v は一定の罰点で，開始ギャップには $u+v$，延長ギャップには u の罰点が科される．つまり，延長ギャップには開始ギャップより v 少なく罰点を科して，ギャップが連続して入りやすいように数学的な工夫がされている．

このように，マルチプルアライメント問題は，アミノ酸残基の保存や置換に一定の指標を与え，その総和（sum-of-pairs）が最大となるものが最も確からしいアライメントであるというようにモデル化される．つまり，マルチプルアライメント問題を解くことは，アライメントのスコア $SP(A)$ の最大化に関する組み合わせ最適化問題を解くこととなる．

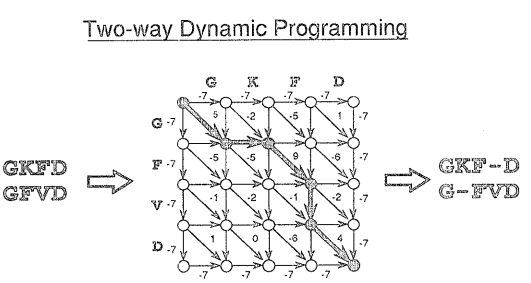

図1 DPによるペアワイズアライメントの解法[1]
Fig. 1 Scheme of pairwise alignment using DP.[1]

3. 従来の手法

3.1 ダイナミックプログラミング

BLOSUM や PAM などのスコアマトリクスとギャップコストのように，アミノ酸残基の保存や置換に一定の評価尺度が与えられれば，その尺度において最適なマルチプルアライメントを厳密に求めるダイナミックプログラミング法（Dynamic Programming: 以下 DP）が知られている．DP は段階的に決定を行なう特徴を持つ最適化問題を解くためのアルゴリズムのひとつである[1]（図1は配列2本のペアワイズアライメントの場合の例）．

配列2本のペアワイズアライメントについて，アライメント全体の評価値を最適化するようなグローバルアライメントや，アライメントのある一部分の評価値を最適化するようなローカルアライメントは，2次元 DP の技術で厳密に解決できる．

Needleman/Wunsch 型のグローバルアライメント[18]と Smith/Waterman 型のローカルアライメント[19]のアルゴリズムの違いについて，2本の配列のペアワイズアライメントを例にして，以下に簡単に説明する．

(1) Needleman/Wunsch 型のグローバルアライメント

2本の配列を $A=a_1a_2...a_n$，$B=b_1b_2...b_m$ として，配列要素 a_i，b_j の類似度を $s(a_i,b_j)$ とする．長さ k の欠失の重みを W_k とする．マトリクス H を以下のようにする．

$$H_{0,0} = 0.$$
$$H_{i,j} = \max\{H_{i-1,j-1} + s(a_i, b_j), \max_{k\geq 1}\{H_{i-k,j} - W_k\}, \max_{l\geq 1}\{H_{i,j-l} - W_l\}\}$$
$$for \ \ 1 \leq i \leq n \ \ and \ \ 1 \leq j \leq m.$$

この操作を $H_{0,0}$ から $H_{n,m}$ まで繰返し，$H_{n,m}$ から選択された位置を逆向きに，$H_{0,0}$ までたどれば（traceback），最適なグローバルアライメントが得られる．

(2) Smith/Waterman 型のローカルアライメント

ローカルアライメントの場合は，前の式を少し変形して，次の式のようにする．

$$H_{k,0} = H_{0,l} = t$$
$$for \ \ 0 \leq k \leq n \ \ and \ \ 0 \leq l \leq m.$$
$$H_{i,j} = \max\{H_{i-1,j-1} + s(a_i, b_j), \max_{k\geq 1}\{H_{i-k,j} - W_k\}, \max_{l\geq 1}\{H_{i,j-l} - W_l\}, t\}$$
$$for \ \ 1 \leq i \leq n \ \ and \ \ 1 \leq j \leq m.$$

つまり，マトリクスの値がある値 t（例えばゼロ）より小さい場合は，強制的に t にしてしまう．それと同時に，t が選ばれたときには，経路をリセットするようにしておくと，リセットされたマトリクスの位置では新たな経路が始まるので，とびとびの断片的な経路ができる[20]．

この操作を $H_{0,0}$ から $H_{n,m}$ まで繰返し，$H_{i,j}$ が最大の位置を探して，その位置から選択された位置を逆向きに，リセットされた位置（$H_{i,j}$ が t）までたどれば（traceback），最適なローカルアライメントが得られる．

また，$H_{i,j}$ がある閾値以上の位置について同じように，traceback が重ならないように行なえば，複数の準最適なローカルアライメントを求めることもできる．

上記の 2 次元の DP を応用すれば，N本の配列のマルチプルアライメントは N次元の DP で原理的には厳密に解決できる．しかし，N次元の DP は，配列の最大長を L とすると，$O(L^N)$ の計算時間とメモリ空間を必要とするので，実用規模の問題（$L = 100 \sim 1000, N = 10 \sim 100$）では計算量が爆発して，現実には計算不可能である．そこで，実用的には従来から近似的な解法がとられてきた．代表的な近似解法は，マルチプルアライメントをペアワイズアライメントの組み合わせとして近似したツリーベース法[3)~5)]である．

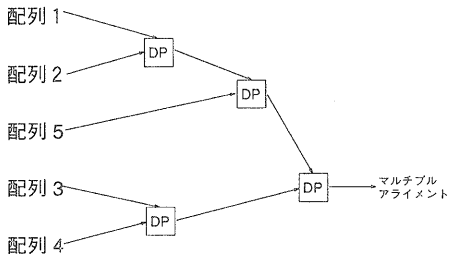

図 2 ツリーベース法（累進法）[3)~5)]
Fig. 2 Scheme of tree-based method.[3)~5)]

3.2 ツリーベース法

ツリーベース法（累進法）[3)~5)]（図 2）は，現在最も広く使われているマルチプルアライメントの解法で，以下のように行なう[2)]．

(1) すべての配列ペアについて 2 次元 DP によるペアワイズアライメントを行ない，配列間距離を求める．

(2) この距離行列から系統樹を作成する．

(3) 系統樹の階層関係に基づいて，隣合った 2 つの葉に相当する配列間のアライメントを求め，得られたアライメントを新たな 1 つの葉とする．この操作をすべての配列が 1 つのアライメントにまとめられるまで繰り返す．

(3) の後半では，得られたアライメントを組み合わせることになり，配列グループ間のアライメントが必要になる．配列間の DP がそのまま使われることも多いが，グループ間の DP[8)]を用いて，グループを複数配列の平均化した “プロファイル” の様に扱った方がより厳密なアライメントが得られる．また，ギャップの分布を考慮した一般化プロファイル法[11)]を組み合わせることにより，より効率的に厳密なアライメントが得られる．

類似した配列同士のアライメントは確実で信頼性が高いので，このように類似度の高い配列から順に組み合わせると，ある程度のアライメントの精度を達成できる．しかし，ツリーベース法では，一度できたアライメントは最後まで残り，後で修正されることがない．特に，比較される配列の類似性が低いと，初期段階の誤りが増幅される傾向があり，解の精度は必ずしも十分ではない[1)]．この欠点に対処したのが反復改善法[7),8)]である．

3.3 反復改善法

反復改善法[7),8)]は一種の山登り探索で，アライメントのスコアが収束するまで，以下の改善サイクルを繰り返す[2)]（図 3）．

(1) 現在のアライメントを 2 つのグループに分ける．

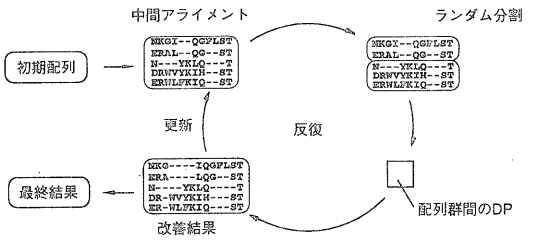

図3 山登り探索の反復改善法[7),8)]
Fig. 3 Scheme of hill-climbing iterative method.[7),8)]

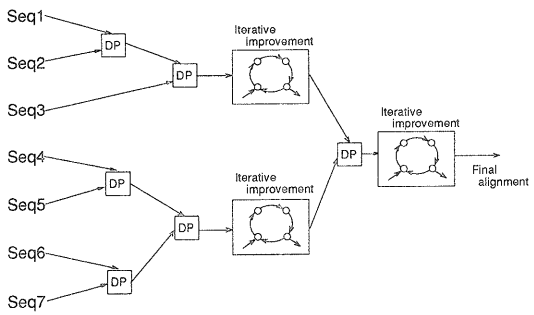

図4 ツリーベース法と(並列)反復改善法の融合[6)]
Fig. 4 Scheme of tree-based iterative method.[6)]

アライメントの本数を N本とすると，可能な分け方は $2^{N-1}-1$ 通りになるが，その中から1つを任意に選ぶ．

(2) 2つのグループ間の最良アライメントをグループ間のDPで求める．

グループ間のDPで求めるアライメントは改悪されることはないので，この改善サイクルは必ず収束する．

ツリーベース法で得られたマルチプルアライメントに反復改善法を適用すると，初期段階の誤ったアライメントが修正され，アライメントの精度を大幅に改善できる（ツリーベース反復改善法[6)]）（図4）．

しかし試行錯誤的な山登り探索による反復改善法は，大規模の問題になると，収束までの改善サイクル数が膨大になり，計算時間が障害となる[1)]．N本の配列からなる配列グループの分割の方法は $2^{N-1}-1$ 通りであるから，配列本数が増えると，分割の方法は指数オーダーで増える．またアライメント過程の後半になると，ほとんどの分割が改善に寄与しないので，無駄な計算を繰り返すことになる．これらの欠点を軽減したのが石川，十時らによる次の並列反復改善法[1)]である．

3.4 並列反復改善法

並列反復改善法[1)]では最良優先探索を行なう．最良優先探索は解空間において注目する状態の近傍を全て探索し，そのうちから最も良い状態へ遷移する探索法

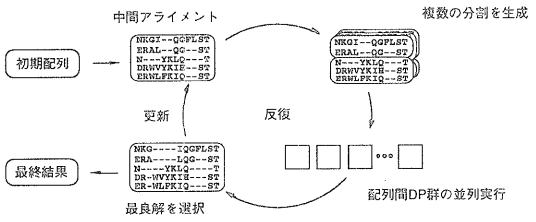

図5 最良優先探索の並列反復改善法[1)]
Fig. 5 Scheme of best-first iterative method.[1)]

である．最良優先探索に基づく並列反復改善法は，近傍探索の部分を並列化したもので，最もナイーブな方法ではアライメントのスコアが収束するまで以下の改善サイクルを繰り返す（図5）．

(1) 現在のアライメントを2つのグループに分割する．アライメントの本数を N本とすると，可能な分け方は $2^{N-1}-1$ 通りになるが，その全ての2分割を作成する．

(2) 2つのグループ間の最良アライメントをグループ間のDPで求める．これを全ての2分割について並列に行なう．

(3) それらの結果を比較し，スコアの最も良いアライメントを選ぶ．

前節で述べた反復改善法と同様に，(2)(3)の前後でアライメントのスコアが改悪されることはないので，スコアは必ず収束する．また，ツリーベース法で得られたマルチプルアライメントに並列反復改善法を適用したツリーベース並列反復改善法[1),6)]（図4）は，前節で述べたツリーベース反復改善法[6)]よりも安定して精度のよい解が得られ，反復改善過程を並列化することにより，より高速に解が求まる．

しかし(1)における2分割の数は配列の本数の増加に対して指数的に増加する．大規模のアライメント問題では，探索空間を効率良く制限するためのヒューリスティクスとして，次の限定分割[1)]を導入して2分割の数を効率良く制限する．

3.4.1 1本抜き限定分割，1〜2本抜き限定分割

複数配列を2つのグループに分割する時に，片方のグループの配列数を，1本（1本抜き限定分割[1)]），または1〜2本（1〜2本抜き限定分割[1)]）とする．配列 N本のアライメント問題の場合，分割法はそれぞれ N 通りと $N(N+1)/2$ 通りになる．ナイーブな反復改善法の分割法が，配列の本数に対して指数オーダーであったのに対して，それぞれ1乗オーダー，2乗オーダーとなるが，解のスコアはそれほど低下しない．1〜2本抜き限定分割は，1本抜き限定分割よりも，平均して若干スコアの良い解を生成する．こうした限定分割

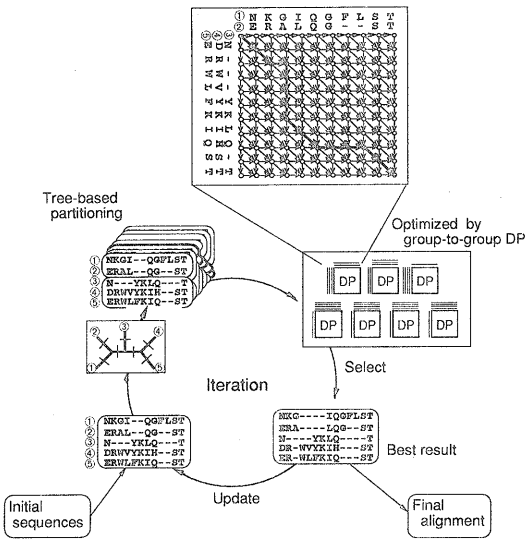

図6 ツリー依存限定分割 並列反復改善法[1)]
Fig. 6 Scheme of best-first iterative method with tree-dependent partitioning.[1)]

が有効なのは, アライメント途中の配列グループは, その配列数が多いとカラムの特徴が平均化されているのに対し, 配列数が少ないとカラムの特徴が比較的よく表れるためと考えられる. カラムの特徴がはっきりしていると, 配列グループ間の DP によるアライメントの最適化が効率良くなされ, スコアの改善が大きい[1)].

3.4.2 ツリー依存限定分割

ツリー依存限定分割[1)]は 1 乗オーダーの分割法で, 1～2 本抜き限定分割をしのぐ効果をあげる. 1～2 本抜き限定の, 1 本抜き限定に対する優位性は, アライメントの過程で良く揃った 2 本が一緒に処理されることにある. そこで, ツリー依存限定分割では, 反復改善の過程で良く揃った配列グループを, 改善サイクルごとに抽出し, それらと残りの配列間でアライメントを行なう（図6）. ツリー依存限定分割では, 各改善サイクルの始めにその時点でのアライメントのツリーを配列の類似性に基づいて作成し, ツリー上で近い配列は, なるべく同じ配列グループに含まれるように, 分割を決める. 具体的には, 得られたツリーに対して, ツリーの枝の任意の 1 か所を切断して分けられる 2 つの配列グループを, 改善の対象とする分割に選ぶ. この分割法はツリーの形状によらず, 配列 N本に対して $2N-3$ 通りある. 末端の枝が切断される場合は, 1 本と残りの分割になるので, ツリー依存限定分割には, 1 本抜き限定分割が含まれている.

4. 並 列 化

4.1 実 装

我々は PVM ライブラリと MPI ライブラリを利用して, それぞれマスター・スレーブ方式でツリーベース並列反復改善法の近傍探索の部分を並列化し, これを Sun SPARCcenter 2000E（PU 数:20, 主記憶:5GB 共有）, SGI PowerChallenge（PU 数:12, 主記憶:2GB 共有）, Hitachi SR2201（PU 数:256, 主記憶:64GB 分散）上に実装した. 近傍探索におけるヒューリスティクスとしてツリー依存限定分割を用いた.

具体的には以下のように行なう.

(1) マスタープロセスは, 使用する PU 台数分のスレーブプロセスを生成する.

(2) マスタープロセスは, 現在のアライメントの 2 分割をツリー依存限定分割で複数組生成し, それぞれの組を各スレーブプロセスに配る. ここでは, マスタープロセスは各スレーブプロセスに対して 1 タスクずつを投げ, 処理が終わったスレーブプロセスに新しい処理を投げるようにする. 各スレーブプロセスは, グループ間の DP によってアライメントの最適化を行なう. これを実行するタスクがなくなるまで行なう.

(3) 計算結果はマスタープロセスに集められ, マスタープロセスは最良のアライメントを選ぶ.

並列反復改善法では, 与えられたアライメントの配列本数で 2 分割の数, すなわち並列に実行可能なタスク数が決まる. 一般的にはタスク数は使用できる PU 数よりも多い. ツリー依存限定分割では 2 分割される 2 つのグループの配列本数に差があり, 並列の粒度のばらつきが大きいが, (2) では PU の負荷を平均化する負荷分散が行なわれている.

4.2 並列化の評価及び考察

129 本のリポカリンファミリーのアミノ酸配列から, タスクの粒度が異なるように, 配列長 100 と配列長 214 の配列長の異なる 2 種類のデータを作成し, それらを用いて並列化効率の評価を行なった（図7, 図8）. 評価は PVM ライブラリを用いたプログラムの方を使用して Hitachi SR2201（256PU）上で行なった（Sun SPARCcenter 2000E（20PU）と SGI PowerChallenge（12PU）は, 使用できる PU 数が少なく, 少ない PU 数での並列化効率には大きな差がでなかったため, 評価結果の詳細は割愛する）. 使用 PU 数 P の並列化効率 E_Pは, 1PU での計算時間を T_1, 使用 PU 数 P での計算時間を T_Pとすると, $E_P = T_1/(T_P P)$ と定義する. ツリー依存限定分割では, 2 分割の分け方

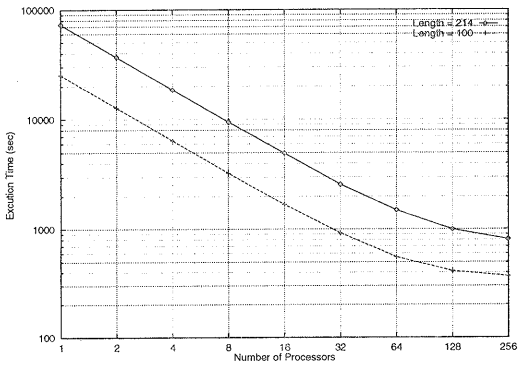

図7 SR2201 上での使用 PU 数と計算時間の関係
(使用 PU 数にはマスタープロセスは含まれない)
Fig. 7 Relation between the number of processors and execution time.

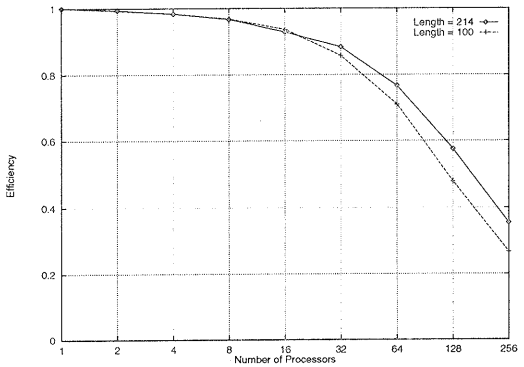

図8 SR2201 上での使用 PU 数と並列化効率の関係
(使用 PU 数にはマスタープロセスは含まれない)
Fig. 8 Relation between the number of processors and efficiency.

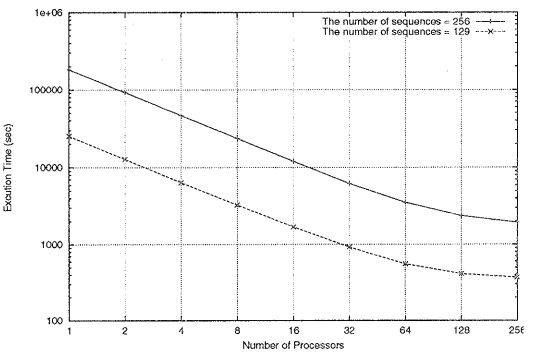

図9 SR2201 上での使用 PU 数と計算時間の関係
(使用 PU 数にはマスタープロセスは含まれない)
Fig. 9 Relation between the number of processors and execution time.

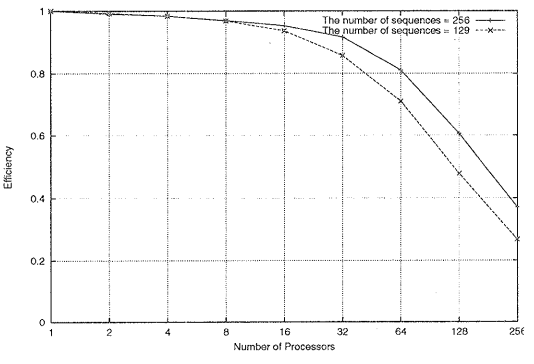

図10 SR2201 上での使用 PU 数と並列化効率の関係
(使用 PU 数にはマスタープロセスは含まれない)
Fig. 10 Relation between the number of processors and efficiency.

は配列 N本に対して $2N-3$ 通りであるので, 129 本に対しては 2 分割の分け方, すなわちタスク数は 255 である.

図 7, 図 8 に示すように, 配列長 214 のデータでは, 1PU 使用した場合約 20 時間かかっていた計算が, 256PU 使用した場合は約 13 分に減り, 約 90 倍の高速化が達成された. 2 種類のデータを比較すると, タスクの粒度の大きいデータ（配列長 214）の方が, 粒度の小さいデータ（配列長 100）よりも並列化効率が良い. これは, タスクの粒度が大きい方がアライメント計算に時間がかかり, 全体の計算時間の対する通信時間の割合が減るためだと思われる. 実用的な問題では, 今回の評価データに比べて配列長が長い場合が多く, タスクの粒度も大きくなるので, 並列化効率はさらに改善されると考えられる.

また 2 種類のデータとも PU 数が 64 以上になると並列化効率が大幅に低下する. その原因の 1 つとしては, タスク数が 255 と少ないことが考えられる. 例えばスレーブの PU 数が 255 の時には 1PU 当たりのタスク数が 1 となり, 前節で述べた並列の粒度のばらつきを抑える負荷分散が機能しないため, 計算の終盤でアイドル状態になる PU が増加することが考えられる. そこで今度は, タスクの粒度が同じでタスク数が多くなるように, 配列長 100 のリポカリンファミリーのアミノ酸配列から配列本数を 256 に増やしたデータを作成し, 並列化効率の評価を行なった（図9, 図10）. 配列本数が N の場合のタスク数は $2N-3$ となるので, 256 本の場合のタスク数は 509 となる. 評価はこれまでと同じように PVM ライブラリを用いたプログラムの方を使用して Hitachi SR2201 上で行なった.

図9, 図10 に示すように, タスク数が多いデータ（配列本数 256, タスク数 509）の方が, タスク数が少ない

表1 各手法のアライメントスコアと処理時間の比較
Table 1 Relation between alignment score and execution time.

手法	スコア	処理時間 (sec)	PU数
ツリーベース法	-1052897	163	1
ツリーベース反復改善法 山登り探索(逐次)	-682549	10251	1
本研究の手法 最良優先探索(並列)	-674054	73224	1
		1493	64
		807	256

データ（配列本数129, タスク数255）よりも並列化効率が良い. タスク数がさらに多いデータでは, 並列化効率はさらに改善されると考えられる. また並列化効率が低下する原因として他には, 1つのマスタープロセスから逐次的に同期通信を行なっているためのボトルネックも考えられ, タスク分配機構の改善も課題である.

5. 従来法との比較及び考察

従来法の「ツリーベース法」[3)~5)]及び「ツリーベース反復改善法」[7),8)]と, 本研究の「ツリーベース並列反復改善法」のアライメントの精度と処理時間の関係を比較した（表1）. 表1では, スコアが大きいほどアライメントの精度が良いことを示す. 従来法の「ツリーベース法」には, 系統樹をUPGMA法（平均距離法）で作成し, 2つのグループ間の最良アライメントをグループ間のDPで求めてマルチプルアライメント形成するプログラムを独自に作成して用いた. マルチプルアライメントのプログラムで現在最も広く使われているClustalW[4)]も「ツリーベース法」を基本にしているが, 生物学的な見地からいろいろな工夫がなされているため, 本論文が用いた「アミノ酸残基の保存や置換に一定の指標を与え, その総和（sum-of-pairs）が最大となるものが最も確からしいアライメントであるというモデル」とは異なるために, 本論文で用いた評価関数では比較できない. 従来法の「ツリーベース反復改善法」では山登り探索を1本抜き限定分割でラウンドロビン方式で逐次に行なった. 本研究の「ツリーベース並列反復改善法」では最良優先探索をツリー依存限定分割で並列に行なった. データは, 129本のリポカリンファミリーのアミノ酸配列（最大配列長214, 平均ホモロジー24%）を使用した.

表1の例に示されるように, 従来法の「ツリーベース法」は処理時間は速いが, 精度が極端に悪い. 本研究の手法が精度が一番よく, 並列化により処理時間も抑えられており, 最も実用的な手法といえる. 本研究の手法は保存領域以外の相同性の弱い領域もかなり正確にアライメントするので, 見かけのスコア差以上に生物学的な価値があり, 特に進化系統樹の研究に非常に有効であると生物学者から評価を得ている.

6. A*アルゴリズムによる探索空間の効率的な刈り込み

ツリーベース並列反復改善法の中で行なっている, 2つのグループ間のペアワイズアライメントは, グループ間の2次元DPの技術で厳密に解決できる[8),11)]. DPを用いると, 最適解を求めるには, 2次元のグラフ（図1）上を網羅的に全探索する必要がある. 全探索DPでは最適解が保証されるが, 無駄な計算が多くなる. ペアワイズアライメントでは, 最適解のパスは2次元グラフの対角線近傍に存在する場合が多いため（配列間の類似性が高い場合）, 対角線から離れた領域を強制的に削除して, 計算量を削減する近似も提案されている. しかしこの方法では, 最適解のパスが実際には対角線から離れた領域に存在する場合（配列間の類似性が低い場合）は, 最適解が求まらず, 解が極端に悪くなる危険性がある. ペアワイズアライメントの解が悪いと, 反復改善法の中で悪い局所解に陥る. 最終的なマルチプルアライメントの精度を良くするには, ペアワイズアライメントで良い解を求めることが必要である.

ここでA*アルゴリズム[12)]を用いると, 2次元グラフ上で必要な部分だけを探索して最適解を求めることができる. 全探索DPと比較すると, 不必要な部分を探索しないので, 処理時間を節約できる. また, 必ず最適解が求まるため, 強制的に探索空間を削除するDPのように, 解が極端に悪くなる危険性がない.

6.1 A*アルゴリズムのペアワイズアライメントへの適用

DPでの探索では, コストの合計が最大になる経路を求めた（3.1節の図1）. 説明上の便宜性から, A*アルゴリズムでの探索では, 2次元グラフの各エッジに与えられたコストの正負を逆転させ, コストの合計が最小になる経路を求める最短経路問題に探索空間を変換することとする.

グラフの任意のノード n から最終ノードまでの最短経路のコストの実際値と推定値をそれぞれ $h(n)$ と $\hat{h}(n)$ とする. A*アルゴリズムでは, 最適解を保証するためには, $\hat{h}(n) \leq h(n)$ が成り立つような $\hat{h}(n)$ を設定する必要がある. このような $\hat{h}(n)$ が与えられると, 展開の候補ノードの中で, 出発ノードからの最短経路のコストに $\hat{h}(n)$ を足し合わせた値が最小のノードを

展開していくと，少ないノード数展開で最短経路が求まる．もしも，$\hat{h}(n) = h(n)$ ならば，不必要なノードを展開せずに最短経路を求めることができるので，$\hat{h}(n)$ を $h(n)$ に近づけることが，効率化の鍵である．

2次元グラフのノード $n(i,j)$ における $\hat{h}$ は，理想的には，次のように定義するのがよい．横，縦方向に配置させる配列の長さをそれぞれ l_x, l_y, 2次元グラフのエッジのコストを $C(i,j)$, ギャップペナルティを P' とすると，

$l_x - i \geq l_y - j$ の時，

$$\hat{h} = \sum_{m=j+1}^{l_y} \min_{i+1 \leq k \leq l_x} C(k,m) + ((l_x - i) - (l_y - j)) \times P'$$

$l_x - i < l_y - j$ の時，

$$\hat{h} = \sum_{m=i+1}^{l_x} \min_{j+1 \leq k \leq l_y} C(m,k) + ((l_y - j) - (l_x - i)) \times P'$$

ただし，エッジのコストの最小値の候補には，ギャップコストが与えられている横（縦）方向のエッジも含めた．斜め方向のエッジの場合と違って，横（縦）方向のエッジを通る場合には，将来必ず縦（横）方向のエッジを余分に通ることになる．そこで，そのコストには縦（横）方向のギャップコストの最小値を加算しなければならないが，ギャップコストの最小値は i,j に依存せず一定なので，ギャップコストの最小値の2倍とすればよい．また，ギャップペナルティ P' もギャップコストの最小値とした．

しかしこのままでは，ノードが展開される度に，$\hat{h}$ の計算に配列長の2乗オーダーの計算量が必要になり，$\hat{h}$ の計算に時間がかかりすぎて，A*アルゴリズムの効果がなくなる．そこで $\hat{h}$ の計算量を削減するために，実際には，$\min_{i+1 \leq k \leq l_x} C(k,m)$ を $\min_{1 \leq k \leq l_x} C(k,m)$ に，$\min_{j+1 \leq k \leq l_y} C(m,k)$ を $\min_{1 \leq k \leq l_y} C(m,k)$ に近似する．この近似によって，k の範囲が i,j に依存しなくなるので，$1 \leq m \leq l_y$ に対する $\min_{1 \leq k \leq l_x} C(k,m)$ と，$1 \leq m \leq l_x$ に対する $\min_{1 \leq k \leq l_y} C(m,k)$ は，あらかじめ前処理として1度だけ計算しておけばよい．またそれと同時に，$\sum_{m=j+1}^{l_y} \min_{1 \leq k \leq l_x} C(k,m)$ と $\sum_{m=i+1}^{l_x} \min_{1 \leq k \leq l_y} C(m,k)$ も，前処理として計算しておけるので，ノードが展開される度に $\hat{h}$ を計算する必要がなくなる．$\hat{h}$ の計算は2乗オーダーであるが，DPの2乗オーダーの計算に比べると非常に軽い計算なので，前処理として1度だけ計算することによって，A*アルゴリズムの効果が現れる．

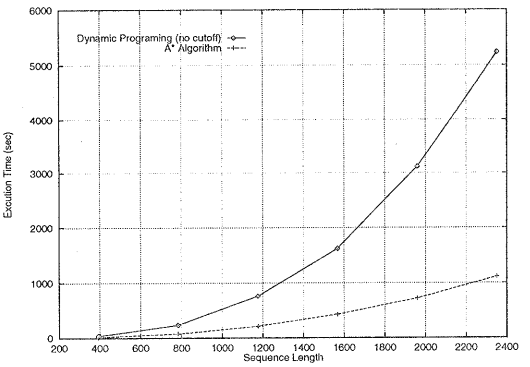

図11 A*アルゴリズムと DP の処理時間の比較
(横軸は配列長，縦軸は計算時間)
Fig. 11 Execution time of A* algorithm compared with DP.

なお k 個連続したギャップのコスト $uk+v$ を単純な比例コスト（$v=0$: linear cost）[12]ではなく，本論文の2.2節の「モデル化」で説明されている切片つきコスト（$v \neq 0$: affine cost）[16),17)]にすると，ギャップコスト計算が複雑化し，A*アルゴリズムにおけるヒープ管理でのメモリ消費量も多くなるが，我々は後者を実現した．比例コストの場合は，ギャップコストはパスに依存せず一定になるので，各ノードには1つのパスを対応させるだけでよい．一方，切片つきコストの場合は，ギャップコストはパスに依存するので，各ノードには縦方向，横方向，斜め方向の3つのパスを対応させなければならない．したがって，切片つきコストの場合は，比例コストの場合に比べて探索空間が3倍に増加し，ヒープ管理のノード数も3倍に増加する．また切片つきコストの場合は，各ノードは，直前までの連続するギャップの数をパスの情報として余分に持たなければならないので，ヒープ管理でのメモリの消費量は大幅に増加する．

6.2 A*アルゴリズムの評価及び考察

2つのグループ間のペアワイズアライメントを A* アルゴリズムと全探索 DP（対角線から離れた領域を削除しない）を用いて行ない，処理時間を比較した（図11）．テスト用のデータは22本のキナーゼ配列（平均ホモロジー 29%）のアライメント結果を，11本ずつの2つのグループに分けて使用した．図11に示すように，全探索 DP の処理時間に比べて A*アルゴリズムの処理時間は，配列長が長くなるほど短縮される．長さ 392 では全探索 DP の処理時間の 0.44 倍であるが，長さ 2352 では 0.21 倍まで低下する．また，A*アルゴリズムの効率は，配列の類似性が高いほど，向上することが容易に予想される．なぜなら，配列の類似

性が高いと最適パスが2次元グラフの対角線周辺の領域に限局されるからである．このように，A*アルゴリズムの効率は，配列長と配列の類似性に依存する．本論文では詳細は割愛するが，種々のデータを用いた実験を通じて，図11とほぼ同じ傾向を示し，全探索DPに対して処理時間が0.2～0.5倍程度に短縮できることが計測された．

なお，反復的に2次元のA*アルゴリズムを用いる方法ではなく，N次元のA*アルゴリズムを用いてマルチプルアライメントを求める方法[21)~23)]もあるが，やはり，組み合わせの数が爆発するために，大規模問題には対応できない．

7. まとめ

アミノ酸配列のマルチプルアライメントの問題は，アミノ酸残基の保存や置換，欠失や挿入に一定の指標を与え，その総和（sum-of-pairs）が最大となるものが最も確からしいアライメントであるというようにモデル化できる．このモデルは，総和の最大化に関する組み合わせ最適化問題を解くこととなり，大規模問題では，組み合わせの数が爆発するため，アライメントの精度と計算時間との間にトレードオフの関係が存在している．

我々は，大規模なマルチプルアライメントの問題を高精度に解くために，「ツリーベース並列反復改善法」をPVMライブラリとMPIライブラリを用いて，それぞれで汎用の並列計算機上に並列実装した．またA*アルゴリズムを適用して探索空間の効率的な刈り込みを実現した．今回行なった並列化と探索空間の刈り込みにより処理時間が大きく抑えられた．本実験では，並列化により約90倍（Hitachi SR2201, 256 PU使用），探索空間の刈り込みにより約5倍（配列長2352の場合）の速度向上が計測された．あわせると約450倍の速度向上が見込まれる．さらに配列長が長い問題ではより大きな速度向上が期待できる．高速化により，従来は計算時間が膨大なために困難とされていた本数の多いアライメントの精度の向上を，現実的な計算時間内で実現することを可能とした．特に配列の類似性が低い場合のアライメントの精度の向上が大きく，進化距離が遠い配列を研究する際の有力な道具となる．

謝辞 本研究の性能評価のために貴重なアミノ酸配列データをご提供頂いた，生物分子工学研究所 藤 博幸 博士に深謝します．

参考文献

1) 石川幹人，十時泰，戸谷智之，星田昌紀，広沢誠：並列反復改善法によるタンパク質の配列解析，情報処理学会論文誌，Vol. 35, No. 12, pp. 2816–2830 (1994).
2) 後藤修：マルチプルアライメントは生体高分子情報の交差点，生物物理，Vol. 38, No. 2, pp. 52–56 (1998).
3) Feng, D. and Doolittle, R.: Progressive sequence alignment as a prerequisite to correct phylogenetic trees, *J. Mol. Evol.*, Vol. 25, pp. 351–360 (1987).
4) Higgins, D., Bleasby, A. and Fuchs, R.: CLUSTAL V: improved software for multiple sequence alignment, *Comput. Applic. Biosci.*, Vol. 8, pp. 189–191 (1992).
5) Barton, J.G.: Protein multiple sequence alignment and flexible pattern matching, *Methods in Enzymology*, Vol. 183, Academic Press, pp. 403–428 (1990).
6) Hirosawa, M., Totoki, Y., Hoshida, M. and Ishikawa, M.: Comprehensive study on iterative algorithms of multiple sequence alignment, *Comput. Applic. Biosci.*, Vol. 11, No. 1, pp. 13–18 (1995).
7) Berger, M. and Munson, P.: A novel randomized iterative strategy for aligning multiple protein sequences, *Comput. Applic. Biosci.*, Vol. 7, pp. 479–484 (1991).
8) Gotoh, O.: Optimal alignment between groups of sequences and its application to multiple sequence alignment, *Comput. Applic. Biosci.*, Vol. 9, pp. 361–370 (1993).
9) Haussler, D., Krogh, A., Mian, I. and Sjoelander, K.: Protein modeling using hidden Markov models: analysis of globins, *Proc. 26th Annu. Hawaii Int'l. Conf. Syst. Sci.*, Vol. 1, pp. 792–802 (1993).
10) Tanaka, H., Asai, K., Ishikawa, M. and Konagaya, A.: Hidden Markov models and iterative aligners: study of their equivalence and possibilities, *Proc. 1st Int'l. Conf. Intelli. Syst. Mol. Biol.*, pp. 395–401 (1993).
11) Gotoh, O.: Further improvement in methods of group-to-group sequence alignment with generalized profile operations, *Comput. Applic. Biosci.*, Vol. 10, No. 4, pp. 379–387 (1994).
12) Araki, S., Goshima, M., Mori, S., Nakashima, H., Tomita, S., Akiyama, Y. and Kanehisa, M.: Application of parallelized DP and A* algorithm to multiple sequence alignment, *Proc. Genome Informatics Workshop IV*, Universal Academy Press, pp. 94–102 (1993).
13) 木村資生：分子進化の中立説，紀伊国屋書店 (1986).
14) Henikoff, S. and Henikoff, J.: Amino acid sub-

stitution matrices from protein blocks, *Proc. Natl. Acad. Sci. USA*, Vol. 10, pp. 10915–10919 (1992).
15) Dayhoff, M., Schwartz, R. and Orcutt, B.: A model of evolutionary change in proteins, *Atlas of Protein Sequence and Structure*, Vol. 5, No. 3, Nat. Biomed. Res. Found., Washington DC, pp. 345–352 (1978).
16) Gotoh, O.: An improved algorithm for matching biological sequences, *J. Mol. Biol.*, Vol. 162, pp. 705–708 (1982).
17) 後藤修: 核酸・蛋白質一次構造の計算機による解析, 日本物理学会誌, Vol. 38, No. 6, pp. 477–480 (1983).
18) Needleman, S. and Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequences of two proteins, *J. Mol. Biol.*, Vol. 48, pp. 443–453 (1970).
19) Smith, T. and Waterman, M.: Identification of common molecular subsequences, *J. Mol. Biol.*, Vol. 147, pp. 195–197 (1981).
20) 美宅成樹, 金久實 (編): ヒトゲノム計画と知識情報処理, 培風館 (1995).
21) Ikeda, T. and Imai, H.: Fast A* algorithm for multiple sequence alignment, *Proc. Genome Informatics Workshop 1994*, Universal Academy Press, pp. 90–99 (1994).
22) Gupta, S., Kececioglu, J. and Schaeffer, A.: Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment, *J. Comput. Biol.*, Vol. 2, pp. 459–472 (1995).
23) Kececioglu, J., Altschul, S. and Lipman, D.: Msa 2.1 : A program for computing multiple alignments. source codes (1994). (http://www.ibc.wustl.edu/ibc/msa.html).

(平成10年10月30日受付)
(平成10年12月28日再受付)
(平成11年 2 月 2 日採録)

十時 泰

昭和37年生. 昭和62年名古屋大学理学部物理学科卒業. 平成2年(株)情報数理研究所入社. 平成5年～7年(財)新世代コンピュータ技術開発機構に出向. 平成11年理化学研究所ゲノム科学総合研究センター研究員. 現在に至る. 生物情報科学(特にDNA／タンパク質配列情報解析)に関する研究およびシステム開発に従事.

秋山 泰(正会員)

昭和36年生. 平成2年慶應義塾大学大学院理工学研究科電気工学専攻博士課程修了. 工学博士. 同年通産省電子技術総合研究所研究官. 平成4年京都大学化学研究所助教授. 平成8年技術研究組合新情報処理開発機構並列応用つくば研究室室長. 現在に至る. 並列計算機を用いたタンパク質立体構造および遺伝子配列情報解析等の研究に従事. 電子情報通信学会, 日本生物物理学会, 分子生物学会, 神経回路学会, IEEE各会員.

鬼塚健太郎

昭和38年生. 平成2年東京都立大学理学研究科物理学専攻修士課程修了. 同年松下電器産業(株)入社. (財)新世代コンピュータ技術開発機構(ICOT)に出向, 同機構の開発する並列論理型コンピュータの遺伝子情報処理分野への応用に関する研究に従事. 平成6年より松下技研(株)において音声認識の研究, 平成7年より松下電子工業(株)において画像処理プロセッサの開発に従事, 平成8年技術研究組合新情報処理開発機構に出向, 並列応用つくば研究室にて並列計算機上でのタンパク質立体構造解析に関する研究に従事. 日本生物物理学会会員.

野口 保(正会員)

昭和32年生. 昭和58年東京農工大学大学院工学研究科応用物理学専攻修士課程修了. 昭和59年富士通(株)入社. 同年富士ファコム制御(株)に出向. コンピュータケミストリ分野のシステム開発に従事. 平成元年蛋白工学研究所(PERI)に出向. タンパク質立体構造予測の研究に従事. 平成3年より富士通(株)にてバイオ分野のシステム開発に従事. 平成6年から平成7年にかけて九州工業大学情報工学部受託研究員. 平成8年技術研究組合新情報処理開発機構主任研究員. 並列応用つくば研究室にて, 並列計算機を用いたタンパク質立体構造予測の研究に従事. 日本物理学会会員.

斎藤 稔（正会員）

昭和29年生．昭和60年名古屋大学大学院理学研究科物理学専攻博士課程修了．理学博士．昭和61年名古屋大学理学部物理学科助手．昭和63年蛋白工学研究所主任研究員．長距離クーロン力を高速に計算する方法（PPPC法）と，これに基づく分子動力学法のプログラム（COSMOS90）を開発した．COSMOS90によって，水中の蛋白質をクーロン力をカットオフせずにシミュレートすることに初めて成功した．平成8年新情報処理開発機構主任研究員．COSMOS90, AMBER, Barnes-Hut treecodeの並列化を行う．平成10年弘前大学理工学部教授．現在に至る．日本物理学会，日本化学会，日本生物物理学会，各会員．

安藤 誠（正会員）

昭和42年生．平成4年慶應義塾大学大学院理工学研究科計算機科学専攻修士課程修了．同年日本鋼管（株）入社．データベース（関係型，オブジェクト指向など）のデータモデルに関する研究に従事．平成6年同社より米国コンベックスコンピュータ社（現ヒューレットパッカード社）に派遣．並列計算機Exemplarシリーズの OS 開発グループに所属．平成8年技術研究組合新情報処理開発機構に出向．並列応用つくば研究室にて，並列計算機上でのタンパク質立体構造解析等の研究に従事．