Efficient Sequence Comparison Using Binary Codes

Hossein Kamel Rahimi

***Abstract*—In this paper, we propose an efficient way of finding the exact distance in sequence comparison by using Huffman coding method for alphabets with uniform symbol probabilities. The approach is proposed as a refinement for word pair comparison in D2 statistics, though it can readily be generalised. Two given sequences with identical lengths are encoded to Huffman binary codes by which we are able to calculate Hamming Distance using binary operations efficiently. This method is applied on D2 statistics to compare *k-tuples* faster than its original version. The evaluation on emprical sequences showed that the method is faster than original D2; especially, when re-using the encoded sequences which resulted in better performance.**

***Index Terms*—Performance, experimentation, string and sequence processing, huffman encoding, hamming distance, D2 statistics, string comparison, binary codes, bitwise operation.**

I. Introduction

Finding similarities and differences of bio-sequences such as DNA and amino acids is a fundamental problem in biological science. Vinga and Almeida argue that "Sequence analysis is a discipline that grew enormously in recent years in response to the overwhelming burst in data generated by molecular biology initiatives" [1]. As Bioinformatics is an essential ingredient for the success of human genome projects, powerful computer systems and sophisticated algorithms are required [2]. Sequence comparison has played a crucial role in molecular bio-sequence analysis [3] and is perhaps the most well-known and intensively studied problems in modern molecular biology [4]. Numerous sequence comparison algorithms have been developed to address the overwhelming amount of molecular data generated by sequencing of whole genomes and molecular libraries [1], [5].

Alignment-free sequence comparison regimes are frequenly used to compare genomic sequences and detect regularity regions [6]. The earliest and the most widely alignment-free approach is D2 statistics [7] which mesasures the number of words shared between two biological sequences called *k-word* (also known as *k-tuple* or *k-grams*) [8], [9].

Among the more important aspects of sequence comparison is the determination of sequences which 'almost match' the query, those which lie within some neighbourhood of the perfect match. This paper aims to propose a heuristic software solution for rapid determination of approximate matches between word pairs in D2 statistics. By encoding the base symbols through fixed codes according to the application, we allow the comparison to be computed via bitwise operations. The library sequence to be search against and the query is encoded only on one run and the rest of the computation approximates similar matches based on the encoded sequence as many times as needed with variant queries.

This paper is organized as follow paragraphs. In the next section, we introduce the main ideas in Section II before introducing the method formally in Section III and Section IV. Results are presented in Section V, and conclusions in Section VI.

Manuscript received January 26, 2015; revised March 24, 2015.

The author is with the Faculty of Science and Engineering, Queensland University of Technology, Australia (e-mail: H.Kamelrahimi@qut.edu.au).

II. Huffman Coding

The proposed approach relies on the well-knownentropy encoding method Huffman coding which is a common technique for data compression [10]. Various Huffman applications have been proposed and the method is still widely used. Mcintyre and Pechuraapplied Huffman to test data over various source programs in four different languages where the data was read from files [11]. Nourani and Tehranipour used Huffman technique to reduce test data volume, power dissipation and pattern delivery time in the field of Integrated Circuits [12]. Also, Tang utilised Huffman encoding methods for a proposed method of multimedia encryption for vast amounts of MPEG video data [13].

Since efficiency and effectiveness are the key aims in bio-sequence comparison algorithms, the aim of the proposed method is to utilize Huffman Binary Encoding as a mechanism of rapidly identifying exact and approximate matches of two bio-sequences and comparing pairs of bio-sequences much quicker than previous approaches. Encoded in Huffman binary codes, a bio-sequence could be compared rapidly regardless of the sequence's actual symbols because the core of the proposed algorithm, being Huffman Binary code, could be operated at machine bit level. Therefore, this research topic is seeking an accurate, fast, and efficient way of bio-sequence comparison which has been crucial in bioinformatics science for many years.

III. Huffman Binary Encoding

According to Huffman, the symbol or sequence symbol associated with a given message is called the *message code* [10]. In Bioinformatics, sequence *k*-mers are considered the *message code* which are the members of a set of alphabetic characters

$$\Sigma = \{A, C, G, T\}$$

Let $P(i)$ be the probability of i-th message, then

$$\sum_{i=1}^{N} P(i) = 1 \quad (1)$$

$$N = |\sum|$$

The basic type of the Huffman method is the 'static type', where the frequencies of message codes are constant; while in 'dynamic type', the frequencies change. Knuth [14] describes that "a Huffman tree with nonnegative integer weights can be represented in such a way that any weight can be increased or decreased". In this preliminary study, it is assumed that occurrence probabilities of the *k*-mers throughout the sequence are identical in such a way that satisfies equation (1) as equation (2):

$$P(A) = P(C) = P(G) = P(T) = \frac{1}{N}, \quad (2)$$

an assumption which is only mildly in error for most sequence data, and one that may be modified for high GC content sequences.

Having two bio-sequences S_1 and S_2 over the alphabet set $\sum = \{A, C, G, T\}$, we assign static binary codes 00,01,10 and 11 to A,C,G and T respectively:

$$S_1 = a_1 a_2 a_3 \ldots a_w$$
$$S_2 = b_1 b_2 b_3 \ldots b_w$$
$$ai \in \sum \text{ and } bi \in \sum$$
$$1 \le i \le w$$

Message codes are

$$A=00,\ C=01,\ G=10,\ T=11$$

At this point, we are able to compute the Huffman code for a given sequence by substituting any *k*-mer to its correspondent message code. For instance, given sequence S_1 =TCATG, the Huffman Binary code is H_1=1101001110.

A. D2 Statistics

Definition: The D2 statistic is defined as the number of words of pre-specified length k between two given sequences **A**=(A_1, …, A_m) and **B**=(B+1, …, B_n) where A_i and B_j belonging to the alphabet set $\sum$ of size $|\sum|$ =N and also let f_a be the occurrence probability of letter 'a', though in this work uniform probability for each letter is considered as $f_a = \frac{1}{N}$. Also, it is assumed that each sequence is generated independently and identically (i.i.d) of the letters in alphabet $\sum$ [8]. Let $Y(i, j)$ be the k-word match indicator variable indicating if the words starting at position i in A and word at position j in B match [5], [16].

$$D2(A,B) = \sum_{(i,j)\in I} Y(i,j)$$

where index set $I = \{(i,j): 1 \le i \le n-k+1, 1 \le j \le m-k+1\}$. Similarly, the approximate word match statistic

$$D2(t) = \sum_{(i,j)\in I} Y(t)(i.j)$$

where $Y_{(i,j)}{}^{(t)} 0 < t < k$ is the k-word match indicator variable allowing up to *t* mismatches.

IV. Method

The fundamental aspect of the method is the encoding function in which each base is substituted with its corresponding *message code*. Bradford [16] used the idea of *fixed binary codes* to measure Levenshtien [17] *edit distance* between a set of encoded strings with the respect to a given super-sequence, though the method is only reliable to the existence of the super-sequence. Although Bradford's work does not address the performance, in particular, for empirical sequences, its concept is promising to be studied for wider string comparison projects. In this paper, the idea of encoding strings to binary codes is developed to compare sequence pairs in respect to the facts that

1) Bitwise operations are highly efficient by which total comparison between sequence pairs is performed fast and (see section V-A)

2) Since most of the computation cost is paid as sequences are encoded, and needs to be done only once, the comparison process could be repeated comparatively rapidly with little computation [16]. (see Section V-B)

Given two sequences with identical length *N*, we aim to find the Hamming Distance [18] H_d between two Huffman Binary Codes associated to the sequences.

$$s_1 = a_1 a_2 a_3 \ldots a_N$$
$$s_2 = b_1 b_2 b_3 \ldots b_N$$

Let h_1 and h_2, the Huffman binary bit vectorsof s_1 and s_2 and H_d (equation 3) is the Hamming Distance bit vector calculated from Exclusive OR operation of h_1 and h_2 and represents those positions where there are mismatches comparing h_1 and h_2,.

$$Hd = XOR(h_1, h_2) \quad (3)$$

Despite true bits in H_d could be pronounced as distances between h_1 and h_2 bit vectors, it does not demonstrate Hamming Distance between s_1 and s_1, as each *k*-mer has been substituted with 2 bits and some substitutions involve more bits than others. In order to deal with this issue, we introduced normalized *D* as *Dz* (equation 4). We apply an OR operation on each adjacent bit-pair representing one *k*-mer

$$Dz(i) = OR\big(D(i), D(i+1)\big)\ where\ i = 2k \quad (4)$$

For computing *Dz* in an efficient way, bitwise operation is again required. Let D_s be the shifted version of *D* by one to the left and let $\underline{Z} = OR(D, DS)$. Due to the differences between *D* and Ds, then it could be claimed that

$$\underline{Z}(i) = OR\big(D(i), D(i+1)\big)\ where\ 1 \le i \le N-1$$

To satisfy equation (4), bits at indexes $2k+1\ where\ 0 \le k \le N-1$ positions in $\underline{Z}$ should be eliminated which here we call them *noise bits*; let $\acute{1}$ a finite sequence of "10"s with the length equal to $\underline{Z}$.

$$\acute{1} = 1010101010 \ldots$$

where

$$|\acute{1}| = \left| \underline{Z} \right|$$

and

$$Dz = AND(\underline{Z}, \text{Ī})$$

By counting true bits in *Dz*, Hamming distance H_d between S_1 and S_2 is revealed:

$$Hd(s_1, s_2) = cardinality(Dz)$$

In equation (4), cardinality *Dz* is the number of true bits found in Dz. For example, considering S_1=ACGTC and S_2= GCATG, we then have

$$h_1 = 0001101101$$

$$h_2 = 1001001110$$

Then

$$D = XOR(h_1, h_2) = 1000100011.$$

and *D* is shifted to the left by one as

$$Ds = 0001000110$$

$$\underline{Z} = OR(D, Ds) = 100110011$$

Since the length of $\underline{Z}$ is 10, thus is defined as a finite sequence of "10"s with the identical length of $\underline{Z}$.

$$\text{Ī} = 1010101010$$

To eliminate *noise bits* at odd index position

$$Dz = AND(100110011, 1010101010) = 1000100010$$

Finally Hamming distance between two sequences s_1 and s_2

$$Hd(s_1, s_2) = cardinality(Dz)$$

By replacing their corresponding values

$$Hd(ACGT, GCATG) = cardinality(1000100010) = 3$$

Hamming Distance of two sequences is 3 which was computed using bitwise operations.

A. Algorithm

Definition: Given a genome $g=\{s_1, s_2, \ldots, s_n\}$ on an alphabet $\sum = \{A, C, G, T\}$, $\left|\sum = 4\right|$, s_i is a sub-sequence of *g* with the length |*N*| where $1 \le i \le n$ and $P(A) = P(C) = P(G) = P(T) = \frac{1}{4}$ and let bit vector *h* as Huffman code where each alphabet in ∑ is encoded to binary codes as A← 00, C← 01, G←10 and T←11. Also, let *D* and *Dz* be bit vectors – the 'Distance Vector' and 'Normalized Distance Vector' respectively.

HuffmanEncoder(Sequence *s***)**

k←1; *l*←1 *N*←Length of *s*;
  While (*k*≤*N***)**
  D(*l*) and *D*(*l*+1)←binary Code of *s*(*k*);
  k←*k*+1; *l*←*l*+2;
  EndWhile();
EndEncode;

h_i←**HuffmanEncoder(**s_i**);**
h_j ← **HuffmanEncoder(**s_j**);**
D ← **XOR(**h_i,h_j**)**;
i←1;*j*←1;*N*← |*D*|;
While(*i* ≤*N*)
$D_z(j)$←**OR(***D(i),D(i+1)***)**
i←*i*+2; *j* ←*j*+1;
EndWhile
k←0;
While(*k* ≤*N*)
Ī *(k)*←"1";
Ī *(k+1)*←"0";
EndWhile

D_z←**AND(**D_z, Ī**);**
Return D_z.cardinality();

V. Experiments

Experimental evaluations were performed using custom implementations to evaluate the speed of the proposed approach. Given the library sequence to be searched against and the query to be searched for, let $\lambda(s)$ and $\lambda(q)$ be the elapsed CPU time for encoding *s* and *q* to fixed binary codes respectively. Furthermore, let $\gamma(s, q)$ be the required CPU time to search the encoded query *q* over the encoded library sequence *s*, then denote total execution time

$$\Delta = \lambda(s) + \lambda(q) + \gamma(s, q) \quad (5)$$

In this section, *k-tuple* comparison is performed using string comparison and the proposed approach to compute D2 total execution time approach (section A). Then CPU time will be recorded for the proposed comparison approach over encoded library sequence and the result is evaluated with section B. Finally, the effect of various *k* over CPU time and accuracy will be demonstrated (section *C*).

A. D2 String Comparison vs. Proposed Approach

In this section, the D2 value between the encoded library sequence s with the length $50 \le N \le 50,000$ and query with the length 300 and word size *k* = 3 are computed to examine the efficieny of the proposed approach by capturing total execution time Δ. One common approach to find the dissimilarity between *k-tuples* in D2 statistics is to take the minimum of all window distances for each pair $W = (W(L), W(Q))$, where $W(L)$ and $W(Q)$ are the *k-tuples* in library sequence and query sequence respectively [19]. In Fig. 1, we used the proposed binary method and string compariosn to compare *k-tuples* and recored total execution time for each sequence lengths in regard to equation (5).

B. Efficiency over Encoded Library Sequences

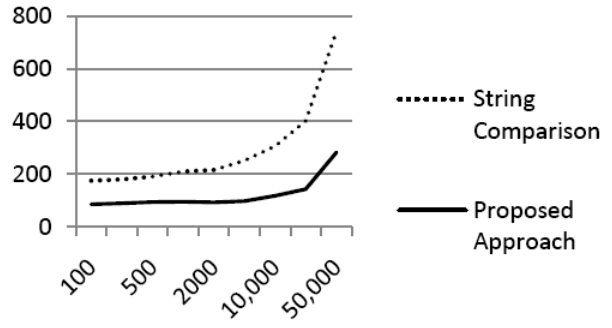

Fig. 1. Total execution time Δ for D2 computation over proposed binary approach and string comparison.

Several query searches could be performed over a given library sequence which has been already encoded in section A to address *reusability* aspect of this approach rather than running the encoding procedure several times over the same library sequence. Hence, in equation (5), let $\lambda(s)=0$ and denote new total execution time

$$\Delta^* = \lambda(q)+\gamma(s,q) \tag{6}$$

Equation (6) demonstrates that encoding time for queries $\lambda(q)$ should be included as queries could be different. In this experiment, different queries with identical length 50 have been selected in order to have the same impact on query encoding time $\lambda(q)$ in (6) with respect to the total time Δ analysis with uniform data characteristics. In regard to Table I, the figures show that this approach could be even more efficient while re-using the encoded library sequence over various query searches.

TABLE I: THE PROPOSED APPROACH TOTAL EXECUTION TIME (MS) BEFORE ENCODING Δ AND AFTER ENCODING Δ^*

	N=50	*N*=100	*N*=200	*N*=500	*N*=1,000	*N*=2,000	*N*=5,000	*N*=10,000	*N*=20,000	*N*=50,000
Δ	87	85	88	93	95	92	97	117	142	282
Δ^*	0.023	0.024	0.323	1.560	2.160	2.663	9.471	26.239	53.651	193.293

Comparing Fig. 1 and Table I, the performance of the approach is effected whether encoder is included (Fig. 1) or excluded (Table II) despite the fact that both scenarios are faster than the regular approach. Table II also demonstrated that reusing the encoded library sequence could decrease the computation cost dramatically.

C. K-Selection

TABLE II: *N*=1000 AND *Q*=300

	K=3	*K*=6	*K*=9	*K*=12	*K*=15	*K*=18	*K*=21
Δ^*	70	88	97	89	94	94	111
D2	20	44	68	92	110	125	140
INDEX	2804	2804	2804	2804	2804	2804	2804

In this section, the effect of applying various *k*'s over total execution time Δ^* for sequence length *N*=1000,10,000 and 100,000 and query length *Q*=300, 1000 and 3,000 is examined respectively (Table II, III and IV). For each experiment, total execution time Δ^*, D2 value and index position where the most similar sequence segment with the minimum dissimilarity have been recorded. In accordance with minimum of all window distances discussed in section A the smaller D2 value yields a more similar sequence segment, though in regard to Table II, III and IV, the results show that selecting various *k* does not have a huge impact on the performance. Also in Table IV, an exact match has been found as D2=0. Furthermore, for all *k*'s, it was found that the approach could recognize the identical index positions.

TABLE III: *N*=10,000 AND *Q*=1000

	K=3	*K*=6	*K*=9	*K*=12	*K*=15	*K*=18	*K*=21
Δ^*	107	121	119	132	118	197	223
D2	19	46	73	98	119	140	161
INDEX	38808	38808	38808	38808	38808	38808	38808

TABLE IV: *N*=100,000 AND *Q*=3000

	K=3	*K*=6	*K*=9	*K*=12	*K*=15	*K*=18	*K*=21
Δ^*	230	295	212	232	223	298	201
D2	0	0	0	0	0	0	0
INDEX	387994	387994	387994	387994	387994	387994	387994

VI. CONCLUSIONS

In this paper, an efficient method for sequence pair comparison using Huffman coding was proposed. The genome sequences were viewed as text documents and applying fixed binary codes to the base symbols allowed the comparison to be performed with the help of bitwise operations, which enhances the performance. This simple approach appears to perform well over a wide range of sequences and conditions and offers superior performance to commonly used methods. It was also shown that the exact distance degree of sequence pairs could be achieved using this approach. The experiments showed that applying this approach could speed up original D2 statistics. In addition, re-using encoded sequences would decrease execution time dramatically so that the total computation focuses on the comparison of binary codes rather than time-consuming encoding procedure.

Several attempts might be done to improve the core of the proposed approach in future. Firstly, storing and re-using the encoded sequences is the major step which helps to decrease total execution time; instead of parsing the actual sequences.

Since the encoded sequences are in binary format, less space is required for storage, yet an efficient methodology for storing and retrieving the binary codes is needed. Moreover, this approach only works for alphabet sets with uniform symbol probabilities. Since many bio-sequence comparisons are performed with different symbol probabilities, the encoding procedure is required to be improved for further applications.

ACKNOWLEDGMENTS

The authors would like to thank Dr. James M. Hogan for the comments and providing sample data.

REFERENCES

[1] S. Vinga and J. Almeida, "Alignment-free sequence compariosn-A review," *Bioinformatics*, vol. 19, no. 4, pp. 513-523, 2003.
[2] R. Fuchs, "From sequence to biology: The impact on bioinformatics," *Bioinformatics*, vol. 18, no. 4, pp. 505-506, 2002.
[3] K. Sang, J. Ren, G. Reinert, M. Deng, M. S. Waterman, and F. Sun, "New development of alignment-free sequence comparison: Measures, statistics and next generation sequencing," *Briefing in Bioinformatics Advance Access*, pp. 1-11, 2013.
[4] E. Behnam, M. S. Waterman, and A. D. Smith, "A geometric interpretation for local alignment-free," *Computational Biology*, vol. 20, no. 7, pp. 471-485, 2013.
[5] S. Foret, M. R. Kantorovitz, and C. J. Burden, "Asymptotic behaviour and optimal word size for exact and approximate word matches between random sequences," *BMC Bioinformatics*, vol. 5, pp. 1-9, 2006.
[6] S. Foret, S. R. Wilson, and C. J. Burden, "Characterizing the D2 statistic: Word matches in biological sequences," *Statistical Applications in Genetics and Molecular Biology*, vol. 8, no. 1, 2009.
[7] X. Liu, L. Wan, J. Li, G. Reinert, and M. S. Waterman, "New powerful statistics for alignment-free sequence comparison under a patterntransfermodel," *Journal of Theoretical Biology*, vol. 284, issue 1, pp. 106-116, 2011.
[8] M. R. Kantorovitz, G. E. Robinson, and S. Sinha, "A statistical method for alignment-free comparison of regulatory sequences," *Bioinformatics*, vol. 23, no. 13, pp. 249-255, 2007.
[9] G. Reinert, D. Chew, F. Sun, and M. S. Waterman, "Alignment-Free sequence comparison (I): Statistics and power," *Journal of Computational Biology*, vol. 16, no. 12, pp. 1615-1634, 2009.
[10] D. A. Huffman, "A method for the construction of minimum-redundancy codes," in *Proc. the I.R.E*, 1952, pp. 1908-1102.
[11] D. R. Mcintyre and M. A. Pechura, "Data compression using static huffman code-decode tables," *Communications of the ACM*, vol. 28, no. 6, 1985.
[12] M. Nourani and M. H. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applictaions," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 1, pp. 91-115, 2005.
[13] L. Tang, "Methods for encrypting and decrypting MPEG video data efficieny," *ACM MULTIMEDIA 96*, Carneigie Mellon University, pp. 219-229, 1996.
[14] D. E. Knuth, "Dynamic huffman coding," *Journal of Algorithm*, vol. 6, pp. 163-180, 1983.
[15] C. Burden, S. Foret, and S. Wilson, "K-word matches: An alignment-free sequence compariosn method," *International Conference on Pattern Recognition (ICPR 2008)*, IEEE Computer Society, Melbourne, Australia, pp. 235-238.
[16] J. H. Bradford, "Sequence matching with binary codes," *Information Processing Letter*, vol. 34, pp. 192-196, 1990.
[17] V. I. Levenshtein, "Binary codes capabale of correcting, deletions, insertions, and reversals," *Soviet Physics-Doklady*, vol. 10, no. 8, pp. 845-848, 1966.
[18] R. W. Hamming, "Error detectiong and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147-150.
[19] T.-J. Wu, Y.-H. Huang, and L.-A. Li, "Optimal word sizes for dissimilarity measures and estimation of the degree of sissimilarity between DNA sequences," *Bioinformatics*, vol. 21, no. 22, pp. 4125-4132, 2005.

Hossein Kamel Rahimi finished his bachelor degree in software engineering in Azad University of Iran in September 2006 and he graduated with the master degree of IT with the major of software architecture at Queensland University of Technology in Brisbane, Australia in December 2013. During his studies, he was awarded "Certificate of Dean's List for Excellent Academic Performance" in July 2013 and December 2013. He was also recognized as the top student for his academic performance from Golden Key International Honour Society in July 2013. He is interested in software optimization and bioinformatics, particularly, in sequencing and sequence comparisons of genomes.