RESIDUE NUMBER SYSTEM BASED OPTIMIZATION OF SMITH-WATERMAN ALGORITHM

HASSAN KEHINDE BELLO[1*], RAFIU MOPE ISIAKA[2], & KAZEEM ALAGBE GBOLAGADE[3]
[1]Department of Computer Science, Federal Polytechnic, Offa
[2]Department of Computer Science, Kwara State University, Malete, Nigeria
[3]Department of Computer Science, Kwara State University, Malete, Nigeria
E-mail: hassanbello@fedpoffaonline.edu.ng

Abstract
One of the greatest global challenges in bioinformatics is the computational time and memory utilization in biological sequence alignment (BSA). Several studies on the computation of BSA showed that the run time and its corresponding memory are on the high side. A number of algorithms that have been attempted to optimize BSA speed include fast alignment search tool algorithm (FASTA), basic local alignment search tool (BLAST), Needleman-Wunsch and Smith-waterman algorithm (SWA). Evidence has also shown that SWA is the most widely used BSA algorithm due to its accuracy which is, however, found to be relatively slower. Recently, there is a rapid exponential growth in International biological database size which necessitates the urgent optimization of SWA. In this work, the attributes of Residue Number System (RNS) such as carry free addition, borrow free subtraction, digit by digit multiplication without partial product and parallel computation are leveraged to enhance the performance of SWA. The implementation was carried out on MATLAB 2016 version RA (9.0.341360). The metrics used for evaluation are processing time and memory utilization. The results were finally compared with SWA computation. The computational time of SWA and SWA-RNS based are 23.34 and 7.53 respectively, while the memory space utilized respectively are 12.86 and 2.53. The results demonstrate the great computing power of RNS in the operation of SWA with an improvement of 51.21% and 67.12% in speed and storage requirement respectively.

Keywords: Biological sequence alignment, Smith-waterman algorithm, Residue number system, Computational time, Memory space.

Introduction
The advent of bioinformatics has brought many innovations in living organisms. These innovations are possibly due to the progress traced to sequence alignment techniques. In all living organisms, the nucleus of a cell contains genetic information like chemical known as deoxyribonucleic acid (DNA). This DNA stores biological information (Watson & Crick, 1953) which is responsible for uniqueness in living organisms. It is made up of chemical building block called nucleotide (Barbora, Michael, Norbert, Martin & Marc, 2015) which contains nitrogen molecules Adenine, Cytosine, Guanine and Thymine associated with the following abbreviations A, C, G and T respectively (Amal & Safwat, 2015). The comparison between these sequences of nucleotides is known as sequence alignment. Sequence alignments play a vital role in the history of evolutionary development (Bain, 1986). This is where a query sequence is arranged in parallel with the subject sequence from a large database in order to find the similarities between them (Durbin, Eddy, Krogh & Mitchison, 1998). Bioinformatics Scientists used the result of the comparison to deduce biological information of newly discovered sequence from a set of previously known sequences. In addition, if Bioinformatics Scientists are able to discover similarity between DNA sequences of two different species, then, evolutionary trend between them can be predicted. Another vital usage of sequence alignment is that the relationship between disease and its inheritance can also be marked out. This is done by aligning a specific DNA sequence of individuals with the disease to those of normal persons. If correlations can be found in the DNA comparison, then it can be used to identify level of vulnerability of certain disease, where new drugs may be designed as means to treat the

disease (ChiWai, Kin-Hong & Philip, 2005). This means, a better understanding of human DNA proffers a better personal treatment. A number of BSA algorithms on state-of-the-art among others are FASTA, BLAST, Needleman-wunsch and Smith-waterman algorithm (SWA). SWA is the most popular BSA algorithm widely used on state-of-the-art because of its accuracy. It is also used for identifying relationships between strings of genetics; however, the computational cost and memory space requirement are complex and intolerable. Several techniques have been proposed to optimize the algorithm like field programmable gate array (FPGA), graphic processing unit (GPU), single instruction multiple data (SIMD), systolic array, recursive variable expansion (RVE), code profiling, etc. Still, neither of these techniques achieves the expectation. Recently, there is enormous usage and fast growth of bioinformatics in various applications, coupled with the exponential growth in size of international biological database (Luscombe, Greenbaum & Gearstein, 2001) which requires fast, reliable, accurate and efficient sequence alignment algorithm in the field of computational biology (Sara, Sameh, Arabi & Keshk, 2017), for this reason, the urgent acceleration of SWA becomes paramount.

Consequently, the unique attributes of residue number system (RNS) which include, carry free addition, borrow free subtraction, digit by digit multiplication without partial product and parallel computation which would help the application faster in most system are also explored in this work.

Technical Background

Short or smaller sequences can be aligned manually, but with recent fast growing of biological database, manual method of alignment will be difficult and inappropriate. Instead, researchers on state-of-the-art designed biological alignment algorithms for the purpose of fast and efficient alignment. SWA and RNS are used in this study. The SWA is used as the algorithm that computes the alignment while RNS is used as the accelerator that enhances the processing speed and minimizes the space complexity.

The Smith-waterman algorithm

The Alignment algorithm of two sequences K and T is given by Smith-waterman (Smith & Waterman, 1981) as in equation (1). Where $M_{i,j}$ is the maximum alignment value between the two sequences.

$$M(i,j) = \text{Max} \begin{cases} 0 \\ M(i-1,j-1) + S(x_i,y_j) \text{ match/mismatch} \\ M(i-1,j) + g \\ M(i,j-1) + g \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

At $M(0,0) = 0$, $M(0,j) = g \times j$ and $M(i,0) = g \times i$, where $1 \leq i \leq n, 1 \leq j \leq m$. Where g is the penalty for gap insertion in any of the sequence and M(i,j) is the score for match or mismatch, depending upon whether K[i] = T[j] or K[i] ≠ T[j].O(m + n) is the time complexity for the initialization step, where m is the total number of elements in sequence K and n is the total number of elements the sequence T.

Order of computing values in SWA

Step 1: Initialization

At the initialization step, the matrix M(i,j) will be initialized with $M_{0,j} = 0$ and $M_{i,0} = 0$, for all i and j as shown in Figure 1.

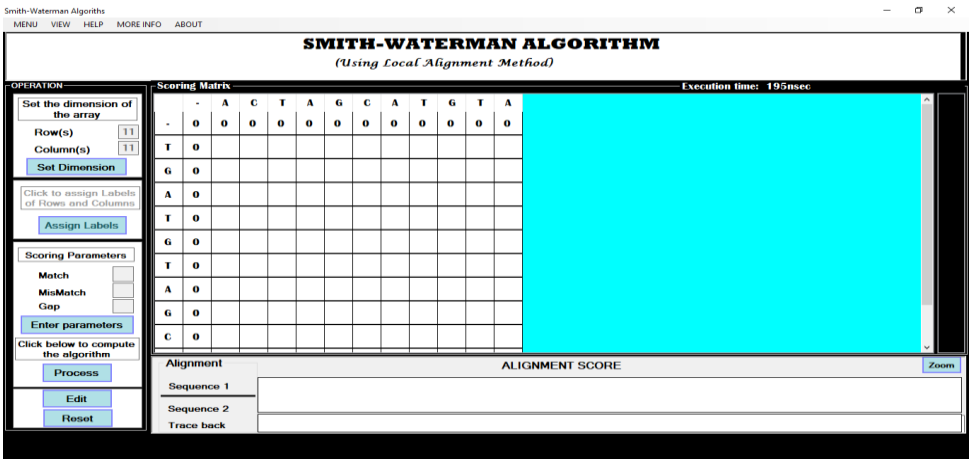

Figure 1. Matrix initialization

Step 2: Matrix filling

Equation 1 will be used to fill up all the entries in the matrix M(i,j) based on the given scoring parameter (Figure 3). Here, O(mn) is the time complexity, at this step the total number of elements in the matrix is given by mn.

Step 3: Trace-back

At the end of step 2 above, the cell with the highest score will occupy the bottom right of the matrix and is traced back to get the optimal local alignment (Figure 5). The time complexity of the trace-back is given by O(m + n).At the end of step 3, the total time complexity of Smith-Waterman algorithm is given by O(m+n) + O(mn) + O(m+n) = O(mn). Also, the space complexity of the algorithm is given by O(mn) because the size of the matrix is m x n.

The Residue Number System

Residue Number System (RNS) is defined by a set of relatively prime integers called the moduli. The moduli set is represented as $\{m_1, m_2, m_3 \ldots m_n\}$ where m_k is the k^{th} modulus. Each integer can be represented as a set of smaller integers called the residue (Korishma & Sun, 1993; Omar, 2011; Youssef, Emam & Abdul-Elghany, 2012). It is defined over an interval of relatively prime modulus sets $\{P_1,P_2,P_3 \ldots P_n\}$, where the greatest common divisor (gcd) (Taylor, 1984) between any two of P_i and $P_j = 1$, $i \neq j$ i.e gcd $(P_i, P_j) = 1$. Let us represent a number X as $\{x_1, x_2, x_3, \ldots, x_n\}$. This illustration is distinctive for any integer X in the range [0, M-1]. Where M is the product of the given moduli sets (i.e. $M = P_1 \times P_2 \times P_3 \times \ldots \times P_n$). RNS offers flexibility in digit-by-digit computation; this makes addition, subtraction and multiplication faster and efficient r is the RNS of a number X with respect to m, if and only if r is the leftover after several removals of all multiples of m from X, where m is the moduli.

Related Works

Many authors in the past have used various techniques to improve BSA algorithm. In 2012, Nur-Farah, Nur-Dalilah, Syed, Zulkifli and Abdul-Karimi proposed a Smith-Waterman software version using FPGA which was implemented on EP4CE115F29C7 FPGA. Thirty-two tests were conducted ranging from 2x2 base-pair to 64x64 base-pair to measure the run-time of the software. The result is such that the runtime is reducing from 3.67 to 1.07 times less during the test which is vice-versa with the number of cells. Meanwhile, the average runtime for each cell is ranging from 0.03492 to 0.0450 per cell. It was concluded that the runtime is depending on the iterative computational method used.

In 2016, Fereshteh and Shiva presented a paper on the efficient design of residue to binary converter, using CRT on the moduli $\{2^n, 2^{2n}+ 1, 2^n+ 1, 2^n-1\}$. A comparative analysis was

carried out between the previous and the proposed design. The results showed that the proposed design is having 15.02%-time complexity less than the previous design.

Recently, Ernst, Vlad-Mihai and Zaid in 2017 proposed a highly optimized SWA implementation on FPGA using OpenCL. This clearly reduced the code. The implementation is fast and efficient with GCUPS of 214 and frequency of 193MHz. The technique is restricted to linear systolic array and the theoretical values are meaningful only if all the processing elements are busy performing with few alignments.

In 2018, Fei, Dan, Lina, Xin and Chunlei, proposed SWA sequence alignment application with backtracking on Field Programmable Gate Array. The design was able to get rid of data dependency, decrease storage requirements and speed up between 3.6 and 25.2 over the previous SWA on a general-purpose computer platform.

Methodology

In this work, we applied RNS with $\{2^{2n+1}-1, 2^{n-1}, 2^{2n}-1\}$ as choice of our moduli set to compute SWA. The dynamic range (DR) is the product of $2^{2n+1}-1$, 2^{n-1} and $2^{2n}-1$. The data used are Homo sapiens nucleotides of Eukaryota (seq1) and Metazoa (seq2) from National Centre for Biology Information (NCBI), ("Analysis of the DNA sequence", 2006). Converters are designed for data conversion and processed by RNS. MATLAB version 2016 is used to carry out the implementation.

Step 1: SWA-RNS based forward converter design

Design of SWA-RNS forward converter involves use of the moduli $\{2^{2n+1}-1, 2^{n-1}, 2^{2n}-1\}$ set to convert the range of conventional numbers to residue numbers (Table 1).

From $\{2^{2n+1}-1, 2^{n-1,} 2^{2n}-1\}$, DR is M = 31*2*15 = 930.

$$-\frac{M}{2} \leq X \leq \frac{M}{2} - 1 \qquad \text{...} (2)$$

$-\frac{930}{2} \leq X \leq \frac{930}{2} - 1$ Range of X is $-465 \leq X \leq 464$

The inputs variables (M(i,j-1), M(i-1,j), M(i-1,j-1), S(i,j) and g) in conventional numbers are converted to residue number system by binary to RNS converter (BRC).

Table 1: Residue Table for MOD(31,2 15)

Conventional Number (X)	-465	-464	-463	…	-1	0	1	…	463	464
MOD(31,2,15)	0,1,0	1,0,1	2,1,2	…	30,1,4	0,0,0	1,1,1	…	29,1,3	30,0, 4

Step2: SWA-RNS based Processor design

SWA-RNS processor design involves many components and stages (figure 2). It includes 1 look-up-table (LUT), 3 adders (ADDER) and 3 comparators (COMP) before obtaining the optimal value $M_{i,j}$

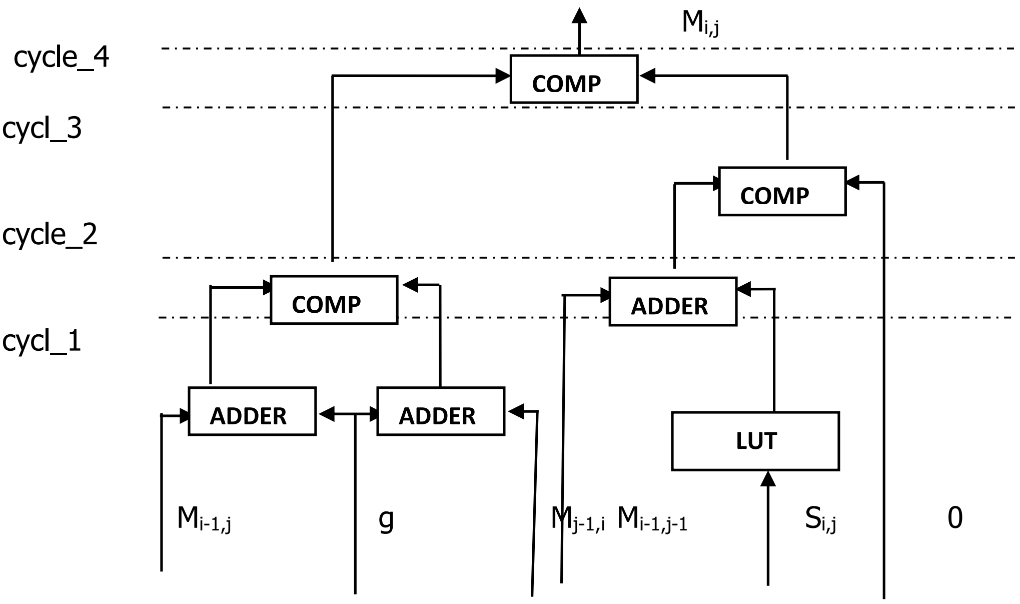

Figure 2: Computational circuit for matrix $M_{i,j}$

The logic circuit comprises of three adders, three comparators and one look-up table as shown in Figure 2. Each value in a cell is computed through the following steps:
(1) The components $M_{i-1,j}$ are added to g (upper addition);
(2) The components $M_{1,j-1}$are added to g (left addition);
(3) The comparator selects the higher of (1) and (2);
(4) The components $M_{i-1,j-1}$ are added to $S_{i,j}$ (diagonal addition);
(5) The comparator compares the result of (4) with zero and selects the higher;
(6) The comparator finally compares the higher of (3) and (5) then picks the higher of the two values.

Step 3: Design of a reverse converter
In this work, Chinese Remainder Theorem (CRT) algorithm is used to convert from residue to binary number (RBC). The CRT algorithm is given as

$$X = \left| \sum_{i=1}^{n} | x_i M_i^{-1} \right| M_i|_M \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (3)$$

Where$M = \prod_{i=1}^{n} mi, \bar{M} = \frac{M}{mi}$; $|M_i^{-1} \text{ x } M_i \ | = 1$

The reverse conversion of the residue 13,1,0 with respect to the moduli set {31,2,15} is computed as follows: $x_1 = 31$, $x_2 = 1$ and $x_3 = 0$; $m_1 = 31$, $m_2 = 2$ and $m_3 = 15$;
then M = 31 * 2 * 15 = 930
$M_1 = 30$, $M_2 = 465$ and $M_3 = 62$
$|M_1^{-1}|_{m1} = |30^{-1}|_{31} = 30$
$|M_2^{-1}|_{m2} = |465^{-1}|_2 = 1$
$|M_3^{-1}|_{m3} = |62^{-1}|_{15} = 8$
$|X|_M = |\sum^{n}_{i=1}|x_i M_i^{-1}|_{mi} M_i|_M$
$|X|_M = |\sum^{3}_{i=1}|x_i M_i^{-1}|_{mi} M_i|_M$
$|X|_M = |x_1|M_1^{-1}|_{m1} M_1 + x_2 |M_2^{-1}|_{m2} M_2 + x_3|M_3^{-1}|_{m3} M_3 |_M$
$= | (13)(30)(30) + (1)(1)(465) + (0)(8)(62) |_{930}$
$= | 11700 + 465 + 0 |_{930}$
$= | 12165|_{930}$
$= 75.$

MATLAB Implementation of steps 1 to 3 above

Two sequences (Seq1 and Seq2) are used in the implementation on MATLAB where Seq1: TGATGTAGCGA and Seq2: ACTAGCATGTA, match = 2, mis-match = 1, gap = -1 and n = 2 in 2^{2n+1}-1, 2^{n-1},2^{2n}-1 gives 31,2 and 15. The implementation is carried out on MATLAB 2016 version RA (9.0.341360). Results are shown in Figures 4, 5 and 6.

Results and Discussion

The result of computed SWA, with alignment value of 14 is shown in figure 3.

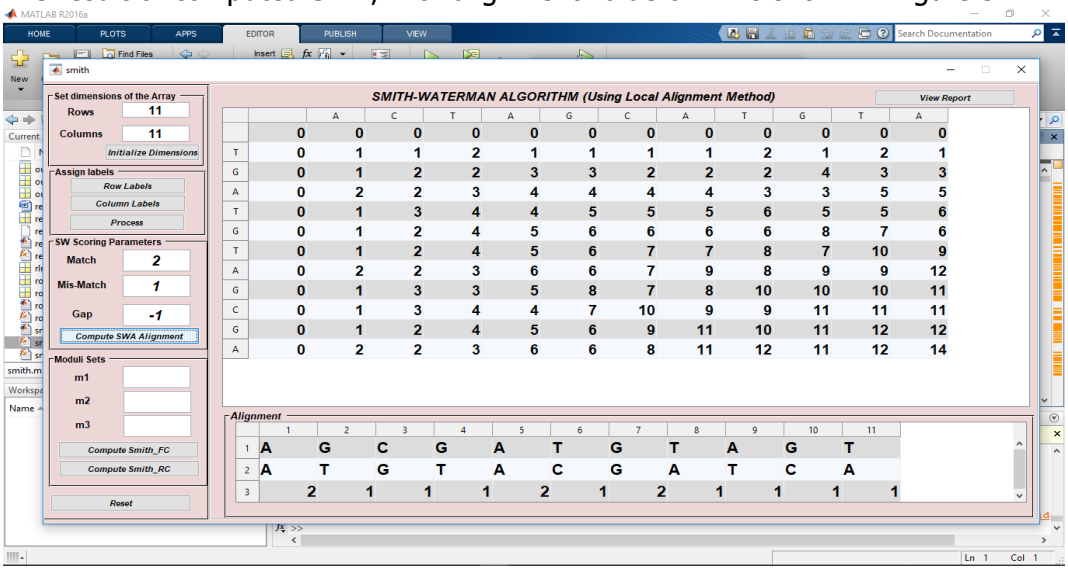

Figure 3: Output of SWA

MATLAB forward conversion of SWA without using RNS is shown in figure 3.5b with alignment score of 14.

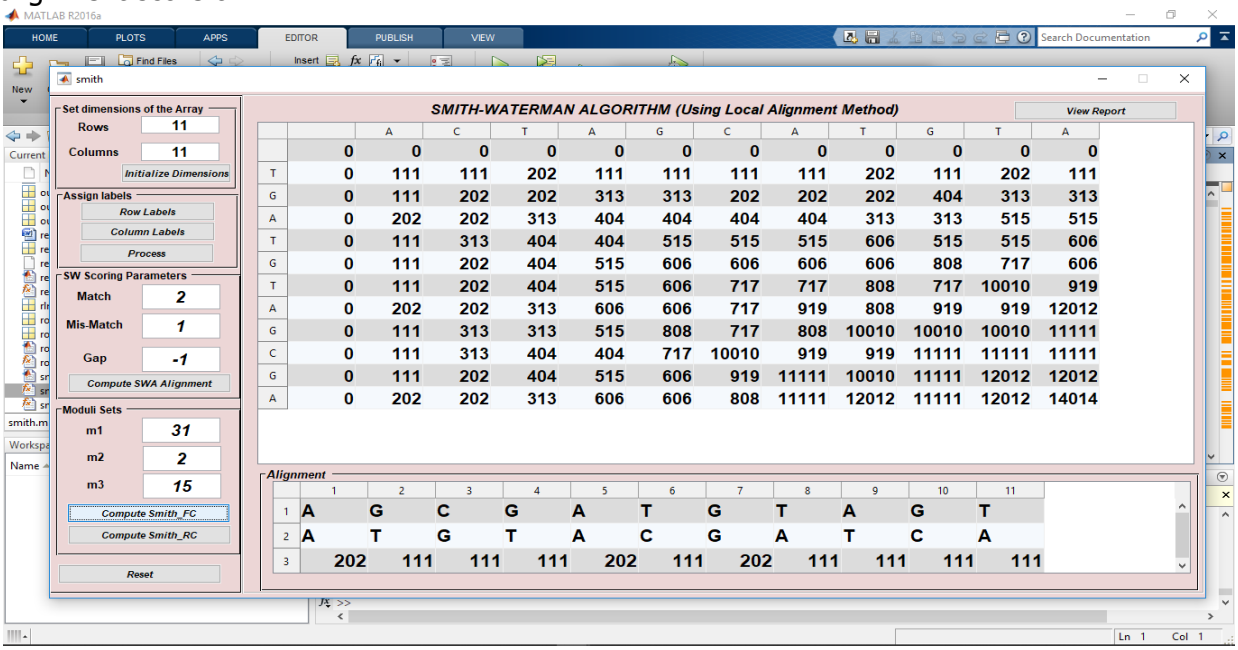

Figure 4: SWA-RNS based Forward conversion output

MATLAB forward conversion of SWA with RNS application (SWA-RNS based) is shown in figure 4.

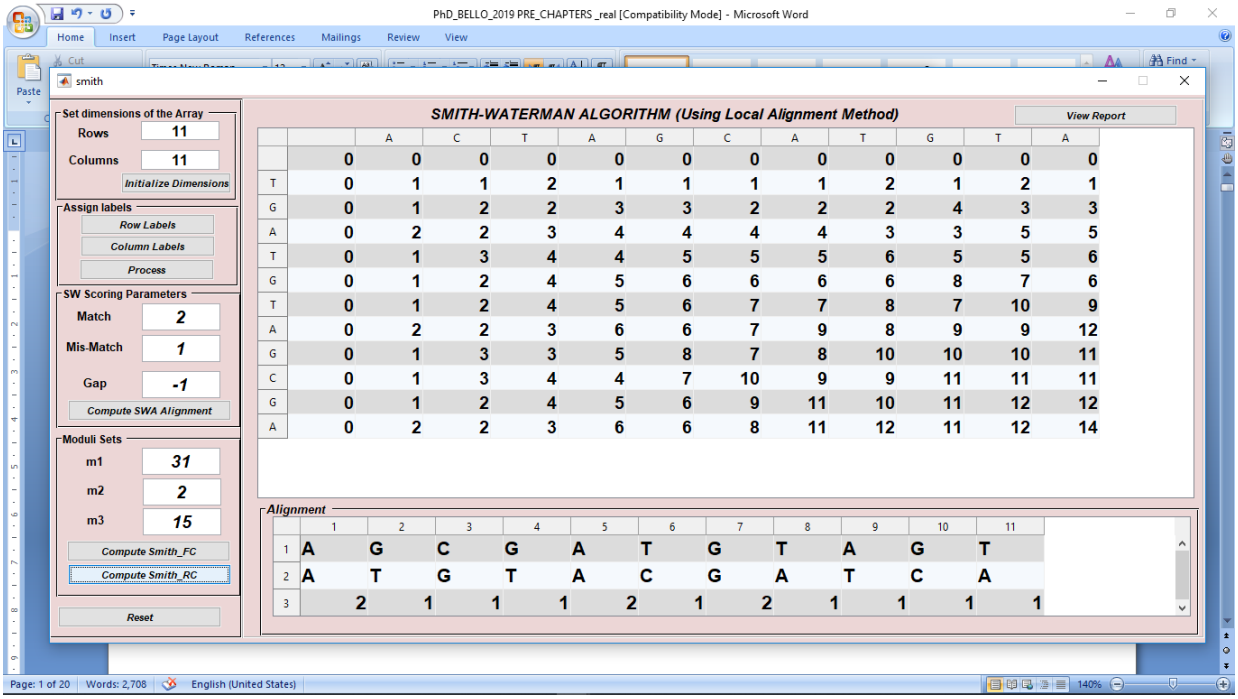

Figure 5: SWA-RNS based Reverse Conversion output

The final result of SWA-RNS based is produced with alignment score of 14 depicted on figure 5.

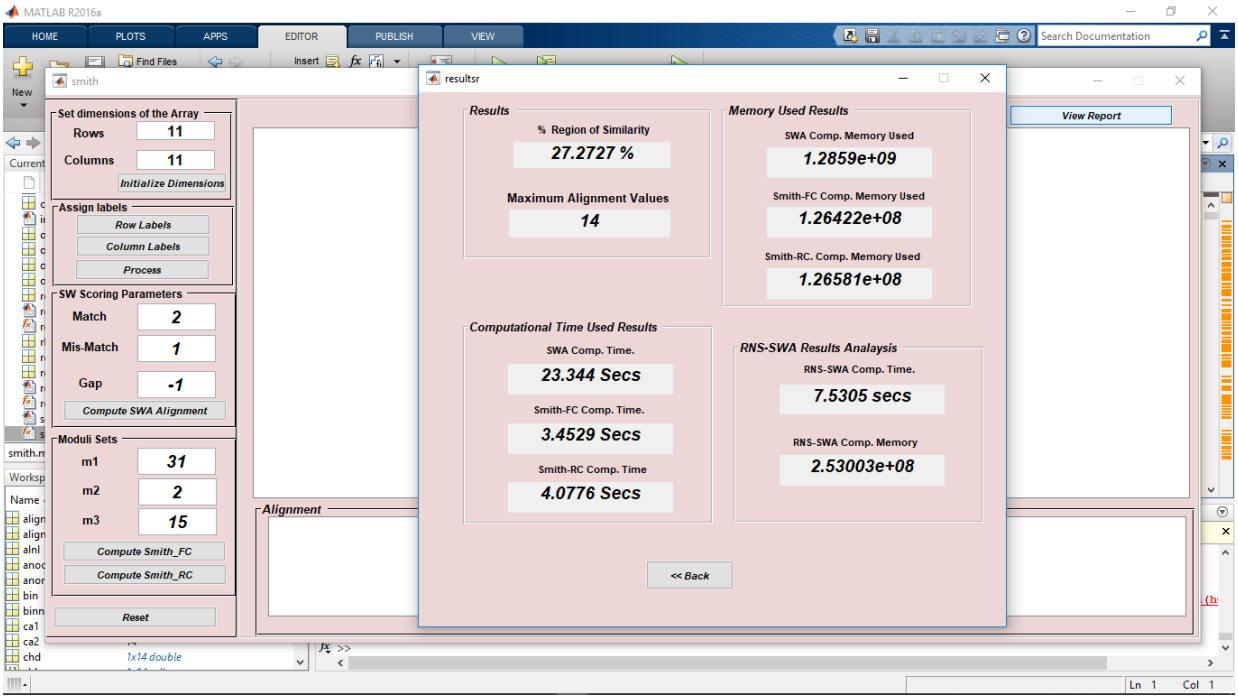

Figure 6: Report analysis

From Figure 6:
Maximum alignment value = 14
Percentage region of similarity = 27.27%

Table 2: Results from MATLAB

Match = 2, Mis-match = 1, gap = -1; m1 =31, m2 = 2 and m3 =15
MATLAB 2016 version RA (9.0.341360)

Computational Time (sec)		Computational Memory used (dcb)	
SWA	SWA-RNS based	SWA	SWA-RNS based
23.34	7.53	12.86	2.53
(%) Improvement 51.21%		% Improvement 67.12%	

Results from MATLAB application in the implementation is shown on Table 2. Computational time improvement 51.21%; Memory used improvement 67.12%.

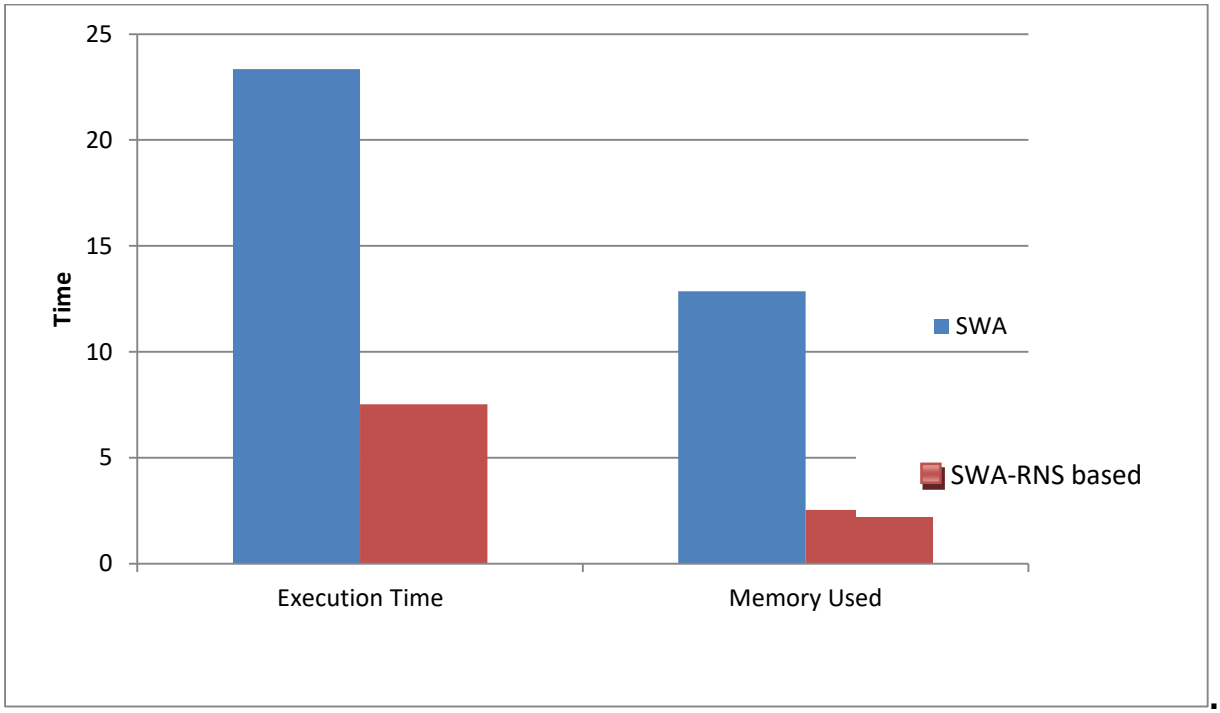

Figure 7: Memory vs. execution time

Conclusion

RNS is becoming popularly used in literature because of its unique attributes which can influence speed on any application that employs operations of addition, subtraction and multiplication. It has been suitably used in Digital Signal Processing, Digital Image Processing, RSA algorithm, Cryptography and Fourier applications. Our findings have shown that RNS can be used in bioinformatics to enhance processing speed and reduce space complexity in SWA. This is a great hope for bioinformatics community.

References

Amal., K., & Safwat, H. (2015). Hiding secret information in DNA sequences using silent mutations. *British Journal of Mathematics & Computer Science, 11*(5),1-11. Article no. BJMCS.19561.

Analysis of the DNA sequence Homo sapiens golgin A6 family. Retrieved Sept 9, 2019, from

Bain, W. (1986). MULTAN: A program to align multiple DNA sequences. *Nucleic Acids Res, 14,* 159-179.

.

Barbora, K., Michael, K., Norbert, L., Martin, F., & Marc, B. (2015). Visualization of biomolecular structures. Eurographics Conference on Visualization (eurovista)

ChiWai, Y., Kin-Hong, L., & Philip, L. (2005). A Smith-waterman systolic cell, Chapter in Lecture Notes in Computer Science· January 2005.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). Biological sequence analysis: Probabilistic Models for Proteins and Nucleic Acids, Cambridge University Press, Cambridge UK.

Ernst, J. H., Vlad-Mihai, S., & Zaid, A. (2017). High performance streaming smith-waterman implementation with implicit synchronization on intel FPGA using OpenCL. IEEE 17th International Conference on Bioinformatics and Bioengineering,492-496

Fei, X., Dan, Z., Lina, L., Xin, M., & Chunlei, Z. (2018). Fpga-sw: Accelerating large-scale smith–waterman sequence alignment application with backtracking on fpga linear systolic array. *Interdisciplinary Sciences: Computational Life Sciences, 10*(1), 176-188.

Fereshteh, B. G., & Shiva, T. (2016). An efficient new CRT based reverse converter for the 4-moduli set $\{2^n, 2^{2n}+1, 2^n+1, 2^n-1\}$. *International Journal of Computer Science and Information Security (IJCSIS), 14*(12), 226-233.

Korishma, H., & Sun, J. D. (1993). On theory and fast algorithm for error correction in residue number system product codes. *IEEE Trans computers, 42*, 840-852.

Luscombe, N. M., Greenbaum, D., & Gearstein, M. (2001). What is bioinformatics? An Introduction and overview. Yearbook of Medical informatics, USA (2001), 83-96.

Nur-Farah, A. S., Nur-Dalilah, A. S., Syed, A. M., Al-Junid, Z. A., & Abdul, K. H. (2012). Software implementation of Smith-waterman algorithm in FPGA. Faculty of electrical engineering Universiti Teknologi MARA 40450, Shah Alam, Malaysia.

Omar, A. (2011). Data conversion in residue numbers system. A thesis submitted to the Department of Electrical & Computer Engineering McGill University Montreal.

Sara, S., Sameh, S., & Arabi, E. K. (2017). An efficient and precise position-based multiple sequence alignment technique. 2nd International Conference on Advanced Technology and Applied Sciences (ICaTAS 2017), 1-6.

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences (PDF). Journal of Molecular biology p195-197.

Taylor, F. J. (1984). Residue Arithmetic: A Tutorial with Examples, Computer, 17(5), 50-62

Watson, J. D., & Crick, F. H. (1953). Molecular structure of nucleic acids; A structure for deoxyribose nucleic acid. *Nature, 171*(4356),737–738.

Youssef, M. I., Emma, A., & Abdul-Elghany. (2012). Multi-layer data encryption using RNS in DNA sequence. *International Journal of security and its applications, 6*, 1-12.