Counting of a Degenerate Word in Random Sequences

Wei-Mou Zheng[*,1,2] and Ke-Song Liu[2]

[1]*Beijing Genomics Institute, Shenzhen (BGI-SZ), Shenzhen 518083, China*

[2]*Institute of Theoretical Physics, Academia Sinica, Beijing 100190, China*

Abstract: Overlap of words complicates the problem of word counting. The technique of the imbedded Markov chain (IMC) uses an enlarged state space to recover the Markovian property for the word counting problem. A degenerate word represents multiple non-degenerate words. An efficient way to decompose a degenerate word according to its overlapping pattern for implementing IMC is proposed.

Keywords: Imbedded Markov chain; degenerate motifs; word counting; DNA motifs.

1. INTRODUCTION

A very common task in biological sequence analysis is to identify motifs or signals in a particular sequence set. The transcription regulatory site discovery is an example. The methods of motif discovery depend on the ways for representing motifs. A popular way to characterize a motif is to use a position-specific weight matrix (PSWM). Motif matrix or PSWM measures the propensity of each base at each position of the motif. Alignment and motif matrix updating based techniques then find multiple local alignments among input sequences to discover motifs [1-9].

Another way for representing motifs is to use the consensus sequence. The motif is written as a string of IUPAC characters. Motifs are identified as over- and under-represented oligomers. Consensus sequences can be identified by combinatorial or enumerative methods, which count all possible words of a certain length in an input sequence set and then use statistics to evaluate over- or under-represented words [10, 11].

Statistical studies on the distribution of the word locations along a sequence and word frequencies have been an active field of research [12-16]. Despite the existence of known results, due to the lack of easy implementary algorithms, methods for exact computation of count statistics have been not so popular.

A commonly used description of binding site motifs is by means of degenerate consensus strings. For example, *Hin*dII binds to site `GT(A/G)(C/T)AC` [17]. An IUPAC symbol or character other than the four letters {`A`,`C`,`G` and `T`} provides a means to indicate alternative choices. Such situations are often met, e.g. GAL4 binding site $CGGN^{11}CCG$ [18], the C_2H_2 zinc finger of protein domain motif [19] and pattern hunter seed design [20]. However, the problem of exactly counting degenerate words in random sequences has been rarely discussed [21].

*Address correspondence to this author at the Beijing Genomics Institute, Shenzhen (BGI-SZ), Shenzhen 518083, China; E-mail: zheng@itp.ac.cn

There exists a powerful combinatorial method, the Goulden-Jackson cluster method [22] as well as many other mathematical results for dealing with exact word counting problems. However, they are not easily suitable for direct numerical calculation. Recently, an efficient exact motif discovery has been propoed based on a compound Poisson approximation [23]. A fast approach for exact Markovian probability functions for motif occurrences using a DFA (deterministic finite-state automata)-only approach was discussed in ref. [24]. An approach for exact computation of statistics for word occurrences in random sequences is the so-called imbedded Markov chain (IMC) technique [12, 14, 25, 26]. Using IMC, we have proposed an efficient algorithm to exactly calculate first, second moments of word counts and the probability for a word to occur at least once in random texts generated by a Markov chain [27]. Asymptotic approximations for word count statistics have been also derived. Here, after explaining the concepts of IMC, we extend the IMC to deal with counting of a degenerate word.

2. IMBEDDED MARKOV CHAIN FOR WORD COUNTING

The overlap among words introduces correlation. This complicates the word counting problem even for sequences which are generated independently at each position (M0-model). The IMC scheme enlarges the state space to 'digest' the correlation caused by word overlap. Let us explain the IMC scheme with a simple example of counting a single binary word X = `HHTHH` in alphabet {`T`,`H`} (head and tail for coin tossing). Word X can overlap with itself, sharing common substring `H` or `HH`. Appending `THH` or `HTHH` to an occurrence of X will get another occurrence of X.

For simplicity, let us consider a sequence written in `T` and `H`, where each letter occurs independently of its position with the probabilities p_0 and p_1, respectively. The Markov chain imbedded on this original M0 chain is realized by designing the ending states for sequences as the first six strings in the second column of Table **1**. Every sequence is assigned to one and only one state by successively comparing the strings one by one with the suffixes of the sequence. When the first coincidence between a string and a

1874-1363/10 2010 Bentham Open

suffix is found the state of the sequence is marked by the string. For example, the state of HHTHHH is HH (or 3 in index), while that of HHTHHT is HHT (or 2). Every state by appending one more letter T or H transits to another state as its unique poststate. For state $0 < k < 5$ there is a trivial poststate $k-1$ since five states are created by taking prefixes of HHTHH and ordering by lengths. The states, their indices, and the two poststates of each states are all listed in Table **1**. By representing the states as nodes, a Markov graph representing state transitions may be drawn. This directed graph is given by the 6×6 binary matrix t whose entries $t_{ij} = 1$ if j is the poststate of i, and $t_{ij} = 0$ otherwise. A sequence viewed with these ending states is an IMC.

Table 1. The Ending States, their Indices and Poststates of X = HHTHH. H-Poststates and T-Poststates Indicates Poststates by Appending H and T, Respectively

Index	String	H-Poststate	T-Poststate
0	HHTHH	HH	HHT
1	HHTH	HHTHH	T
2	HHT	HHTH	T
3	HH	HH	HHT
4	H	HH	T
5	T	H	T
4′	TH	HH	HT
5′	HT	TH	TT
5″	TT	TH	TT

For M2-model, state 4 should be replaced by 4′, and 5 by 5′ and 5″. Correspondingly, the poststate of 1 or 2 by appending T is 5′ (HT).

Let us introduce $P(n,l,k)$, the probability for an l long sequence which has the ending state k and n occurrences of X. The IMC implies the following recursion relations

$$P(n,l,j) = \sum_i \mu_j t_{ij} P(n,l-1,i), \quad \text{if } j \neq 0, \tag{1}$$

$$P(n,l,0) = \sum_i \mu_0 t_{i0} P(n-1,l-1,i), \tag{2}$$

where μ_j denotes the probability for the last letter of the string for state j, e.g. $\mu_0 = p_1$. The initialization of P is simply $P(0,1,4) = p_1$, $P(0,1,5) = p_0$ and $P(n,1,k) = 0$ otherwise. In principle, all statistical quantities can be derived from the recursion relations.

It is easy to generalize the procedure to deal with the problem of counting multiple words, say $X_1 = HHTHH$ and $X_2 = HHTT$. The union of the ending states from X_1 and those from X_2, ranked according to the lengths of strings, forms the set of joint ending states. Count n becomes pair n_1 and n_2 for X_1 and X_2, respectively. The binary transition matrix t and recursion relations can be also derived.

By adding extra states of single letters the above scheme work for a alphabet larger than two. Random sequences are usually described by a homogenous Markov chain model of order m (Mm). For M1-model, we need only to interpret μ_i by replacing p_i with the corresponding conditional probabilities of M1. For a general Mm-model, all the strings of length m should be included as IMC states. In a view consistent with M0, state strings shorter than m should be prefixed and split into more strings of length m as substates, but matrix t is still the same. For the example of counting X in an M2-sequence, as shown at the bottom of Table **1**, the modification is

$$4\ H \to 4'\ TH, \qquad 5\ T \to 5'\ HT, 5''\ TT.$$

The modification for recursion relations is also quite straightforward. We shall consider only M0, and discuss Mm again in last section.

3. LEAST DEVELOPED SET FOR A DEGENERATE WORD

Let us take DNA sequences as example. The fundamental nucleotides are A, C, G and T. An IUPAC symbol or character other than these four letters implies more choices, e.g., $Y = \{C,T\}$, $B = \{C,G,T\}$, and $N = \{A,C,G,T\}$. The number of fundamental letters represented by an IUPAC symbol x is the degeneracy d_x of x. The degeneracy of a string is the product of degeneracy of each letter in the string. A letter or string whose degeneracy is greater than one is called degenerate, and otherwise non-degenerate. If the letters represented by letter x are all represented by another distinct letter y, we say y contains x, and express this as $y \rhd x$ or $x \lhd y$; in this case x and y 'match' each other. Thus, $C \lhd Y \lhd B \lhd N$. Similarly, we may extend the definitions to strings. If two strings I and J of the same length satisfy $I \lhd J$ we call I a realization of J, and say that I and J 'match' each other. We shall use I also to indicate the set of all its realizations. The process going from J to I is 'developing' of J. When no confusion is caused, a realization may include the case of identity. String Z is a realization of two strings I and J if $Z \lhd I$ and at the same time $Z \lhd J$.

The overlap among words complicates the problem of word counting. String I overlaps J if and only if there exist a proper suffix of I and a proper prefix of J with the same length which are identical. The relation of overlapping is asymmetric between two different strings. For example, $I = AACCG$ overlaps $J = ACCGG$, but J does not overlap I. For a single string, self-overlap can occur. A non-degenerate string $S = s_1 s_2 \ldots s_n$ overlaps with its d-shift if $s_1 s_2 \ldots s_{n-d} = s_{d+1} s_{d+2} \ldots s_n$, which forces periodicity with period d for S; specifically $s_i = s_j$ if $i = j \pmod d$.

A degenerate string $R = r_1 r_2 \ldots r_n$ represents multiple non-degenerate strings of the same length. A direct use of all the non-degenerate words it represents is generally inefficient and unnecessary. It is our main task to find the least developed set of R which includes all the necessary overlap patterns of R's realizations. The overlap of degenerate strings is defined in the sense of match. That is, strings X and Y overlap if there exist their respective realizations X' and Y' which overlap in the sense of identity. (Correspondingly, the periodicity of a degenerate string is also in the sense of match.) When degenerate string R overlaps with its d-shift there exists $t_1 t_2 \ldots t_{n-d}$ as a realization of $r_1 r_2 \ldots r_{n-d}$ and $r_{d+1} r_{d+2} \ldots r_n$. Whenever these two substrings of R are not identical, string $R_{<d} = r_1 \ldots r_d t_1 \ldots t_{n-d}$ of length n may be created as a realization of R. (Since only the transitions *to* the target

word are involved in the word counting we ignore realization $R_{>d} = t_1 \ldots t_{n-d} r_{n-d+1} \ldots r_n$, which is resulted from a left shift. Realization $R_{>d}$ is relevant to suffixes of the target word instead of prefixes.) When $R_{<d}$ is identical to R, no new string other than R is created; this can be ignored. Generally, string $R_{<d}$ is of a lower degeneracy than R. In this way R is split into $R_{<d}$ and its complement. String $R_{<d}$ overlaps R (under d-shift). All the distinct members of $R_{<d}$ with $1 < d < n$ other than R form the 'primary set' $\mathcal{R}^{(1)}$ of R.

Replacing any letter x of a string X by a letter which contains x preserves the property of overlap of X. Thus, if two realizations of degenerate string R overlap, R must be self-overlapping (under the same d-shift). Any overlapping realization of R (under a right shift) must be also a realization of some string in the primary set $\mathcal{R}^{(1)}$ since $\mathcal{R}^{(1)}$ is created with all possible right d-shifts. It is seen then that a realization of any two strings taken from $\mathcal{R}^{(1)}$ overlaps at least one string of $\mathcal{R}^{(1)}$. Such realizations form the secondary set $\mathcal{R}^{(2)}$ of R. Realization of a string from $\mathcal{R}^{(2)}$ and another string from $\mathcal{R}^{(1)}$ gives a string of the tertiary set $\mathcal{R}^{(3)}$, realization of strings from $\mathcal{R}^{(3)}$ and $\mathcal{R}^{(1)}$ forms $\mathcal{R}^{(4)}$, and so on and so forth until no new strings are found. Any realization of two strings taken from $\mathcal{R}^{(2)}$ is included in $\mathcal{R}^{(3)}$ or $\mathcal{R}^{(4)}$, hence can be ignored. In this way all overlapping patterns of R are obtained. At the same time R is split into finer and finer partition.

After appending R to the whole set obtained above, and removing identical strings, the set, which may be called the raw set, might be redundant. To establish an exclusive partition of R according to its overlapping patterns, we should carefully interpret the meaning of degenerate strings. Recall that in the construction of $R_{<d}$ we should actually interpret R as the complement of $R_{<d}$. We sort the strings of the raw set in ascending order of degeneracy of strings and then in that of letters at successive positions. The strings with the lowest degeneracy are of the highest ranks, and are exclusive to each other. (Otherwise the 'intersection' of any two non-exclusive strings would give a realization of even lower degeneracy, and leads to a contradiction.) For exclusiveness, any higher rank strings which are contained in a lower rank string should be removed from the latter. This can be done efficiently as follows. Assume that the size of the raw set is m with R being the m-th member. We establish an $m \times m$ binary matrix I of inferiority: $I_{ij} = 1$ if the i-th string contains the j-th, and $I_{ij} = 0$ otherwise. It is always true that $I_{ij} = 0$ for $i < j$. We then replace I_{ii} by the degeneracy of string i. Staring with $i = 1$, successively at each i, whenever a nondiagonal $I_{ij} = 1$ is met, we subtract the already updated I_{jj} from I_{ii}. Finally, the diagonal entries give the real degeneracy of strings. After removing the strings with vanishing degeneracy, the remainders form the ordered least developed set $\mathcal{R}$ of R. The order implies the exclusiveness; at the same time the partition of R is complete. That is, any non-degenerate realization X of R belongs to one and only one string in the ordered list. String X is compared with the list from the top one by one. When the first time it is found that a string contains X, then X belongs to that string of the list.

Once the least developed set is obtained, we have to append all distinct prefixes of all the strings in the set to form the full set of the IMC states. In doing this the order of strings of the developed set should not distorted, and longer prefixes are always of a higher rank than shorter ones. In fact, there is an extra complexity related to degeneracy. Directly taking a prefix of R (with length k, $1 \le k < n$), we may form the least developed set of the prefixes as before for R. On the other hand, we can take prefixes of the same length k from the least developed set $\mathcal{R}$ of R. Often the two sets of prefixes are identical, but the size of the former can be larger. It is the former that should be appended in any case. An example will be given later. Finally, we append the single letter states which so far do not exist in the set of states. This ends the construction of IMC states. When a sequence is assigned an ending state by comparing its suffixes to state strings successively from top, a caution related to degeneracy is that the string of the assigned state should contain or coincide with a suffix of the sequence; this is stronger than just match, and will be explained with an example.

3.1. Examples

Example 1 ***Degenerate word R = AGRRRAG***

For simplicity, we consider alphabet of {A,G}. (Adding C and T is rather trivial.) For $R =$ AGRRRAG, overlap occurs at $d = 2$, 3. The raw set is also the least developed set of R; it consists of $R_{<2}$, $R_{<3}$ and R. The IMC states created from R are the 19 states listed in the first row of Table **2**. Their poststates by appending A and G are listed in second and third rows, respectively. Appending G to AGRRR leads to AGRRRG. Although RRRG matches AGAG, the latter does not contain the former, so AGRRRG does not belong to state AGAG. Finally, AGRRRG is identified as state G. The effective degeneracies of AGAGRAG, AGRAGAG and AGRRRAG are respectively 2, 2 and 4; their sum is 8, the total degeneracy of R.

Here, both AGAGA and AGRRA are not prefixes of $\mathcal{R}$. They belong to the least developed set of AGRRR and have to be added to the set of IMC states. (For $R' =$ AGRRR, $R'_{<2} =$ AGAGR, $R'_{<3} =$ AGRAG, $R'_{<4} =$ AGRRA. The realization of AGAGR and AGRRA further gives AGAGA. This ends the construction of the raw developed set of AGRRR, which turns to be non-redundant. The effective degeneracies of the five ordered members are respectively 1, 1, 2, 3 and 1.).

Example 2 Degenerate word R = AGRRRGA

For AGRRRGA, overlap occurs at $d = 2$, 3, 4.

$\mathcal{R}^{(1)} = \{AGAGRGA, AGRAGGA, AGRRAGA\}$,
$\mathcal{R}^{(2)} = AGAGAGA$, and

$\mathcal{R} = \{AGAGAGA, AGAGRGA, AGRAGGA, AGRRAGA, AGRRRGA\}$.

The total 23 states of the IMC, which simply consist of $\mathcal{R}$, all the prefixes of $\mathcal{R}$ and an extra single letter state G, and their poststates are listed in Table **3**.

4. CONCLUDING REMARKS

The technique of IMC treats words, either overlapping or non-overlapping, in the same simple and straightforward way. A degenerate word represents multiple non-degenerate words. A main task in implementing IMC for a degenerate word is to decompose the word into different overlapping patterns. In the final analysis, overlap is a concept in the

Table 2. The IMC States of *R* = AGRRRAG

IMC state	AGAGRAG	AGRAGAG	AGRRRAG,	AGAGRA	AGRAGA	AGRRRA,	AGAGA	AGAGR			
A-poststate	AGRAGA	AGAGA	AGA	AGRRA	AGRA	A	AGAGRA	AGAGRA			
G-poststate	AGR	AGAGR	AGR	AGAGRAG	AGRAGAG	AGRRRAG	AGAG	AGRR			
IMC state	AGRAG	AGRRA	AGRRR,	AGAG	AGRA	AGRR,	AGA	AGR,	AG,	A,	G
A-poststate	AGRAGA	AGRRRA	AGRRRA	AGAGA	AGRRA	AGRRA	AGRA	AGRA	AGA	A	A
G-poststate	AGR	AG	G	AGAGR	AGRAG	AGRRR	AGAG	AGRR	AGR	AG	G

Table 3. The IMC States of *R* = AGRRRGA

IMC state	AGAGAGA	AGAGRGA	AGRAGGA	AGRRAGA	AGRRRGA,	AGAGAG	AGAGRG	AGRAGG	AGRRAG					
A-poststate	AGRA	A	AGRRA	AGRA	A	AGAGAGA	AGAGRGA	AGRAGGA	AGRRAGA					
G-poststate	AGAGAG	AGRRAG	AGRAG	AGAG	AG	AGAGR	AGRRR	AGRR	AGR					
IMC state	AGRRRG,	AGAGA	AGAGR	AGRAG	AGRRA	AGRRR,	AGAG	AGRA	AGRR,	AGA	AGR,	AG,	A,	G
A-poststate	AGRRRGA	AGRA	AGRA	AGA	A	A	AGAGA	AGRRA	AGRRA	AGRA	AGRA	AGA	A	A
G-poststate	G	AGAGAG	AGAGRG	AGRAGG	AGRRAG	AGRRRG	AGAGR	AGRAG	AGRRR	AGAG	AGRR	AGR	AG	G

sense of identity. An overlap of a degenerate word in the sense of matching unfold the degeneracy, and split the word into two exclusive sets: one supporting the overlap and one not supporting. Overlap among words is an asymmetric relation. Correspondingly, left and right shift of a degenerate word to itself usually have different meaning.

By viewing the nodes of IMC Markov graph for M0-model as compound nodes, the same Markov graph works for an Mm-model. Thus, the framework for Mm-model is almost the same as that for M0-model. Consider Example 1 of subsection 3.1 for M3-model. It is necessary to further expand states according to non-degenerate triples. Since the effective degeneracy of an M0 IMC state string is generally smaller than its apparent degeneracy the string admits only a few ending triples. It is easy to assign any triple an M0 IMC state ($\in\{AGA, AGR, AG, A, G\}$). (Starting from single letter M0 states A and G, and then appending A, G twice according to matrix t can also obtain this triple assignment.) Appending a letter A or G, we may trace the admissible triples for each M0 IMC state. By developing the suffix of length m of any state string, say state AGRRRA, its admissible triples can be also determined. In this case, AGRRRA is developed as: AGRAAA, AGRAGA, AGRGAA, AGRGGA; their associated states are respectively AGRRRA, AGRAGA, AGRRRA, AGRRRA. Thus, the triples of AGRRRA are AAA, GAA and GGA. We see that an Mm model inherits the M0 IMC states, but at the same time every M0 state is endowed with several m-tuples for labeling substates and applying right transition rates. The recursion relations can then be extended to Mm.

Finally, we make a brief remark further on the IMC. For the above problem of counting a word of length K, a trivial M1 model is to consider an enlarged alphabet $\{A,C,G,T\}^K$. All these 4^K K-tuples may be called fine-grained states for the M1. Correspondingly, the IMC states are coarse-grained. A projection operator Π may be introduced to connect the former with the latter. The IMC satisfies $XT\Pi = X\Pi t$, or simply $T\Pi = \Pi t$, where X is an arbitrary fine state, while T and t the transition matrices for fine and coarse states, respectively [28]. The least developed set for a degenerate word provides the minimal set of coarse-grained states which preserves the Markovian property. Missing of any states in the least developed set, say AGAGA and AGRRA in Example 1 of Section 3.1, will break the constraint $T\Pi = \Pi t$. Adding states by splitting a degenerate state will preserve the Markovian property, but reduce the efficiency. (The set of coarse-grained states after adding or removing states can still be complete and exclusive in the sense of assigning any sequence an ending state.)

The size of the least developed set for a degenerate word is strongly dependent on the specific pattern of the word. As an example, for the case of the so-called structured motif like $CGGN^{11}CCG$, the size of the set of self-overlapping realizations may be determined by the method of generating function. The simplest example is ATGN^{k}CCG. There is no overlap between the two 3-mers ATG, CCG, and themselves. The generating function is

$$a^3E\left[(1+n+n^2+\ldots)a^3\right]\frac{1}{1-n}b^3 \to \frac{x^6}{1-x}E\left(\frac{x^3}{1-x}\right), \qquad E(y)\equiv\frac{1}{1-y}, \tag{3}$$

where a^3 stands for ATG, b^3 for CCG, n for N, and a full-length string is of the form $a^3n^ia^3\ldots n^ja^3n^hb^3$ with $i,j,\ldots,h\in\{0,1,2,\ldots\}$. (The function E without the factorial coefficients is close to the ordinary exponential function.) The size equals the coefficient of the term x^l, where l is the width of the degenerate string, i.e., $6+k$. Similarly, the generating function for GGGN^{k}GGG is

$$\left(a^3+a^4+\ldots\right)E\left[(n+n^2+\ldots)(a^3+a^4+\ldots)\right]\to\frac{x^3}{1-x}E\left[\frac{x^4}{(1-x)^2}\right],$$

and that for CGCN^{k}ATG is

$$(a^3+a^5+\ldots)E\left[\frac{1}{1-n}\frac{a^3}{1-a^2}\right]\frac{1}{1-n}b^3\to\frac{x^6}{(1-x)(1-x^2)}E\left[\frac{x^3}{(1-x)(1-x^2)}\right].$$

Generally, one should consider the self-overlap of the head string and the overlap of the head string with the tail. For example, the generating function for $\mathtt{CGGN}^{k}\mathtt{GGC}$ is

$$a^3E\left[(1+n+n^2+\ldots)a^3\right]\cdot\left[(1+n+n^2+\ldots)b^3+b+b^2\right]\rightarrow\frac{x^4}{1-x}E\left(\frac{x^3}{1-x}\right).$$

Our procedure to create the least developed set of IMC states for the word counting problem is valid for a general form of degenerate word, which includes the structured motif like $\mathtt{CGGN}^{11}\mathtt{CCG}$ as a special case. We have discussed only single degenerate word. The extension to multiple degenerate words does not raise any significant difficulty.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China and the National Basic Research Program of China (2007CB814800).

REFERENCES

[1] G. D. Stormo, and G. W. Hartzell,3rd, "Identifying protein-binding sites from unaligned DNA fragments", *Proc. Natl. Acad. Sci. USA.*, vol. 86, pp.1183-1187, 1989.

[2] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment", *Science*, vol. 262, pp.208-214, 1993.

[3] T. L. Bailey, and C. Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers", In *Proc. Second Int. Conf. Intel. Sys. Mol. Biol.*, Menlo Park, CA: AAAI Press, 2004, pp.28-36.

[4] S. T. Jensen, X. S. Liu, Q. Zhou, and J. S. Liu, "Computational discovery of gene regulatory binding motifs: a bayesian perspective", *Stat. Sci.*, vol. 19, pp.188-204, 2004.

[5] G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole, "Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes", *Nucleic Acids Res.*, vol. 32, pp. W199-W203, 2004.

[6] W. M. Zheng, "Genomic signal enhancement by clustering", *Commun. Theor. Phys.*, vol. 39, pp.631-634, 2003.

[7] W. M. Zheng, "Relation between weight matrix and substitution matrix: motif search by similarity", *Bioinformatics*, vol. 21, pp. 938-943, 2005.

[8] P. G. S. da Fonseca, C. Gautier, K. S. Guimarães, and M.-F. Sagot, "Efficient representation and *P*-value computation for high-order Markov motif", *Bioinformatics*, vol. 24, pp. i160-i166, 2008.

[9] A. A. Sharov and M. S. H. Ko, "Exhaustive search for over-represented DNA sequence motifs with CisFinder", *DNA Res.*, vol. 16, pp.261-273, 2009.

[10] J. Van Helden, B. Andre, and J. Collado-Vides, "Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies", *J. Mol. Biol.*, vol. 281, pp. 827-842, 1998.

[11] S. Sinha, and M. Tompa, "Discovery of novel transcription factor binding sites by statistical overrepresentation", *Nucleic Acids Res.*, vol. 30, pp. 5549-5560, 2002.

[12] J. Kleffe, and U. Langbecker, "Exact computation of pattern probabilities in random sequences generated by Markov chains", *Comput. Appl. Biosci.*, vol. 6, pp. 347-353, 1990.

[13] J. Kleffe, and M. Borodovsky, "First and second moment of count of words in random texts generated by Markov chains", *Comput. Appl. Biosci.*, vol. 8, pp. 433-441, 1992.

[14] J. C. Fu, and M. V. Koutras, "Distribution theory of runs: a Markov chain approach", *J. Am. Stat. Assoc.*, vol. 89, pp.1050-1058, 1994.

[15] S. Robin, and J. J. Daudin, "Exact distribution of word occurrences in a random sequence of letters", *J. Appl. Probab.*, vol. 36, pp. 179-193, 1999.

[16] G. Reinert, S. Schbath, and M. S. Waterman, "Probabistic and statistical properties of words: an overview", *J. Comput. Biol.*, vol. 7, pp. 1-46, 2000.

[17] P. D'haeseleer, "What are DNA sequence motifs?", *Nat. Biotechnol.*, vol. 24, pp. 423-425, 2006.

[18] J. Van Helden, A. F. Rios, and J. Collado-Vides, "Discovering regulatory elements in non-coding sequences by analysis of spaced dyads", *Nucleic Acids Res.*, vol. 28, pp. 1808-1818, 2000.

[19] C. O. Pabo, E. Peisach, and R. A. Grant, "Design and selection of novel Cys2His2 zinc finger proteins", *Annu. Rev. Biochem.*, vol. 70, pp. 313-340, 2001.

[20] B. Ma, J. Tromp, and M. Li, "PatternHunter --- faster and more sensitive homology search", *Bioinformatics*, vol. 18, pp. 440-445, 2002.

[21] J. Zhang, X. Chen, and M. Li, "Computing exact p-value for structured motif", *Lect. Notes Comput. Sci.,* vol. 4580, pp. 162-172, 2007.

[22] I. Goulden and D. M. Jackson, *Combinatorial Enumeration*, New York: John Wiley, 1983.

[23] T. Marschall, and S. Rahmann, "Efficient exact motif discovery", *Bioinformatics*, vol. 25, pp. i356-i364, 2009.

[24] P. Ribeca, and E. Raineri, "Faster exact Markovian probability functions for motif occurrences: a DFA-only approach", *Bioinformatics*, vol. 24, pp. 2839-2848, 2008.

[25] G. Nuel, "Effective *p*-value computations using Finite Markov Chain Imbedding (FMCI): application to local score and to pattern statistics", *Algorithms Mol. Biol. (BMC)*, vol. 1, pp. 5:1-5:14, 2006.

[26] J. Zhang, B. Jiang, M. Li, J. Tromp, X. Zhang, and M. Q. Zhang, "Computing exact *P*-values for DNA motifs", *Bioinformatics*", vol. 23, pp. 531-537, 2007.

[27] G. Shan, and W. M. Zheng, "Counting of oligomers in sequences generated by Markov chains for DNA motif discovery", *J. Bioinform. Comput. Biol.*, vol. 7, pp. 39-54, 2009.

[28] G. Kotsalis, and M. Dahleh, "*Model reduction of irreducible Markov chains*", Proceedings of 42nd IEEE Conference on Decision and Control, 2003, vol. 6, pp. 5727-5728,

Received: November 8, 2009 Revised: January 12, 2010 Accepted: February 16, 2010