Vol.2019-MPS-123 No.55
Vol.2019-BIO-58 No.55
2019/6/19

k最近傍法を用いた構造予測向け配列アラインメント生成手法の高速化

成井 政人[1,a)]　牧垣 秀一朗[1]　石田 貴士[1]

概要：ホモロジーモデリング法は立体構造既知のタンパク質を元に立体構造を予測する手法であり、配列アラインメントの精度が立体構造予測の精度に大きく影響する。Makigaki らは k 最近傍法を用いて配列アラインメントの精度を改善し、ホモロジーモデリング法の精度を向上させることに成功した。しかしながら、この手法は多くの k 最近傍法計算が必要なために計算コストが高い問題があった。
本研究では k 最近傍法を用いた配列アラインメント生成手法を高速化する方法を提案した。既存の近似 k 最近傍法高速化手法に対し、配列アラインメント生成用のデータセットを用いて速度と精度を評価した。結果として、最適なアルゴリズムとハイパーパラメータを特定し、精度の低下を抑えながら速度を 47 倍に高速化した。

キーワード：k 最近傍法, 配列アラインメント, 高速化

Acceleration of sequence alignment generation method for structure prediction using k nearest neighbor method

Abstract: Homology modeling is a method for predicting a three-dimensional structure of a protein based on three-dimensional structures of other proteins and their sequence alignments. In the method, the accuracy of the sequence alignment greatly affects the accuracy of the three-dimensional structure prediction. Recently, Makigaki et al. proposed a method to improve the accuracy of sequence alignment using the k-nearest neighbor method (k-NN) and succeeded to improve the accuracy of the homology modeling. However, the method had a problem of high computational cost because the method requires huge number of k-NN predictions. In this research, we proposed a method to accelerate the sequence alignment method using k-NN . We tested several fast but approximate k-NN acceleration methods and evaluated both computing time and accuracy for a dataset for the sequence alignment generation. Finally, we found optimal algorithm and the hyperparameters, and the speed of search was increased approximately 47 times while suppressing the decrease in accuracy.

Keywords: k Nearest neighbor, Sequence alignment, Acceleration

1. 序論

1.1 タンパク質立体構造予測

タンパク質は生化学の分野で非常に重要な生体高分子であり、その機能を解明することで創薬などに役立てることができる。タンパク質の機能はその立体構造が大きく関わっており、立体構造の決定は生命科学における重要な研究の一つになっている [1]。

タンパク質立体構造を実験的に求める主な手法として X 線結晶解析や核磁気共鳴法を用いた方法などがあるが、X 線結晶解析は 1 つのタンパク質の構造を決定するために結晶化するだけで数年を要する場合があるなどどちらも時間的コストが大きく、実験機材も高価で金銭的なコストも大きい。そのため、タンパク質の立体構造のデータベースである Protein Data Bank（PDB）[2] に登録されている立体構造のデータ数は 148,268 件（2019 年 1 月現在）[3] に留まっている。

[1] 東京工業大学 情報理工学院 情報工学系 知能情報コース
Graduate Major in Artificial Intelligence, Tokyo Institute of Technology
[a)] narui@cb.cs.titech.ac.jp

一方でタンパク質のアミノ酸配列は次世代シーケンサーの登場などにより高速な解析が可能となったため多くの情報が蓄積されるようになり、アミノ酸配列のデータベースである Universal Protein Resource（Uniprot）[4] には多くのアミノ酸配列の情報が記載されている。Uniprot のデータ数は 139,694,261 件（2019 年 1 月現在）[5] であり、PDB のデータ数に対して非常に大きなものになっている。

このような現状を踏まえ、コンピュータを用いてタンパク質のアミノ酸配列からタンパク質立体構造を予測する手法が研究されてきた。

1.2 ホモロジーモデリング法

コンピュータでの計算によりタンパク質の立体構造を予測する手法の一つとしてホモロジーモデリング法 [6] がある。アミノ酸配列が類似しているタンパク質は立体構造も類似しているという性質があり、ホモロジーモデリング法はこの性質を利用した手法である。ホモロジーモデリング法では、まず立体構造既知のタンパク質が登録されたデータベースに対し、クエリと配列類似性をもつタンパク質を検索する。次に検索で見つけたタンパク質の立体情報と配列アラインメントを用いてクエリの立体構造を予測する。この手法は相同性のあるタンパク質がデータベース中にある場合は高精度の予測が可能な手法である。

1.3 k 最近傍法を用いた配列アラインメント生成手法

Makigaki らはこの配列アラインメントを改良するため、機械学習に基づいた生成手法 [7][8] を提案した。配列アラインメントを作成する際の残基置換スコアの計算には通常 BLOSUM62 などの置換行列が用いられる。しかし、置換行列ではなく教師付き機械学習を用いてスコア付けを行うことでよりホモロジーモデリング法に適したアラインメントを生成することができる。

図 1 は彼らの手法の概略図である。この手法では、トレーニングデータセットのラベルを生成するために、構造の類似したタンパク質間の構造アラインメントを使用する。そのため、この手法により生成された配列アラインメントは構造アラインメントを行なった場合と似たアラインメントとなり、ホモロジーモデリング法に適したアラインメントが生成されることが期待される。特徴量としては 2 つのタンパク質の PSSM が用いられ、残基ペアの一致を予測するために標的残基とその周辺の PSSM が使用される。

1.3.1 残基置き換えスコア予測モデルの作成

(1) 正解となる構造アラインメントを生成するために構造的に類似したタンパク質の情報を SCOP データベース [9] から取得する。
(2) 取得したデータから TM-align[10] により構造アラインメントを生成する。
(3) PSSM を作成するために UniRef90 データベース [4] に対して 3 イテレーションの PSI-BLAST[11] を実施する。

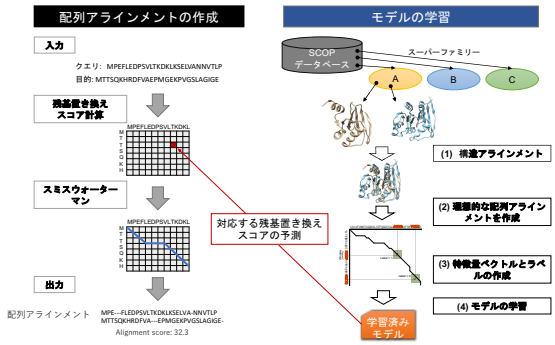

図 1 k 最近傍法を用いた配列アラインメント生成手法（[7] を改変）

1.3.2 データセットの定義

機械学習を行うためにこの情報を数値ベクトルにエンコードする。

(Q, T) をクエリ、ターゲットのベクトルとし、Q_i を Q の i 番目の要素とする。すると、特徴ベクトル $V_{x,y}$ はクエリとターゲットの特徴ベクトルの結合で表され、式 1 で定義される。

$$\mathbf{V}_{x,y} = (\mathbf{P}_x^{query}, \mathbf{P}_y^{target}) \tag{1}$$

アミノ酸の種類を 20, タンパク質の長さを N とする時、PSSM の長さは $20 \times N$ である。P は PSSM 行の連結であり式 2 で定義される。

$$\mathbf{P}_i = (p_{i-\frac{w}{2}}, \ldots, p_i, \ldots, p_{i+\frac{w}{2}}) \tag{2}$$

ここで、w はウインドウ幅であり、p_i は i 番目の PSSM 要素である。また、i は式 3 を満たす。

$$x - \frac{w}{2} \leq i \leq x + \frac{w}{2}, y - \frac{w}{2} \leq i \leq y + \frac{w}{2} \tag{3}$$

$i \leq 0$ で定義される “パディング” 領域については, $|Q| > i$ と $|T| > i$, p_i を 0 に割り当てる.

また、Q_x と T_y におけるラベル $L_{x,y}$ を式 4 のように定義する。

$$L_{x,y} = \begin{cases} 1, & \text{if } Q_x \text{ matches } T_y \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

図 2 はラベル定義の概略図である。

1.3.3 配列アラインメントの計算

作成されたスコアから Smith-Waterman 法 [12] を用いて配列アラインメントを生成する。Smith-Waterman 法ではダイナミックプログラミングにより残基置き換えスコアが最大となるようなアラインメントを生成することができる。

Vol.2019-MPS-123 No.55
Vol.2019-BIO-58 No.55
2019/6/19

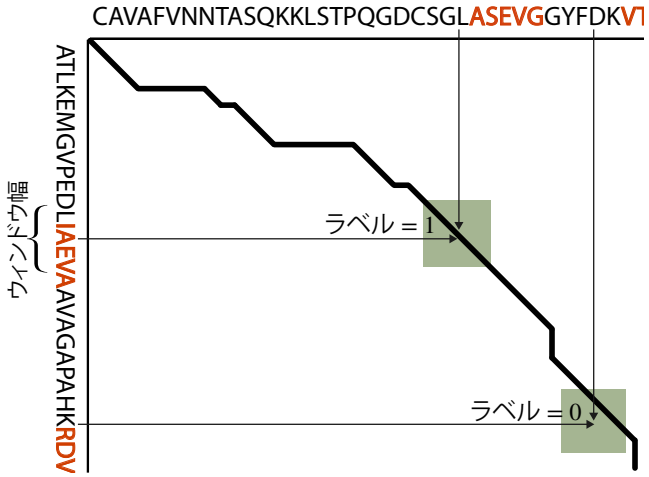

図 2　ラベルの定義（[7] を改変）

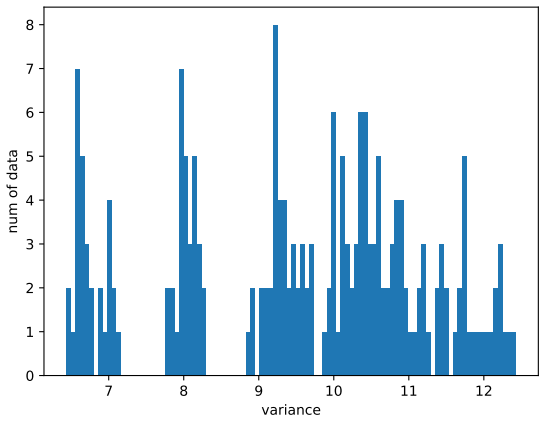

図 3　データセットの軸ごとの分散

1.4　Makigaki らの手法の問題点

BLOSUM62 を用いる場合に比べ、k 最近傍法を用いる場合は予測のたびに残基置換スコアを生成するため予測時の計算コストが大きい。結果としてアプリケーション全体として計算時間が長いという問題がある。Makigaki らの手法では長さが Q と T である 1 組のアミノ酸配列のアラインメントを生成するために QT 回の k 最近傍法の予測が必要である。線形探索の計算量は探索対象のベクトル長が N、ベクトル次元が D の場合は $\mathbf{O}(ND)$ であるため、全体の計算量は LR 回の k 最近傍法の計算量である $\mathbf{O}(NDQT)$ となる。このため大量の配列アラインメントを生成する場合や、生成されたアラインメントに基づきデータベースを検索する際の k 最近傍法の予測回数は膨大になり、計算量も大きくなる。

1.5　研究の目的

本研究では Makigaki らの手法の実行時間の大部分を占めるアミノ酸残基置換スコアの計算を高速化することで計算コストを抑えることを目的とする。k 最近傍法には既存の高速化手法があり、Makigaki らの手法に使われている k 最近傍法にも適用が可能である。既存手法を吟味し、適切な k 最近傍法高速化手法を適用することでアミノ酸残基置換スコア計算の高速化を行う。

2.　提案手法

2.1　概説

本研究では Makigaki らの立体構造予測手法の高速化を行う。その手法として実行時間の大部分を占めるアミノ酸残基置換スコアの計算を高速化した。アミノ酸残基置換スコアの計算に用いられる k 最近傍法の高速化には既存の高速化手法を複数検討し、立体構造予測手法の性質やデータセットに合ったものを適用することで実現した。

2.2　アプリケーションで用いられる kNN の特性

Makigaki らの手法では長さが M と N である 1 組のアミノ酸配列のアラインメントを生成するために MN 回の k 最近傍法による予測が必要である。このため、データベースに対し大量のクエリを計算する場合では、k 最近傍法の予測回数は膨大なものになる。一方で、学習はデータベースに対して 1 度行えばよく、k 最近傍法の予測回数に対し学習回数がかなり少なくなる。そのため、学習にかかる時間は重視せず予測にかかる時間が短いアルゴリズムを特定することを目的にする。

2.3　k 最近傍法高速化手法の検討

k 最近傍法高速化手法として、ライブラリが提供されており有力な手法である kd-tree[13]、randomized kd-tree[14]、hierarchical kMeans-tree[15] を検討する。

2.3.1　kd-tree

今回のデータセットは 200 次元とそこまで高次元でないことや、木構造を用いるアルゴリズムで最も基本的なものであることを踏まえ、このアルゴリズムを採用する。

2.3.2　Randomized kd-tree

通常の kd-tree では分割する軸に分散が最大の軸を選ぶことが多い。しかし、多くのデータセットでは分散が最大の軸と 2 番目以降の軸で分散に大きな差がないことが多く、最大の軸を選び続けることが最善の戦略とは限らない。この考えを元に、軸の選び方を変えた複数の木構造を用いるようにしたのが randomized kd-tree である。

図 3 は Makigaki らの手法で使われるデータセットの軸ごとの分散を示した図である。図の作成に用いたデータセットは計算時間の都合から 10,000 分の 1 にランダムサンンプリングしたものを用いている。

図 3 より今回のデータセットも分散が最大の軸と 2 番目以降の軸で分散に大きな差がないことから randomized kd-tree が有効と考え、この手法を採用した。

データ数	2,355,695
次元	200
正例の割合	7.479%

表 1 データセットの詳細

2.3.3 Hierarchical kMeans-tree

検討は、ホモロジーモデリングの精度を低下させないために、近似近傍探索による精度低下が AUC 0.05 程度に収まる範囲で行う。Muja らの研究 [15] より近似による精度低下が AUC 0.05 以内では、randomized kd-tree より hierarchical kMeans-tree が良い性能を示す可能性があるので hierarchical kMeans-tree を実験対象に含めた。

2.4 データセット

Makigaki らの手法で使われているアミノ酸残基置換スコアを予測するデータセットを使用する。これは Makigaki らの手法では 10 分の 1 にランダムサンプリングしたものを使用しているが、計算時間の関係から検討には 100 分の 1 にランダムサンプリングしたサブセットを用いる。データセットはスパースではなく、各値は整数値である。また、ラベルは構造アラインメントにより得られた配列アラインメントで中心残基が一致するなら 1、そうでないなら 0 になるように付与されている。

100 分の 1 にランダムサンプリングしたデータセットの詳細を表 1 に示す。

2.5 評価

データセットをランダムに 10-fold cross validation を行った上で検討を行い、その平均値を結果として使用する。

また、今回 k 最近傍法の二値分類の結果は利用せず、近傍 k 個の内ラベルが 1 であるものの割合をスコアとして利用している。本来は Makigaki らの手法全体の速度で計測をすべきだが、k 最近傍法に比べ DP の計算は高速で、k 最近傍法にかかる時間が 99 %以上を占めるため、本研究では k 最近傍法の予測にかかった時間のみを C++標準ライブラリ chrono を用いて計測した。

精度評価は後述する AUC を使用した。これはデータセット中に含まれるデータの正例と負例の割合が非常に偏っているためである。

3. 実験

3.1 実験環境

実験環境を表 2 に示す。また、実験に用いたソフトウェアのバージョンを表 3 に示す。

プログラムは計算機のノードを占有して実行した。また、データはメモリに展開され、予測中はディスクアクセスが行われない状態で実験を実施した。

計算機	TSUBAME3.0 f node
CPU	Intel Xeon E5-2680 V4 Processor（Broadwell-EP, 14 コア, 2.4GHz）× 2
メモリ	256GB
OS	SUSE Linux Enterprise Server 12 SP2

表 2 計算機環境

ソフトウェア	バージョン
gcc	5.5.0
python	2.7.15
scikit-learn	0.20.1

表 3 ソフトウェアのバージョン

3.2 実装

各アルゴリズムの実装は FLANN 1.9.1 を使用した。

FLANN[15] は近似 k 最近傍法のライブラリである。k 最近傍法のライブラリは多数あるが、近似近傍探索が可能であることや、実装されているアルゴリズムの多彩さ、拡張しやすさから FLANN を採用した。

今回の実験では FLANN のソースコードから gcc コンパイラでビルドしたものを使用した。ビルドにはライブラリの提供する cmake ファイルを用いた。コンパイルオプションも cmake ファイルに準じるが、C++コンパイラのオプションとして`-std=c++11` を追加している。

また、Makigaki らの手法で用いられる k 最近傍法の k の値は 1000 であるので、以下の実験では $k = 1000$ であるものとする。

3.3 ハイパーパラメータ探索

k 最近傍法ライブラリ FLANN の Python 実装を使用して scikit-learn 準拠の識別器モデルを作成した。この識別器モデルを用いて scikit-learn の GridSearchCV にてハイパーパラメータ探索を行った。データセットは Makigaki らの手法で使われているアミノ酸残基置換スコアを予測するデータセットを 1000 分の 1 にランダムサンプリングしたサブセットを用いる。

1000 分の 1 データセットに対して線形探索を行った結果は探索時間が 1,826 秒で AUC が 0.701 であった。この結果から、今回は AUC 0.05 程度の性能低下を許容し、AUC 0.695 を達成するまでにかかった探索時間が最も短いパラメータを最適なパラメータとする。

3.3.1 kd-tree

kd-tree では以下のパラメータを GridSearchCV を用いて調節した。

- 探索ノード数の上限（checks）[512, 1024, 2048, 4096, 8192, 16384, 32768]

結果は図 4 のようになった。

実験結果から、checks = 32768 を最適なパラメータと判断した。

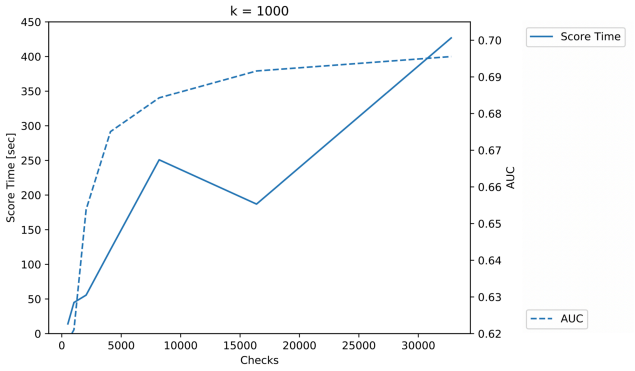

図 4　kd-tree におけるハイパーパラメータ探索の結果

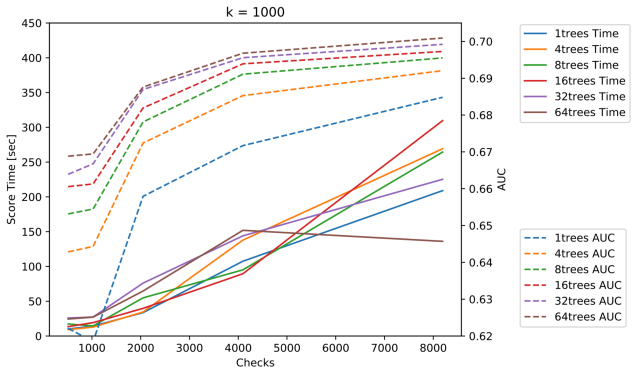

図 5　Randomized kd-tree におけるハイパーパラメータ探索の結果

3.3.2　Randomized kd-tree

Randomized kd-tree では以下のパラメータを GridSearchCV を用いて調節した。

- 探索ノード数の上限（checks）[512, 1024, 2048, 4096, 8192]
- 構築する kd-tree の数（trees）[1, 4, 8, 16, 32, 64]

結果は図 5 のようになった。

実験結果から、checks = 4096、trees = 64 を最適なパラメータと判断した。

3.3.3　Hierarchical kMeans-tree

Hierarchical kMeans-tree では以下のパラメータを GridSearchCV を用いて調節した。今回は構築時間を重視しないので kMeans クラスタリングの初期化には k-means++法 [16] を用い、イテレーション回数は Muja らの研究 [15] を元に余裕を持たせ 50 とした。

- 探索ノード数の上限（checks）[512, 1024, 2048, 4096, 8192]
- 分割を中止する閾値と kMeans クラスタリングのクラスタ数（branching）[16, 32, 64, 128, 256, 512, 1024]

結果は図 6 のようになった。

実験結果から、checks = 8192、branching = 1024 を最適なパラメータと判断した。

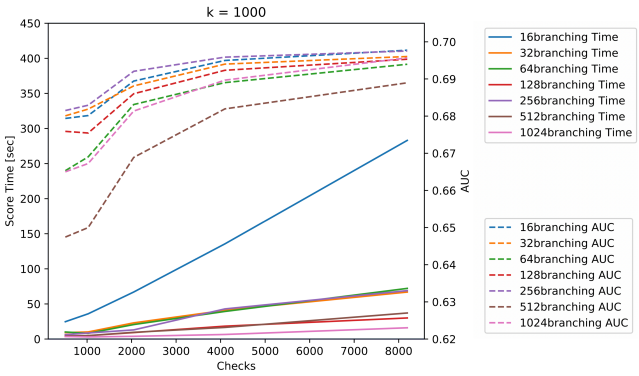

図 6　Hierarchical kMeans-tree におけるハイパーパラメータ探索の結果

アルゴリズム	探索時間 [sec]	AUC
Brute Force	47,226	0.721
kd-tree	$> 86,400$	-
Randomized kd-tree	998	0.716
Hierarchical kMeans-tree	2,510	0.715

表 4　100 分の 1 データセットの探索結果

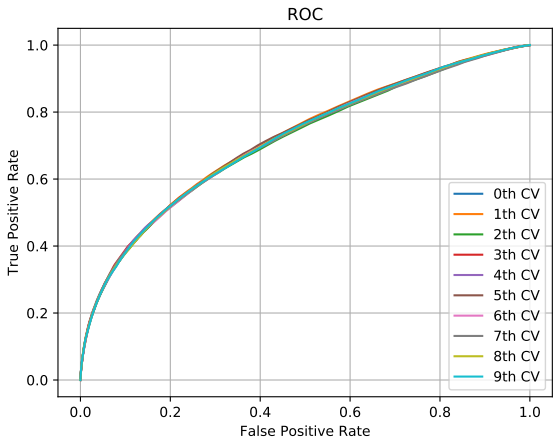

図 7　Randomized kd-tree の ROC

3.4　k 最近傍法の高速化手法の検討

最適化されたハイパーパラメータを用いて、既存の k 最近傍法の高速化手法に対し予測速度と精度を評価する。

100 分の 1 データセットに対して実験を行った結果は表 4 のようになる。kd-tree に関しては計算機の実行時間制限である 24 時間以内に計算が終了しなかったため、AUC は計測できなかった。

最も良好な結果を示した randomized kd-tree の 10-fold cross validation の各分割の ROC 曲線を重ねると図 7 のようになり、ほぼ一致することがわかった。

3.5　結果の考察

探索するノードの上限が増加するにしたがって探索時間が増加することが予測されるが、いくつかのケースで上限が増加した時に探索時間が減少した。これは探索するノー

アルゴリズム	探索時間 [sec]	AUC
Brute Force	1826	0.701
kd-tree	1341	0.701
Randomized kd-tree	82.31	0.698
Hierarchical kMeans-tree	60.28	0.696

表 5 1000 分の 1 データセットの探索結果

ドが増えたことで探索の早い段階で良い解が見つかりその後の探索で探索範囲の枝刈りがうまくいった可能性が考えられる。また、探索するノードの上限が増加するにしたがって精度が向上することが予測されるが、これに関しても探索するノードが増えた時に精度が減少する場合がある。これに関しては k 最近傍法の探索精度と AUC との相関が想定より小さいためと考えられる。

100 分の 1 データセットに対して行った実験結果である表 4 より、既存の k 最近傍法の高速化手法のうち、randomized kd-tree が 100 分の 1 データセットに対しては最も適していたと言える。

一方で、1000 分の 1 データセットに対して行ったパラメータ探索に注目すると、hierarchical kMeans-tree が顕著に良い性能を示していることがわかる。実験と同様にして 10000 分の 1 データセットでハイパーパラメータ探索をし、最良パラメータで 1000 分の 1 データセットに対して探索を行った結果を表 5 に示す。なお、hierarchical kMeans-tree についてパラメータ探索により得られた最適な訪問ノード数は 2048 であったが、AUC が 0.691 と低い値になったのでノード数は 4096 で計測した結果を用いた。この結果より 1000 分の 1 データセットでは hierarchical kMeans-tree が最適なアルゴリズムであると言える。

以上から、データセット数によって最適なアルゴリズムが異なることがわかる。Makigaki らの手法で用いられるデータセットは 10 分の 1 にランダムサンプリングしたデータセットであるため、より近い 100 分の 1 データセットの結果を採用し、randomized kd-tree が最も適切なアルゴリズムとした。

4. 結論

4.1 本研究の成果

本研究では既存の k 最近傍法の高速化手法を検討し、最良の結果であった randomized kd-tree を用いることで Makigaki らの立体構造予測手法に用いられる k 最近傍法部分を高速化し、立体構造予測手法全体での計算時間を短縮した。

今後の課題としては、randomized kd-tree を今回使用したデータセットに対して最適化させるなどが考えられる。最適化の手法としては、PSSM を連結したデータセットに適切な距離関数を定義するなどが考えられるだろう。また、k 最近傍法の精度が Makigaki らの手法の精度に与える影響を調べ、適切な近似度合いを評価、適用し、手法全体として高速化することなどが挙げられる。

参考文献

[1] Yang, Y., Gao, J., Wang, J., Heffernan, R., Hanson, J., Paliwal, K. and Zhou, Y.: Sixty-five years of the long march in protein secondary structure prediction: The final stretch?, *Briefings in Bioinformatics*, Vol. 19, No. 3, pp. 482–494 (online), DOI: 10.1093/bib/bbw129 (2018).

[2] Gilliland, G., Berman, H. M., Weissig, H., Shindyalov, I. N., Westbrook, J., Bourne, P. E., Bhat, T. N. and Feng, Z.: The Protein Data Bank, *Nucleic Acids Research*, Vol. 28, No. 1, pp. 235–242 (online), DOI: 10.1093/nar/28.1.235 (2000).

[3] : PDB Data Distribution by Structural Genomics Centers, `https://www.rcsb.org/stats/distribution_structural-genomics-centers`.

[4] UniProt Consortium, T.: UniProt: the universal protein knowledgebase, *Nucleic Acids Research*, Vol. 46, No. 5, pp. 2699–2699 (online), DOI: 10.1093/nar/gky092 (2018).

[5] : UniProtKB/TrEMBL UniProt release 2019_01, `https://www.uniprot.org/statistics/TrEMBL`.

[6] Rodriguez, R., Chinea, G., Lopez, N., Pons, T. and Vriend, G.: Homology modeling, model and software evaluation: Three related resources, *Bioinformatics*, Vol. 14, No. 6, pp. 523–528 (online), DOI: 10.1093/bioinformatics/14.6.523 (1998).

[7] Makigaki and Ishida: Sequence alignment using machine learning for accurate template-based protein structure prediction. Bioinformatics, to be appeared.

[8] : 第 57 回 BIO 研究発表会, `https://www.ipsj.or.jp/kenkyukai/event/bio57.html`.

[9] Murzin, A. G., Brenner, S. E., Hubbard, T. and Chothia, C.: SCOP: A structural classification of proteins database for the investigation of sequences and structures, *Journal of Molecular Biology*, (online), DOI: 10.1016/S0022-2836(05)80134-2 (1995).

[10] Zhang, Y. and Skolnick, J.: TM-align: A protein structure alignment algorithm based on the TM-score, *Nucleic Acids Research*, Vol. 33, No. 7, pp. 2302–2309 (online), DOI: 10.1093/nar/gki524 (2005).

[11] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J.: Gapped BLAST and PSI-BLAST:a new generation of protein database search programs, *Nucleic Acids Research*, Vol. 25, No. 17, pp. 3389–3402 (online), DOI: 10.1093/nar/25.17.3389 (1997).

[12] Smith, T. and Waterman, M.: Identification of common molecular subsequences, *Journal of Molecular Biology*, Vol. 147, No. 1, pp. 195 – 197 (online), DOI: https://doi.org/10.1016/0022-2836(81)90087-5 (1981).

[13] Bentley, J. L.: Multidimensional binary search trees used for associative searching, *Communications of the ACM*, Vol. 18, No. 9, pp. 509–517 (online), DOI: 10.1145/361002.361007 (1975).

[14] Silpa-Anan, C. and Hartley, R.: Optimised KD-trees for fast image descriptor matching, *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 1–8 (online), DOI: 10.1109/CVPR.2008.4587638 (2008).

[15] Muja, M. and Lowe, D.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration., *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and*

tions, Vol. 1, pp. 331–340 (2009).
[16] Arthur, D. and Vassilvitskii, S.: k-means++: the Advantages of Careful Seeding, *Proc ACM-SIAM symposium on discrete algorithms.*, Vol. 8, pp. 1027–35 (online), DOI: 10.1145/1283383.1283494 (2007).