アルゴリズム 52－4
（1996．7．24）

複数生物学的配列のアラインメントのパラメトリック的考察

渋谷 哲朗　今井 浩

東京大学大学院理学系研究科情報科学専攻
〒113 東京都文京区本郷 7-3-1

分子生物学において、DNA や蛋白質などの文字列のアラインメントの問題は重要な問題である。しかし、得られた最適解が生物学的に最適なアラインメントであるとは限らないために、パラメータを変化させて、最適解をいろいろ出して見るなどの必要がある。この時、パラメータの変化をどのようにするかについての研究は 2 本のアラインメントの問題ではなされているが、複数アラインメントの問題ではなされていなかった。本論文では、この複数アラインメントの問題において、ギャップのペナルティに関してパラメトリック的な実験、考察を行なう。

On Parametric Multiple Alignments of Biological Sequences

Tetsuo Shibuya　Hiroshi Imai

Department of Information Science, Faculty of Science, University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo 113, Japan

The alignment problem of DNA or protein sequences is very important in molecular biology. In this problem, the obtained optimal solution is not always the biologically best alignment. Thus, it is required to vary parameters and check the varying optimal alignments. The way to vary parameters has been studied well on the problem of only two sequences, but not in the multiple alignment problem. This paper does parametric analysis of gap penalty on the multiple alignment problem through experiments.

1 はじめに

一般的にアラインメントの問題は与えられた文字同士の得点表に基づいて、最適解を求める最適化問題と考えられる。この問題は分子生物学の遺伝子情報処理の分野で、3 次元構造予測や機能予測などで使われたりする、非常に重要な問題である。

アラインメントの問題は、小規模の 2,3 本の配列の問題では、ダイナミック・プログラミング (DP) がよく用いられる。この方法ではサイズ n の d 本の配列のアラインメントの問題で、$O(n^d)$ の時間計算量・空間計算量を必要とし、より多くの本数を対象にしたアラインメントでは適用は難しい。

A*アルゴリズムはよく知られた探索手法であるが、強力な評価値を用いると、探索領域を非常に減らすことができることが知られている。アラインメントの問題でも、適当な評価値を用いることによって、5,6 本の配列のアラインメントでもかなり容易に解き得ることが示されている [6, 7]。本研究では、この手法を主に用いている。

ギャップのペナルティなどのパラメータをかえることによって、最適解はかわるが、生物学的に最適なアラインメントに対応するパラメータは個々の例によって様々であることが知られている。従って、いろいろなパラメータを変えて、最適解を求める必要性がある。それにともない、パラメータ空間において、どの最適解がどの領域を占めるか、という研究が、2 本のアラインメントの問題の場合ではよく研究されている [4, 5, 9, 11, 10, 12]。しかしながら、3 本以上の複数アラインメントについては、まだパラメータを変えた時の最適解の変化については研究はなされていない。

本研究では、複数配列ののアラインメント問題について、ギャップのペナルティーに関してそのようなパラメータ空間の領域分割を行なう。また、それに必要な計算時間や、パラメータを変えるのに従って、どのように最適解を解く計算量が変わるかなどについて実験を通して考察を行なう。

2 アラインメントの問題

d 本の文字列が与えられた時に、文字同士の似ている度合を考えて、適当にギャップをいれて整列させる問題を文字列アラインメント問題という。2 本のアラインメントの場合では、文字同士の似た度合（蛋白質ならばアミノ酸同士の遷移確率をもとにしたものなど [2]）を表したスコアを与えて、各列のスコアを計算し、その和からギャップに関するペナルティーをたしたものを考える。例えば

```
Score of     Score of     Score of
 LF-     =    L      +     F      +  ギャップの
 LLE          L            L       ペナルティ
```

となる。2 本のアラインメントの場合、ギャップのペナルティは長さ x のギャップに対して $a+bx$ のような式で与えることが多い (affine ギャップと呼ばれる) が、複数アラインメントの場合では最適解を求めることが極めて困難なので、bx と簡単化した場合を考える (linear ギャップと呼ばれる)。

2 本の場合はこれを最大化したものを、それ以上の本数の場合は、2 本ずつの組のスコアの和を最大化したものを、最適解とする最適化問題としてとらえることができる。2 本以上の場合では、例えば、

```
 TKLF--      TKLF--                    TKLF--
 KALLER      KALLER      KALLER
 HS-FTH   =           +  HS-FTH   +    HS-FTH
 のスコア    のスコア    のスコア       のスコア
```

となる。

d 本のアラインメントの問題は、d 次元格子状グラフでの最短路問題に変換することができる(図 1)。$V = \{(x_1, \ldots, x_d) | x_i = 0, 1, \ldots, n_i\}$ で、$E = \{(v, v+e) | v \in V, e \in [0,1]^d, e \neq \mathbf{0}\}$ であるようなグラフ $G = (V, E)$ を考える。各枝の長さを対応する文字同士あるいは文字とギャップのペナルティと同じものと考えると、最長路問題として、とらえることができ、正負を逆にすることによって、最短路問題に帰着できる。(ここで、最短路問題に帰着するのは、A*アルゴリズムを使用するためである。)

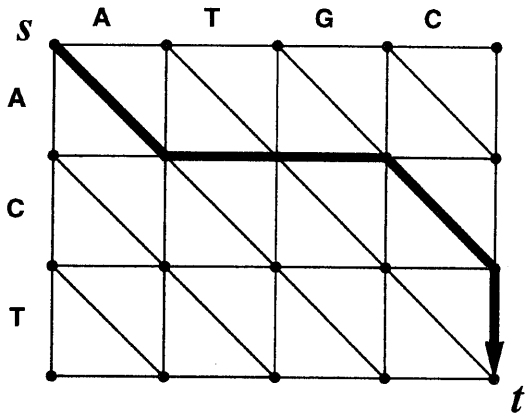

図 1: ATGC と ACT のアラインメントを表すグラフ。太線で描かれた s-t のパスは ATGC- と A--CT というように整列させたアラインメントに対応している。

このような非常に規則的な構造を持ったグラフ上での問題であるため、小規模な問題では DP が非常に有用である。しかしながら、大規模になると、必要となるメモリーのサイズが非常に莫大なものとなるため、A*アルゴリズムなどを使うとよいことが知られている。

ダイクストラ法は始点から近い点から順に探索して行くアルゴリズムであるが、A*アルゴリズムは始点から

(始点 s から点 v までの最短路長)+(その点から終点 t までの最短路長の下限値 $h(v)$)

の小さいものから順に探索して行くアルゴリズムである。この下限値として、2 本のアラインメントでは、荒木ら [1] によって、また、それ以上の複数アラインメントでは池田ら [6, 7] によって提案されたものが存在している。

- $d = 2$ の時： スコア表をもとに下限値を計算する。
- $d > 2$ の時： 2 本ずつの組の部分問題の最適解に対応する最短路長の和を下限値として用いる。

3 パラメトリック問題について

アラインメントの最適解が、ギャップのペナルティの変化などによってどのように変化するか、ということを調べるには、単にパラメータを少しずつ変化させて、実際に計算する、というのは無謀である。なぜならば、わずかな変化では最適解に変化が全くないことも考えられるし、逆に、変

化させた量より小さい範囲内で、重要な変化が起きているにもかかわらず、気がつかないこともあり得るからである。

このような背景から、2本アラインメントの場合のパラメータ空間をどの解が最適であるかによって領域分割する研究がいろいろとなされている [4, 5, 9, 11, 10, 12]。変化させるパラメータが1つだけの場合はかなり容易にこの領域分割をすることが可能である [4, 10]。

linear のギャップのペナルティに関して考えてみる。1つのギャップに対するペナルティを b とする。ギャップを g_i 個持つある一つのアラインメント a_i のスコア $s_i(b)$ は、

$$s_i(b) = s_i(0) + g_i \cdot b$$

と表せられる。ここで、$b_1 < b_2$ なる b_1,b_2 について、$b = b_1$ での最適なアラインメントのうちギャップの数が最大のものを a_1、$b = b_2$ での最適なアラインメントのうちギャップの数が最小のものを a_2 とする。この2つのアラインメントは $b = b_3 = -\frac{s_1(0)-s_2(0)}{g_1-g_2}$ において、同じスコアを持つ。もし、$b = b_3$ においてこの二つのアラインメントよりよい解が存在しなければ、$b_1 \leq b \leq b_2$ では、この二つのアラインメントのどちらかが必ず最適解として存在し、その境界は b_3 であることがわかる。もし、そうでないのであれば、b_3 と b_1,b_2 の間において、再帰的に同じことを行なうことによって、この $b_1 \leq b \leq b_2$ の間を完全に領域分割することが可能である。

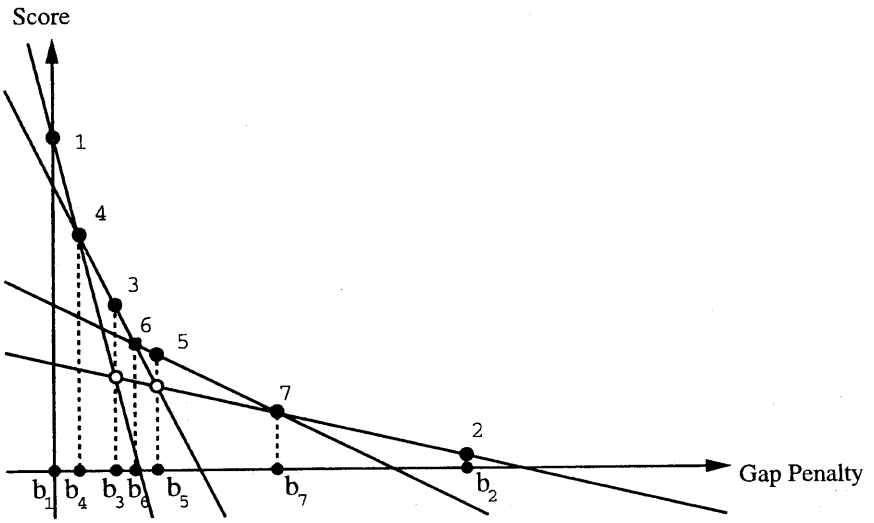

図 2: b_1 と b_2 の間の領域分割の例。ここでは、4つの領域に分割されている。再帰計算の回数は7回である。なお、添字は計算の順番と同じである。

この領域分割については、領域の数を n とすると $2n-1$ 回最適解を計算すればよいだけなのでかなり効率的である。これは、コストがギャップに関して線形であることによっている。

ここで、ギャップの数の最大のもの、最小のもの、というものが必要となってくる。これは、DPでも、A*アルゴリズムでも、スコアとギャップの個数の組 (s, g) に関して辞書的順序で最短のパスを求めるだけなので、適当な拡張によって、解くことが可能である [4]。

しかしながら、最適解が少なければ、全部列挙することも可能である。最適解よりΔ以上悪くない準最適解すべてを列挙する効率的なアルゴリズムが存在する [3, 8] ので、これが最適解すべての列挙に使える。このアルゴリズムは出力に対して線形時間のオーバーヘッドしかかからないの

で、最適解が少ない場合は、こちらの方が速い場合も多いと考えられる。パラメトリック解析は最適解を詳しく調べることが目的なので、少なくとも、最適解すべてを出力する方が好ましい、ということもあり、本論文では、こちらのアルゴリズムを採用している。

パラメータが2つ以上の場合のパラメータ空間でもこのような領域分割は可能であるが、計算量がかなり増加することが知られており [4]、実際に適用することを考えた場合には、実用的ではないと考えられる。

4 ギャップペナルティに関するパラメトリック実験

実際のアミノ酸配列 EF-1α数本を用いて、実際に計算機実験を行なった。EF-1αは長さ約430程の非常に類似度の高いグループである。スコア行列としては、PAM-250 を用い、すべての計算機実験は Sun Ultra 1(メモリ：128Mbytes) 上で行なった。

表 1: 実験に用いた蛋白質配列。d 本の実験では上から d 本を使っている。

Sequences			Pairwise Scores				
Species	Protein	Length	Met	Tha	Thc	Sul	Ent
Halobacterium marismortui (Hal)	EF-TU	421	1329	1314	1221	1109	1099
Methanococcus vannielii (Met)	EF-TU	428		1336	1247	1150	1176
Thermoplasma acidophilum(Tha)	EF-1α	424			1311	1261	1233
Thermococcus celer (Thc)	EF-1α	428				1132	1130
Sulfolobus acidocaldarius (Sul)	EF-1α	435					1192
Entamoeba histolytica (Ent)	EF-1α	430					

ギャップのペナルティは通常、スコア行列の中で最小の値と同じにとることが多く、その値はこの場合−8 である。0 以上のギャップのペナルティは全く意味がないし、0 に近い値も意味がない。そこで、$d = 2$ から $d = 6$ の複数アラインメントの問題において、基本的には−16 と 2 の範囲で領域分割を行なった結果を表 2に示す。ただし、後で説明する理由から、$d \geq 5$ では、大きい側の範囲を若干狭くしている。

表 2の見方であるが、各 d に関して、1 行目は領域の境界となっているギャップのペナルティである。ただし、左右の端の値は計算した範囲の端であって、領域の境界であるとは限らない。2 行目はその値における最適解の数である。3,4 行目はギャップの数が最大のもの、最小のものの解の数である。ここに、-が入っているものは、ギャップの数がすべて等しい、すなわち、実際の領域の境界ではないということを表している。なお、この表において、境界を浮動点小数で書いているが、実際の計算はすべて有理数で行なっている。

この表からわかることとしては、すでに 2 本のアラインメントの問題ではわかっていたことではあるが [4, 9, 11, 10, 12]、d が大きくなっても、領域の大きさはギャップペナルティが大きいほど小さくなることがわかる。

また、少なくともここで計算したすべての場合において、最適解が複数あることが判明した。しかも、殆んどの最適解の個数が 2 の冪乗またはその倍数であるようなことが多いようである。最適解のひとつのある部分を適当に変えてもスコアが同じ、というような部分が複数ある時に、それぞれの部分の変え方 2 通りであれば、容易にその最適解と同じスコアのアラインメントを 2 の冪乗個得ることができる。これがその理由であると考えられるが、実際、解を表示してみると、そのようになっていることが多いことがわかった。

表 2: ギャップのペナルティに関するパラメトリック実験結果

$d=2$	Gap penalty	-16	-5	-3	-2.5	-2			
	#Solutions	4	12	24	192	576			
	#Max	-	8	16	8	32			
	#Min	-	4	8	16	8			
$d=3$	Gap penalty	-16	-3.5	-3	-2.75	-2.5	-2.2	-2	
	#Solutions	8	16	24	32	72	48	256	
	#Max	-	8	16	16	16	32	96	
	#Min	-	8	8	16	16	16	32	
$d=4$	Gap penalty	-16	-8	-3.83	-3.5	-2.5	-2.33	-2.25	-2
	#Solutions	16	32	32	32	32	48	160	4608
	#Max	-	16	16	16	16	32	128	384
	#Min	-	16	16	16	16	16	32	128
$d=5$	Gap penalty	-16	-7.5	-4	-3.38	-3.17	-3	-2.88	-2.75	-2.5
	#Solutions	2	4	4	4	4	4	12	8	24
	#Max	-	2	2	2	2	2	4	4	4
	#Min	-	2	2	2	2	2	2	4	4
$d=6$	Gap penalty	-16	-6.5	-4.5	-4	-3.5				
	#Solutions	4	16	8	8	4				
	#Max	-	4	4	4	-				
	#Min	-	4	4	4	-				

$d \geq 5$ でギャップペナルティの大きい側の計算範囲を狭めているが、これは、計算時間の問題による。ギャップのペナルティを大きくすると（すなわち絶対値を小さくすると)、ギャップの許容範囲が大きくなり、そのために最適解の数が多いだけでなく、A*アルゴリズムによる探索範囲が広がるために計算時間がかなり増加するためである。たとえば、$d=4$ のアラインメントでの A* アルゴリズムで探索された点数をギャップペナルティを -16 から -1.5 の間でギャップペナルティを 0.5 きざみでプロットしたものが図 3である。

この表を見ることにより、ギャップペナルティが大きくなるにつれ、爆発的に探索点数が増えていることがわかる。この探索領域の爆発的増加のために、128Mbyte のメモリでは、d が大きいとギャップペナルティが -2 付近の計算ができなかった。A*アルゴリズムは元来、相似性の高いアラインメントにおいて効果的であると言われている。対象の相似性が高いにもかかわらず、A*アルゴリズム探索領域が大きいということは、ギャップのペナルティの設定が悪い、ということでもある。このようなことから、ギャップのペナルティを -3 あたりより大きく設定するのは意味がないものと考えられる。

5 今後の課題など

本論文では、ギャップのペナルティの変化に応じての最適解の変化を観察することに成功した。しかし、最適なアラインメントを決めるパラメータはギャップのペナルティ以外にもいろいろあり、さらなる研究が必要である。また、パラメータを変えた際の準最適解の変化なども、調査する

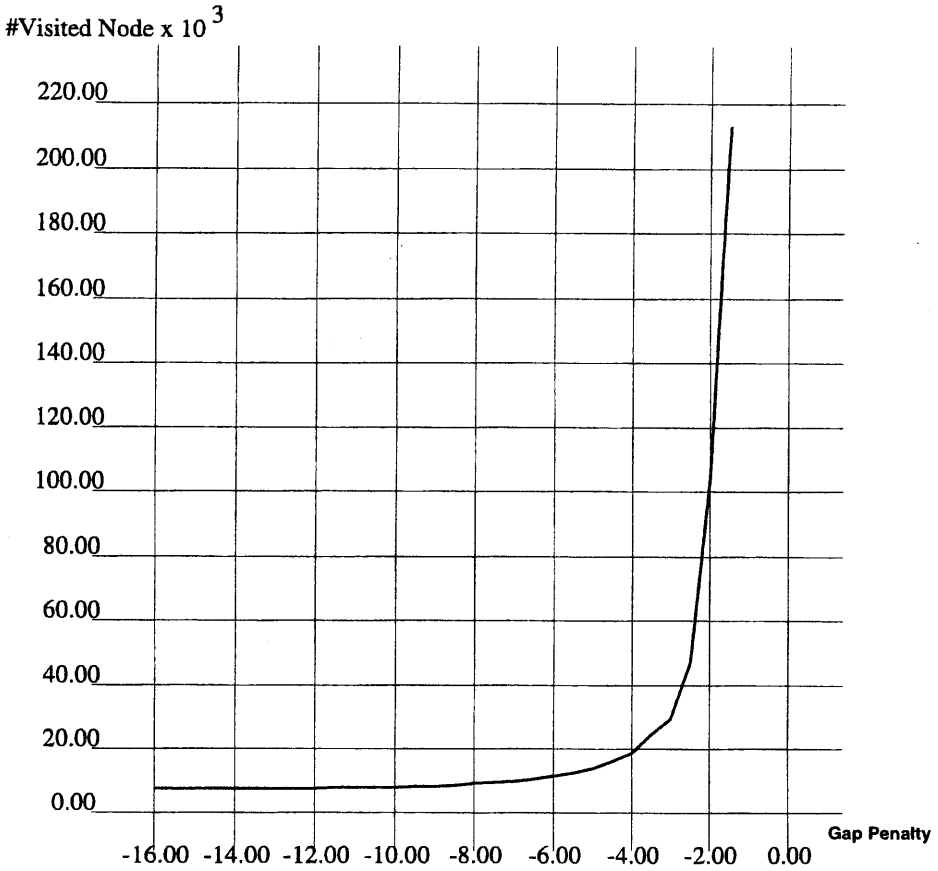

図 3: ギャップペナルティの変化に伴う A*アルゴリズムでの探索点数の変化

必要があると思われる。また、これらの手法はアラインメントの問題以外にも適用できる可能性があり、その方向性での研究も必要かと思われる。

謝辞

本研究は文部省科学研究費重点領域「ゲノムサイエンス」の援助を受けた。

参考文献

[1] S. Araki, M. Goshima, S. Mori, H. Nakashima, S. Tomita, Y. Akiyama and M. Kanehisa, "Application of Parallelized DP and A* Algorithm to Multiple Sequence Alignment," *Proceedings of Genome Informatics Workshop IV*, pp.94-102, 1993.

[2] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, *Atlas of Protein Sequence and Structure,* National Biomedical Research Foundation, Washington D. C., 1978.

[3] David Eppstein, "Finding the k Shortest Paths" *Proceedings IEEE Foundation of Computer Science*, 1994, pp154-165.

[4] D. Gusfield, K. Bakasubramanian and D. Naor, "Parametric Optimization of Sequence Alignment," *Proceedings of 3rd ACM-SIAM Annual Symposium on Discrete Algorithms,* pp. 432-439, 1992.

[5] X. Huang, P. A. Pevxner and W. Miller, "Parametric Recomputing in Alignment Graphs," *Proceedings of 5th Annual Symposium on Combinatorial Pattern Matching,* pp. 87-101, Springer-Verlag LNCS 807, 1994.

[6] T. Ikeda, "Applications of the A* Algorithm to Better Routes Finding and Multiple Sequence Alignment," *Master's Thesis,* Dept. of Info. Sci., Univ. of Tokyo, 1995.

[7] T. Ikeda, and H. Imai, "Fast A* Algorithms for Multiple Sequence Alignment," *Proceedings of Genome Informatics Workshop V*, pp.90-99, 1994.

[8] T. Shibuya and H. Imai, "Suboptimal Alignments of Multiple Biological Sequences," *IPSJ SIG Notes SIGAL51*, Information Processing Society of Japan, pp. 1-8, 1996.

[9] M. Vingron and M. S. Waterman, "Sequence Alignment and Penalty Choices: Review of Concepts, Case Studies and Implications," *J. of Mol. Biol.* 235: pp. 1-12, 1994.

[10] M. S. Waterman, "Introduction to Computational Biology: Maps, sequences and genomes," Chapman & Hall, 1995.

[11] M. S. Waterman, "Parametric and ensemble sequence alignment algorithms," *Bull. Math. Biol.* 56: pp. 743-767 1994.

[12] M. S. Waterman, M. Eggert and E. Lander, "Parametric sequence comparisons," *Proc. Natl. Acad. Sci. USA,* 89: pp. 6090-6093, 1992.