Vol.2015-MPS-103 No.68
Vol.2015-BIO-42 No.68
2015/6/25

三次元空間における効率良い近似点集合マッチングと分子パターン照合への応用

佐々木 耀一[1,a)]　渋谷 哲朗[2]　伊藤 公人[3]　有村 博紀[1]

概要：本論文では三次元近似点集合マッチング問題の実際的に効率良いアルゴリズムについて研究する．入力として 3 次元のデータ点集合 T と，サイズ k のパターン点集合 P，正実数 $r > 0$ が与えられたとき，P との最小二乗和平均平方根距離（MinRMSD）が r 以下となるデータ点のサイズ k の部分集合をすべて見つけるアルゴリズムを与える．このアルゴリズムは順列の列挙に基づくため最悪時に $O(kn^k)$ 時間を要するが，解の網羅性で利点をもっており，より精密なアルゴリズムと組合せることで分子パターン照合などの応用に用いることができる.MinRMSD の上限関数を用いた分枝限定法による高速化によって，実際のインフルエンザウイルスの PDB データを用いた評価実験を行った．

キーワード：パターン照合，3D 点集合マッチング，分枝限定法，列挙アルゴリズム，最小二乗和平均平方根距離

Efficient Approximate 3-Dimensional Point Set Matching and Its Application to Molecular Pattern Matching

Sasaki Yoichi[1,a)]　Shibuya Tetsuo[2]　Ito Kimihito[3]　Arimura Hiroki[1]

Abstract: In this paper, we study a practically efficient algorithm for 3-D approximate point set matching based on enumeration. While this algorithm requires $O(kn^k)$ time in the worst case due to exhaustive enumeration, it is practically efficient due to branch-and-bound search and has advantage of generating all answers. To evaluate the proposed algorithm, we ran computer experiments on real influenza viruses data from PDB.

Keywords: 3D pointset matching, pattern matching, branch and bound, enumeration algorithm, RMSD.

1. はじめに

1.1 背景

高分子やタンパク質の三次元構造においての検索技術は，分子生物学において近年，重要な役割を担っている．例えば，あるタンパク質の構造が与えられ，その性質が知られていないときに，構造が似ているものは性質も似ているという理論から，既に構造と性質が知られているタンパク質との構造的に類似しているかを調べることにより，性質を予想することができる [1]．また，近年の急速なデータベースの増大に伴い，より早い時間での検索技術が必要とされている．

タンパク質はアミノ基で構成された鎖状構造である．従って，その構造は C_α と呼ばれる特定の原子の三次元座標の点列によって表される．そのような分子を鎖状分子と呼ぶ．また，DNA や RNA，グリカンのように生細胞においても，多くの重要な鎖状分子がある．

1.2 本研究の目的

本論文では，データを鎖状構造つまり点列の入力ではなく，点集合の入力とし，その中から与えられたパターンの

[1] 北海道大学大学院情報科学研究科
[2] 東京大学医科学研究所ヒトゲノム解析センター
[3] 北海道大学人獣共通感染症リサーチセンター
[a)] ysasaki@ist.hokudai.ac.jp

点列と構造が似ている部分点列をデータの点集合の中から見つけ出すという近似点集合マッチング問題を考えていく．この問題は，データとして n 個の点を要素として持つ点集合 T と，パターンとして k 個の点からなる点列 P，非負実数 $r > 0$ を入力とし，パターンに対する類似度が r 以下となるデータ中のサイズ k の部分点集合を抽出する問題である．

ナイーブなアルゴリズムは，データベースの点集合から可能な k 個の点を抽出し点列として，その後，パターンとの類似度を求め，閾値より小さければ解となる類似部分点集合として，出力する．この動作を繰り返すことにより，データベースの全ての部分点列に対して近似マッチングを見つけることができる．計算量は $k \cdot {}_nP_k = k \cdot n \cdot (n-1) \cdots (n-(k+1))$ より $O(n^k)$ 時間となり，データの内容に関わらず，常に $O(kn^k)$ 時間を要する．作業領域は $O(k)$ 領域である。

本稿では，この問題を解決するために，最小 RMSD スコアの上限関数を用いた探索の枝刈りによる効率化を提案する．素朴なアルゴリズムの問題点は，再帰的な計算により解となる候補集合を探索していたが，探索木の葉，すなわち，候補集合のサイズが k に達したときに初めて，最小 RMSD による解の検査を行うことである．

初めに，前綴りに関する最小 RMSD の上限関数（upper-bound function of minimum RMSD for prefices）を導入する．われわれは，この関数が正しく最小 RMSD スコアの上限関数となっていることを示す．

次にこれにより，上限関数による分枝限定法（branch-and-bound method） を用いた点集合マッチングアルゴリズムを提案する．これは，空な候補集合 ε から探索を開始し，再帰関数 FIND の各呼び出しにおいて，データベースからデータ点を一個選び追加するかどうか決めるのに，その時点での候補集合とパターン集合の前綴りの上限値と閾値 r の比較をその都度行い，明らかに成功しない探索の枝刈りを早めに行うものである．

最後に，提案アルゴリズムを C++言語により実装し，計算機実験を行うことにより有効性を検証した．結果から，すべての候補集合を訪問する素朴なアルゴリズムと比較して，提案手法は上限関数による枝刈りによって，広い範囲のパラメータで素朴なアルゴリズムより，１０倍から１００倍程度高速であることが観察された．

1.3 関連研究

本論文の関連研究として，Schwartz と Sharir [3] はデータベースとパターンを共に点列の入力とし，RMSD 指標によるマッチングを考察し，データベースサイズを N としたときに $O(N \log N)$ 時間で実行するアルゴリズムを提案している．また，Shibuya [2] は，同じマッチング問題を，データベースサイズに対して線形時間で実行するアルゴリズムを提案している．その他に，Akutsu [4] らは，ハッシュを用いた手法を提案しており，$O(N)$ 期待計算時間を実現している．また，Shibuya [1] は，幾何接尾辞木 (geometric suffix tree) と呼ばれる，三次元構造に対する索引データ構造を示している．これは，文字列に対する索引構造である接尾辞木 (suffix tree) [6], [7] を基に，タンパク質のような複雑な三次元構造に対して適応させたデータ構造である．このデータ構造は，RMSD 指標による効率よい近似部分構造マッチングを実現し，データベース中に存在するタンパク質構造同士の頻出する類似部分構造の発見にも用いることができる．

2. 準備

2.1 基本的定義

本稿では，三次元空間 $\mathbb{R}^3$ を考える．長さ n の**点列**（point sequence）S は，$P = (s_1, \ldots, s_n) \in (\mathbb{R}^3)^n$ である．ここに，各 $i = 1, \ldots, n$ に対して，$s_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ は i 番目の点の三次元座標を表すベクトルである．ここに，$|P| = n$ で P の**長さ**を表す．任意の $1 \le i \le j \le n$ に対して，$P[i \ldots j] = (s_i, s_{i+1}, \ldots, s_j)$ で P の添え字 i から j までの**部分点列**を表す．点列 P の長さ $1 \le i \le n$ の**前綴り**（prefix）とは，最初の i 個からなる列 $P[1..i]$ である．

三次元の回転行列 R によって点列 P を回転した構造は $R \cdot P = (Rs_1, \ldots, Rs_n)$ で与えられる．回転行列 R と移動ベクトル v は三次元空間の点に対する**変換** $f_{R,v}(P) = R \cdot P + v$ を与える．以後，文脈から明らかならば，組 (R, v) とそれが表す変換を同一視する．ベクトル v の**転置ベクトル**を v^t で表し，行列 A の**転置行列**を A^{t} で表す．$trace(A)$ は行列 A の**トレース**を与え，$|v|$ はベクトル v の**ノルム**を与える．

2.2 二つの点列間の最小平方平均二乗和距離

初めに，Shibuya [1] で三次元点列間の類似度として用いられている二つの点列間の最小平方平均二乗和距離を定義する．

三次元座標で表された長さ n の二つの点列を $S = (s_1, \ldots, s_n)$ と $T = (t_1, \ldots, t_n)$ とおく．任意の回転行列 R と移動ベクトル v が与えられたとき，S と T の間の R と v で決まる変換のもとでの**平方平均二乗和距離**を式

$$RMSD_{R,v}(S,T) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |s_i - (R \cdot t_i + v)|^2} \quad (1)$$

によって定義する．

このとき，二つの長さ n の点列 S と T 間の**最小平方平均二乗和距離（最小 RMSD スコア**ともいう）を，

$$MinRMSD(S,T) = \min_{R,v} RMSD_{R,v}(S,T) = \min_{R,v} \sqrt{\frac{1}{n}\sum_{i=1}^{n} |s_i - (R\cdot t_i + v)|^2} \quad (2)$$

と定義する．上式で，R と v は全ての回転行列と移動ベクトルの上を動く変数である．すなわち，最小 RMSD スコア $MinRMSD(S,T)$ は，全ての回転行列 R と移動ベクトル v に対する $RMSD_{R,v}(S,T)$ の値の最小値である．

2.3 点列間の最小 RMSD スコアの計算方法

ここで，Shibuya [1] に基づいて，二つの点列間の最小 RMSD スコアの線形時間アルゴリズムを与えておこう．点列 S と T に対して，$RMSD_{R,v}(S,T)$ を最小とする回転行列，移動ベクトルをそれぞれ $\hat{R}$, $\hat{v}$ とする．最小 RMSD スコアは，以下に示す方法で，$O(n)$ 時間で計算可能である．

回転行列 R が固定されていれば，$R\cdot T$ の重心を S の中心に移動するような v が $RMSD_{R,v}(S,T)$ を最小化することが知られている [2]．つまり $\hat{R}$ が与えられれば $\hat{v}$ は $O(n)$ 時間で計算可能である．

二つの点列 S,T の重心を原点に合わせるように移動した後，$RMSD_{R,v}(S,T)$ を最小化する回転行列 R を考える．これは，

$$RMSD'_R(S,T) = \sqrt{\sum_{i=1}^{n} |s_i - (R\cdot t_i)|^2} \quad (3)$$

を最小化する $\hat{R}$ を求めることと同じである．ここで，$J = \sum_{i=1}^{n} s_i \cdot t_i^t$ と置くと，これは明らかに $O(n)$ 時間で計算可能である．従って $RMSD'(S,T)$ は $\sum_{i=1}^{n}(s_i^t s_i + t_i^t t_i) - 2\cdot trace(R\cdot J)$ と表すことが出来る．また，$U\sum V$ を J の特異値分解 (SVD) とすると $R = VU^T$ のとき最大となる [5]．三次元座標においては J が 3×3 行列であり，3×3 に対する SVD は $O(3^3)$ 時間で計算可能であることから $\hat{R}$ は J から定数時間で求められる．

以上より，$\hat{v}$，$\hat{R}$ がそれぞれ $O(n)$ 時間で求められることから，$MinRMSD$ は定義式より $O(n)$ 時間で求められることは明らかである．

2.4 二つの点集合間の最小平方平均二乗和距離

次に最小 RMSD スコアを，要素間に順序のない集合同士の距離に拡張する．$P = P[1..n]$ と $Q = Q[1..n]$ を，要素数 n の三次元空間の点集合とする．ただし，これらの点集合は，点列として表されているものとし，要素である点の順番は任意に固定するとする．

順列 $\pi : \{1,\ldots,n\} \to \{1,\ldots,n\}$ に対して，π による長さ n の点列 $P = P[1..n]$ の並べ替えを $\pi(P) = (P[\pi(1)],\ldots,P[\pi(n)])$ と定める．

Algorithm 1 最小 RMSD スコアに基づく点集合マッチング列挙問題の素朴なアルゴリズム

```
1: procedure MAIN(P, T, r)
   入力： パターン P[1..k]，テキスト T[1..n]，正実数 r.
   出力： T 中の最小 RMSD スコア r に関する P の全ての異なる出現位置.
2:     FIND(Q, P, T, |Q|, |T|) を呼び出す;
3: end procedure
4:
5: procedure FIND(Q, P, T, k, n)
6:     if |Q| = k then                    ▷ すべての点を選び終わった
7:         if MinRMSD(P[1..k], Q[1..k]) ≤ r then
8:             マッチング位置の集合 Q を出力する;
9:         end if
10:    else
11:        for j = 1, ..., n do
12:            x = T[j];
13:            if x ∉ Q then
14:                R := Q の末尾に要素 x を加えて得られる点列;
15:                FIND(R, P, T, k, n) を再帰的に呼び出す;
16:            end if
17:        end for
18:    end if
19: end procedure
```

点列として表されたサイズ n の集合 S と T 間の**最小平方平均二乗和距離**（点集合間の**最小 RMSD スコア**と呼ぶ）を，すべての順列 π と，回転行列 R，移動ベクトル v に関する最小の RMSD 値

$$MinRMSD(P,Q) = \min_{\pi}\min_{R,v} RMSD_{R,v}(\pi(P),Q) \quad (4)$$

と定義する．すなわち，$MinRMSD(P,Q)$ は，全ての並べ替えに関して点列 P と Q の最小 RMSD スコアをとった最小値である．

2.5 幾何学的距離に関する近似点集合マッチング問題

本論文で考察する最小 RMSD スコアに関する点集合マッチング問題を次のように定義する．三次元空間を考える．

データ点集合とは，n 個の点からなる点集合 $T = \{t_1,\ldots,t_n\}$ である．**パターン点集合**とは，k 個の点からなる点集合 $P = \{p_1,\ldots,p_k\}$ である．T と P の要素を，それぞれ，データ点およびパターン点と呼ぶ．ただし，以下では常に $k \le n$ であり，データ点集合とパターン点集合は，点列として表されるものと仮定する．

非負実数 $r > 0$ に対して，パターン点集合 P がデータ点集合 T に**最小 RMSD スコア** r **でマッチする**とは，k 個のデータ点からなるある集合 $Q \subseteq T$ が存在して，

$$MinRMSD(P,Q) \le r$$

が成立することである．このとき，T の部分集合 Q を，T における P **の出現位置**という．

定義 2.1 最小 RMSD スコアに関する点集合マッチング探索問題は，データ点集合 T と，パターン点集合 P，非

負実数 $r > 0$ を受け取り，T 上の最小 RMSD スコア r でのパターン P の出現位置 $Q \subseteq T$ を一つ見つける問題である．

上記の問題の変種として，P が T に最小 RMSD スコア r でマッチするかどうかを YES または NO で答えるものを**点集合マッチングの決定問題**と呼ぶ．また，全ての出現位置 $Q \subseteq T$ をもれなく，かつ重複なしに見つけるものを**点集合マッチングの列挙問題**と呼ぶ．ただし，点列として異なる出現位置 Q は互いに異なるとみなす．

本稿では，とくに断らなければ，点集合マッチングの列挙版問題を考察する．決定版と探索版の問題は，列挙版のアルゴリズムで解くことができる．

2.6 最小 RMSD スコアに基づく点集合マッチングの素朴なアルゴリズム

最小 RMSD スコアに基づく点集合マッチング問題は，Algorithm 1 に示す素朴なアルゴリズムで解くことが出来る．このアルゴリズムは，次のステップで動く．

- 与えられたパターン点集合 P に対して，$k = P$ 個のデータ点 $Q[1], \ldots, Q[k]$ を可能なすべての順番でテキスト点集合 T から取り出す．
- 取り出されたデータ点からなる候補集合 $Q = \{Q[1], \ldots, Q[k]\}$ のそれぞれに対して，P と Q との最小 RMSD スコア値を求め，それが閾値 r より小さければ出現位置として出力する．

このアルゴリズムは，すべての最小 RMSD スコア r に関する P の全てのマッチングを見つけることができる．作業領域は $O(k)$ 領域である。その一方で，マッチングの可能性が少なくても，その計算時間はデータの内容に関わらず常に $O(kn^k)$ 時間を要するという問題をもつ．次節では，この問題の改善に取り組む．

3. 提案手法

3.1 基本アイデア

本節では，前の節で指摘した素朴な点集合マッチング手法の問題点を改善する．そのために，最小 RMSD スコアの上限関数を用いた探索の枝刈りによる効率化を提案する．

素朴なアルゴリズムは，再帰的な計算により解となる候補集合を探索していたが，探索木の葉，すなわち，候補集合のサイズが k に達したときに初めて，最小 RMSD スコアによる解の検査を行っていたため，効率が悪い．

そこで，前綴りに関する最小 RMSD スコアの上限関数（upper-bound function of minimum RMSD for prefices）を導入し，これを用いた分子探索法（branch-and-bound method） を取り入れる．ここで，前綴りに関する最小 RMSD スコアの上限関数とは，サイズが i の候補集合を選択した段階で，パターン P と Q の長さ i の前綴り（prefix）$P[1..i]$ と $Q[1..i]$ の上限関数値が閾値より大きければ，それ以後の探索で，残りの $k-i$ 個の点をどのように T から選んでも，P と Q の最終的な最終 RMSD は閾値 r 以下にならないようなものをいう．

このような上限関数が最小 RMSD スコアに対して構成できれば，空な候補集合 ε から探索を開始し，再帰関数 FIND の各呼び出しにおいて，データベースからデータ点を一個選び追加するかどうか決めるのに，その時点での候補集合とパターン集合の前綴りの上限値と閾値 r の比較をその都度行い，明らかに成功しない探索の枝刈りを早めに行うことが可能になる．そこで，探索が効率化する．

3.2 最小 RMSD スコアの上限関数の導出

手始めに，ウォーミングアップとして，次の最小二乗和距離の上限関数を導出しよう．以下では，k を任意の正整数とし，P と Q を長さ k の任意の点列とする．以下では，$MinSSD$ および MR を，それぞれ，MS と MR と略記することがある．

定義 3.1 長さ k の二つの点列 P と Q の**最小二乗和距離**（minimum sum of squared distance, Min SSD）は，次の量

$$\begin{aligned} &MinSSD(P,Q) \\ &= k \cdot (MinRMSD(P,Q))^2 \\ &= \min_{R,v} \sum_{i=1}^{k} |P[i] - (R \cdot Q[i] + v)|^2 \end{aligned}$$

である．

次の補題は，最小 SSD スコアは系列の長さに関して単調性を満たすことを主張する．

補題 3.1 任意の二つの点列 $P[1..k]$ と $Q[1..k]$ に対し，もし $1 \le i \le j \le k$ ならば，$MinSSD(P[1..i], Q[1..i]) \le MinSSD(P[1..j], Q[1..j])$ が成り立つ．

（証明）補題のためには．次の不等式を示せば十分である：$MinSSD(P[1..k], Q[1..k]) \ge MinSSD(P[1..k-1], Q[1..k-1])$．ここで，回転行列と移動ベクトル R, v に関する点 p と q の距離の二乗を $SD_{R,v}(p,q) = |p - (R \cdot q + v)|^2$ と表記すると，最小 SSD スコア $MS = MinSSD(P[1..k], Q[1..k])$ の定義より次が成り立つ．

$$\begin{aligned} MS &= MinSSD(P[1..k], Q[1..k]) \\ &= \min_{R,v} \sum_{i=1}^{k} SD_{R,v}(P[i], Q[i]) \\ &= \min_{R,v} \left\{ SD_{R,v}(P[k], Q[k]) + \sum_{i=1}^{k-1} SD_{R,v}(P[i], Q[i]) \right\} \\ &\ge \min_{R,v} \{SD_{R,v}(P[k], Q[k])\} \\ &\quad + \min_{R',v'} \left\{ \sum_{i=1}^{k-1} SD_{R',v'}(P[i], Q[i]) \right\} \\ &= SD_{R,v}(P[k], Q[k]) \\ &\quad + MinSSD(P[1..k-1], Q[1..k-1]) \end{aligned}$$

上の最後から2行目の導出は，最良スコア $MinSSD(P[1..k], Q[1..k])$ を与える変換を固定して得られる前綴りの組 $P[1..k-1]$ と $Q[1..k-1]$ のスコアは，この前綴りの組に対して直接求めた最良スコアより良くはならないことから示される．次に，$SD_{R,v}(P[k], Q[k]) \geq 0$ なので，次を得る．

$$MS \geq MinSSD(P[1..k-1], Q[1..k-1]).$$

得られた不等式を繰り返し適用すると，補題を得る．（証明終わり）

これを用いると，最小 RMSD スコアについても次の単調性が示せる．

補題 3.2 長さ $i \geq 1$ の任意の点列を P と Q とする．

$$MinRMSD(P[1..i], Q[1..i]) \geq \left(\frac{i-1}{i}\right)^{1/2} MinRMSD(P[1..i-1], Q[1..i-1])$$

が成り立つ．

（証明） 以下では，*MinSSD* および *MinRMSD* を，それぞれ，*MS* と *MR* と略記する．補題 3.1 より，

$$MS(P[1..i], Q[1..i]) \geq MS(P[1..i-1], Q[1..i-1]) \quad (5)$$

が成立する．次に，式 (5) において，*MinRMSD* の定義より，*MS* を *MR* で置き換えると，

$$i \cdot MR(P[1..i], Q[1..i])^2 \geq (i-1) \cdot MR(P[1..i-1], Q[1..i-1])^2 \quad (6)$$

となる．両辺の値は正なので，式 (6) の両辺の平方根をとって整理すると，補題の不等式が得られる．（証明終わり）

次の補題は，探索の途中で得られた長さ i の前綴りにおけるテストと，最終的なテストの関係を示す．

補題 3.3 (*MinRMSD* の疑似単調性) 任意の正整数 $1 \leq i \leq n$ に対して，不等式

$$MinRMSD(P[1..k], Q[1..k]) \geq \left(\frac{i}{k}\right)^{1/2} MinRMSD^{1/2}(P[1..i], Q[1..i])$$

が成立する．

（証明）簡便のため，以下では *MinRMSD* を *MR* と書く．このとき，補題の式の左辺からスタートして，$i = n, \ldots, 2$ に対して，補題 3.2 の不等式を適用していくと，次の不等式を得る．

$$\begin{aligned} & MR(P[1..k], Q[1..k]) \\ & \geq \left(\tfrac{k-1}{k}\right)^{1/2} MR(P[1..k-1], Q[1..k-1]) \\ & \geq \left(\tfrac{k-1}{k}\right)^{1/2} \left(\tfrac{k-2}{k-1}\right)^{1/2} MR(P[1..k-2], Q[1..k-2]) \\ & \cdots \\ & \geq \left(\tfrac{k-1}{k}\right)^{1/2} \left(\tfrac{k-2}{k-1}\right)^{1/2} \cdots \left(\tfrac{i}{i+1}\right)^{1/2} MR(P[1..i], Q[1..i]) \\ & = \left(\tfrac{i}{k}\right)^{1/2} MR(P[1..i], Q[1..i]) \end{aligned}$$

これより，補題が得られた．（証明終わり）

Algorithm 2 最小 RMSD スコアに基づく点集合マッチング列挙問題の改良版アルゴリズム

```
1: procedure Main(P, T, r)
   入力： パターン P[1..k]，テキスト T[1..k]，正実数 r.
   出力： T 中の最小 RMSD スコア r に関する P の全ての異なる出現位置.
2:     Find(ε, 0, P, T, |Q|, |T|) を呼び出す;
3: end procedure
4:
5: procedure Find(Q[1..i], i, P, T, k, n)
6:     if i = k then                          ▷ すべての点を選び終わった
7:         if MinRMSD(P[1..k], Q[1..k]) ≤ r then
8:             マッチング位置の集合 Q を出力する;
9:         end if
10:    else
11:        for j = 1, ..., n do
12:            x = T[j];
13:            if x ∈ Q then
14:                continue
15:            end if
16:            R := Q の末尾に要素 x を加えて得られる点列;
17:            if UB_MinRMSD_{i+1,k}(R, P[1..i+1]) > r then
18:                continue
19:            end if
20:            Find(R, i+1, P, T, k, n) を再帰的に呼び出す;
21:        end for
22:    end if
23: end procedure
```

補題 3.3 に基づいて，最小 RMSD スコアに関する上限関数を与える．

定義 3.2 (*MinRMSD* の上限関数) 任意の正整数 $1 \leq i \leq k$ に対して，上限関数を次のように定める：

$$UB\_MinRMSD_{i,k}(P[i], Q[i]) = \left(\tfrac{i}{k}\right)^{1/2} MinRMSD(P[i], Q[i]). \quad (7)$$

この上限関数を使った枝刈りでは，長さ i の前綴り $P[1..i]$ と $Q[1..i]$, に対して上限関数の値を計算し，$UB\_RMSD_{i,k}(P[1..i], Q[1..i], i, k) > r$ が成立するならば，これ以上の探索を行わない．

次の定理は，この上限関数を用いた枝刈りでは，正解を誤って枝狩りしてしまうことはないことを保証する．

定理 3.1 (*MinRMSD* の上限関数の健全性) 任意の正実数 $r > 0$ と正整数 $1 \leq i \leq k$ に対して，$UB\_RMSD_{i,k}(P[1..i], Q[1..i], i, k) > r$ ならば $MinRMSD(P, Q) > r$ が成立する．

ただし一般には，上記の定理の逆は成立しない．

3.3 アルゴリズム

Algorithm 2 に，提案の上限関数による分枝限定を用いた点集合マッチングアルゴリズムを示す．このアルゴリズムは，再帰に基づいて，すべての候補集合を列挙し，最小 RMSD スコアが閾値 r 以下の解を正しく出力する．これ

は，2 節で与えた素朴な点集合マッチングアルゴリズムと，ほぼ同じステップからなるが，17 から 19 行目で，上限関数を用いた枝刈りを行う．補題 3.3 より，この枝刈りは健全であり，いかなる正解を見落とすこともない．手続きの 13 行目のテストより，同じ並べ替えを重複して列挙することがないことが保証される．したがって，次が言える．

定理 3.2 点集合マッチングアルゴリズム Algorithm 2 は，与えられたデータ点集合 T とパターン集合 P に対して，最小 RMSD スコアが閾値 r 以下となる P のすべての出現位置を正しく見つける．

アルゴリズムの使用領域は $O(k)$ 領域である．計算時間に関しては，総計算時間が $O(kn^k)$ 時間であること以外はわからない．ただし，素朴なアルゴリズムが入力に無関係に常に $\Theta(kn^k)$ の時間を要するのに対して，提案アルゴリズムは早期終了の可能性をもっている点で実際には利点がある．

4. 計算機実験

本節では，3 節の提案手法の有効性を評価するために，計算機実験を行った．

4.1 データ

今回の実験で使用するデータベースには，H10N8 と呼ばれる A 型インフルエンザウイルスのデータを使用した．これは，RCSB Protein Data Bank[*1] からウイルスの三次元座標が記されたデータファイル中（3722 個の原子を含む）から 477 個あるアミノ酸の C_α 炭素だけを取り出して作成した．

4.2 方法

2 節の素朴な手法（naive）と 3 節の提案手法（pruned）を C++で実装し，CPU Intel Core i5 2.6 GHz，メモリ 8 GB，コンパイラ Apple LLVM version 6.0 (clang-600.0.54)，最適化オプション -O3 の計算機環境で行った．

実験では，入力サイズ T，パターンサイズ P，距離閾値 r を変化させて，各プログラムを 1 回ずつ走らせて，計算時間を計測した．各サイズのデータとパターンは，それぞれ，データベースファイルから先頭から指定した数の行を取り出して作成した．

4.3 結果

実験結果を示す．

4.3.1 実験 1：入力データサイズに対する計算時間

図 1 に，入力サイズ T を 10 からはじめて 10 刻みで 150 まで変化させて，計算時間を調べた結果を示す．これより，提案アルゴリズムは，素朴なアルゴリズムに対して 10 倍から 100 倍程度高速であった．

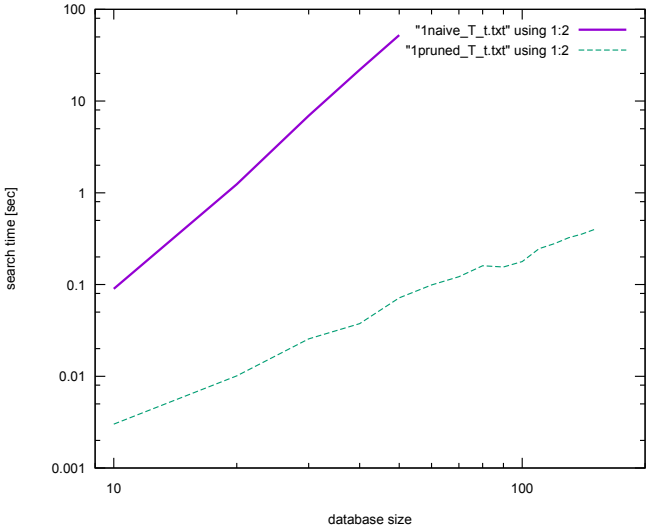

図 1 実験 1：入力データサイズに対する計算時間．パターン長 $P = 4$，閾値 $r = 0.1$ を用いた．

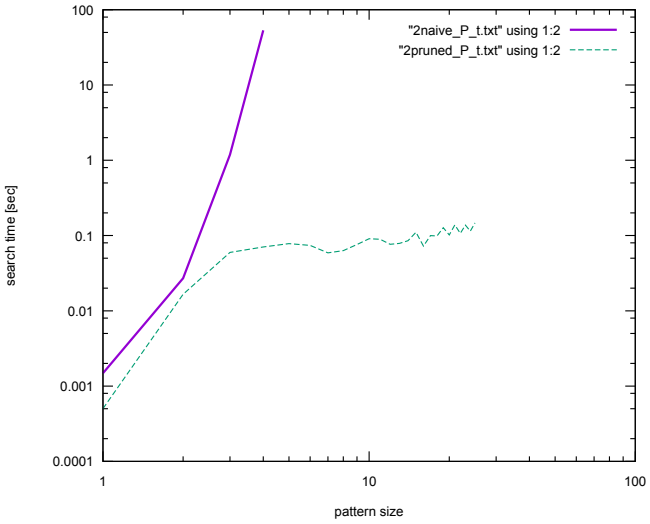

図 2 実験 2：パターンサイズに対する計算時間．データベース長 $T = 50$，閾値 $r = 0.1$ を用いた．

4.3.2 実験 2：パターンサイズに対する計算時間

図 2 に，パターンサイズ P を 1 からはじめて 1 刻みで 25 まで変化させて，計算時間を調べた結果を示す．$P = 1 \sim 2$ までは，両方のアルゴリズムは同じような計算時間だが，$P = 3$ より大きくなるにつれ素朴なアルゴリズムが相対的に大きな時間を要した．

4.3.3 実験 3：距離閾値に対する計算時間

図 3 に，距離閾値を $r = 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 4.0, 8.0$ のように変化させて，計算時間を調べた結果を示す．素朴なアルゴリズムは，R によらず常に同数の全ての候補集合を訪問することが構成からわかる．そのため，R によらず 1 秒程度の計算時間を要している．一方，提案アルゴリズムは，探索の途中でも閾値 R の上界を用いた枝刈りを行うため，特に小さな R ほど枝刈りの効果が発揮され，最大 50 から 60 倍程度の高速化が観察された．

[*1] http://www.rcsb.org/pdb/home/home.do

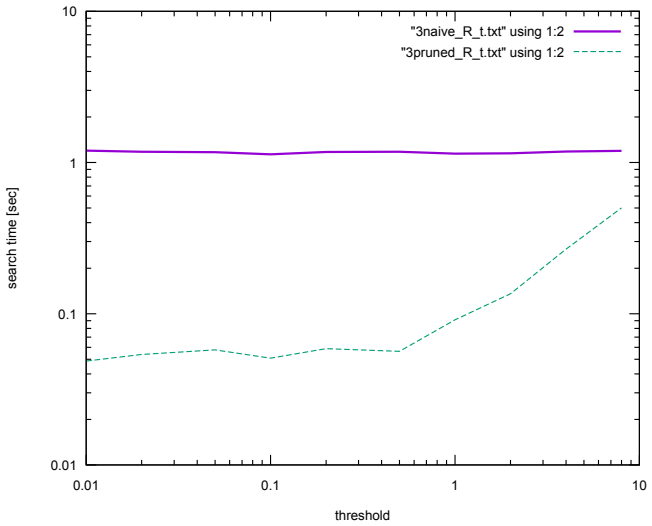

図 3 実験 3：距離閾値に対する計算時間．データベース長 $T = 50$，パターン長 $P = 3$ を用いた．

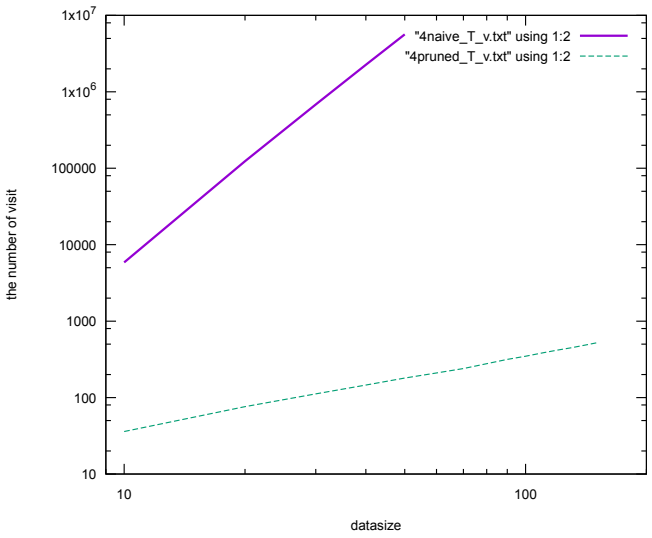

図 4 実験 4：枝刈りの効果．パターン長 $P = 4$，閾値 $r = 0.1$ を用いた．

4.3.4 実験 4：入力データサイズに対する訪問ノード数

今度は，図 4 に．枝刈りの効果を調べるために，入力サイズ T を 10 からはじめて 10 刻みで 150 まで変化させて，二つのアルゴリズムの探索における訪問ノード数を比較した結果を示す．こちらも，距離閾値 R を変化させた場合と同様に，広い T の範囲で枝刈りの効果が観察された．

5. 結論

本節では，最小 RMSD スコアに関する 3 次元空間での点集合マッチングを考察し，列挙に基づくアルゴリズムを与え，上限関数を用いた枝刈りによる高速化について議論した．実験結果からは，すべての候補集合を訪問する素朴なアルゴリズムと比較して，提案手法は上限関数による枝刈りによって，広い範囲のパラメータで素朴なアルゴリズムより高速であることが観察された．

今後の課題として，提案のアルゴリズムの計算時間について，適当なデータとパターンの生成モデルの仮定のもとでの，平均時間など，詳細な解析は以後の課題である．また応用として，本稿の手法に基づいた，分子データベースや，3D モデルデータベース，時空間系列データにおける近似検索への適用は興味深い課題である．

参考文献

[1] Shibuya Tetsuo, ”Geometric Suffix Tree : Indexing Protein 3-D Structures,” Jounal of the ACM, Vol.57, No.3, Article 15, 2010.
[2] Shibuya Tetsuo, ”Searching Protein 3-D Structures in Linear Time,” RECOMB 2009, LNCS 5541, 115, 2009.
[3] Jacob T. Schwartz, Micha Sharir, ”Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves,” Intl. J. of Robotics Res. 6, 2944, 1987.
[4] Akutsu Tatsuya, Onizuka Kentaro, Ishikawa Masato, ”New Hashing Techniques and Their Application to a Protein Strucure Database System,” Proc. Hawaii Int. Conf. System Sciences 5, 197-206, 1995.
[5] Gilbert Strang, ”Introduction to Linear Algebra, 3rd Edition,” Wellesley-Cambridge Press, 2003.
[6] Dan Gusfield, ”Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology,” Cambridge University Press, 1997.
[7] Esko Ukkonen, ”On-line construction of suffix-trees,” Algorithmica, Vol. 13, No. 3, 249-260, 1995.