ICOT Technical Report: TR-0900

TR-0900

並列反復改善法によるタンパク質の配列解析

石川　幹人、十時　泰、
戸谷　智之、星田　昌紀（松下）、
広沢　誠（かずさDNA）

November 25, 1994

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Institute for New Generation Computer Technology

並列反復改善法によるタンパク質の配列解析[1]

石川 幹人，戸時 泰，戸谷 智之[2]
(財) 新世代コンピュータ技術開発機構 (ICOT)

星田 昌紀[3]
松下電器産業（株）マルチメディアシステム研究所

広沢 誠[4]
（財）かずさＤＮＡ研究所

Abstract

Protein sequence analysis—such as multiple sequence alignment is an important methodology for revealing new frontiers in molecular biology. Although high-dimensional dynamic programming can rigorously solve multiple sequence alignment problems in principle, its heavy requirements on computer resources make other approximate methods appropriate. The tree-based method, a typical approximation, tends to accumulate errors when aligning sequences of low similarities. It was recently found that iterative improvement could correct such errors, but that too many iteration cycles were needed to solve a practical-scale problem. To reduce the execution time, we parallelized the method with best-first search and multi-agent hill-climbing approaches. We also incorporated restricted partitioning techniques which decrease the number of processing elements and the execution time required for convergence. As a result, practical-scale alignment problems were solved within a practical amount of time, with scores higher than those obtained by conventional tree-based methods.

1 はじめに

近年、DNA（デオキシリボ核酸）の塩基配列やタンパク質のアミノ酸配列の分析手順が確立され、それらのデータを蓄えたデータベースが急速に膨らんでいる。たとえば現在 GenBank[1)] に蓄えられている塩基配列のデータ量は、およそ一億数千万塩基であるが、米欧日で取り組んでいるヒトゲノム計画は、三十億塩基もある人間の全DNAを読み取ろうとしている。現在の実験分析技術をもってすれば、来世紀初頭にもその読み取りが達成される可能性が高い。しかし現状では、配列データを次々と蓄えてはいるが、それらの十分な利用がなされているわけではない。それは配列を読み取る実験技術の進歩に比べ、配列情報がいかなる意味を持つかを探る情報処理技術が未熟であるためといえる。そのため今後の分子生物学の画期的な進歩には、情報処理技術の確立が急務とされている[2)]。

こうした情報処理のうちで、最も基本的な技術は、配列間の類似性を調べる解析処理である。ここでよく問題とされるのは計算量である。というのは、あるひとつの配列と、ある観点から類似とみなすべき配列をデータベースから見つけるには、数万から数百万の配列との類似性を評価しなければならない[3)]。さらに、数個の配列にわたって共通の特徴を抽出したいとなれば、組合

[1]Protein Sequence Analysis by the Parallel Iterative Improvement Method

[2]Masato Ishikawa, Yasushi Totoki, Tomoyuki Toya [Institute for New Generation Computer Technology]

[3]Masaki Hoshida [Multimedia Systems Research Laboratory, Matsushita Electric Industrial Co., Ltd.]

[4]Makoto Hirosawa [Kazusa DNA Research Institute]

こうしたDPを多次元化することで、複数配列のマルチプルアライメントを行うことが、原理的には可能である。たとえば、配列3本のアライメントには、3次元のネットワークに基づいたDPの処理を行えば良い[23),24)]。DPは、原理的には一度に何本の配列でも同時に処理でき、与えられた評価値における最適なマルチプルアライメントが得られるはずである。ところがn本の配列を同時にアライメントするn次元のDPは、概して、配列のn乗の計算量とn乗のメモリー量が必要であり、現実的に可能なのは3次元までである。

DPを多次元化すると、ギャップコストの扱い方も複雑になる。カラムに1個でもギャップのある配列があったら、そのカラムの評価値はゼロにするという簡便な方法[24)]もなされているが、ギャップの多いアライメントの品質があまりよくない。ギャップコストの扱いは、処理時間とのトレードオフであり、議論の余地がある[25)]。一方、高速化手法として、アライメントに長いギャップが入ることはあまりないことから、高次元ネットワークのコーナー部分の処理を、ペアワイズアライメントの結果に基づいて大幅にカットする方法が提案されている[26)]。この方法により、問題によっては4次元以上のDPが可能となる。しかし、それでも、配列10本以上の実用的なマルチプルアライメントに適用すると、膨大な計算時間が必要となり、DPの多次元化では、一般のマルチプルアライメントの問題には対応できない。

2.2 ツリーベース組合せ法

現在、マルチプルアライメントの問題を解くのに、最も広く使われているのはツリーベース組合せ法である。この方法は、2次元DPを利用して、配列を似ているものから順にツリー状に組み合わせ、マルチプルアライメントを生成する。2次元DPの高速性と、類似した配列同士のアライメントの確実性から、現実の問題に適応可能なスピードと、ある程度の解の品質を達成している。最近出回っている配列解析ツールも、多くはこうした方法をとっている[27)]。

最初に図4を用いてツリーの作成法を説明しよう。系統樹作成と違って、ツリーの作成時にはアライメントは済んでないのであるから、始めにすべての配列対について2次元DPによるアライメントを行って、配列間の類似性スコアを求める。そうして得られた三角表において、(a)まず、最も類似性の高い配列対（図の場合は配列1と配列2）をツリーの枝にして結線する。その結合した配列対の、残りの配列に対する類似性は、先のスコアの平均値として、三角表を更新する。そしてまた，(b)最も類似性の高い配列対（こんどは配列3と配列4）を三角表から探しだし、結線する。これを繰返す(c)と、ツリーが完成する(d)。この手順はUPGMA[28)](unweighted pair-group arithmetic average clustering)と呼ばれている。

ツリーが完成したならば、ツリーの枝に相当する2つの配列群を組合せて、マルチプルアライメントを作っていく（図5）。そうすると最後には、全体のアライメントが得られる。配列群を組合せる際に、ペアワイズのDPだけを行う方法と、配列群間のDPを使う方法とがある。ペアワイズDPだけを行う代表的な方法[29)]では、配列群の組合せ時に、双方の配列群から各1本をとった配列対のなかで、最も類似性の高い配列対のペアワイズアライメントに基づいて、配列群同士のアライメントを行う。配列群間のDPを使う方法[30)]では、2つの配列群をそのまま2次元DPする。そのDP時には、比較されるカラムの類似性が、カラムに含まれる文字についての類似性の和として定義される。カラムの特徴量をプロファイル[31)]に表現して類似性を算出する簡便法もある。

ツリーベース組合せ法の問題点は、初期の組合せで起こったアライメントのエラーが、後々まで波及して、最後に得られる解の精度が低下することである。とくに、配列の類似性が低く、始めに作ったツリーが間違っていた場合には、マルチプルアライメントの品質が大幅に悪くなる。図6は、図1の問題をツリーベース組合せ法で解いた例であるが、不十分なアライメントである

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カラムに置くことを許容して処理を行う。現在最も広く使われている類似性評価尺度は、表１に示したもので、Dayhoffマトリックス[13]と呼ばれる。この尺度は、当時知られていたアライメントをもれなく調べ、該当アミノ酸対の遺伝的変異が偶然に対して如何に大き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。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。この評価尺度は算出法が極めて妥当なものの、決められてからもう１０年以上も経過し、算出に使用したデータが古くなっている。最近、最新のデータで評価尺度を算出し直そうという動きもある[14), 15)]。

マルチプルアライメントされた結果は次のように利用できる。第一に、アライメントした配列のなかに、構造・機能がよくわかっているものが含まれていれば、アライメントした結果からわかる配列の類似性から、未知の構造・機能を推測することができる。第二に、先に述べたように、重要な配列の部分であるモチーフを見出せる。モチーフは構造や機能を特徴づけたり、データベースから新たな知見を引き出す手がかりになる。第三に、マルチプルアライメントから図２のような進化系統樹を描くことができる。系統樹の形成は、アライメントにおけるアミノ酸置換数から、配列の進化距離を推定して行うのであるが、いくつかの推定方法が提案されている[16)]。図２は、最も簡便な平均距離法を用いて描いたものである。このようにマルチプルアライメントは、タンパク質の研究、そして分子生物学の研究の中で重要な役割を担っている。

2　従来のアライメント法

代表的な配列解析法であるマルチプルアライメントは、長らく熟練した生物学者の手作業に依存してきたが、近年になって計算機による自動化や、支援も徐々に行われてきている。これまでの代表的研究を以下に概説するが、より詳しくは最近の総説[17)]を参照されたい。

2.1　ダイナミックプログラミング

最適化問題を解くためのアルゴリズムのひとつにダイナミックプログラミング（DP）があり、早くから文字列のマッチングに適用された[18)]。配列２本のペアワイズアライメントは、DPによって高速に最適解を求めることができる。図３に、短い配列２本のアライメントを解くDPの概念図を示した。たとえば、GKFD, GFVDという２つの配列をアライメントする場合、この２つの配列を図のような２次元のネットワークの辺に対応させる。斜め方向のアーク（矢）は、そのアークの位置に対応する２つのアミノ酸の類似度がスコアとして割り振られる。図の類似度には前述のDayhoffマトリクスを用いている。また縦および横方向のアークはギャップに対応し、ギャップを挿入するときの適当なコストが割り振られる。このように問題を定式化すると、最適なアライメントを求めることは、このネットワーク上のスコア最良の経路を求めることに対応する。スコア最良の経路は左上の端から右下の端に向かって、各ノードに至る最適経路を段階的に決定していくことにより求められる。この２次元のDPの計算量は、ネットワークの面積、つまり、配列の長さの２乗のオーダーである。

進化の過程で、配列の挿入・削除は数文字まとめて起こることから、配列比較の際のギャップコストは、ギャップの長さに依存させるのが望ましい[19)]。もっとも一般的なギャップコストはギャップの長さkに対して、$a+bk$のような一次式の関係を与える方法である[20)]。この関係を導入しても、計算量のオーダーを上げない実装法が考案されている[21)]。また、最適解が得られる保証がなくなってしまうが、ネットワークのコーナー（図では左下と右上の部分）を問題に応じてカットし、計算量を削減することもよく行われる[22)]。

ロセス間通信や同期処理の記述が容易な、並列論理型プログラミング言語のKL1[36]である。またマルチプルアライメントの評価スコアには、アミノ酸の類似度評価にDayhoffマトリクス、長さkのギャップコストに$-7-k$、アウトギャップ（配列の先端や終端に付加されるギャップ）のコストにゼロを用いている。DPによる最適化計算の詳細は、関連の文献[37]を参照されたい。

3.1 並列化の方法

並列化には、最良優先探索とマルチ山登りの２つの方法を用いた。最良優先探索の並列反復改善法[38]は、次に示す手順で改善を行う（図８）。

1. ギャップのない配列群を初期アライメントとする。
2. 初期アライメントに対して、可能なサブアライメントへの分割$2^{n-1}-1$通りを生成する。
3. サブアライメント同士の、DPによるアライメントの最適化を、各々のプロセッサで並列に行う。
4. それらの結果を比較し、スコアの最も良い解を、次の改善サイクルの初期アライメントとする。

あるサイクルで、スコアに改善がみられなかったら、その時点のアライメントを最終解とする。

並列計算機上の最良優先探索の実装は、次のようにした。たとえば、図１のような９本のアライメント問題は分割数が255通りあるので、マスタープロセッサが255台のスレーブプロセッサに、各分割問題を分配して、それぞれ並列に最適化させる。その後、マスタープロセッサは、計算が済んだものから順に解を回収し、全てが集まったならば最良解を判定して、次の改善サイクルを開始する。こうした問題解決を実行している途中の、各プロセッサの稼働状況が、図９に示されている。縦軸は256台のプロセッサ（一番上がマスタープロセッサ）、横軸は時間で、２秒ごとの稼働率が色で示されている。最良優先探索並列では、改善サイクルごとにスレーブプロセッサの同期をとっているため、各サイクルの終りに、待ち時間（青色の部分）が発生する。実行時間全体でスレーブプロセッサの稼働率は67％であった。

マルチ山登りの並列反復改善法では、各プロセッサで異なった乱数を用いて独立に反復改善法を行い、その中で最もスコアの良い解を全体の解とする。使用プロセッサ数は任意であるが、最良優先探索並列と比較するときは、配列９本のアライメントに対してスレーブプロセッサ数を255台としている。マスタープロセッサは必ずしも必要ないが、我々の実装では、定期的（80秒ごと）に解を回収しその時点の最良解をモニタするために、マスタープロセッサを１台割り当てている。

マルチ山登り並列を実行中のスレーブプロセッサは、ほとんどつねにフル稼働である。各スレーブプロセッサは改善サイクルが終り次第、システムタイマーを調べ、80秒を越えるごとに、現在のアライメントの状態をマスタープロセッサへ送っている。スレーブプロセッサによっては、そのモニタ解の通信が競合して待たされることもあるが、それでも、待ち時間はごくわずかで、全体のスレーブプロセッサの稼働率は99％以上であった。

3.2 並列効果の比較

図１のアライメント問題を、逐次反復改善法と、２つの並列反復改善法とで解いた場合の性能比較を、図１０に示した。また、従来のツリーベース組合せ法で得られるスコアのレベルも合

ことがわかる。というのは、全体にギャップがたくさん入ってしまって、アライメントが長くなったうえに、中央にあるモチーフA.Kについては、完全にはアライメントできていない。

品質の改善のため、アライメント結果から得られる系統樹が、始めのツリーと異なっていた場合には、その系統樹に従って、再度ツリーベース組合せ法を行う方法が提案されている[32]。その方法は、エラー発生の低減に有効だが、いったん発生したエラーを修正することはできない。また、3次元のDPを用いて同時に比較する配列数を増やし、エラーを吸収しながら組合せたアライメント結果を、さらにシミュレーテッドアニーリングを用いて全体的に修正する手法[33]も開発された。しかし今度は、実用的問題を解くには実行時間がかかり過ぎる傾向があった。

このように、遺伝子情報の研究促進のためには、ツリーベース組合せ法の問題点を克服し、かつ実用規模の問題が妥当な時間内に解けるマルチプルアライメントの解法が、求められていた。

2.3 反復改善法

最近、アライメント過程の初期段階で生じるエラーを修正できる、反復改善法が提案された[34]。この方法は次に示す手順で、配列群間のDPを反復的に適用することによりマルチプルアライメントを行う（図7）。

1. ギャップのない配列群を初期アライメントとする。
2. 初期アライメントを、ランダムに2つのサブアライメントに分割する。
3. サブアライメント同士のアライメントを、DPを適用して最適化する。
4. その結果を、次の改善サイクルの初期アライメントとする。

収束条件に適当なサイクル数を決め、そのサイクル数にわたってスコアに改善がみられなかったら、その時点のアライメントを最終解とする。

反復改善法の基本的アイデアは次のところにある。マルチプルアライメント全体の評価値が、それに含まれるすべての配列対の評価値の和（配列ごとに重みがついていてもよい）で与えられるという条件のもとでは、あるマルチプルアライメントを任意に2つの配列群に分け、それらの間で互いに2次元DPを行うと、得られるアライメントは、もとのマルチプルアライメントより、スコアが良い（少なくとも等しい）。つまり、配列群間のDPを繰返すと、マルチプルアライメントのスコアが初期状態から次第に良くなっていくのである。しかし、次第に良くなっていくといえども、どこまでも単調に良くなるわけではない。乱数の与え方によりアライメントが不適当な方向へ進むと、比較的悪い局所最適値に陥ってしまうこともある。

反復改善法の大きな問題は、やはり実行時間である。実用規模の問題になると膨大な改善サイクルを要する。n本の配列からなる配列群の分割の仕方は$2^{n-1}-1$通りであるから、20本以上の配列をアライメントしようとすると、分割数は50万通り以上にのぼってしまう。とくに、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、収束までの試行サイクル数もたいへん大きなものとなる。

3 並列計算機を用いた反復改善法

我々は、実用規模のマルチプルアライメント問題に対して、ツリーベース組合せ法以上の品質の解を与えることを目指し、反復改善法の並列化を試みた。そして、その実験システムを、分散メモリ型の並列計算機PIM/m[35]（要素プロセッサ256台構成）上に実装した。使用言語は、プ

と、ますますその傾向が大きくなってくる。

こうした特徴を捉え、１本抜き限定分割法と１，２本抜き限定分割法を開発した。１本抜き限定分割法では、配列群を２つに分割するときに、小さい配列群の配列数を必ず１本とする。配列n本のアライメント問題の場合、分割数はn通りである。１，２本抜き限定分割法では、小さい配列群の配列数を必ず１本か２本とし、配列n本のアライメント問題の分割数は$n(n+1)/2$通りになる。従来の反復改善法の分割数が、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。それでも、解のスコアはそれほど低下しない（後述）。もちろん、１，２本抜き限定分割法は、１本抜き限定分割法よりも、平均して少しスコアの良い解を生成する。

こうした限定分割が有効なのは、次の理由からだろう。アライメント途中の配列群は、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て、配列数が少ないとカラムの特徴が比較的よく表れる。カラムの特徴がはっきりしていると、配列群間のアライメントの最適化が効率良くなされ、スコアの向上が大きい。

次に、なんとか１乗オーダーの分割数で、１，２本抜き限定分割法以上の効果をあげられないかと考え、第３の限定分割法を模索した。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、一緒に最適化されることに由来すると思われた。それならば、反復改善の過程で良く揃った配列群を、改善サイクルの都度に抽出し、それらと残りの配列間で最適化をすれば効果的と推測された。こうした発想から、ツリー依存の限定分割法が開発できた。

ツリー依存限定分割法では、各改善サイクルの始めにその時点でのアライメントの状態を調べ、全配列対のアライメントスコアを三角表にする。それに基づいてUPGMA法でツリーを描画し、ツリー上で近い配列はなるべく同じ配列群に含まれるように、分割を決めるのである。具体的には、図１２のように、ツリーの任意の１か所を切断して分けられる２つの配列群を、最適化の対象とする分割に選ぶ。この分割数はツリーの形状によらず、配列n本に対して$2n-3$通りである。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。

4.2　限定分割導入時の並列効果比較

図１と同様規模の配列９本のアライメント30問について実行し、限定分割法による並列効果を検討した。

表２に、最良優先探索の並列反復改善法に、各種限定分割法を導入した場合の性能比較を示す。限定分割法の導入により、実行プロセッサ数（マスタープロセッサを数に含めていない）が大幅に減少しているにもかかわらず、解のスコアと実行時間は、全数分割（限定分割を導入してない並列実行）と大きな差はない。全数分割は、限定分割より解の平均スコアが少し良いが、実行時間が多くかかっているのを考慮すれば、大きな優位性を見い出せない（図１３も参照されたい）。また全数分割は、稼働率が低く、サイクル当り実行時間も大きくなる傾向がある。その理由は、DPを行うときに、比べられる配列群が４本と５本のように均等に近くなっていると、１本と８本などに比べ最適化計算に時間がかかる。全数分割では、均等に近い分割を処理しているプロセッサの数が多く、その計算をしているプロセッサのうちのひとつが、サイクルのネックとなりやすい。

表２で、限定分割同士を比較すると、ツリー依存分割のスコアが最も良い。ツリー依存分割は、全数分割と同じ理由で稼働率がやや低いが、平均改善サイクル数がやや少ない。またツリー依存分割は、１，２本抜きよりプロセッサ数が少ないのにもかかわらず、良い解を早めに生成で

わせて示した。図の逐次法の曲線は、マルチ山登り並列の平均をプロットしたもので、逐次法を255回、異なった乱数で行ったものの平均に相当している。図を見ると、いずれの反復改善法においても、ツリーベース組合せ法で得られるレベルを越えてアライメントが改善されること、逐次反復改善法が並列化手法で高速化されていることがわかる。

逐次反復改善法では、実行時間8000秒（反復改善サイクルにして約630回に相当）の解の平均スコアは2447であったが、その値に最良優先探索並列では124秒（総反復改善サイクルにして2550回）で、マルチ山登り並列では1063秒（総反復改善サイクルにして約21000回）で至っている。実行速度として比較すれば、それぞれ、64.5倍、7.5倍となったことに相当するが、並列化手法では、総反復改善サイクル数が大幅に増えている。これは、膨大な無駄計算に支えられた結果として、計算時間が低減したことを意味している。

最良優先探索並列を、マルチ山登り並列と比較すると、前者はプロセッサの待ち時間は多いものの、無駄計算が少なく、早めに良い解が得られる。図１０の実験では、最良優先探索並列は、182秒で2544のスコアの最終解を生成したが、マルチ山登り並列の解は、2544に達するのに2702秒を要した。しかし、さらに時間をかけると、その解は2544を越えて良くなって行った。マルチ山登り並列は、時間をかければ最良優先探索の解のスコアを上回るようである。最良優先探索の解は局所最適解に陥っていると予想されるので、マルチ山登りの解の方が最終的には良くなる可能性が高い。

それを確かめるために、次の実験を行った。配列９本のアライメント問題の場合、分割数はわずか255通りなので、全通りの分割についていずれも改善がないことを確認したうえで、収束を判断できる。そのような収束条件を各プロセッサに設定したうえで、図１と同じ規模の30種のアライメント問題について、255台並列のマルチ山登りを適用してみた。その結果、マルチ山登り並列は、全ての問題において最良優先探索の解のスコアを上回った。また、収束時に各プロセッサが持っていた解のスコアをそれぞれ調べたところ、そのうちの53.4％は、最良優先探索の解のスコアに満たなかった。最良優先探索並列は、逐次反復改善法の平均的な解に至っていると判断できる。

4　限定分割並列反復改善法

配列20本以上の実用規模のアライメント問題は、並列化だけでは対処できない。なぜなら、反復改善法の探索空間は、配列の本数の増加に対して、指数的に広がるからである。そこで、探索空間を効率良く制限するヒューリスティクスである限定分割法を３種類開発し、並列反復改善法に導入した。

4.1　限定分割法とその妥当性

我々は反復改善法の実験を繰り返すうちに、配列数が不均等に分割されたサイクルのスコア改善が比較的大きいことに気づいた。つまり、１本と残り（１本抜き）とか、２本と残り（２本抜き）のように配列が分割されたサイクルは、配列の本数が均等に分割されたサイクルよりも、スコア改善度合が平均して大きいのである。さらに、各分割を等確率に選ぶ、反復改善法の本来の選択方法では、数が少ない不均等の分割が選ばれにくく、スコア改善が遅いことがわかった。

図１１は、その傾向を示すグラフである。上段にはサイクル当りのスコア改善度合が、下段には試行されたサイクル数が、それぞれ、分割が何本抜きであったかによって区別されて示されている。図を見ると、１本抜きや２本抜きの改善度合が大きいのに、それらが試される回数は相対的に小さいことがわかる。試行回数のグラフは二項分布になっているので、配列本数が増える

5　おわりに

これまでの結論を要約すると次のようになる。我々は反復改善法を並列化し、実行性能の向上を図った。はじめに、配列９本程度の小規模な問題で、逐次反復改善法と２種の並列反復改善法とを比較した結果、255 台規模の並列実行で、最良優先探索並列で数十倍程度、マルチ山登り並列で数倍程度の高速化が得られた。そのうえ、得られる解の平均スコアも向上した。また、マルチ山登り並列は、十分な実行時間をかけると、最良優先探索並列の解のスコアを平均して上回ることがわかった。

さらに、独自の限定分割法を導入することにより、解の品質の大きな低下を招くことなく、使用プロセッサ数、収束までにかかる時間を大きく削減できた。その結果、配列２２本程度の実用規模の問題も、反復改善法で実用的な時間内に解決可能となり、従来用いられていたツリーベース組合せ法に比べ、安定して高品質の解を生成できるようになった。並列反復改善法のなかでも、マルチ山登り並列のツリー依存限定分割法が最も良い性能を示した。しかし、実用規模の問題であると探索空間が広まり、マルチ山登り並列の収束性が落ち、最良優先探索の高速性の特徴も相対的に有用となってくる。

今回取り上げた２つの手法の他にも、いくつかの並列化手法、あるいはそれらの折衷案が考えられる。たとえば、つねに最良から数個の解候補を保存しながら探索する、いわゆるビームサーチを行うと、最良優先探索にマルチ山登り的要素を入れたこととなり、高速性を保ったまま、解のスコアを向上させることが可能になる。また、マルチ山登りに、２つの解の一部分をつなぎ合わせて良い解を得る、いわゆる遺伝的アルゴリズムのオペレーションを導入すれば、一層効果的である[39),40)]。要素プロセッサの数に余裕がある場合は、DP の内部を並列化[41)~43)]して高速化したり、いろいろな初期状態からマルチ山登りをして、良い解に収束する可能性を高めるのもよい。

反復改善法とツリーベース組合せ法を融合させ、ツリーベースに配列を組み合わせたたびに、その状態を反復改善法で修正する方法も提案されていた[44)]。しかし、この方法も反復改善に大きな時間がかかるため、実用規模の問題に適応できず、これまであまり注目されて来なかった。ところが、我々の限定分割法を導入すれば実用規模の問題に対処可能となるうえ、並列化手法による実行時間の低減も大きく、一転して有望な手法となっている[45)]。

最後に評価スコアについて考察する。山登りの収束解が非常に多様であることからわかるが、解空間はかなりマルチピークである。評価スコアに局所的な平滑処理を加え、解空間をなだらかにしてピークを減らし、大局的な最適解へ至り易くすることが考えられる。それを行う際には、スコア計算にかかる演算の増大と、変更されたスコアの生物学的な妥当性の検討が必要である。

また、今回使用したペアワイズアライメントのスコアを総計するタイプの評価スコアは、一般的ではあるが、必ずしもすべてのマルチプルアライメントの評価に最適とは言えない。用途に応じていくつかの評価スコアが提案されている[46)]が、生物学上の多様なニーズを満たす万能なものはないとされている。そこで、インタラクティブにアライメントするツール[47)]や、生物学者が行うアライメントの修正操作をパターン化して、ルールベースシステムにまとめるアプローチ[48)]なども重要である。これらを、本論文で議論したスコアを最適化するアライメントプログラムと、合わせて検討することで、生物学的に有用なトータルシステムがまとめられるであろう。

謝辞

研究の機会を与えて下さった ICOT 内田俊一研究所長、新田克己第２研究部長に感謝いたします。また、後藤修氏には多くの貴重な助言をいただき、とくに謝意を表します。なお、本研究

きる傾向がある、

図１３には、マルチ山登り並列の反復改善法に各種限定分割法を導入した場合の、平均スコアの向上履歴が、グラフで表示されている。比較のため、表２に示した最良優先探索並列の４手法による結果と、従来のツリーベース組合せ法による結果も合わせて示してある。図を見ると、限定分割を導入した各手法は、いずれも収束性が早まり、解が高速に得られているのがわかる。それらは、点線で示したツリーベース組合せ法の解（平均生成時間 69 秒）のレベルを、ツリーベース組合せ法の数倍の時間内に上回っている。

マルチ山登り限定分割法を、分割数と同じ台数（９／４５／１５）で並列にした試行の性能を、最良優先探索並列の各分割法の性能と比較すると、いずれもマルチ山登り並列が、時間はかかるものの、最良優先探索並列の解のスコアを上回っている。マルチ山登りの１，２本抜き限定分割は、無駄な分割の試行が多くなり、４５台程度では、このグラフの時間内では収束に至っていない。それでも、最良優先探索並列の１，２本抜き限定分割の解を平均して上回っている。

さらに、これら３つのマルチ山登り限定分割法を、255 台並列に拡張すると、収束性が大きく高まる。とくに、ツリー依存限定分割、１本抜き限定分割の実行速度は、最良優先探索並列の各解の生成時間と比べても遜色ない。最終収束解のスコアは、同じ 255 台並列でもツリー依存限定分割が最も良い。

4.3　実用規模の問題解決

図１と同様の配列を２２本に増やした実用規模のアライメント問題を 10 問作成し、限定分割法を導入した並列実行を行った（全数分割は分割数が 200 万を越えるため行えない）。図１４では、それらの平均スコアを図１３と同様なかたちで比較している。その結果、次の点で図１３と同様な傾向が見られた。（１）限定分割を導入した並列反復改善法の各手法が、点線で示したツリーベース組合せ法の解（平均生成時間 388 秒）のレベルを、早期に上回っている。（２）時間をかければ、マルチ山登り並列が最良優先探索並列の解のスコアを上回る。（３）同じ 255 台のマルチ山登り並列でも、ツリー依存限定分割が平均して最も良い解を与える。

問題の規模が上がったことにより、図１３と異なる傾向も見られた。（１）マルチ山登り並列の１，２本抜きは、試行する分割数が増大したため、収束性が大幅に落ちた。（２）探索空間が広がったために、比較的良い解が早く得られる最良優先探索の特徴が際だった。（３）最良優先探索のうち、ツリー依存分割は、処理する配列の本数が増えて要素プロセッサの稼働率が 50 %までに低下し、実行時間が相対的に上がってしまった。

この実験結果からすると、この程度の実用規模の問題を解く場合は、次のように解決手法を選ぶとよい。解決にそれほど時間をかけたくないときは、最良優先探索並列の１，２本抜き／１本抜き限定分割が良い。解決にある程度時間をかけられるときは、マルチ山登り並列のツリー依存／１本抜き限定分割が良い。

今回使用した問題のうちの１問を、最良優先探索並列の１，２本抜き限定分割で実行した結果を、我々が開発したアライメント表示ツールに出力し、図１５に図示した。各文字をアミノ酸の性質によっておおよそ色分けをしており、アライメントされた各カラムの特徴が一目でわかる。また、中段の棒グラフは、各カラムの類似性の評価スコアを示している。明るい棒は比較的スコアの高いカラムを示している。なお、この画面では、アライメント結果の右側が少し切れているが、それは画面をスクロールして見るように設計されている。

15) Jones, D.T., Taylor, W.R. and Thornton, J.M.: The rapid generation of mutation data matrices from protein sequences, *Comput. Appl. Biosci.*, Vol.8, No.3, pp.275-282 (1992).

16) 根井正利：分子進化遺伝学，培風館 (1990).

17) 石川幹人，金久實：配列データの多重アラインメント法，新生化学実験講座第１６巻，日本生化学会編，東京化学同人， pp.323-342 (1993).

18) Needleman, S.B. and Wunsch, C.D.: 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.F. and Waterman, M.F.: Identification of common molecular subsequences, *J. Mol. Biol.*, Vol.147, pp.195-197 (1981).

20) Fitch, W.M. and Smith, T.F.: Optimal sequence alignments, *Proc. Natl. Acad. Sci.*, Vol.80, pp.1382-1386 (1983).

21) 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).

22) Ukkonen, E.: Algorithms for Approximate String Matching, *Inf. Control*, Vol.64, pp.100-118 (1985).

23) 戸谷，星田，石川，新田：並列３次元ダイナミックプログラミング法による蛋白質の配列解析，情報処理学会第５回プログラミング研究会 (1991).

24) Murata, M. Richardson, J.S. and Sussman, J.L.: Simultenious comparison of three protein sequences, *Proc. Natl. Acad. Sci.*, Vol.82, pp.3073-3077 (1985).

25) Altschul, S.F.: Gap Costs for Multiple Sequence Alignment, *J. Theor. Biol.*, Vol.138, pp.297-309 (1989).

26) Carrillo, H. and Lipman, D.: The multiple sequence alignment problem in biology, *SIAM J. Appl. Math.* Vol.48, pp.1073-1082 (1988).

27) Higgins, D.G., Bleasby, A.J. and Fuchs, R.: CLUSTAL V: improved software for multiple sequence alignment, *Comput. Appl. Biosci.*, Vol.8, No.2, pp.189-191 (1992).

28) Sneath, P.H.A. and Sokal, R.R.: *Numerical Taxonomy*, Freeman and Company (1973).

29) Feng, D.-F. and Doolittle, R.F.: Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees, *J. Mol. Evol.*, Vol.25, pp.351-360 (1987).

30) Barton, J.G.: Protein Multiple Sequence Alignment and Flexible Pattern Matching, *Methods in Enzymology*, Vol.183, Academic Press, pp.403-428 (1990).

31) Gribskov, M., McLachlan, A.D. and Eisenberg, D.: Profile analysis: Detection of distantly related proteins, *Proc. Natl. Acad. Sci.*, Vol.84, pp.4355-4358 (1987).

は文部省重点領域研究「ゲノム情報」との協力のもとに行いました。

参考文献

1) Benson, D., Lipman, D.J. and Ostell, J.: GenBank, *Nucleic Acids Res.,* Vol.21, No.13, pp.2963-2965 (1993).

2) 金久，新田，小長谷，田中：知識情報処理技術とヒトゲノム計画，人工知能学会誌， Vol.6, No.5, pp.630-640 (1991).

3) 五條堀，森山，内藤，河合：大量ＤＮＡデータを対象とした遺伝情報のコンピュータ解析，情報処理， Vol.31, No.7, pp.878–886 (1990).

4) Coulson, A.F.W., Collins, J.F. and Lyall, A.: Protein and nucleic acid sequence database searching: a suitable case for parallel proccessing, *Comput. J.,* Vol.30, No.5, pp.420-424 (1987).

5) Butler, R. *et al.*: Aligning Genetic Sequences, *Strand: New Concepts in Parallel Programming,* Prentice-Hall, New Jersey (1990).

6) Miller, P.L., Nadkarni, P.M. and Carriero, N.M.: Parallel computation and FASTA: confronting the problem of parallel database search for a fast sequence comparison algorithm, *Comput. Appl. Biosci.,* Vol.7, No.1, pp.71-78 (1991).

7) Istvanick, W. *et al.*: Dynamic Methods for Fragment Assembly in Large Scale Genome Sequencing Projects, *Proc. 26th Annual Hawaii Int. Conf. Syst. Sci.,* Vol.1, pp.534-543 (1993).

8) Archambaud, D., Faudemay, P. and Greiner, A.: RAPID-2, An Object-Oriented Associative Memory Applicable to Genomic Data Processing, *Proc. 27th Annual Hawaii Int. Conf. Syst. Sci.,* Vol.5, pp.150-159 (1994).

9) Bernstein, F.C. *et al.*: The Protein Data Bank: A Computer-based Archival File for Macromolecular Structures, *J. Mol. Biol.,* Vol.112, pp.535-542 (1977).

10) Barker, W.C. *et al.*: The PIR-International Protein Sequence Database, *Nucleic Acids Res.,* Vol.20 (Supplement), pp.2023-2026 (1992).

11) Hanks, K.S., Quinn, A.M. and Hunter, T.: The Protein Kinase Family, *Science,* Vol.241, pp.42-52 (1988).

12) 木村資生：分子進化の中立説，紀伊国屋書店 (1986).

13) Dayhoff, M.O., Schwartz, R.M. and Orcutt, B.C.: 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).

14) Gonnet, G.H., Cohen, M.A. and Benner, S.A.: Exhaustive Matching of the Entire Protein Sequence Database, *Science,* Vol.256, pp.1443-1445 (1992).

```
(a) problem
CABL :KLGGGQYGEVYEGVWKKYSLTVAVKTLKEDTMEVEEFLKEAAVMKEIKHPNLVQLLGVCTREPPFYIITEFMTYGNLLDY
FER  :LLGKGNFGEVYKGTLKDKTSVAVKTCKEDLPQELKIKFLQEAKILKQYDHPNIVKLIGVCTQRQPVYIIMELVSGGDFLT
TRK  :ELGEGAFGKVFLAECHNLLPEQDKMLVAVKALKEASESARQDFQREAELLTMLQHQHIVRFFGVCTEGRPLLMVFEYMRH
GCPK :SLRGSSYGSLMTAHGKYQIFANTGHFKGNVVAIKHVNKKRIELTRQVLFELKHMRDVQFNHLTRFIGACIDPPNICIVTE
PKC1 :VLGKGNFGKVILSKSKNTDRLCAIKVLKKDNIIQNHDIESARAEKKVFLLATKTKHPFLTNLYCSFQTENRIYFAMEFIG
CGMPK:TLGVGGFGRVELVQLKSEESKTFAMKILKKRHIVDTRQQEHIRSEKQIMQGAHSDFIVRLYRTFKDSKYLYMLMEACLGG
CAMK :ELGKGAFSVVRRCVKVLAGQEYAAKIINTKKLSARDHQKLEREARICRLLKHPNIVRLHDSISEEGHHYLIFDLVTGGEL
FUSED:LVGQGSFGCVYKATRKDDSKVVAIKVISKRGRATKELKNLRRECDIQARLKHPHVIEMIESFESKTDLFVVTEFALMDLH
WEE1 :LLGSGEFSEVFQVEDPVEKTLKYAVKKLKVKFSGPKERNRLLQEVSIQRALKGHDHIVELMDSWEHGGFLYMQVELCENG

(b) result
CABL :KLGGGQYGEVYEGVWK---------KYSLTVAVKTLKED----TMEVEEFLKEAAV---MKEIK-HPNLVQLLGVCTREPPFYIITEFMTYGNLLDY
FER  :LLGKGNFGEVYKGTLK----------DKTSVAVKTCKED--LPQELKIKFLQEAKI---LKQYD-HPNIVKLIGVCTQRQPVYIIMELVSGGDFLT-
TRK  :ELGEGAFGKVFLAECH-NLLP---EQDKMLVAVKALKEA---SESARQDFQREAEL---LTMLQ-HQHIVRFFGVCTEGRPLLMVFEYMRH------
GCPK :SLRGSSYGSLMTAHGKYQIFANTGHFKGNVVAIKHVNKK---RIELTRQVLFELKH---MRDVQ-FNHLTRFIGACIDPPNICIVTE----------
PKC1 :VLGKGNFGKVILSKSK---------NTDRLCAIKVLKKDNIIQNHDIESARAEKKVFLLATKTK-HPFLTNLYCSFQTENRIYFAMEFIG-------
CGMPK:TLGVGGFGRVELVQLK--------SEESKTFAMKILKKRHIVDTRQQEHIRSEKQI---MQGAH-SDFIVRLYRTFKDSKYLYMLMEACLGG-----
CAMK :ELGKGAFSVVRRCVKV---------LAGQEYAAKIINTKK-LSARDHQKLEREARI---CRLLK-HPNIVRLHDSISEEGHHYLIFDLVTGGEL---
FUSED:LVGQGSFGCVYKATRK---------DDSKVVAIKVISKRG-RATKELKNLRRECDI---QARLK-HPHVIEMIESFESKTDLFVVTEFALMD-LH--
WEE1 :LLGSGEFSEVFQVEDP--------VEKTLKYAVKKLKVKF-SGPKERNRLLQEVSI---QRALKGHDHIVELMDSWEHGGFLYMQVELCENG-----
      .LG.G.FG.V......          ......A.K.....   ..........E...   ..... H.....................E....
```

図1 マルチプルアライメントの例

Fig.1 An example of multiple sequence alignment.

32) Corpet, F.: Multiple sequence alignment with hierarchical clustering, *Nucleic Acids Res.*, Vol.16, pp.10881-10890 (1988).

33) 石川，星田，広沢，戸谷，鬼塚，新田，金久：並列推論マシンを用いたタンパク質の配列解析，情報処理学会情報学基礎研究会, 23-2 (1991).

34) Berger, M.P. and Munson, P.J.: A novel randomized iterative strategy for aligning multiple protein sequences, *Comput. Appl. Biosci.*, Vol.7, pp.479-484 (1991).

35) Nakashima, H. *et al.*: Architecture and Implementation of PIM/m, *Proc. Fifth Gener. Comput. Syst. '92,* pp.425-435 (1992).

36) Hirata, K. *et al.*: Parallel and Distributed Implementation of Concurrent Logic Programming Language KL1, *Proc. Fifth Gener. Comput. Syst. '92,* pp.436-459 (1992).

37) Gotoh, O.: Optimal Alignment between Groups of Sequences and Its Application to Multiple Sequence Alignment, *Comput. Appl. Biosci.*, Vol.9, No.3, pp.361-370 (1993).

38) Ishikawa, M. *et al.*: Protein Sequence Analysis by Parallel Inference Machine, *Proc. Fifth Gener. Comput. Syst. '92,* pp.294-299 (1992).

39) Ishikawa, M. *et al.*: Parallel Iterative Aligner with Genetic Algorithm, *Proc. Genome Informatics Workshop IV,* Universal Academy Press, pp.84-93 (1993).

40) Tajima, K.: Multiple Sequence Alignment Using Parallel Genetic Algorithms, *Proc. Genome Informatics Workshop IV,* Universal Academy Press, pp.183-187 (1993).

41) Iyengar, A.K.: Parallel DNA Sequence Analysis, MIT/LCS/TR-428 (1988).

42) Araki, S. *et al.*: Application of Parallelized DP and A* Algorithm to Multiple Sequence Alignment, *Proc. Genome Informatics Workshop IV,* Universal Academy Press, pp.94-102 (1993).

43) Ukiyama, N. and Imai, H.: Parallel Multiple Alignments and Their Implementation on CM5, *Proc. Genome Informatics Workshop IV,* Universal Academy Press, pp.103-108 (1993).

44) Subbiah, S. and Harrison, S.C.: A Method for Multiple Sequence Alignment with Gaps, *J. Mol. Biol.,* Vol.209, pp.539-548 (1989).

45) 星田，石川，広沢，戸谷，十時：並列反復改善法によるタンパク質配列のアライメント，情報処理学会第２７回情報学基礎研究会， pp.13-24 (1992).

46) Altschul, S.F. and Lipman, D.J.: Trees, Stars, and Multiple Biological Sequence Alignment, *SIAM J. Appl. Math.,* Vol.49, pp.197-209 (1989).

47) Schuler, G.D., Altschul, S.F. and Lipman, D.J.: A Workbench for Multiple Alignment Construction and Analysis, *PROTEINS,* Vol.9, pp.180-190 (1991).

48) Hirosawa, M., Hoshida, M. and Ishikawa, M.: Protein Multiple Sequence Alignment using Knowledge, *Proc. 26th Annual Hawaii Int. Conf. Syst. Sci.,* Vol.1, pp.803-812 (1993).

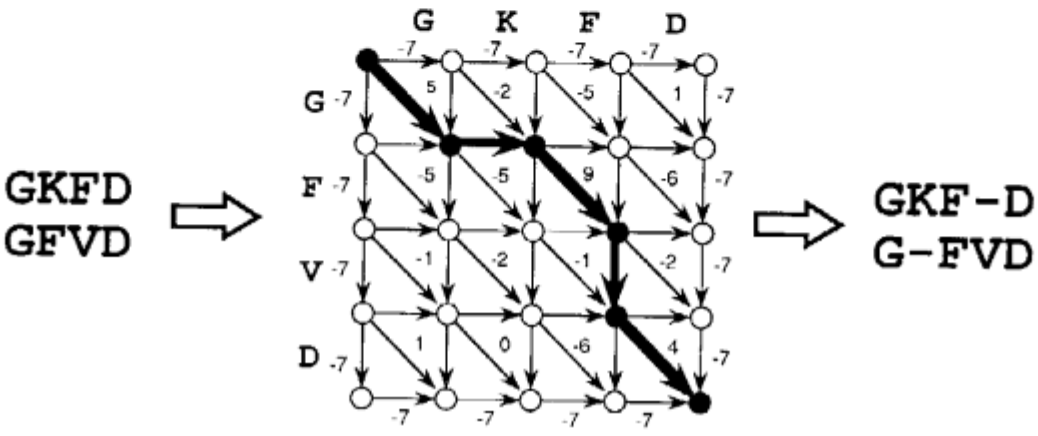


図3　2次元ダイナミックプログラミング

Fig.3 Two-dimensional dynamic programming.

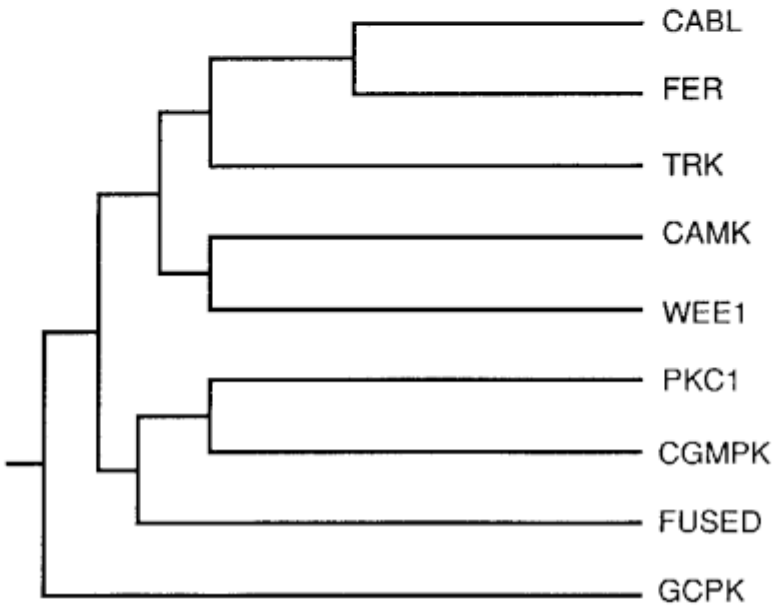


図 2 進化系統樹の例

Fig.2 An example of evolutional tree.

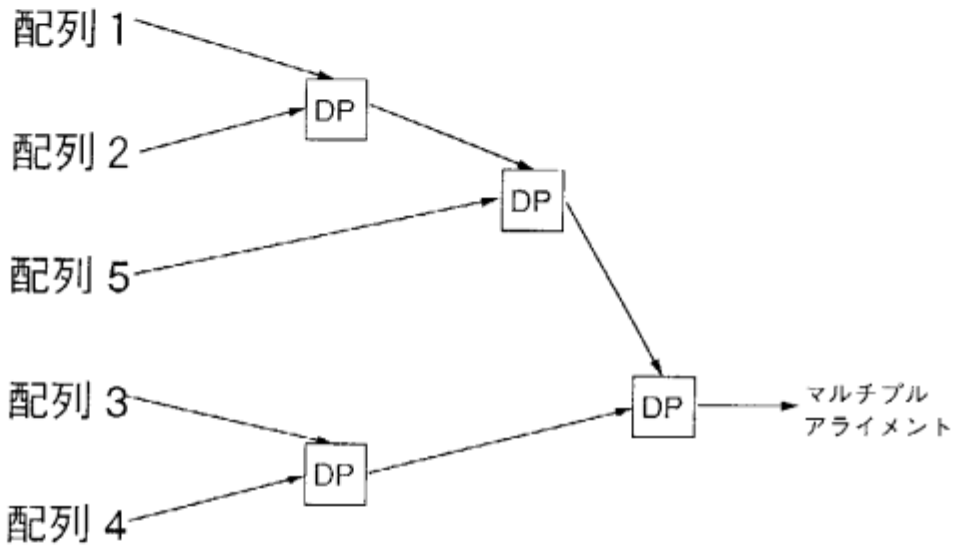
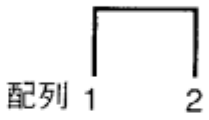


図 5 ツリーベース組合せ法

Fig.5 The tree-based method.

(a)

配列	1	2	3	4
5	914	761	598	66
4	799	832	881	
3	558	754		
2	1029			

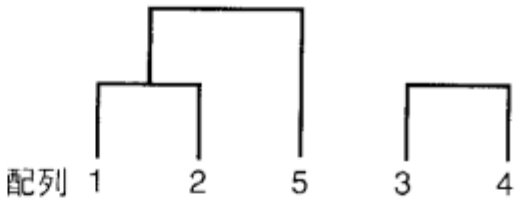


(b)

配列	1+2	3	4
5	837	598	66
4	815	881	
3	656		

(c)

配列	1+2	3+4
5	837	332
3+4	735	



(d)

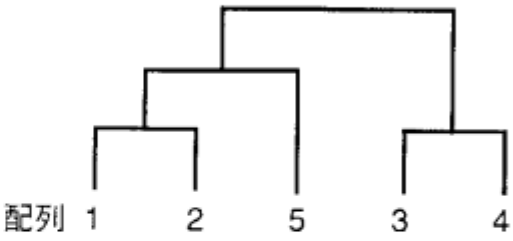


図4 ツリーの作り方（UPGMA 法）

Fig.4 UPGMA clustering.

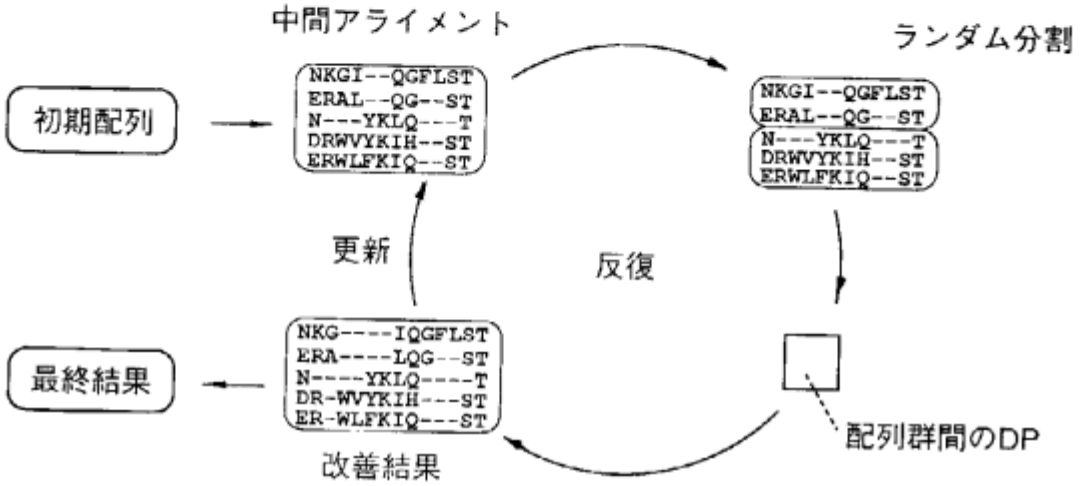


図 7 反復改善法

Fig.7 The iterative improvement method.

```
CABL :----------KLGGGQYGEVY----EGVWKK---YS-LTVAVKTLKED----TMEVEEFLKEAAVM---KEIK-HP-NLVQLLGVCTREPPFYIITE--FMTYGNLLDY-
FER  :----------LLGKGNFGEVY----KGTLKD----K-TSVAVKTCKED-LPQELKI-KFLQEAKIL---KQYD-HP-NIVKLIGVCTQRQPVYIIME--LVSGGDFLT--
TRK  :----------ELGEGAFGKVFLAECHNLLPEQ--DK-MLVAVKALKEA--SESARQ-DFQREAELL---TMLQ-HQ-HIVRFFGVCTEGRPLLMVFE--YMRH-------
GCPK :SLRGSSYGSLMTAHGKY-QIF--ANTGHFKG------NVVAIKHVNKK--RIELTR-QVLFELKHM---RDVQ-FN-HLTRFIGACIDPPNICIVTE------------
PKC1 :----------VLGKGNFGKVI----LSKSKN---TD-RLCAIKVLKKDNIIQNHDIESARAEKKVFLLATKTK-HP-FLTNLYCSFQTENRIYFAME--FIG--------
CGMPK:----------TLGVGGFGRVE----LVQLKS---EESKTFAMKILKKRHIVDTRQQEHIRSEKQIM----QGA-HSDFIVRLYRTFKDSKYLYMLMEACLGG--------
CAMK :----------ELGKGAFSVVR--RCVKVLAGQEYAA-KIINTKKLSA-----RDHQ-KLEREARIC---RLLK-HP-NIVRLHDSISEEGHHYLIFD--LVTGGEL----
FUSED:----------LVGQGSFGCVY----KATRKD---DS-KVVAIKVISKR-GRATKELKNLRRECDIQ---ARLK-HP-HVIEMIESFESKTDLFVVTE--F----ALMDLH
WEE1 :----------LLGSGEFSEVF--QVEDPVEK----T-LKYAVKKLKVK-FSGPKERNRLLQEVSIQ---RALKGHD-HIVELMDSWEHGGFLYMQVE--LCENG------
               .LG.G.FG.V.    ......    . ...A.K.....  ...... ....E....   .... H. ...................E  ...
```


図6 ツリーベース組合せ法によるアライメント

Fig.6 A tree-based multiple alignment.

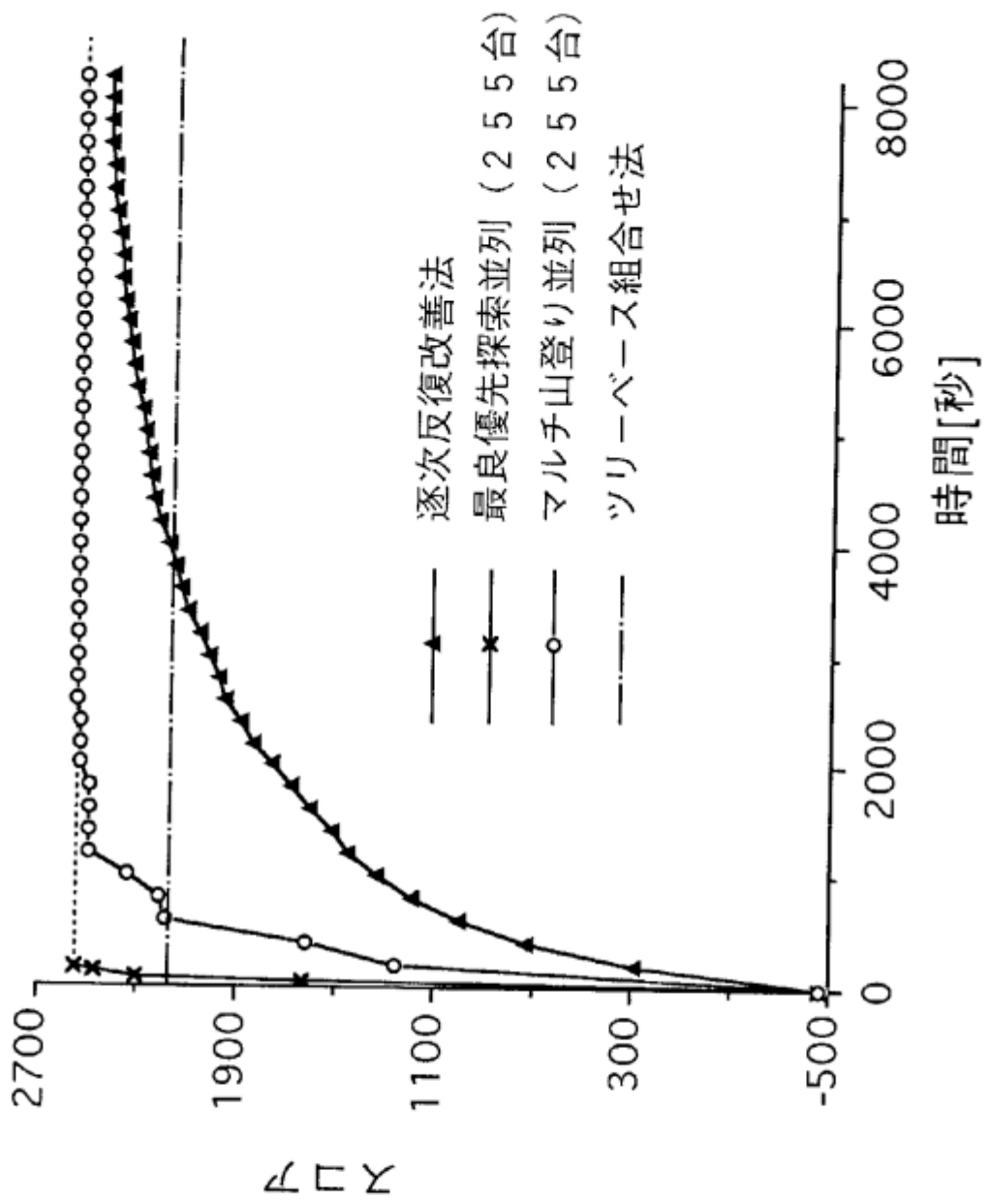


図１０ 逐次／並列反復改善法の改善履歴比較

Fig.10 Comparing improvement histories obtained by sequential/parallel methods.

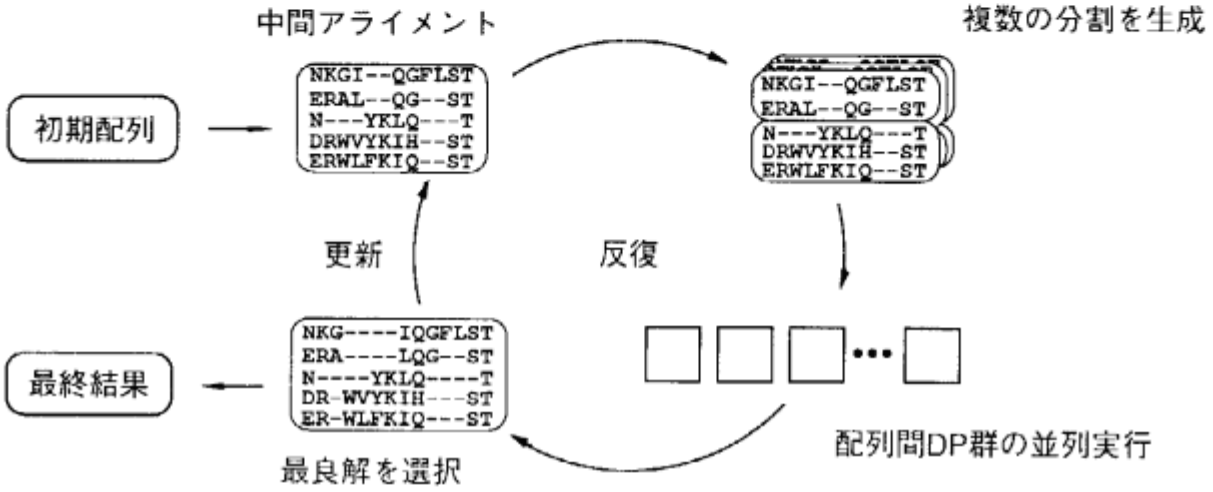


図8 最良優先探索並列反復改善法

Fig.8 The best-first parallel iterative improvement method.

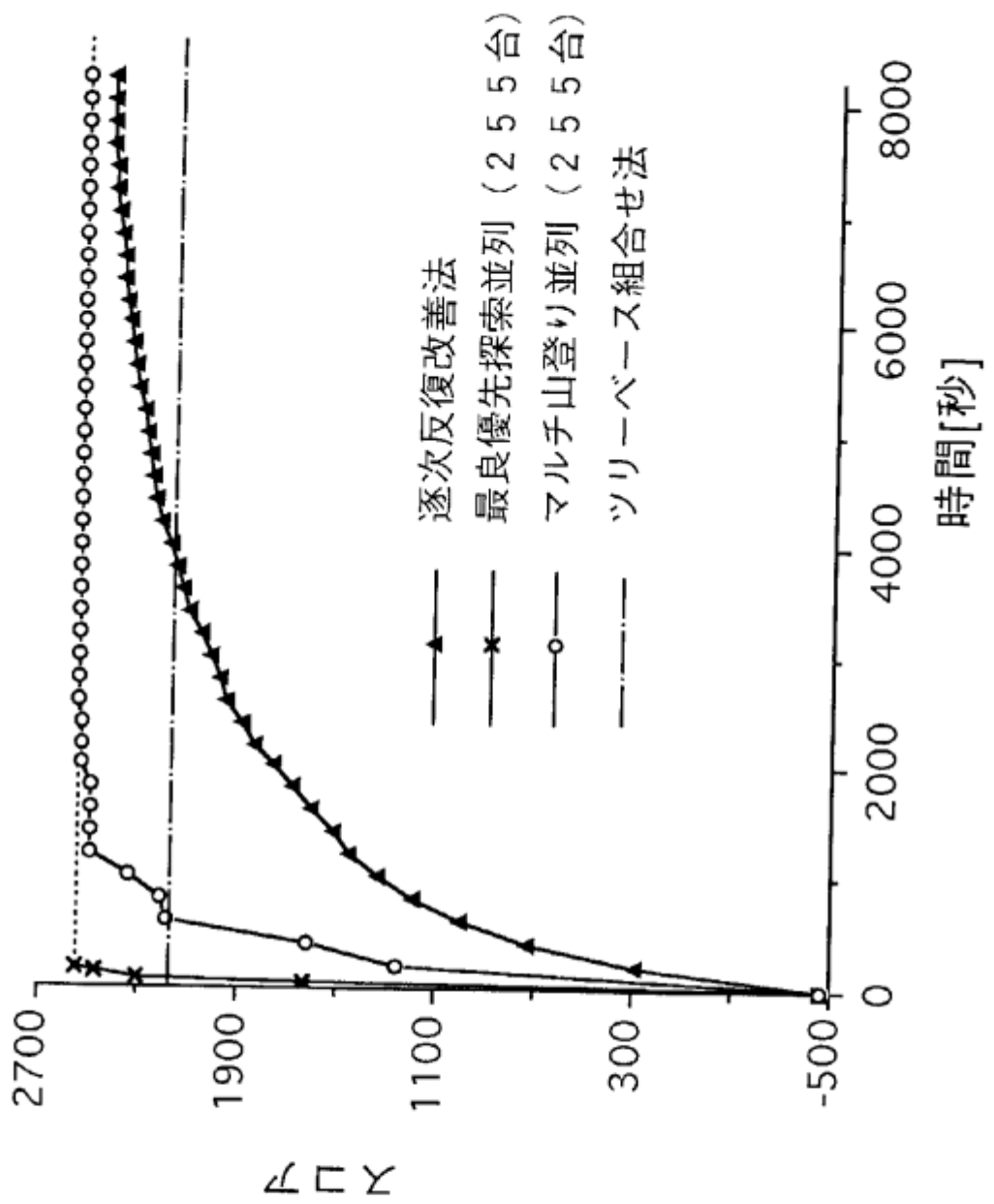


図１０ 逐次／並列反復改善法の改善履歴比較

Fig.10 Comparing improvement histories obtained by sequential/parallel methods.

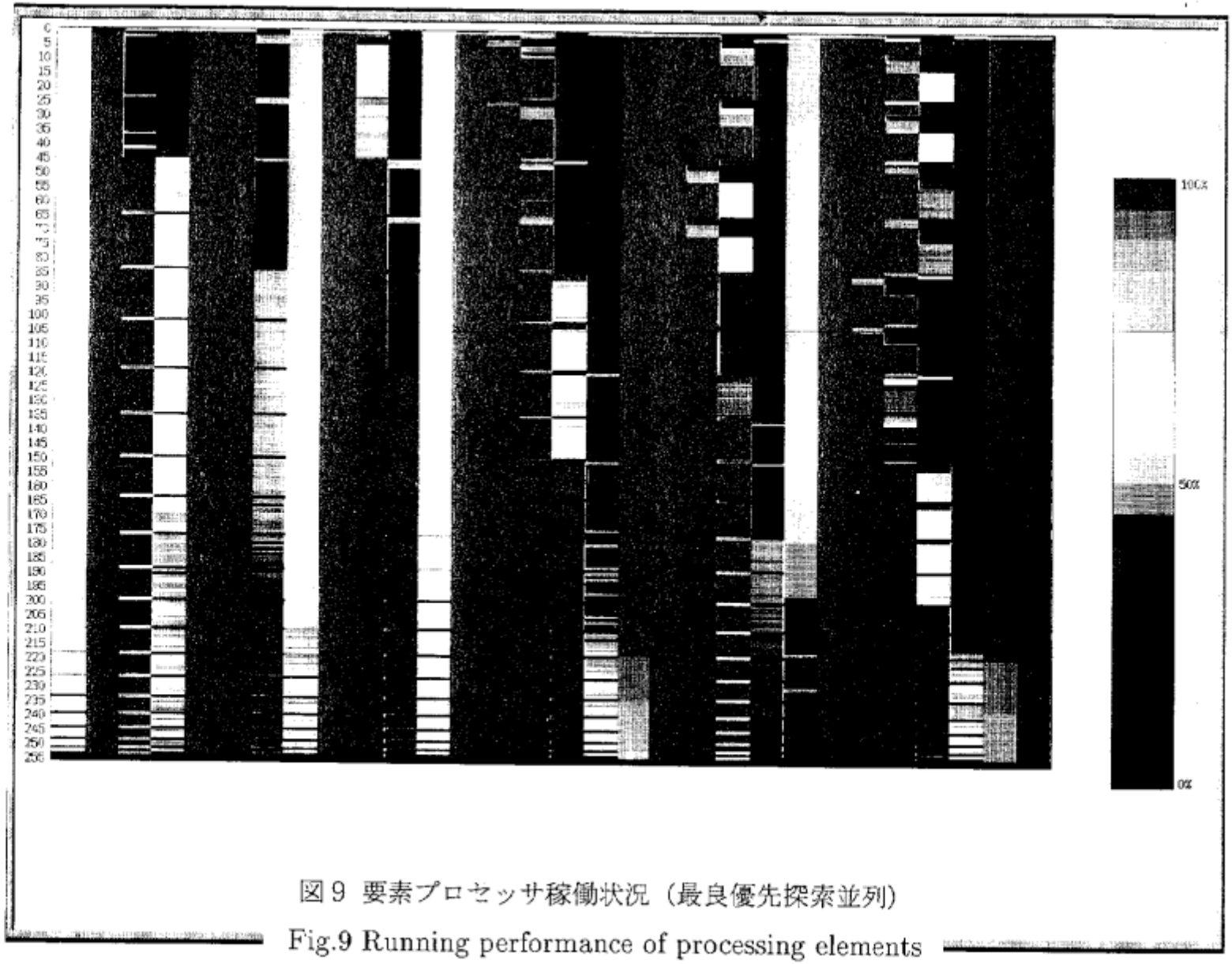


図9 要素プロセッサ稼働状況（最良優先探索並列）

Fig.9 Running performance of processing elements (best-first search).

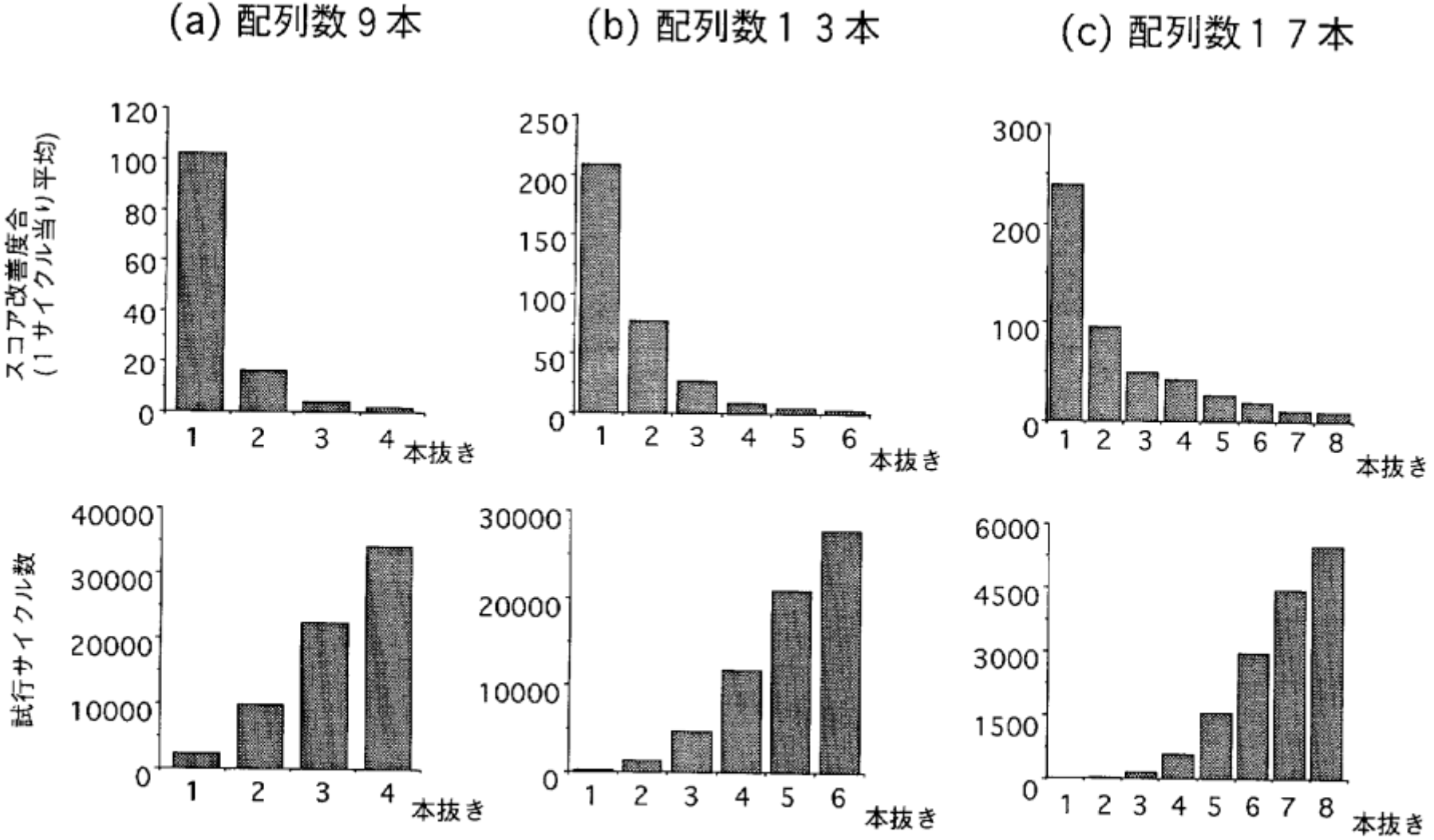


図１１ 改善度合の各分割による比較

Fig.11 Comparing improvement dependent on partitions.

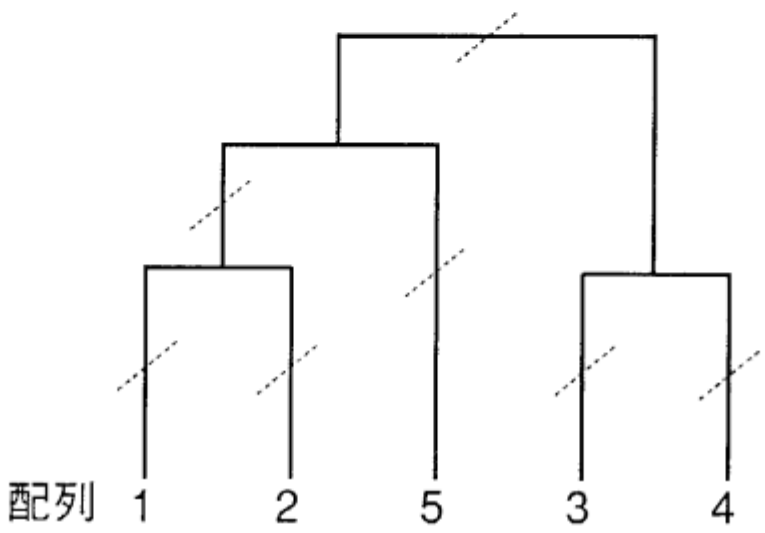


図１２ ツリー依存限定分割

Fig.12 Tree-dependent partitioning.

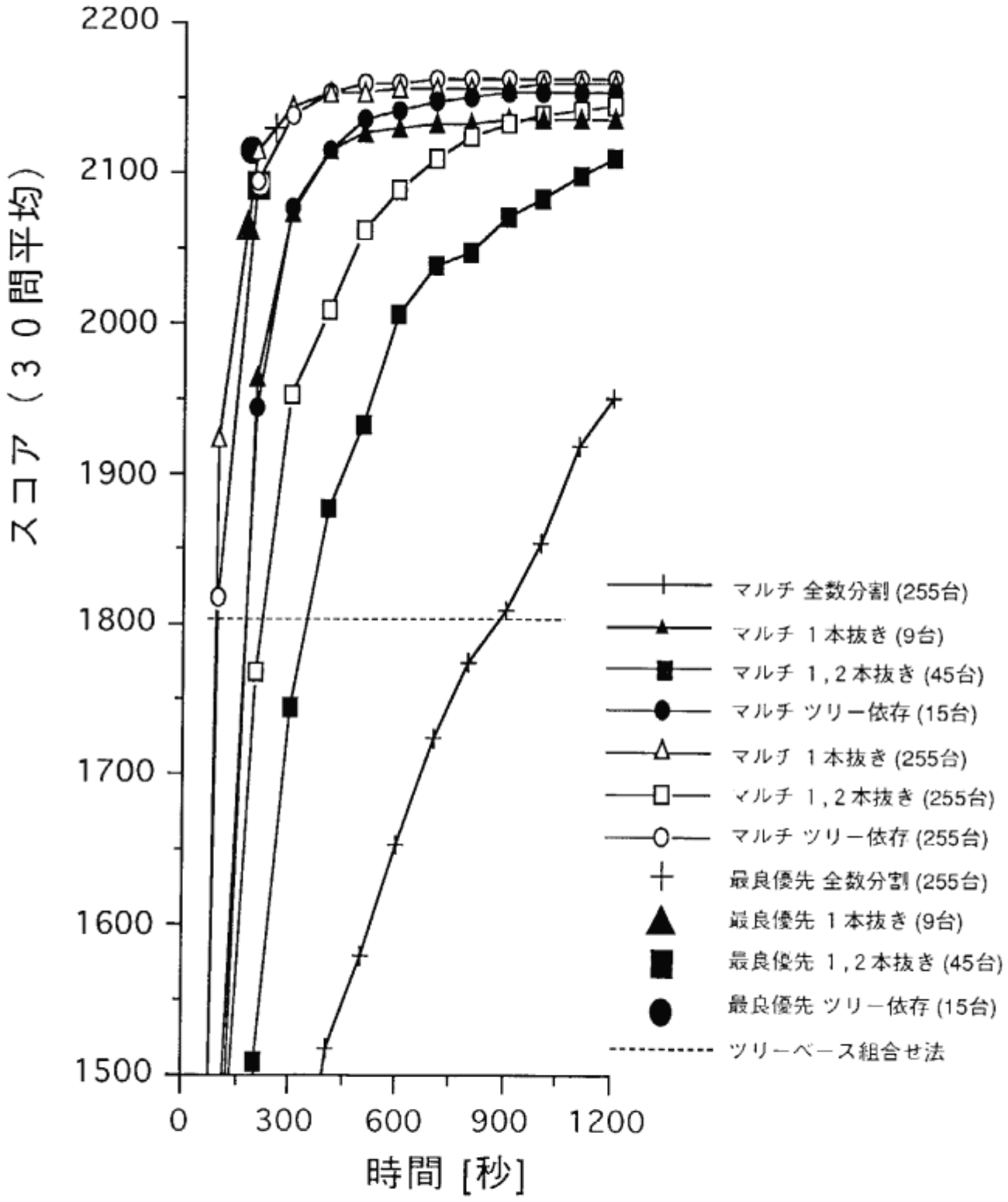


図１３ 限定分割による改善履歴比較（配列９本）

Fig.13 Comparing improvement histories obtained by restricted partitioning (9-sequence alignment).

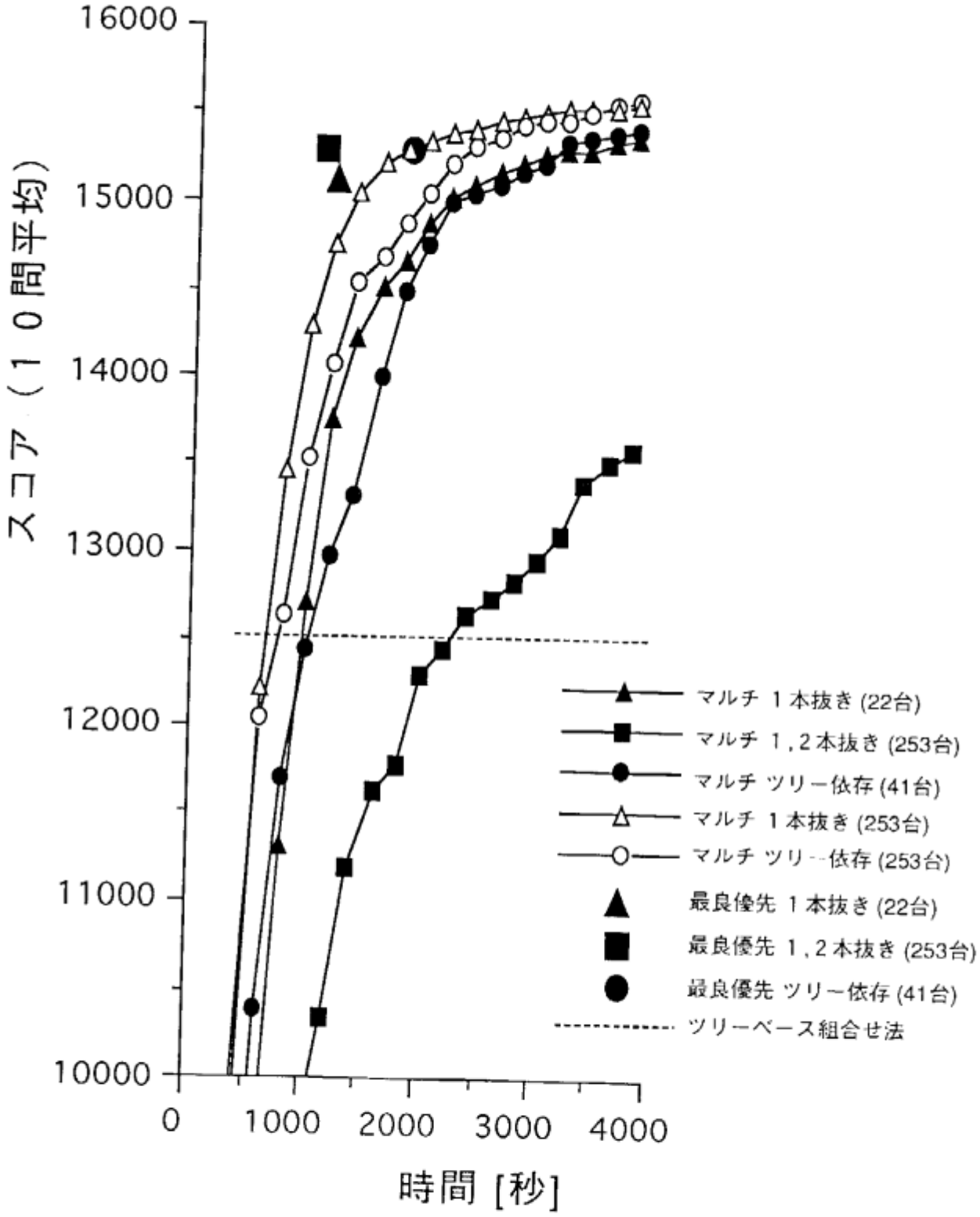


図１４ 限定分割による改善履歴比較（配列２２本）

Fig.14 Comparing improvement histories obtained by restricted partitioning (22-sequence alignment).

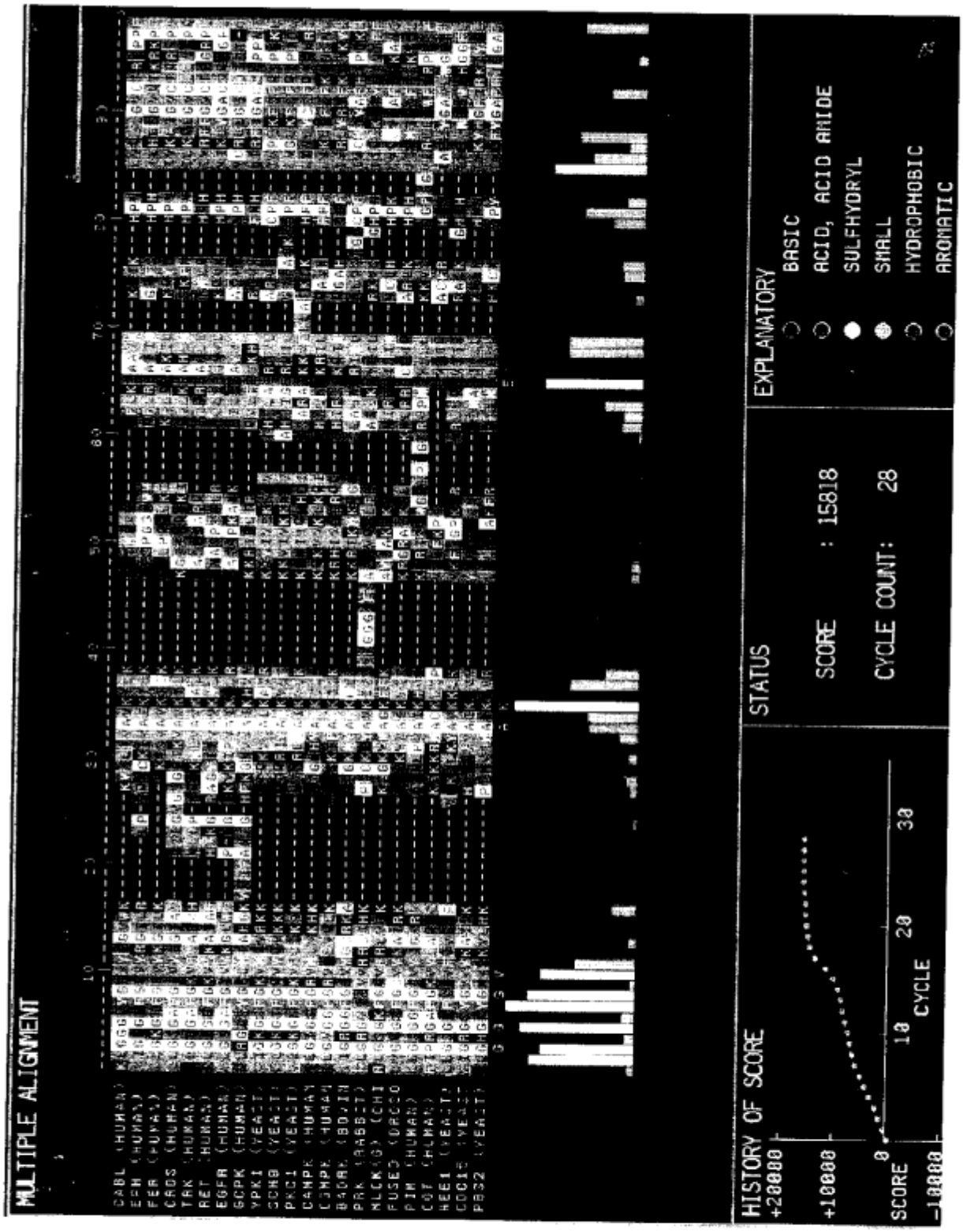


図１５ アライメント結果の例

Fig.15 An example of resulting alignment.

アミノ酸の名称	略記法	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
システイン	C	12																			
セリン	S	0	2																		
トレオニン	T	-2	1	3																	
プロリン	P	-3	1	0	6																
アラニン	A	-2	1	1	1	2															
グリシン	G	3	1	0	-1	1	5														
アスパラギン	N	-4	1	0	-1	0	0	2													
アスパラギン酸	D	-5	0	0	-1	0	1	2	4												
グルタミン酸	E	-5	0	0	-1	0	0	1	3	4											
グルタミン	Q	-5	-1	-1	0	0	-1	1	2	2	4										
ヒスチジン	H	-3	-1	-1	0	-1	-2	2	1	1	3	6									
アルギニン	R	-4	0	-1	0	-2	3	0	1	-1	1	2	6								
リシン	K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5							
メチオニン	M	-5	-2	-1	-2	-1	3	-2	-3	-2	-1	-2	0	0	6						
イソロイシン	I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	2	-2	2	5					
ロイシン	L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6				
バリン	V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	2	-2	2	2	4	2	4			
フェニルアラニン	F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9		
チロシン	Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	4	2	-1	-1	2	7	10	
トリプトファン	W	-8	2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17

表1 アミノ酸間の類似性スコア

Table 1 Similarity score between amino acids.

	全数分割	限定分割		
		1本抜き	1, 2本抜き	ツリー依存
実行プロセッサ	255台	9台	45台	15台
解の平均スコア	2129	2064	2091	2116
平均実行時間	246秒	177秒	197秒	182秒
平均改善サイクル数	16.9回	15.6回	16.1回	14.7回
サイクル当り実行時間	14.6秒	11.3秒	12.2秒	12.4秒
プロセッサの稼働率	67%	78%	74%	68%

表2 最良優先探索並列反復改善法の比較

Table 2 Comparing best-first parallel iterative improvement methods.