GMR

Genome sequence compression based on optimized context weighting

M. Chen[1,2], J.J. Shao[3] and X.M. Jia[2]

[1]Department of Electronics, Yunnan University, Kunming, Yunnan, China
[2]Information Security College, Yunnan Police Officer Academy, Kunming, Yunnan, China
[3]Science and Technology College, DianChi College of Yunnan University, Kunming, Yunnan, China

Corresponding author: J.J. Shao
E-mail: minkeychen@163.com / xuemingjia@163.com

Genet. Mol. Res. 16 (2): gmr16026784
Received April 23, 2017
Accepted May 7, 2017
Published May 18, 2017
DOI http://dx.doi.org/10.4238/gmr16026784

ABSTRACT. Context weighting is an important technology for genome compression. In this study, we discuss the relationship between the weighting of context models and the weighting of the description lengths corresponding to their respective context models. It indicates that weighting of context models is equivalent to the weighting of their description lengths. With these discussions, we present the weights optimization algorithm based on the minimum description length, and suggest implementing the least-square algorithm for the optimization of the weights. The proposed optimization algorithm is used in the compression of bacterial genome sequences. The experiment results indicate that by using the proposed weights optimization method, our context weighting-based genome compression algorithm can achieve better performance than context weighting-based algorithms reported in the literature.

Key words: Genome sequence compression; Context modeling; Context weighting; Least square

INTRODUCTION

The size of the genome data set has increased exponentially in the past decade owing to the utilization of rapid DNA sequencing technology (Deorowicz and Grabowski 2011). The required memory to store such amount of genome data becomes larger and larger. The compression algorithms can enhance the storage efficiency of the genome data. Moreover, the hereditary information of a species is hidden in its genome sequence, which is not always understood sufficiently by researchers. Thus, in genome sequence compression, none of the bases in a sequence can be ignored, suggesting that genome sequence compression is advantageous.

There are two types of algorithms for DNA sequence compression: substitution algorithms and algorithms based on the entropy coding technology. Grumbach and Tahi (1993) proposed the LZ77-based algorithm *BioCompress* to compress early genome sequences. In this algorithm, if a repetitive sequence is recorded in a dictionary then it is detected, and Fibonacci coding is used to assign the codeword for the item index of this repetitive sequence in the dictionary. In coding sequences that are not contained in the dictionary, each base is encoded with a 2-bit codeword. The *BioCompress* algorithm and other similar algorithms give the basic structure of the substitution algorithms. These algorithms differ from each other in the way they encode the non-repetitive sequences or those sequences that are not in the dictionary. Grumbach and Tahi (1994) implemented a Markov model with order 2 to describe the statistic features of the bases in the non-repetitive sequences and suggested the entropy coding to encode bases in these sequences. Matsumoto et al. (2000) used the entropy coding technology based on context modeling is to compress the non-repetitive sequences, although this technology is just the complement of the substitution algorithm. Substitution algorithms cannot always ensure high compression efficiency. The performance of substitution algorithms greatly rely on their dictionaries. The size of a dictionary is directly associated with the cost for coding an item index, i.e., if the size increases then the cost for coding an item index also increases, which in turn affects the compression efficiency. Moreover, the dictionary can be configured to improve the compression performance. However, in this case, the cost to encode the dictionary cannot be ignored. Hence, the size of the dictionary should be considered to evaluate the performance of such substitution algorithm. For example, although the algorithm in (Deorowicz et al., 2013) could achieve the best compression results than any other existing algorithms, the genome database used in this algorithm as the dictionary was too large to be encoded at a reasonable cost.

The context modeling-based entropy coding algorithms, a family of lossless compression algorithms with high compression efficiency, are proposed to compress genome sequences. Tabus et al. (2003) proposed that context modeling can be used for DNA sequence compression. However, the correlations among the bases were weak because of the existence of the indels and the fracture subsequences. It means that the conditional probability distributions constructed by directly using the neighboring bases of the current base as the contexts are not suitable enough to obtain good compression efficiency. One intuitive method to tackle this problem is context weighting. Some context models with different orders are constructed, and the context model that was obtained by weighting these models is used as the coding model with improved coding performance. Some algorithms utilize this modeling strategy. Cao et al. (2007) presented a method in which the “expert

models" (XM) were constructed to describe the correlation among bases in a sequence. Each XM is actually one of the Markov models (with different orders). In the coding process, the coding model is obtained by weighting all these XMs. Pinho et al. (2009) found that the compression results by using the context models with some finite orders are better than the results obtained by using the context models with high orders. The context models with high orders are not necessary in the context modeling for genome sequence compression. However, as many as possible correlations among neighboring bases are needed to obtain a better coding model in context modeling. Context weighting can balance this conflict since different neighboring bases are used to construct different context models, which are weighted to obtain one coding model. In this way, the coding model can utilize the correlations among more neighboring bases but maintain a limited order. Thus, context weighting is one of the efficient modeling methods for genome sequence compression. However, the performance of context weighting is closely related to the weight selection. In all previous researches, the optimization of the weights was not discussed directly. In the method proposed by Pinho et al. (2009), the weights are determined based on the experience. As an improvement, in (Pinho et al., 2011), the weights are obtained with the help of a filtering operation. There are two problems here. The first is that the impulse responses of the filters are determined manually. The second is that the optimized filters do not directly produce optimized weights. In Cao et al. (2007), although the weights are related to the average code lengths of respective models, there is no method to optimize the weights directly.

In context modeling, each conditional probability distribution in a context model is estimated by using a count vector that is obtained through counting the number of various bases in the training genome sequences with the same context event. Rissanen (2001) coded a training sequence using an adaptive model, and the codelength was referred to as the description length of the training sequence for the given model. For a given sequence and a given context model, the coding performance of the model can be determined by the description length of the sequence under this model. The conditional probability distribution that corresponds to the count vector with smaller description length may lead to shorter code length in the coding process. In this way, minimizing the description length can be used as the objective of the weights optimization.

In this study, the relationship between the weighting of context models and the weighting of the description lengths is discussed, which indicates that the weighting of context models is equivalent to the weighting of the description lengths of these models. Then, the least-square algorithm is used to implement the weights optimization. The proposed weights optimization method is used to help the compression of bacterial genome sequences to improve the coding efficiency.

It is worth to notice that only the compression algorithm is discussed in this study. Our study here does not aim to develop the compression tool for genome sequence. Pinho and Pratas (2014) present MFCompress that is used for compressing FASTA. The core algorithm of MFCompress is based on the context modeling technology. For a tool, problems such as the compression of the symbol "N" and the representation of the head of one genome sequence file, should be considered. However, our work contains no such details. We pay more attention to the compression algorithm for the bacterial genome sequence. We use the symbol "A", which is similar to the operation in the algorithms, instead of "N" that is present in the sequence (Pinho et al., 2009, 2011).

MATERIAL AND METHODS

Context weighting

Genome sequences consist of four types of bases: adenine (A), thymine (T), guanine (G), and cytosine (C). Let $x_0, \ldots, x_t, \ldots, x_n$ represent a genome sequence, and $x_t \in \{A, T, G, C\}$ denote the current base to be coded. Context-based entropy coding is employed to compress the genome sequence. During context modeling, conditional probability distributions $P(x_t|x_{t-1}, \ldots, x_{t-K})$ are constructed using the past bases $x_{t-1}, \ldots, x_{t-K}$ of x_t as contexts. Each combination of $x_{t-1}, \ldots, x_{t-K}$ is a context event and K is the order of the context model. These conditional probability distributions are referred to as context model. Meanwhile, different choice of $x_{t-1}, \ldots, x_{t-K}$ or different order of K leads to different context models (different conditional probability distributions). However, when one base is being encoded, only one of these distributions in the context model is used to drive the arithmetic encoder. The conditional probability distribution which is chosen to be used is determined by the combination of $S_c = x_{t-1}, \ldots, x_{t-K}$ for the current base x_t (actually, by its current context event S_c). Many distributions are considered for coding x_t when different context models are constructed. During this period, context weighting is suggested to utilize these distributions to form one coding distribution by weighting.

Let $P(x_t \mid s^c_i)$ denote the conditional probability distribution corresponding to the context event s^c_i in the ith context model, and w_i denote the value of its weight for context weighting. N denotes the number of context models participated in weighting. Then, context weighting, which aims to obtain the coding model with better coding efficiency, can be represented as Equation 1:

$$P(x_t|S)=\sum_{i=1}^{N} w_i * P(x_t|s^c_i) \qquad \text{(Equation 1)}$$

where $P(x_t|S)$ is the conditional probability distribution used to drive the arithmetic encoder to assign the codeword for the current base $x_t|$. However, in practice, these conditional probability distributions, including $P(x_t \mid s^c_i)$ and $P(x_t|S)$, are known in advance. They should be estimated by using their corresponding count vectors to calculate probabilities in these distributions. In this estimation procedure, the count vector v, which consists of counting number n_i, is obtained by counting the past bases encoded, and n_i represents the number of symbols which holds value i. Then probability can be estimated by calculating $n_i / sum(n_i)$.

Meanwhile, in practice, the weighting of context models is actually implemented by weighting count vectors that corresponds to the conditional probability distributions $P(x_t \mid s^c_i)$ respectively. An example below is employed to explain this procedure.

Consider a context-weighting scheme with two models. Let s^C_1 and s^C_2 denote the current context events in their respective models. The corresponding conditional probability distributions are $P(x_t \mid s^c_1)$ and $P(x_t \mid s^c_2)$. Let w_1 and w_2 denote the weights of $P(x_t \mid s^c_1)$ and $P(x_t \mid s^c_2)$, with the weights satisfying $w_1 + w_2 = 1$. Two count vectors CV_1 and CV_2, which correspond $P(x_t \mid s^c_1)$ and $P(x_t \mid s^c_2)$, respectively, are listed on the left side of Equation 2. Multiplied by weights, cv_1 and cv_2 become the vectors listed on the right side of Equation 2.

$$\begin{array}{lcccc} & 0(A) & 1(T) & 2(G) & 3(C) \\ \mathbf{CV_1}: & n_0 & n_1 & n_2 & n_3 \\ \mathbf{CV_2}: & m_0 & m_1 & m_2 & m_3 \end{array} \qquad \begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ w_1\mathbf{CV_1}: & w_1n_0 & w_1n_1 & w_1n_2 & w_1n_3 \\ w_2\mathbf{CV_2}: & w_2m_0 & w_2m_1 & w_2m_2 & w_2m_3 \end{array} \quad \text{(Equation 2)}$$

After weighting, the count vector cv corresponding to $P(x_t|S)$ has the form given in Equation 3.

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ \mathbf{CV}: & w_1n_0+w_2m_0 & w_1n_1+w_2m_1 & w_1n_2+w_2m_2 & w_1n_3+w_2m_3 \end{array} \quad \text{(Equation 3)}$$

Then $P(x_t|S)$ can be estimated by Equation 4:

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ P(x_t|S): & \dfrac{w_1n_0+w_2m_0}{w_1V_1+w_2V_2} & \dfrac{w_1n_1+w_2m_1}{w_1V_1+w_2V_2} & \dfrac{w_1n_2+w_2m_2}{w_1V_1+w_2V_2} & \dfrac{w_1n_3+w_2m_3}{w_1V_1+w_2V_2} \end{array} \quad \text{(Equation 4)}$$

where $V_1 = n_0 + n_1 + n_2 + n_3$ denotes the total number of training bases in CV_1 and $V_2 = m_0 + m_1 + m_2 + m_3$ denotes the total number of training bases in CV_2.

According to Chen et al. (2013), L_1, which is the description length of the count vector CV_1, can be calculated by Equation 5:

$$L_1 = \log(V_1 - 1)! - \sum_{i=0}^{3} \log(n_i - 1)! - \log(4-1)! \quad \text{(Equation 5)}$$

L_2, which is the description length of the count vector CV_2, can be calculated similarly.

To calculate the description length by Equation 5, the term log $(V_1–1)!$ or log $(n_i–1)!$ can be represented as log $V_1!$ - log V_1 or log $n_i!$ - log n_i. When Stirling's formula

$$n! \approx n^{(n+\frac{1}{2})} e^{-n} \sqrt{2\pi} \quad \text{(Equation 6)}$$

is used to approximately calculate factorials and when log is the natural logarithm, L_1 and L_2 can be calculated using Equations 7 and 8, respectively:

$$L_1 = V_1 \log V_1 - n_0 \log n_0 - n_1 \log n_1 - n_2 \log n_2 - n_3 \log n_3 - \frac{1}{2} \log \frac{V_1}{n_0 n_1 n_2 n_3} - \sigma \quad \text{(Equation 7)}$$

and

$$L_2 = V_2 \log V_2 - m_0 \log m_0 - m_1 \log m_1 - m_2 \log m_2 - m_3 \log m_3 - \frac{1}{2} \log \frac{V_2}{m_0 m_1 m_2 m_3} - \sigma \quad \text{(Equation 8)}$$

where $\sigma = \log 3! + 3 \log \sqrt{2\pi}$ is a constant. The description length L of the count vector CV (Equation 3) can also be calculated as Equation 5. To simplify our representation, let C_i denote

the ratio of the weighted total number of training bases in the count vector CV_i [corresponding to $P(x_t \mid s^c_i)$] to the total number of bases in the weighted count vector CV. Let $t^{(i)}_j$ denote the ratio of the weighted total number of training bases with value j in the count vector CV_i [corresponding to $P(x_t \mid s^c_i)$] to the total number of bases with value j in the weighted count vector CV. For instance, parameters c_1, $t^{(1)}_2$, and $t^{(1)}_2$ are calculated as follows:

$$c_1 = \frac{w_1 V_1}{w_1 V_1 + w_2 V_2} \qquad t_2^{(1)} = \frac{w_1 n_2}{w_1 n_2 + w_2 m_2} \qquad t_2^{(2)} = \frac{w_2 m_2}{w_1 n_2 + w_2 m_2}$$

when all weights w_i are given, all parameters c_i and $t^{(i)}_j$ are determined. The description length L, which corresponds to the count vector CV (Equation 3), can also be calculated as in Equation 9, in which, the first line actually equals to $w_1 L_1 + w_2 L_2$.

$$\begin{aligned} L = {} & w_1 (V_1 \log V_1 - \sum_{j=0}^{3} n_j \log n_j - \frac{1}{2} \log \frac{V_1}{\prod_{j=0}^{3} n_j} - \sigma) + w_2 (V_2 \log V_2 - \sum_{j=0}^{3} m_j \log m_j - \frac{1}{2} \log \frac{V_2}{\prod_{j=0}^{3} m_j} - \sigma) \\ & + w_1 V_1 \log \frac{w_1}{c_1} - w_1 \sum_{j=0}^{3} n_j \log \frac{w_1}{t_j^{(1)}} - \frac{w_1}{2} \log \frac{\prod_{j=0}^{3} t_j^{(1)}}{c_1 {w_1}^2} + w_2 V_2 \log \frac{w_2}{c_2} - w_2 \sum_{j=0}^{3} m_j \log \frac{w_2}{t_j^{(2)}} - \frac{w_2}{2} \log \frac{\prod_{j=0}^{3} t_j^{(2)}}{c_2 {w_2}^2} \end{aligned} \quad \text{(Equation 9)}$$

Let Q denote the last line in Equation 9, then

$$Q = w_1 V_1 \log \frac{w_1}{c_1} - w_1 \sum_{j=0}^{3} n_j \log \frac{w_1}{t_j^{(1)}} - \frac{w_1}{2} \log \frac{\prod_{j=0}^{3} t_j^{(1)}}{c_1 {w_1}^2} + w_2 V_2 \log \frac{w_2}{c_2} - w_2 \sum_{j=0}^{3} m_j \log \frac{w_2}{t_j^{(2)}} - \frac{w_2}{2} \log \frac{\prod_{j=0}^{3} t_j^{(2)}}{c_2 {w_2}^2} \quad \text{(Equation 10)}$$

and L can be represented as

$$L = w_1 L_1 + w_2 L_2 + Q \quad \text{(Equation 11)}$$

Q can be viewed as some kind of weighting cost. If the value of Q is much smaller than the value of $w_1 L_1 + w_2 L_2$, then the weighting cost Q can be ignored in the optimizing procedure. In this study, many experiments are executed to test whether this cost could be omitted. For coding every base, the current weighting cost is calculated first and is compared with the value of $w_1 L_1 + w_2 L_2$. Fortunately, the experiment results indicate that the value of weighting cost is smaller than $w_1 L_1 + w_2 L_2$. With increasing number of bases coded, the value of $w_1 L_1 + w_2 L_2$ becomes larger, but the value of weighting cost remains in a reasonable area. In these experiments, the maximum value of Q equals to 48.95 compared to the value of $w_1 L_1 + w_2 L_2$, which is 112,465.34. The minimum value of Q equals to 0.0876, and the value of $w_1 L_1 + w_2 L_2$ equals to 1,024,584.28. Consequently, the weighting cost provides little influence for context weighting. These experiments and representation (Equation 11) imply that context weighting is equivalent to the weighting of description lengths of count vectors, which corresponds to distributions participated in weighting. Moreover, optimization of the

weights can be achieved by minimizing the description length L of the weighted count vector CV. Based on this observation, the method for the optimization of the weights can be described as follows: for the given count vectors CV_1 and CV_2, which are obtained from the past bases in a sequence, find the weights that can minimize the description length L for the count vector CV. In the next section, we discuss the implementation of our weights optimization method employing the least square algorithm to help achieving the optimization of the weights.

Weights optimization

From Equation 11, we found that the description length L can be represented approximately as $L \approx w_1L_1 + w_2L_2$.

On the basis of this approximation, the least-square algorithm can be used to optimize weights w_1 and w_2. Let $L = (L_1, L_2)^T$ denote the observing vector consisting of description lengths from count vectors that participated in weighting. Let $W = (w_1, w_2)^T$ denote the corresponding weights vector. Then $\hat{L}$, which is considered as the estimated value of the description length L, be represented in the vector form as

$$\hat{L} = \mathbf{L}^T\mathbf{W} \qquad \text{(Equation 12)}$$

The objective of our optimization is to minimize $\hat{L}$. However, in order to minimize $\hat{L}$ by using the least-square algorithm, its corresponding ideal value L^* should be given in advance.

Moreover, according to Wu et al. (2011), the description length $\hat{L}$ can also be represented as:

$$\hat{L} = (w_1V_1 + w_2V_2)H(x_t \mid S) + \Delta \qquad \text{(Equation 13)}$$

where $H(x_t|S)$ is the entropy of the conditional probability distribution $P(x_t|S)$ and Δ represents the model cost. The value of $(w_1V_1+w_2V_2)H(x_t|S)$ represents the ideal code length for coding these $w_1V_1+w_2V_2$ bases. However, this code length cannot be achieved in practice. Thus, we choose $L^* = (w_1V_1+w_2V_2)H(x_t|S)$ as the ideal value of the description length $\hat{L}$ in our weights optimization process. Apparently, $L^* < \hat{L}$. It is impossible to obtain shorter description length than L^*. Therefore, L^* can be the objective of the minimization of $\hat{L}$. Let

$$e = |\hat{L} - L^*| \qquad \text{(Equation 14)}$$

denote the error between $\hat{L}$ and L^*. Minimizing the squared error e^2 is equivalent to minimizing $\hat{L}$, which can be described as

$$\min\{f(\mathbf{W}) = e^2 = |\hat{L} - L^*|^2\} \qquad \text{(Equation 15)}$$

where f(W) is the cost function that is related to W. The minimization of f(W) can be obtained by solving the following equations.

$$\begin{cases} \dfrac{\partial f(\mathbf{W})}{\partial w_i} = 0 \quad i = 1,2 \\ \sum_{i=1}^{2} w_i = 1 \end{cases} \qquad \text{(Equation 16)}$$

The solution of Equation 16 can directly be represented as:

$$\mathbf{W} = \mathbf{R}^{-1} \times \mathbf{d} \qquad \text{(Equation 17)}$$

where, R is the correlation matrix of the observing vector L, and d is the correlation vector between L and L^*. They can be written as

$$\mathbf{R} = \mathbf{L} \times \mathbf{L}^T, \quad \mathbf{d} = \mathbf{L} \times L^* \qquad \text{(Equation 18)}$$

After solving these equations, the optimized weights can be obtained. The coding distribution can also be obtained by using Equation 4.

Since the entropy $H(x_t|S)$ is not known, L^* cannot be calculated directly. In practice, a positive value as small as possible can be used instead. The optimization of weights for coding x_t is based on count vectors that are obtained by counting the past bases $x_0, \ldots, x_{t-1}$ under the same context events as the current context events in different context models. When x_t is coded, these count vectors are actually updated by adding one to the corresponding counts. It implies that the corresponding weights could also be updated after coding x_t. However, such frequent updating leads to high computational complexity. To simplify our algorithm, the weights in this study are not updated after coding every base but are updated until a given number of bases are coded.

Steps of the proposed genome sequence compression algorithm are listed as follows:

Step 1: Construct context models to be weighted by using training sequences (which can be viewed as the past bases).

Step 2: The weights are initialized to equal weights.

Step 3: Use these weights to obtain the weighted distribution for the coding of the current base. If all bases are coded, the algorithm ends; otherwise, go to step 4.

Step 4: Check whether weights should be updated. If updating is required, go to step 5; otherwise, go back to step 3 to code the next base.

Step 5: Use Equation 7 to calculate L_i, use Equations 16 and 17 to update weights, and go back to step 3.

RESULTS

In this study, the proposed optimization algorithm is applied to compress bacterial genome sequences. Context models with different orders are constructed. The genome

sequence *NC_020409* (size 334,969,024 bases) is used as the training sequence to initialize these context models. Four genome sequences *NC_013131*, *NC_014318*, *NC_004691*, and *NC_004532* are used as test sequences.

In algorithm (Cao et al., 2007), the weights for context weighting are determined by the average code lengths of their respective context models. The past 20 bases are encoded respectively by using each context model participated in weighting; the average code length of each model is used as a model parameter reflecting the characteristics of that model. The weight of the corresponding model is calculated on the basis of this model parameter. In experiment 1, we implemented the algorithm used by Cao et al. (2007), and the same method was used to calculate weights. However, in our implementation, the description lengths of respective count vectors are used as the model parameters to calculate the weights instead using the average code lengths. To test the coding efficiency of this substitution, four context models with order 2, 4, 10, and 16 are constructed and their corresponding weights are calculated using the description lengths as the model parameters. Genome sequences *NC_013131* and *NC_014318* are used as test sequences. After coding, the compression results, represented by the term bpb (bits per base), are listed in Table 1. For comparison, the compression results by using the average code lengths as the model parameters are also listed in Table 1.

Table 1. Comparison of coding efficiency with two types of model parameters.

	Size (bases)	bpb results by using the algorithm in (Cao et al., 2007) with different types of model parameters.	
		Average code length (bpb)	Description length (bpb)
NC_013131	10,467,782	1.7922	1.7904
NC_014318	10,236,715	1.7695	1.7663

bpb, bits per base (bits/base).

From Table 1, it can be found that the compression results obtained using the description lengths as the model parameters are similar or slightly better than the results obtained using the average code lengths as the model parameters. It implies that the description length also reflects the characteristics of its corresponding model and can be used to determine the weight in context weighting. However, the weights obtained using both type of model parameters in experiment 1 are not optimized. In the following experiments, the proposed weights optimization algorithm will be used to improve the compression performance of the genome sequences.

In experiment 2, we test compression efficiency with different updating periods. Three context models with orders 2, 4, and 8 are constructed and the compression results with different updating periods (50 and 100 bases) are listed in Table 2. Genome sequences *NC_004691* and *NC_004532* are used as test sequences.

Table 2. Compression results with different updating periods.

Sequences	Size (bases)	Bits per base (bpb) under different updating periods		
		40 bases	50 bases	100 bases
NC_004691	9,267,221	1.7349	1.7363	1.7422
NC_004532	20,063	1.6374	1.6417	1.6658

From Table 2 it can be derived that the compression results with the updating period 50 are better than the results with the updating period 100 for both long and short sequences.

However, the compression results with updating period 40 and 50 are close, indicating that decrease in the updating period does not always provide significant compression efficiency improvement. However, short updating period leads to higher computational burden. Therefore, in our experiments, updating period is set to 50.

The proposed algorithm in experiment 3 is used for weight optimization, and the weighted context model is applied to compress bacterial genome sequences. The same four context models as in experiment 1 are constructed, which were also used by Pinho et al. (2011). Bacterial genome sequences *NC_013131* and *NC_014318* are used as test sequences. Our compression results are listed in Table 3. For comparison, the compression results obtained by using the algorithm XM (Cao et al., 2007) and the algorithm FCM (Pinho et al., 2011) are also listed in Table 3.

Table 3. Compression results by different algorithms.

Sequences	Size (bases)	Bits per base (bpb) by different algorithms		
		XM by Cao	FCM by Pinho	Proposed
NC_013131	10,467,782	1.7922	1.779	1.7216
NC_014318	10,236,715	1.7695	1.739	1.7013

Our algorithm can produce better compression results than either XM or FCM, since the optimization of the weights can lead to higher compression efficiency, as given in Table 3. It also implies that context weighting based on minimizing the description length of past observations can reduce the final code length in the coding of a genome sequence.

Apparently, the proposed algorithm can produce promising results in compressing genome sequences, and the optimization algorithm also ensures that optimized weights are obtained. The designed objective of optimized context weighting is achieved. Moreover, because of the execution of the weight optimization, the execution time for our algorithm is higher than the previous algorithms (Pinho et al., 2009, 2011). Here, the time needed for compressing *NC_013131* is 56.7 s at the platform Intel Core i3 2.0G and 2 GB RAM.

DISCUSSION

In this study, weight optimization algorithm based on the minimum description length is proposed. Context weighting can be implemented by weighting the description lengths of past observations under their respective models. The least-square algorithm is employed to optimize the weights for context weighting. The optimized weights are then used to improve the compression efficiency of genome sequences. Experimental results indicate that the proposed algorithm can lead to better results than the previous reported algorithms.

Conflicts of interest

The authors declare no conflict of interest.

ACKNOWLEDGMENTS

Research supported by the Natural Science Foundation of Yunnan Province under Grant (#2013FD042 and #2015YC003).

REFERENCES

Cao MD, Dix TI, Allison L and Mears C (2007). A simple statistical algorithm for biological sequence compression. Proceedings of the 2007 Data Compression Conference (DCC 2007). Snowbird.

Chen M, Chen J and Guo M (2013). Affinity propagation for the context quantization. *Adv. Mat. Res.* 791: 1533-1536. http://dx.doi.org/10.4028/www.scientific.net/AMR.791-793.1533

Deorowicz S and Grabowski S (2011). Robust relative compression of genomes with random access. *Bioinformatics* 27: 2979-2986. http://dx.doi.org/10.1093/bioinformatics/btr505

Deorowicz S, Danek A and Grabowski S (2013). Genome compression: a novel approach for large collections. *Bioinformatics* 29: 2572-2578. http://dx.doi.org/10.1093/bioinformatics/btt460

Grumbach S and Tahi F (1993). Compression of DNA sequences. Proceedings of the Data Compression Conference, (DCC-93). Snowbird, 340-350.

Grumbach S and Tahi F (1994). A new challenge for compression algorithms: genetic sequences. *Inf. Process. Manage.* 30: 875-866. http://dx.doi.org/10.1016/0306-4573(94)90014-0

Matsumoto T, Sadakane K and Imai H (2000). Biological sequence compression algorithms. *Genome Inform. Ser. Workshop Genome Inform.* 11: 43-52.

Pinho AJ and Pratas D (2014). MFCompress: a compression tool for FASTA and multi-FASTA data. *Bioinformatics* 30: 117-118. http://dx.doi.org/10.1093/bioinformatics/btt594

Pinho AJ, Neves AJR, Bastos CAC and Ferreira PJSG (2009). DNA coding using finite-context models and arithmetic coding. Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. Taipei, 1280-1284.

Pinho AJ, Pratas D and Ferreira PJSG (2011). Bacteria DNA sequence compression using a mixture of finite-context models. 2011 IEEE Statistical Signal Processing Workshop. IEEE, Nice.

Rissanen J (2001). Strong optimality of the normalized ML models as universal codes and information in data. *IEEE Trans. Inf. Theory* 47: 1712-1717. http://dx.doi.org/10.1109/18.930912

Tabus I, Korodi G and Rissanen J (2003). DNA sequence compression using the normalized maxi-mum likelihood model for discrete regression. Proceedings of the Data Compression Conference (DCC-2003). Snowbird, 253-263.

Wu X, Zhai G, Yang X and Zhang W (2011). Adaptive sequential prediction of multidimensional signals with applications to lossless image coding. *IEEE Trans. Image Process.* 20: 36-42. http://dx.doi.org/10.1109/TIP.2010.2061860