ICOT Technical Memorandom: TM-1258

TM-1258

モチーフ検索によるKappa-pの評価

阿比留　幸展（沖電気）

April, 1993

© 1993, ICOT

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

モチーフ検索による Kappa-P の評価

阿比留　幸展
Yukihiro Abiru
沖電気工業（株）総合システム研究所[1]
E-mail:abiru@sokilab.oki.co.jp

1　概要

ICOT で開発した非正規関係データベース管理システム Kappa 上での生物データベースを用いた性能評価について述べる。我々は Kappa の応用プログラムとしてモチーフと呼ばれる機能や構造に対応するパターンを検索する機能を実装した。この検索方式について逐次版の Kappa-II と並列版の Kappa-P[2] 上で、性能評価を行なった。本稿では特に後者の並列版の Kappa-P での台数効果が確認されたことを報告する。

なお、Kappa は非正規関係データベース管理システムであるため、生物データのような複雑な構造をしたデータを格納するのに適しており、我々は Kappa に GenBank や PIR などの主要な分子生物学の公共データベース上のデータを格納した。公共のデータベースについての解説は [1] を参照されたい。

2　分子生物データベース

近年、分子生物学の分野において計算機の役割が重要性になってきている。それは、データの種類・量が非常に莫大なものになってきていること、また、データ構造が複雑なためにもはや生物学的アプローチだけでは不十分になってきているためである。

分子生物データには主に遺伝子情報と蛋白質情報があり、前者には DNA 配列や遺伝子地図、後者にはアミノ酸配列や立体構造、機能などがある [2]。これらは相互依存の関係にあるが、公共のデータベースはそれぞれ独立して作られており、これらにまたがる情報検索は容易ではない。したがって、分子生物学においてデータベースに要求されることは、これらを統合化し、相互に参照しやすい環境を提供することである [3]。

この分野における計算機の役割は、増え続けるデータを効率よく解析してなるべく自動的に知識を得ること、および容易に解析を行えるよう形にデータを格納することである。データ解析で特に注目されているのは、比較的容易に得られるアミノ酸配列から獲得の難しい蛋白質の立体構造を予測し、さらにはその蛋白質の機能を予測することである。

ICOT では次章に述べる Kappa 上に遺伝子情報や蛋白質情報を格納し、統合化された環境を実現し、各種応用プログラムを開発している。この統合化された環境により、本稿でのモチーフ検索のような異なる分子生物学データベース間にまたがる検索も可能となった。

3　非正規関係データベース管理システム Kappa

Kappa は非正規関係データベース管理システムとして ICOT で開発されたものである [4]。Kappa-II では逐次処理を行ない、Kappa-P では並列処理を行なっている。

Kappa の特徴は非正規関係を扱えることである。非正規関係を図 1 に示した NF2 インタフェースと呼ばれる Kappa のユーザインタフェースにより説明する。その特徴としては、以下のようなものがあげられる。

- 属性が入れ子構造を持つことが出来る。たとえば、`date` という属性は`entry_date`,`seq_date`,`text_date`, などという副属性をもつことができる。これにより、属性に構造情報を持たせることが可能となる。
- 属性値として複数の値を持つことができる。従来の (正規型) 関係データベースでは、属性値の数に応じてレコード数を増やす必要があった。

[1] 1992 年 9 月まで (財) 新世代コンピュータ技術開発機構 (ICOT) に所属
[2] 本論文では単に Kappa という場合には Kappa-II と Kappa-P の両者を総称したものをいう

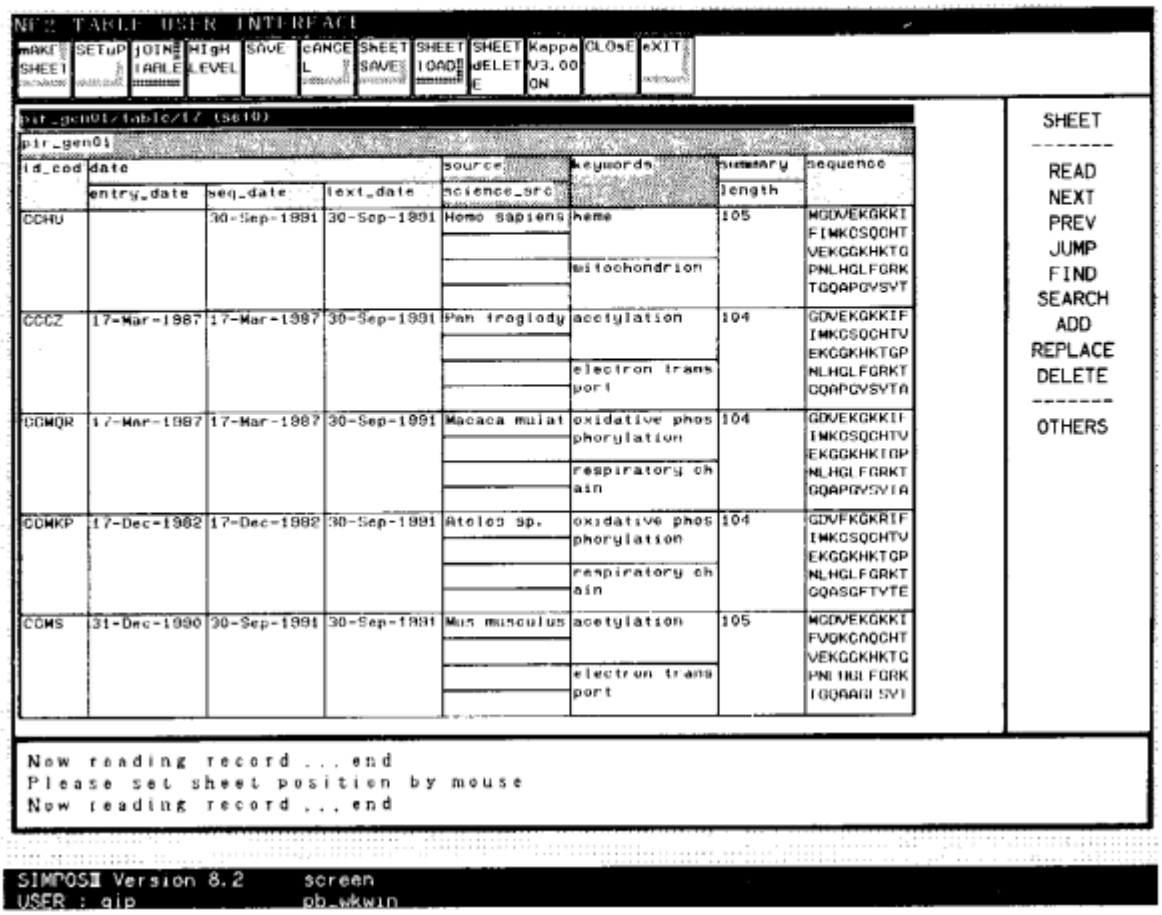

図 1: NF2 インタフェース

なお、NF2 インタフェースは図 1 のシート型の他にカード型、レコード型で表示することもでき、このインタフェースはデータの検索、更新などの機能を持っている。

4 モチーフ検索

4.1 モチーフとは

蛋白質の立体構造や機能を予測する場合には、アミノ酸の配列の中から意味のあるパターンを抽出し、その配列全体としての意味を組み立てていく必要がある。アミノ酸は 20 種類のアルファベットを用いて表現され、配列はそのアルファベットの連続した文字列で表現する。モチーフとは特定の機能や構造に対して共通に現れるパターンである。

モチーフ抽出に関してはマルチプルアライメントという方式が一般的である [5]。マルチプルアライメントは N 次元ネットワークにおける最適経路探索の問題に帰着され、主に DP マッチングという手法を用いて実現されている [6]。ICOT では、DP マッチングの他にも種々の方式を用いたマルチプルアライメントの研究を行なっている。

生物学者は発見したモチーフを正規表現風の文法規則を使って記述してデータベースに格納する。たとえば、Prosite というデータベースには、Consensus pattern という属性中にモチーフが記述してある。この記述の解釈を例を用いて説明すると "[LIVM]-C-{LIVMFYWPCST}-C-D-x(5)-C." は、対象となる文字列中に

(1) L,I,V,M のいずれか一文字があり、次に
(2) C が来て、次に
(3) L,I,V,M,F,Y,W,P,C,S,T のいずれでもない一文字が来て、
(4) C、D が連続し、
(5) 任意の 5 文字の後、C が来る、

を満たす文字列が存在する、という条件である。モチーフの記述としては他には、ある条件が文字列の必ず先頭(後尾)でなければいけないという表現や、上記例にもある繰り返しの回数をor条件で指定するという表現も含まれている。ただし、文字列の数を指定しない変数を含むことは許していない。

4.2 Kappaでのモチーフ検索

Kappaでのモチーフ検索の機能は、モチーフが記述してあるデータベース中のパターンについたモチーフ名を指定し、その条件に該当する他のデータベース中のアミノ酸配列を検索しようというものである。ユーザは検索された結果について、該当するレコード一つまたは集合をとってきたり、レコードの指定した属性のみをとってくる等アプリケーションプログラムにより自由に記述できる。

なお、実際に処理時間の評価を行なったのは、Prositeデータベース中に記述してあるモチーフを条件にPIRデータベース中のアミノ酸配列をチェックし、条件を満たしたレコードのidをとってくる処理についてである。

5 Kappa-IIでの評価

5.1 専用インデクス

モチーフ検索のような文字列検索をデータベースシステム上に実装する場合の問題点は、文字列検索では網羅的な探索が要求され、しかもうまいインデクスがつけられないということにある。我々はアミノ酸の文字20種類のうちの2種類の文字とそのギャップ(二文字の間の距離)の数を与え、その条件を満たすかどうかをチェックするという機能を持つ専用インデクスをKappa-II上に作成した[7]。モチーフ検索を行なう時にこの専用インデクスを参照することにより、対象とするレコードの数を絞り込むことができ、絞り込まれたレコードに対してのみモチーフ検索を行なえば検索の効率が向上すると考えられる。

検索にインデクスを利用するために、本プログラムでは与えられたモチーフの条件を解析することにより、絞り込み演算の条件を作成する。ここでは、二文字とギャップによる条件により絞り込み演算を行なうことをインデクス演算と呼ぶことにする。たとえば、条件"R G-D"は、

- RとGが連続した(ギャップ0)文字列を含み、かつ
- GとDが連続した(ギャップ0)文字列を含み、かつ、
- Rがあって任意のひと文字(ギャップ1)があってDが来る

というインデクス演算用の条件に分解される。"R-G-D"だと、上記のように最大3回までインデクス演算を行なうことが可能である。

本プログラムではギャップは0から4までの5通りまで実装した。インデクス演算は回数を多くすると結果のレコード数は減少する。その結果のテーブルに対してモチーフ検索を行なう。この方法は、レコードが減少されたテーブルに対して検索を行なうため、検索時間の短縮が期待される。

しかしながら、インデクス演算の回数を増やし、対象となるレコード数を減らしたからといって検索全体の処理時間は減少するとは限らない。これは、インデクス演算自身の負荷が大きいためである。したがって本稿では性能評価を、インデクス演算の回数を変化させ、それに応じて絞り込まれたレコード数および全体の処理時間を計測することにより行なう。また、インデクス演算が実行できないような条件のものとの処理時間の比較も行なう。

5.2 対象データ

PIRのバージョン25ではレコード数は約1万8千件、データ全体のサイズは約30MByte、対象となる配列の総計は、約500万である(1レコード平均約280)。また、対象とするモチーフ(条件)は

(1) 単純な文字列

"L-M-A-Q-G-L-Y-N."

(2) インデクス演算を用いない複雑な長い文字列

"[TG]-[STV]-x(8)-[LIVMF]-x(2)-R-x(3)-[DEQNH]-x(7)-[IFY]-x(7)-[LIVMF]-x(3)-[LIVMF]-x(11)-[LIVMF]-x(2)-[LIVMF]."

とする。(1) ではインデクス演算の条件を生成し、演算の回数を変化させる。たとえば、(1) でのインデクス演算の条件は { 文字、文字、ギャップ } で表すと、{L,M,0},{L,A,1},...{Y,N,0} のように最大 25 個生成される。インデクス演算の回数を指定した場合の条件の選び方は、データベース中のアミノ酸 (この場合、アルファベット一文字) の出現頻度の少ない順に選ぶことにした。これは、出現頻度が少なければ少ないほど絞り込みの効果が大きいと考えたためである。また、本プログラムでは or 条件によるインデクス演算を実装しなかったため (2) ではインデクス演算を用いずに検索することになる。これは (1) ではインデクス演算の回数の指定が 0 である場合に相当する。

5.3 実測

実測結果を以下に示す。表 1は (1) と (2) のインデクス演算を行なわなかったもの同士を比較した測定結果である。また、図 2に (1) のインデクス演算の回数を変化させた場合の測定結果を示す。図 2において左側の縦軸は全体の処理時間、右側の縦軸は中間結果のレコード数である。また、横軸は上記インデクス演算の行なわれた回数である。なお、インデクス演算の 19 回目から 24 回目までの計測は省略した。

	(1)	(2)
結果レコード数	9	24
処理時間 (秒)	544	575

表 1: Kappa-II でのインデクスを用いなかった場合の測定結果

5.4 考察およびまとめ

(1) では、or 条件や not 条件などがなく、全条件に対してインデクス演算を適用できる。この例ではインデクス演算の回数が 6 回の時に処理時間が最小値になり、7、8 回目は少し増加し、9 回目に再び処理時間が短くなっている。これは、インデクス演算の回数を多くするとレコード数は絞り込まれて少なくなるが、インデクス演算自身の処理時間がかかるためである。

また、上述のように、一度処理時間が増加してからまた減少している箇所がある。これは 9 回目に行なったインデクス演算による絞り込みの効果が大きく、それ自身の負荷を上回ったからである。このように、絞り込みの効果は、条件によりかなり異なる場合がある。例えば、似たような条件を連続して行なったような場合には絞り込みの効果は少ない。このあたりのインデクス演算の実行順序の決定方法は興味ある課題である。

インデクス演算を行なわなかった (2) については、ほぼ最悪の条件下での測定であり、(1) でインデクス演算を行なわなかった場合と比較してもやや遅くなっている。これは条件の複雑さ、および結果レコード数の差などの理由が考えられる。本稿では、代表的な例として 2 つの条件のみを記載し、30 秒程度の差であったが、実際には 10 数種類検索条件のパターンを変えて測定を行なっている。その結果として、かなりの差 (100 秒程度) が出たものもあった。

以上のことをまとめると

- インデクス演算は中間結果の絞り込みを行なうのに効果的である
- 全体の処理時間を考える場合には、適正な回数を選択する必要がある
- インデクス演算を行なわずに検索を行なう場合には、条件の複雑さ・結果数が処理時間に影響を与える

ということになる。

6 Kappa-P での評価

逐次版の場合、インデクスを利用した場合の検索は条件が良い場合は 2 倍程度の速度向上につながった。しかしながら、この条件というのは検索の対象となるモチーフが単純なものであり、かつ最適な回数の絞り込み演算が行なわれた場合のものである。たとえば、前章の (2) のように or 条件が多い場合や、not 条件が多い場合は、専用インデクスは使用できない。さらに、250 秒という時間は実用的には短い時間とは言えないし、データの量が増えた場合には更に多くの時間がかかることが予想される。実際、PIR のデータは漸次増加しており、PIR のバージョン 30

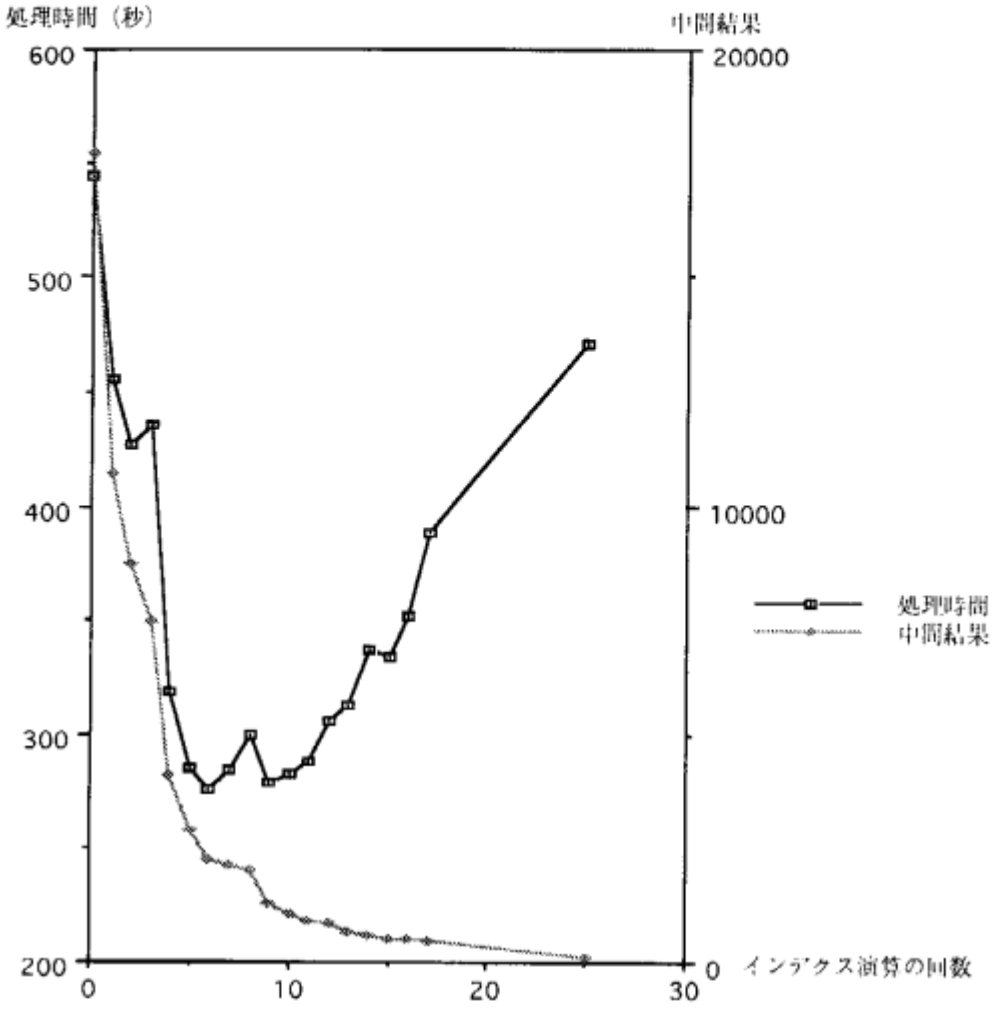

図 2: (1) の処理時間及び中間結果

を Kappa-P に格納した時は Kappa-II の時の約 2 倍、3 万 4 千レコードになった。このような大量のデータを対象にする場合には、処理を並列化して時間の短縮を図る必要がある。

そこで、並列データベース管理システム Kappa-P 上にもモチーフ検索を実装し、評価を行なった。Kappa-P では前章のような特別なインデクスを使わずに条件をチェックすることにした。それは Kappa-P には水平分割されたテーブルごとにフィルタ (対象となるレコードを検索条件によりふるい落とすアプリケーションプログラム) を実行する機能があるためである。配列情報は配列間に依存関係がないため、このフィルタの機能を利用して各プロセッサ毎に独立してモチーフ検索を行なうことが出来る。これにより負荷が分散され、台数効果が期待できる。本章ではこの並列データベース上のモチーフ検索の評価を報告する。なお、Kappa-P の並列処理方式の詳細は [8] を参照されたい。

6.1 対象データ

評価においては、台数の違いによる実行時間の変化および検索パターンの違いによる変化を考察する。ここではパターンは以下の 4 通りをとりあげる。

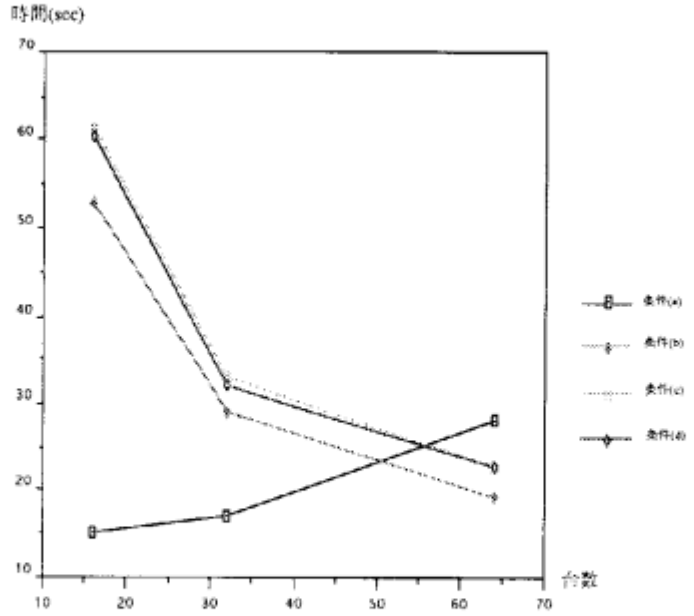

図 3: 並列データベースによるモチーフ検索

(a) 無条件全選択 "x."
(b) 解が存在しない条件 "O."
(c) 単純連続条件 "L-M-A-Q-G-L-Y-N."
(d) 繰り返し、or 条件などを含む複雑な条件
"C-x(3)-[LIVMFY]-x(5)-[LIVMFY]-x(3)-[DENQ]-[LIVMFY]-x(10)-C-x(3)-C-T-x(4)-C-x-[LIVMFY]-F-[ST]-[FY]-x(13,14)-C-x-[LIVMFY]-[RK]-x-[ST]-x(14,15)-S-G-x-[ST]-[LIVMFY]-x(2)-C"

また、測定の対象としたデータは、公共のデータベース PIR のアミノ酸配列データである。このテーブルのデータは約 61Mbyte あり、3 万 4 千のレコードを持っている。また、対象となる配列の総計は約 970 万 (1 レコード平均約 285) である。このテーブルを 16、32、64 のプロセッサに分割し測定を行った。

6.2 測定結果

結果は表 2 の通りである。表中の列の a,b,c,d は上記検索条件、行の数字は分割した台数でその値は秒である。なお、分割せず 1 台での計測は行なっていない。

	a	b	c	d
結果レコード数	33078	0	13	5
16(台)	15(秒)	53(秒)	61(秒)	60(秒)
32(台)	17	29	33	32
64(台)	20	19	23	23

表 2: Kappa-P での測定結果

また、上記表をグラフにしたものを図 3に示す。縦軸は問い合わせを発行してから応答があるまでの処理時間、横軸は分割した台数である。

6.3 考察

本計測では1台での計測を行なっていないため、前章の逐次版の結果から1台での処理時間を推測してみる。前章の条件(1)は、本章で計測した条件(c)と全く同じものであり、この処理時間により考察する。逐次版でのインデクス演算を行なっていない全検索の時間は約500秒であり、これが対象レコードが約1万8千での結果であり、上記計測が3万4千であることから推測すると、約1,000秒かかることになる。したがって、16台、32台の場合には計算通りの台数効果が出ている。

また、上記時間は問い合わせを発行して解が返ってくるまでの時間であるため、単純検索("x.")のように解の検索に負担がかからずかつ解の個数が多い場合は、並列度が大きい場合ほど結果の統合などの負担が大きいことがわかる。

さらに、逐次版では問い合わせの複雑さ、結果レコードの量により処理時間に差が出たのに対し、並列版では上記結果から、全部が解の場合や解が0のような特殊な場合を除いて、検索の条件の複雑さはさほど処理時間に差が発生しないということがわかる。これは各プロセッサ毎の負荷が小さいため、検索条件の複雑さが処理コストの差に出にくいためと推察される。例えば、前章で100秒程度の差が出たパターンの場合でも32台、または64台に分割されたとすれば2、3秒程度の差ということになる。実際、上記例の他に10数種類のパターンで計測を行なってみたが、同じ台数で各パターン毎の差は3秒以内であった。

また、前章で述べた専用インデクスを水平分割されたテーブル毎に作成し、絞り込みを行なう方式も考えられるが、前章でのインデクス演算の負荷、および上記結果の処理時間からすると、絞り込みの効果よりもインデクス演算自身の負荷の方が大きくなることが予想される。このため、本システムではKappa-Pに対応するインデクスは作成しなかったが、データの量が更に増えて、各プロセッサごとの処理の負荷が大きくなった場合はインデクス作成の検討を要する可能性がある。

7 まとめ

分子生物学データを並列データベース上に格納し、モチーフ検索プログラムを実装し、処理速度による性能評価を行なった。文字列検索など、全検索を必要とする応用プログラムはデータが大量になった場合に並列処理が有効であり、実測により効果も確認できた。今後のデータベースへの拡張機能としては類似検索やキーワード検索などが考えられ、検討していきたい。

謝辞

最後に本研究に関して多くの示唆を頂いたICOTの田中秀俊、河村元大岡研究員、横田一正主席研究員に感謝いたします。また、本稿を作成するにあたって多くの助言をいただいた沖電気の総合システム研究所知識情報処理研究部のメンバーにも感謝いたします。

参考文献

[1] 高木利久: 「ゲノムデータベース」、情報処理学会誌 Vol.33, No.10, (1992)

[2] Lesk, A.M. (editor) : *Computational Molecular Biology Sources and Methods for Sequence Analysis*, Oxford Univ. Press (1988).

[3] 田中秀俊: 「分子生物学のデータベース」、情報処理学会研究報告、91-DBS-84-23 (1991)

[4] 河村、横田、金核上: 「知識ベース管理システム Kappa の概要」、人工知能学会誌 Vol.4,No.3, (1989)

[5] Barton, G.J. and Sternberg, M.J.: "A Strategy for the Rapid Multiple Alignment of Protein Sequences: Confidence Levels from Tertiary Structure Comparisons", *J. Mol. Biol.*, 198, pp.327-337 (1987).

[6] Masato Ishikawa,et al:"Protein Sequence Analysis by Parallel Inference Machine",*Proceedings of International Conference on Fifth Generation Computer Systems 1992*, pp.294-299

[7] 平成3年度通商産業省委託業務 電子計算機基礎技術開発成果報告書・ソフトウエア編: pp.371-373

[8] Moto Kawamura, et al.: "Parallel Database Management System:Kappa-P" *Proceedings of International Conference on Fifth Generation Computer Systems 1992*, pp.248-256