電気学会論文誌 D（産業応用部門誌）
IEEJ Transactions on Industry Applications
Vol.136 No.10 pp.686–691　DOI: 10.1541/ieejias.136.686

論　文

NWアルゴリズム拡張による配列アラインメントの高速化についての検討

非会員　尾風　仁*　正　員　山田　親稔*
非会員　宮城　桂*　非会員　市川　周一**

Accelerating Techniques for Sequence Alignment based on an Extended NW Algorithm

Jin Okaze*, Non-member, Chikatoshi Yamada*, Member, Kei Miyagi*, Non-member, Shuichi Ichikawa**, Non-member

（2016年2月15日受付，2016年6月30日再受付）

The NW (Needleman-Wunsch) algorithm is a method of sequence alignment in bioinformatics. The NW algorithm can be applied for global sequence alignment, which is a way of arranging the sequences of DNA to identify regions of similarity. However, the NW algorithm requires a huge number of calculations compared with the SW (Smith-Waterman) algorithm. Many studies have focused on analyzing the output of multiple sequences quickly in three dimensions. However, such methods cannot obtain similarities for whole sequences. In this article, we extend the NW algorithm to three dimensions. The proposed method is expected to provide a fast analysis of high precision data sequences.

キーワード：配列アラインメント，Needleman-Wunsch アルゴリズム
Keywords: sequence alignment，Needleman-Wunsch algorithm

1. はじめに

配列アラインメントは生物学や遺伝子工学の分野で広く用いられており，遺伝子機能や分子進化系統の推定などに活用されている。ある2つの対象配列に対して配列アラインメントを行なった様子をFig. 1に示す。しかしながら，これらの処理は膨大な計算量が必要となり演算に多大な時間がかかる。さらに配列データベースのサイズは年々増大しており，高速化が求められている。FASTAやBLAST（Basic Local Alignment Search Tool）[(1)]などの手法を用いると高速な演算が可能となるが，部分的にヒューリスティックなプロセスを踏むため，精度の高い検索が必要となる場面では正しい結果を返せない可能性がある。また，一般的に画像処理用の演算装置であるGPU（Graphic Processing Unit）を画像処理以外の汎用計算に用いて高速化を図ることをGPGPU

```
AAB24882    TYHMCQFHCRYBNNHSGEKL···GEKTHEHNQCGKAFPT
AAB24881    --------YECNQCGKAFAQHSSLK···GEKPYECNQCGKAFSK
************************************************************
AAB24882    PSHLQYHERTHTGEKPYEC···KPYE----CNQCGKAFAQ--
AAB24881    HSHLQCHKRTHTGEKPYEC···KPYMNVINMVKPLHNS
```

Fig. 1.　A sequence alignment.

（General-purpose computing on GPU）と呼び，GPGPUを配列アラインメントに適用する研究[(2)–(4)]も多く行われている。

また，その応用として，多系列データ解析の高速化を目的として，配列アラインメントの一種であるSW（Smith-Waterman）アルゴリズムの系列数を拡張してGPU上に実装し高速化が提案されている研究[(5)]がある。しかしながら，SWアルゴリズムはローカルアラインメントのため局所的な相関度は求められるが，データ全体の相関度を求めることができない。

本稿では，NW（Needleman-Wunsch）アルゴリズムのGPU実装に向けた3次元への拡張を提案する。これにより，高い精度での多系列データ解析の高速化が期待される。

2. NW（Needleman-Wunsch）アルゴリズム

〈2·1〉 配列アラインメント　配列アラインメントとは，生物学の分野において2本以上の異なるDNA配列やアミノ酸配列を入力として，それらの類似性を比較するための手法の1つである。配列アラインメントはグローバルア

* 沖縄工業高等専門学校
〒905-2192　沖縄県名護市辺野古905
National Institute of Technology, Okinawa College
905, Henoko, Nago, Okinawa 905-2192, Japan
** 豊橋技術科学大学
〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘1-1
Toyohashi University of Technology
1-1, Hibarigaoka, Tempaku-cho, Toyohashi, Aichi 441-8580, Japan

ラインメントとローカルアラインメントに分類される。グローバルアラインメントとは，長さの近い配列を比較する際に用いるもので，配列全体を並び替えるため対象の全体的な類似度を知りたい場合に有効である。ローカルアラインメントは局所的に比較を行うため，配列の部分的な類似度を知りたい場合に用いる。ローカルアラインメントはグローバルアラインメントと比べると計算量が少なく済むため，計算にかかる時間も短くなる場合が多い。また，配列アラインメントはペアワイズアラインメントと多重配列アラインメントに分類される。ペアワイズアラインメントは 2 本の配列を比較し類似度を求めるもので，多重配列アラインメントでは主に 3 本以上の配列を扱う。

〈2・2〉 NW アルゴリズムの概要　NW アルゴリズムは 2 本の配列の共通部分を整列させることにより，それらの相関度を求めるペアワイズアラインメントのアルゴリズムである。グローバルアラインメントの代表的な手法の 1 つであり，2 本の配列間において最高のスコアになるよう整列させる。Fig. 2 に示すようなスコアテーブルを用意し，対象となる 1 つ目の文字列を第 1 列目に，2 つ目の文字列を第 1 行目に代入する。テーブルの左上の要素からスコアを格納していき，一番右下の要素になるまで計算を続ける。その後，右下の要素から左上の要素まで通った経路を逆向きに戻り（トレースバック），その過程で文字列を抽出することにより配列の整列を行う。

スコア計算では SP（Sum of Pair）スコア体系を用いる。マッチ，ミスマッチ，ギャップという 3 つのパラメータを用意し，2 本の配列の文字が一致した場合にはマッチ，不一致の場合はミスマッチ，片方の配列に対してもう 1 つの配列における文字が存在しない場合はギャップと定義する。配列検索の目的に応じてこれらのパラメータ値を調整することが必要であり，ここではマッチは 2 ポイント，ミスマッチは −1 ポイント，そしてギャップは −2 ポイントとする。スコア関数の定義を文字式で表したものを (1) 式に示す。(1) 式において，A と B は任意の文字で $A \neq B$ であり，それぞれ異なる配列における文字列の一部で塩基とする。また，ギャップは記号 “–” を用いて表現している。

$$\begin{cases} s(A,A) = match = 2 \\ s(A,B) = mismatch = -1 \\ s(A,-) = s(-,A) = gap = -2 \end{cases} \quad \cdots\cdots\cdots\cdots\cdots (1)$$

		j =	0	1	2	...	m
	i =	X \ Y	0	y_1	y_2	⋯	y_m
0		0					
1		x_1					
2		x_2					
⋮		⋮					
n		x_n					

Fig. 2.　Score table.

長さが n と m の任意の配列をそれぞれ $X = x_1, x_2, \ldots, x_i$ と $Y = y_1, y_2, \ldots, y_j$ としたとき，NW アルゴリズムの初期条件を (2) 式に示す。文字列同士の比較によるスコア計算は SP スコア体系に基づき，要素を 1 つずつ計算していく。この場合の計算式は，(3) 式により求められる。また，NW アルゴリズムの計算量は $O(mn)$ となる。

$$\begin{cases} NW(0,0) = 0 \\ NW(i,0) = gap \cdot i \\ NW(0,j) = gap \cdot j \end{cases} \quad \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

$$NW(i,j) = max \begin{Bmatrix} NW(i-1,j-1) + s(x_i, y_j) \\ NW(i-1,j) + gap \\ NW(i,j-1) + gap \end{Bmatrix} \cdots\cdots\cdots (3)$$

〈2・3〉 NW アルゴリズムの例　例えば，X = “$ABCD$” と Y = “ACD” という 2 つの配列があるとする。これらの配列を NW アルゴリズムを用いて最大スコアになるように整列させる。(3) 式を用いて $(i, j) = (0, 0)$ から $(i, j) = (4, 3)$ まで 1 格子ずつスコア計算していくと，スコアテーブルは Fig. 3 のようになる。このスコアテーブルをもとにしてトレースバックを行い，対象配列を最大スコアになるように整列させる。トレースバックでは，スコア計算の際に採択したルートを逆向きに選択していくため，この場合，トレースバックは Fig. 4 に示すように $(i, j) = (4, 3)$ から始まり，$(i, j) = (3, 2)$，$(i, j) = (2, 1)$，$(i, j) = (1, 1)$，$(i, j) = (0, 0)$ というように選択していく。そして，選択した格子に対応する文字を抽出していくのだが，$(i, j) = (2, 1)$ から $(i, j) = (1, 1)$ への経路において，配列 X は 1 つ前の要素に移動しているが，配列 Y の要素に関しては移動していない。これは配列 Y にギャップが生じたということであり，この箇所にはギャップの記号 “–” が格納される。このようにして，配列 X = “$ABCD$” と Y = “ACD” を最大スコアになるよう整列さ

	j =	0	1	2	3
i =	X \ Y	0	A	C	D
0	0	0	-2	-4	-6
1	A	-2	2	0	-2
2	B	-4	0	1	-1
3	C	-6	-2	2	0
4	D	-8	-4	0	4

Fig. 3.　Results of score calculation.

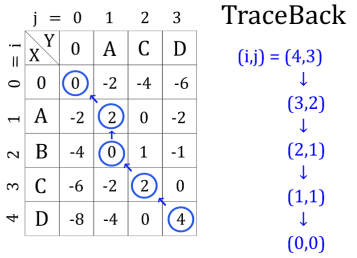

Fig. 4.　Traceback.

せると，X' = “$ABCD$” と Y' = “$A-CD$” および $Score = 4$ を得る。

3. 関連研究

通常，複数の配列を整列させるには，多重配列アラインメントを用いる。多重配列アラインメントのプログラムには，現実に則した結果を得られる ClustalW [6] や，その他にも MAFFT [7] や T-COFFEE [8] などがある。しかしながら，これらのプログラムは累進法を用いており，精度が初めの配列の類似度に依存してしまうため，エラーが多くなる可能性が高い。

GPU を用いて配列アラインメントを高速化している研究も多く行われており，代表的なものでは CUDASW++2.0 [9] などがある。この研究では，配列 DB を対象にして，複数のスコア行列の対角線を同時に処理することで，配列アラインメントの計算過程におけるデータの依存関係を無くし，並列計算することを可能にしている。

4. NW アルゴリズムの 3 次元への拡張

〈4･1〉 アルゴリズム拡張の概要 2 章で述べた NW アルゴリズムを 3 次元に拡張する。つまり，3 本の配列を最高スコアになるように整列させる。スコア計算には Fig. 5 に示すような 3 次元のスコアテーブルを用いる。また，2 次元の NW アルゴリズムと同様に，全ての要素を計算しスコア算出後にトレースバックを行うという手順をとる。3 次元 NW アルゴリズムにおける SP スコア計算は，2 章で述べたアルゴリズムと異なる。任意の要素 $NW(i,j,k)$ に対するギャップ及びマッチとミスマッチの位置関係を Fig. 6 と Fig. 7 に示す。

3 次元に拡張した場合，2 つの配列の文字が一致してもマッチにはならず，3 つの配列の文字が一致した場合にマッチ，不一致の場合にミスマッチ，1 つ以上の配列において文字が存在しない場合にギャップとなる。

3 次元におけるスコア関数の定義を (4) 式に示す。なお，(4) 式において，A，B，C は任意の文字で $A \neq B \neq C$ であり，それぞれ異なる配列における文字列の一部で塩基とする。スコアのポイントやギャップの記号 “–” については先述のアルゴリズムと同様とする。

$$\begin{cases} s(A,A,A) = match = 2 \\ s(A,A,B) = s(A,B,C) = mismatch = -1 \\ s(A,A,-) = s(A,-,-) = gap = -2 \end{cases} \cdots\cdots (4)$$

$$\begin{cases} NW(0,0,0) = 0 \\ NW(i,j,0) = gap \cdot (i+j) \\ NW(0,j,k) = gap \cdot (j+k) \\ NW(i,0,k) = gap \cdot (k+i) \end{cases} \cdots\cdots (5)$$

$$NW(i,j,k)$$

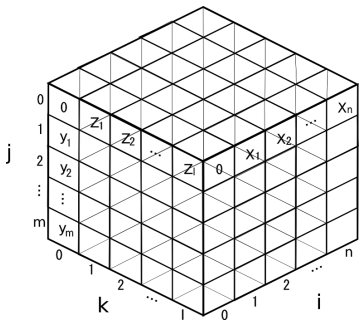

Fig. 5. 3D score table.

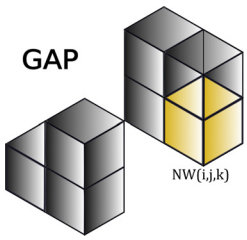

Fig. 6. Location of gap.

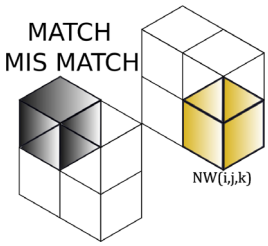

Fig. 7. Location of Match/Mismatch.

$$= max \begin{cases} NW(i-1,j-1,k-1) + s(x_i,y_j,z_k) \\ NW(i-1,j-1,k) + gap \\ NW(i-1,j,k-1) + gap \\ NW(i,j-1,k-1) + gap \\ NW(i,j,k-1) + gap \\ NW(i,j-1,k) + gap \\ NW(i-1,j,k) + gap \end{cases} \cdots\cdots (6)$$

長さが n，m，l の配列をそれぞれ $X = x_1, x_2, \ldots, x_i$，$Y = y_1, y_2, \ldots, y_j$，$Z = z_1, z_2, \ldots, z_k$ としたとき，初期条件を (5) 式に示す。また，Fig. 6 と Fig. 7 より，スコア計算は (6) 式により求められる。この場合，拡張した NW アルゴリズムの計算量は $O(mnl)$ となる。

〈4･2〉 拡張アルゴリズムの例 例えば，X = “BD”，Y = “$ABCD$”，Z = “ABD” という 3 つの配列を 3 次元に拡張した NW アルゴリズムで最高スコアになるように整列させる。(6) 式を用いて $(i,j,k) = (0,0,0)$ から $(i,j,k) = (2,4,3)$ まで 1 格子ずつスコア計算していくと，スコアテーブルは Fig. 8 のようになる。トレースバックでは，Fig. 8 に示したように $(i,j,k) = (2,4,3)$，$(i,j,k) = (2,3,3)$，$(i,j,k) = (1,2,2)$，$(i,j,k) = (1,1,1)$，$(i,j,k) = (0,0,0)$ という経路をたどる。$(i,j,k) = (2,4,3)$ から $(i,j,k) = (2,3,3)$ への経路で X と Z に，$(i,j,k) = (1,2,2)$ から $(i,j,k) = (1,1,1)$ への

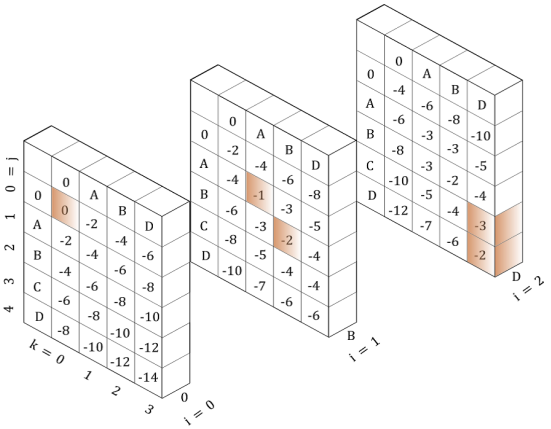

Fig. 8.　Score table and traceback.

経路では X にギャップが発生しているため，これらの箇所にはギャップ “–” が格納される。以上の手順より，配列 $X =$ “BD”，$Y =$ “$ABCD$”，$Z =$ “ABD” を整列させると配列 $X' =$ “$-B-D$”，$Y' =$ “$ABCD$”，$Z' =$ “$AB-D$” 及び $Score = -2$ を得る。

5.　GPU を用いたアルゴリズム高速化の検討

拡張した NW アルゴリズムを高速化する手法について検討する。そのうちの手法のうちの一つとして，GPU を用いた高速化がある。

〈5･1〉 GPGPU　　コンピュータの画像処理を担当するユニットの 1 つである GPU（Graphics Processing Unit）は多数のプロセッサが集まることにより構成されており，プロセッサ 1 つだけの機能に関して言えば CPU の機能に比べ劣るものの，膨大な量の計算を複数のプロセッサで並列処理することによりプログラムへの適用の仕方次第では CPU に比べより高速な演算を行うことができる。GPGPU は，GPU を画像処理以外の汎用数値計算において利用する技術のことである。現在では，医療画像や数値流体力学，環境科学，金融分野など幅広い分野で応用されている。

〈5･2〉 CUDA　　CUDA は GPGPU を行うための統合開発環境のことであり，NVIDIA 社によって提供されている。頂点やピクセルといったグラフィックス関連の概念を考慮せずに，通常の C 言語コンパイラのように扱うことができる。CPU 側をホスト，GPU 側をデバイスと呼び，それぞれで処理を分割して行う。ホスト側でカーネルプログラムを起動し，デバイス側でカーネルプログラムが動作する。そして処理終了時に結果をデバイス側からホスト側に転送する。デバイス側における最小の実行単位をスレッドといい，スレッドをまとめたものをブロック，ブロックを更にまとめたものをグリッド（デバイス）と呼ぶ（Fig. 9）。

また，CUDA プログラミングでは様々なメモリの特性を理解することが求められる。グローバルメモリはアクセス速度が遅いが大容量である。そのため多用すると，パフォーマンスの低下を引き起こしてしまう。共有メモリは 16 KB の小容量だが，アクセス速度が非常に高速で，数百サイクルかかるグローバルメモリに対して，4 サイクルでのアクセスで利用できる。また，処理の前後にはホストとデバイス間のメモリ転送が行われるため，転送に要する時間には注意しておく必要がある [10][11]。

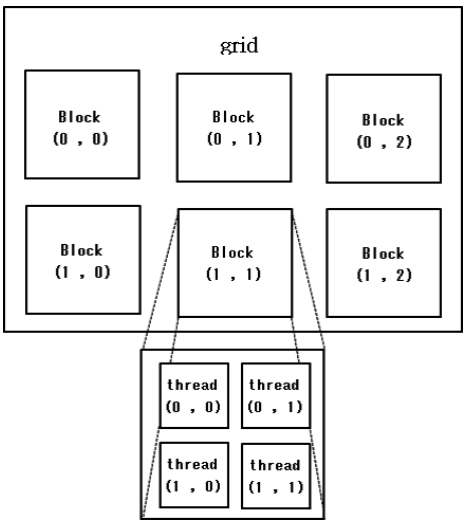

Fig. 9.　Configuration of GPU architecture.

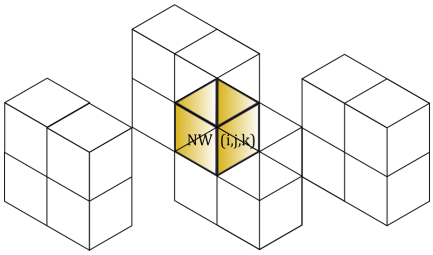

Fig. 10.　Dependencies of elements.

〈5･3〉 提案アルゴリズムの CUDA 実装に関する検討

拡張した NW アルゴリズムを CUDA で実装する際に検討することを以下に示す。

3 次元スコアテーブルにおいて，任意の要素 $NW(i, j, k)$ と依存関係にある格子を Fig. 10 に示す。マッチ，ミスマッチ，ギャップのいずれかに当てはまる格子が依存関係のある要素となる。格子に依存関係があるとその箇所は並列化することができないため，スコアテーブル作成のステップにおいて Fig. 10 の格子は同一ブロックで計算を行うか，シェアードメモリを利用するなどしてデータの共有を行う必要がある。また，依存性のない格子に関しては並列化を行うことができるため，段階的に並列化可能な要素を求め，それぞれに対し並列計算を行う。トレースバック部に関しては，条件分岐が多く，スコア計算に比べ計算量も少ないため，カーネル側で行う。Fig. 11 にホストとデバイス間の処理の流れを示す。ホスト・デバイス間では入力配列のアドレス，配列の大きさ，STEP 数，スコアといったデータのやり取りを行う。

〈5･4〉 実験的評価　　提案したアルゴリズムを文字列を変化させ，CPU 実装と GPU 実装のそれぞれで実行し，実行時間を記録した。使用機器を Table 1 に示す。

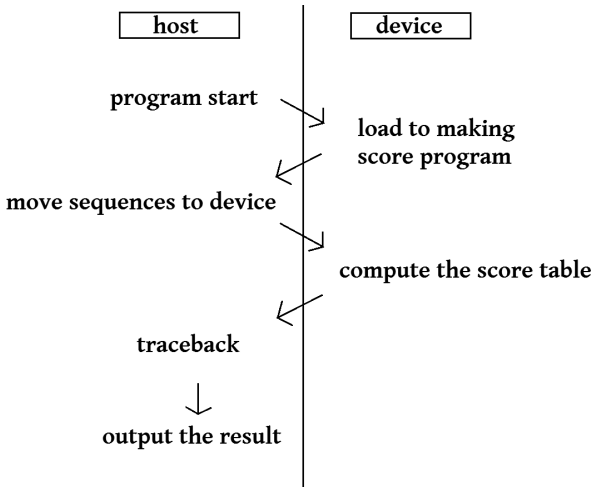

Fig. 11. Processing flow.

Table 1. Environment.

OS	Windows7 Enterprise 64bit
CPU	Intel Core i7-4770K 3.50 GHz
GPU	NVIIA GeForce GTX TITAN
Memory bandwidth	288 [GB/s]
floating-point performance (single-precision)	4.5TFLOPS
floating-point performance (double-precision)	1.3TFLOPS
developmental environment	CUDA7.5

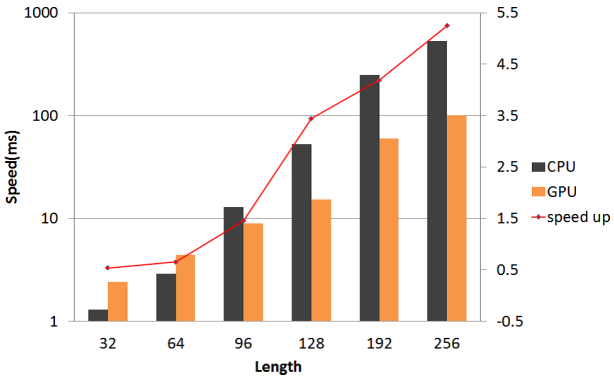

Fig. 12. Execution time of CPU/GPU.

CPU 実装と GPU 実装における実行速度と高速化の割合を Fig. 12 に，Fig. 12 における文字列が 32 から 96 の領域を拡大したものを Fig. 13 に示す。

Fig. 12 より，GPU 実装において，文字列が長くなるに従い CPU 実装に対して高速化される割合が高くなっていることが分かる。これは，提案手法ではステップ数が多くなるに従い並列化可能なブロック数も増加するためだと考えられる。

また，Fig. 13 より，文字列が短い場合は CPU 実装に対して GPU 実装のほうが低速となっていることが分かる。これは，メモリの転送時間が，スコア計算処理時間以上になっているためだと考えられる。

6. まとめと今後の課題

本研究では，高精度での多系列データ解析の高速化を目

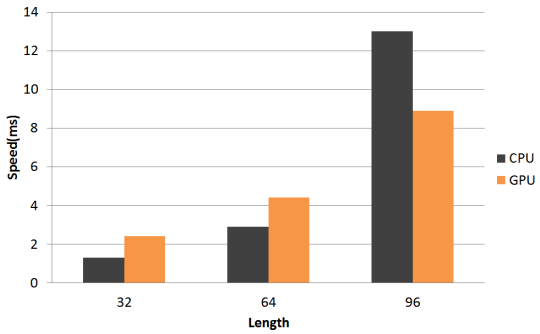

Fig. 13. Execution time of CPU/GPU(32-96).

標とし，NW アルゴリズムを拡張し GPU 上に実装した。その結果，CPU 実装に比べ最大 5.25 倍の高速化が実現できた。今後の課題としては，定量的な精度の算出や，条件分岐を減らすなどといったアルゴリズムや並列化手法改良によるさらなる高速化などが挙げられる。

謝　辞

本研究は，豊橋技術科学大学高専連携教育研究プロジェクト及び日本学術振興会の科学研究費補助金（No.40412902）の支援を受けた。

文　献

（1） NCBI: “BLAST: Basic Local Alignment Search Tool”, http://blast.ncbi.nlm.nih.gov/Blast.cgi
（2） 宗川裕馬・伊野文彦・萩原兼一：「統合開発環境 CUDA を用いた GPU での配列アライメントの高速化手法」, 情報処理学会, pp.17–18 (2008)
（3） 伊野文彦・小谷裕基・萩原兼一：「GPU グリッドによる高速な塩基配列アライメント」, 情報処理学会, p.74 (2007)
（4） 岡田啓佑・伊野文彦・萩原兼一：「遺伝子配列に対するペアワイズアライメントの GPU による高速化」, 情報処理学会研究報告, p.1 (2012)
（5） 須戸里織・吉見真聡・三木光範：「GPU を用いた 3 次元 Smith-Waterman 法の高速化手法の提案」, 信学会, pp.35–39 (2011)
（6） DNA Data Bank of Japan (DDBJ): “ClustalW”, http://clustalw.ddbj.nig.ac.jp/
（7） debian: “Multiple alignment program for amino acid or nucleotide sequences”, https://packages.debian.org/ja-/squeeze/mips/mafft
（8） Swiss Institute of Bioinformatics (SIB): “T-Coffee”, http://tcoffee.vital-it.ch/apps/tcoffee
（9） Y. Liu, B. Schmidt, and D.L. Maskell: “CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions”, BMC Research Notes, pp.1–12 (2010)
（10） J. Sanders + E. Kandrot, 訳：(株) クイープ：“CUDA BY EXAMPLE—An Introduction to General-Purpose GPU Programming—”, （株）イクスプレスジャパン, pp.6–9 (2011)
（11） 岡田賢治：「CUDA 高速 GPU プログラミング入門」, （株）秀和システム, pp.34–41 (2010)

尾　風　　仁　（非会員） 2011 年沖縄工業高等専門学校情報通信システム工学科入学。現在，在学中。

山 田 親 稔（正員） 2000 年琉球大学大学院理工学研究科博士前期課程修了。2004 年同大学大学院博士後期課程単位取得満期修了。同年拓殖大学北海道短期大学専任講師。2007 年沖縄工業高等専門学校情報通信システム工学科助教。2009 年同高等専門学校情報通信システム工学科准教授。2014 年ビクトリア大学（カナダ）客員研究員。2015 年より，沖縄工業高等専門学校情報通信システム工学科准教授。現在に至る。博士（工学）。形式的設計検証，リコンフィギャラブルシステムの研究・教育に従事。IEEE，電子情報通信学会，各会員。

宮 城 桂（非会員） 2008 年高知工科大学情報システム工学科卒業。2010 年同大学大学院修士課程修了。2014 年同大学大学院博士課程修了。同年沖縄工業高等専門学校情報通信システム工学科助教。現在に至る。博士（工学）。自己同期型パイプラインを用いた低消費電力 VLSI の研究に従事。電子情報通信学会会員。

市 川 周 一（非会員） 1985 年東京大学理学部卒業。1987 年同大学大学院理学系研究科修士課程修了。1987 年新技術事業団創造科学推進事業（ERATO）後藤磁束量子情報プロジェクト研究員。1991 年三菱電機（株）LSI 研究所，システム LSI 開発研究所勤務。1994 年名古屋大学工学部助手。1997 年豊橋技術科学大学工学部知識情報工学系講師。2001 年豊橋技術科学大学工学部知識情報工学系助教授。2007 年豊橋技術科学大学工学部知識情報工学系准教授。2010 年豊橋技術科学大学大学院工学系研究科准教授。2011 年沼津工業高等専門学校制御情報工学科教授。2012 年より，豊橋技術科学大学大学院工学系研究科教授。現在に至る。理学博士。並列計算機，並列処理，および専用計算システムアーキテクチャの研究に従事。IEEE（senior member），電子情報通信学会（シニア会員），ACM，情報処理学会，各会員。