Optimized Context Weighting for the Compression of the Un-repetitive Genome Sequence Fragment

Min Chen¹,³ · Rui Li² · LiJun Yang³

Published online: 12 March 2018
© The Author(s) 2018

Abstract The details of context weighting is discussed first and some formulas for describing the progress of context weighting is also given. During the weighting, the weighting cost can influence the performance of context modeling entropy coding. However, the context weighting rely on corresponding counting vectors which are joined in weighting. Meanwhile, the equivalence between context weighting and the weighting of description length implies that the weights can be optimized. The corresponding methods are discussed in details in our paper. At last, the proposed approach is used to compress these simple genome sequences. A large number of experiment results indicate that the corresponding compression results is better than the results of other existing algorithms.

Keywords Context weighting · Weighting cost · Genome compression · Optimized algorithm

1 Introduction

Due to the wide application of the rapid DNA sequencing technology, the storing memory for saving these genome data is becoming larger and larger during the past decade [1]. In order to store the genome data more efficiently, some compression algorithms are proposed
to enhance the storage efficiency. By reducing redundancies contained among bases of a genome sequence, the compression efficiency can be improved. In the research of genome sequence compression, previous researchers focus mainly on the method by using existing compression algorithms to reduce the requirement for storing genome sequence data. However, it is not said that all previous compression algorithms are suitable for genome sequence compression. Although the information have not been understood sufficiently which is contained in different species’ genome sequence. It implies that which base can be dropped or ignored in the compression process is unknown yet. In other words, only those lossless compression algorithms can be suggested for this application.

Broadly speaking, there are three types of genome compression algorithms that can be employed to achieve our compression objective. The first is the substitutional algorithm based on LZ77. The second is the algorithm referred to as referential algorithm. The last is the compression algorithm based on the entropy coding technology. In Grumbach and Tahi [2], BioCompress which is based on LZ77 was proposed to compress genome sequences and the corresponding modified algorithms [3, 4] illustrate the basic pattern of those genome sequence compression algorithms. The subsequent algorithms paid more attention to improve the method for coding those no-repetitive sequences and the sequences which are not in the dictionary. In Grumbach and Tahi [3], the non-repetitive sequences are compressed by the entropy coding with 2-order Markov model. In Deorowicz et al. [4], a further improvement was proposed that the context modeling based entropy coding is used to substitute the Markov model used in [3] for coding those no-repetitive sequences. In fact, these algorithms rely greatly on their dictionaries. The cost for coding contain the cost for coding the dictionary. Moreover, to improve compression performance, the dictionary can be initialized. But it can not achieve better efficiency.

In recent years, the “referential genome sequence compression algorithms” [5, 6] have attracted the attention of more and more researchers for the reason that these algorithms can achieve high compression efficiency which both substitutional algorithms and entropy coding technologies cannot attain in the mammalian genome sequence compression. Meanwhile, these algorithms cooperate with the rapid DNA sequencing technology very well. When a new genome sequence is ready to be compressed, it is aligned with a reference sequence firstly. The location where the difference between the bases in the sequence to be coded and those in the reference sequence occurs is searched. Then the location information where the bases in these two sequences start to differ and the length which denotes the number of different bases are encoded. For decoding, both the sender and the receiver should have this reference sequence, with the encoded information for those differences, the receiver can reconstruct the objective sequence. In this case, the content need to be coded and transmitted will be much less. Consequently, high compression efficiency can be achieved. However, these algorithms rely heavily on the amount of the repetitive sequences contained in the genome sequences. It implies that various compression results can be obtained when genome sequences from different species are input to such an algorithm. For some species, including human, their genome sequence contain many repetitive sequences, the corresponding compression results are excellent. Moreover, if the accuracy of the repetitive sequence alignment is enhanced, the compression efficiency can further be increased. Based on this reason, many algorithms [7–9] pay more attention to improve the accuracy of alignment. Especially in [9], the compression ratio for human genome sequences can reach 80 times. However, these results rely on large number of repetitive sequences contained in the objective sequences. Actually, there are almost 90% repetitive sequences in a complete human genome sequence, which can ensure the algorithm in [9] maintaining an high compression ratio. In this case, the
content need to be coded is small enough and the compression efficiency is enhanced. But for some simple species, such as microorganisms, there are only about 20% or less repetitive sequences in a complete sequence, subsequently, the referential algorithm cannot play its role sufficiently. Thus, in those previous researches [10–13] on the microorganism genome sequence compression, the context modeling based entropy coding technology is suggested instead of the referential algorithm.

The work in [14] illustrate the method for compressing the genome sequence with Context modeling and give the approaches to select which bases to be conditions for constructing the conditional probability distributions. Specially for the work by [21], the “expert models” (XM) are one of efficient context modeling methods. Every XM comes from different models, such as the copy model and the Markov models (with order 1 and 2). Then the context weighting is suggested to merge various context models to construct the probability distribution which is used to assign the code words, such as the words in [14, 15]. However, how to determine the weights is one problem to be tackled.

Actually, during context weighting, the conditional probability distributions are estimated by using their corresponding counting vectors. The description length is one important parameter to illustrate the statistic performance of this context model. It means that if the description length of the counting vector weighted is small, the weighting operation can be considered to be correct. In [16, 17], the approaches to calculate the description are proposed. For inspiring, the weights can also be optimized under the criterion of minimizing the description length.

We discuss the context weighting first and give the description of the weighting operation. Then the weighting cost is discussed in details. The relation between context weighting and the costs are described by using some formulas and tables. At last, the proposed method is used to compress some genome sequences.

2 Methods

2.1 Context Weighting

Genome sequence $x_i, \ldots, x_0$ holds the alphabet $x_i \in Alphabet = \{A, T, G, C\}$. It should be note that only the genome sequences are our objective sequences. Therefore, the uracil (N) is not considered in our compression algorithm. During the context modeling, $P(x_i|x_{t-1}, \ldots, x_{t-K})$ are constructed with the structure of $x_{t-1}, \ldots, x_{t-K}$ for $x_i$ with order $K$. Let $P(x_i|s'_j)$ denote probability distribution in the $i_{th}$ context model with context event $s'_j$ and weights $w_j$. $N$ is the total number of distributions joined in weighting. Then, the context weighting is represented as (1)

$$P(x_i|S) = \sum_{j=1}^{N} w_j * P(x_i|s'_j)$$

$P(x_i|S)$ is the conditional probability distribution weighted for coding. In practice, $P(x_i|S)$ is actually from the estimation by calculating its corresponding counting vectors joined in weighting. For example:

Let $P(x_i|s'_1)$ and $P(x_i|s'_2)$ denote two conditional probability distributions for two counting vectors $CV_1$ and $CV_2$ which are given below. The weights $w_1$ and $w_2$ for $P(x_i|s'_1)$ and $P(x_i|s'_2)$ respectively satisfy $w_1 + w_2 = 1$. Two count vectors $CV_1$ and $CV_2$ are listed
on the left side of (2). Multiplied by weights, $\mathbf{CV}_1$ and $\mathbf{CV}_2$ become the vectors listed on the right side of (2).

$$
\begin{align*}
\mathbf{CV}_1: & \quad 0(A) \quad 1(T) \quad 2(G) \quad 3(C) \\
& \quad n_0 \quad n_1 \quad n_2 \quad n_3 \quad w_1 \mathbf{CV}_1: \quad w_1 n_0 \quad w_1 n_1 \quad w_1 n_2 \quad w_1 n_3 \\
\mathbf{CV}_2: & \quad 0 \quad 1 \quad 2 \quad 3 \\
& \quad m_0 \quad m_1 \quad m_2 \quad m_3 \quad w_2 \mathbf{CV}_2: \quad w_2 m_0 \quad w_2 m_1 \quad w_2 m_2 \quad w_2 m_3
\end{align*}
$$

(2)

Then the distribution $P(x_i|S)$ can be re-write as (3).

$$
P(x_i|S) : \begin{cases}
0 & \frac{w_1 n_0 + w_2 m_0}{w_1 V_1 + w_2 V_2} \\
1 & \frac{w_1 n_1 + w_2 m_1}{w_1 V_1 + w_2 V_2} \\
2 & \frac{w_1 n_2 + w_2 m_2}{w_1 V_1 + w_2 V_2} \\
3 & \frac{w_1 n_3 + w_2 m_3}{w_1 V_1 + w_2 V_2}
\end{cases}
$$

(3)

And $P(x_i|S)$ is calculated by (4)

$$
P(x_i|S) : \begin{cases}
0 & \frac{w_1 n_0 + w_2 m_0}{w_1 V_1 + w_2 V_2} \\
1 & \frac{w_1 n_1 + w_2 m_1}{w_1 V_1 + w_2 V_2} \\
2 & \frac{w_1 n_2 + w_2 m_2}{w_1 V_1 + w_2 V_2} \\
3 & \frac{w_1 n_3 + w_2 m_3}{w_1 V_1 + w_2 V_2}
\end{cases}
$$

(4)

where $V_1 = n_0 + n_1 + n_2 + n_3$ denotes the total number of training bases in $\mathbf{CV}_1$ and $V_2 = m_0 + m_1 + m_2 + m_3$ denotes the total number of training bases in $\mathbf{CV}_2$.

According to Chen and Chen [18], $L_1$, the description length of $\mathbf{CV}_1$, can be calculated by (5):

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

(5)

$L_2$ is also the description length of $\mathbf{CV}_2$ and be calculated similarly.

When Stirling’s formula (6)

$$
n! \approx n^{(n+1)/2} e^{-n} \sqrt{2\pi}
$$

(6)

is used to approximately calculate the factorials and when log is the natural logarithm, $L_1$ and $L_2$ can be calculated with (7) and (8) respectively:

$$
L_1 = V_1 \log V_1 - n_0 \log n_0 - n_1 \log n_1 - n_2 \log n_2 - n_3 \log n_3 - \frac{1}{2} \log \frac{V_1}{n_0 n_1 n_2 n_3} + \sigma
$$

(7)

and

$$
L_2 = V_2 \log V_2 - m_0 \log m_0 - m_1 \log m_1 - m_2 \log m_2 - m_3 \log m_3 - \frac{1}{2} \log \frac{V_2}{m_0 m_1 m_2 m_3} + \sigma
$$

(8)

where $\sigma = - \log 3! - 3 \log \sqrt{2\pi}$ is a constant. The description length $L$ of the count vector $\mathbf{CV}$ can also be calculated similar to (5). To simplify our representation, let $c_j$ denote the ratio between the weighted total number of training bases in the count vector $\mathbf{CV}_j$ (corresponding to $P(x_i|S_j)$) and the total number of bases in the weighted count vector $\mathbf{CV}$. Let
\( t_j^{(i)} \) denote the ratio between the weighted total number of training bases with value \( j \) in the count vector \( CV_i \) (corresponding to \( P(x_i|s^i) \)) and the total number of bases with value \( j \) in the weighted count vector \( CV \). For instance, parameters \( c_1, t_2^{(1)} \) and \( t_2^{(1)} \) are calculated as follows:

\[
\begin{align*}
\hat{c}_1 &= \frac{w_1 V_1}{w_1 V_1 + w_2 V_2},
\hat{t}_2^{(1)} &= \frac{w_1 n_2}{w_1 n_2 + w_2 m_2},
\hat{t}_2^{(2)} &= \frac{w_2 m_2}{w_1 n_2 + w_2 m_2}.
\end{align*}
\]

Apparently, when all weights \( w_i \) are given, all parameters \( c_i \) and \( t_j^{(i)} \) are determined. The description length \( L \) can also be calculated with the equation in (9), in which the first line actually equals to \( w_1 L_1 + w_2 L_2 \).

\[
L = w_1 \left( V_1 \log V_1 - \sum_{j=0}^{3} n_j \log n_j - \frac{1}{2} \log \frac{V_1}{\prod_{j=0}^{3} n_j} - \sigma \right) + w_2 \left( V_2 \log V_2 - \sum_{j=0}^{3} m_j \log m_j - \frac{1}{2} \log \frac{V_2}{\prod_{j=0}^{3} m_j} - \sigma \right) + w_1 \log \frac{w_1}{c_1}
\]

\[
- w_1 \sum_{j=0}^{3} n_j \log \frac{w_1}{t_j^{(1)}} - \frac{w_1}{2} \log \frac{\prod_{j=0}^{3} t_j^{(1)}}{c_1 w_1^2} + w_2 V_2 \log \frac{w_2}{c_2} - w_2 \sum_{j=0}^{3} m_j \log \frac{w_2}{t_j^{(2)}} - \frac{w_2}{2} \log \frac{\prod_{j=0}^{3} t_j^{(2)}}{c_2 w_2^2}
\]

(9)

Let \( Q \) denote the last two lines in (9), then

\[
\begin{align*}
Q &= w_1 V_1 \log \frac{w_1}{c_1} - w_1 \sum_{j=0}^{3} n_j \log \frac{w_1}{t_j^{(1)}} - \frac{w_1}{2} \log \frac{\prod_{j=0}^{3} t_j^{(1)}}{c_1 w_1^2} + w_2 V_2 \log \frac{w_2}{c_2} - w_2 \sum_{j=0}^{3} m_j \log \frac{w_2}{t_j^{(2)}} - \frac{w_2}{2} \log \frac{\prod_{j=0}^{3} t_j^{(2)}}{c_2 w_2^2}.
\end{align*}
\]

(10)

and \( L \) can be represented as:

\[
L = w_1 L_1 + w_2 L_2 + Q
\]

(11)

Actually, if the number of counting vectors is more than two. The similar formula for the description length of the weighted counting vector can be obtained. It is given in (12)
where $N$ denotes the number of $I$-ary counting vectors. The parameters $c_i$ and $t_j^{(i)}$ are determined by (13) and (14) respectively.

$$c_i = \frac{w_i V_i}{\sum_{i=1}^{N} w_i V_i}$$

(13)

And

$$t_j^{(i)} = \frac{w_i n_j^{(i)}}{\sum_{i=1}^{N} w_i n_j^{(i)}}$$

(14)

After derivation, the formula (12) can be represented by (15)

$$L = \sum_{i=1}^{N} w_i L_i + Q$$

(15)

where the weighting cost $Q$ is determined by (16)

$$Q = \sum_{i=1}^{N} \left( w_i V_i \log \frac{w_i}{c_i} - w_i \sum_{j=0}^{l-1} n_j^{(i)} \log \frac{w_i}{t_j^{(i)}} - \frac{w_i}{2} \log \frac{\prod_{j=0}^{l-1} t_j^{(i)}}{c_i w_i^2} \right)$$

(16)

Here $Q$ is referred to as the weighting cost and in most of the time, $Q$ is small enough to be ignored in the weighting process. From (15), it is concluded that the context weighting is almost equivalent to the weighting of the description lengths of those counting vectors. Meanwhile, if the weighting cost is ignored, the weights can be optimized from a linear expression (15). As an example, the two counting vectors are used to illustrate the optimization method.

2.2 The Optimization of Weights

Considering the formula (15), the description length $L$ can be represented approximately as: $L \approx w_1 L_1 + w_2 L_2$.

Then the least square algorithm is suggested to optimize weights $w_1$ and $w_2$. Let $L = (L_1, L_2)^T$ denote the observing vector consisting of all description lengths which come
from all counting vectors joined in the weighting operation. Let $W = (w_1, w_2)^T$ be the corresponding weights vector. Then the estimated value of the description length $\hat{L}$, can be formed in the vector as:

$$\hat{L} = L^T W$$

However, for minimizing $\hat{L}$ by using the LS algorithm, the corresponding ideal value $L^*$ should be given firstly. For tackle this problem, the method is discussed as follows:

According to Wu and Zhai [19], the description length $\hat{L}$ can also be represented as:

$$\hat{L} = (w_1 V_1 + w_2 V_2) H(x_t|S) + \Delta$$

where $H(x_t|S)$ is the entropy of the conditional probability distribution $P(x_t|S)$ and $\Delta$ represents the model cost. Actually, the value of $(w_1 V_1 + w_2 V_2) H(x_t|S)$ represents the ideal code length for coding these $w_1 V_1 + w_2 V_2$ bases. Thus, in our work, $L^* = (w_1 V_1 + w_2 V_2) H(x_t|S)$ is considered as the ideal value of $\hat{L}$. Therefore, $L^*$ can be the objective of the minimization of $\hat{L}$. Let $e = |\hat{L} - L^*|$ denote the difference between $\hat{L}$ and $L^*$. The optimizing objective is to:

$$\min \{ f(W) = e^2 = |\hat{L} - L^*|^2 \}$$

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.

$$\begin{align*}
\frac{\partial f(W)}{\partial w_i} &= 0 \quad i = 1, 2 \\
\sum_{i=1}^{2} w_i &= 1
\end{align*}$$

(21)

The solution of Eq. (16) can directly be represented as.

$$W = R^{-1} \times d$$

(22)

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

$$R = L \times L^T, \quad d = L \times L^*$$

(23)

After solving these equations, the optimized weights can be obtained and the coding distribution can also be obtained by using (4).

3 Results

3.1 Experiment Design

In order to evaluate the performance of our optimization method for the context weighting problem, the proposed algorithm is employed in the microbial genome sequence compression. Four experiments are designed on their corresponding genome sequences to
evaluate various aspects of our algorithm. The testing genome sequences used in our experiments are listed as follows:

CHMPXX: Marchantia polymorpha chloroplast genome DNA
CHNTXX: Nicotiana tabacum chloroplast genome DNA
HEHCMVC: Human cytomegalovirus strain AD169 complete genome
HUMD YSTROP: Homo sapiens dystrophin (DHD) gene, intron 44
HUMGHCSA: Human growth hormone (GH1 and GH2) and chorionic somatomam-
HUMHBB: Human beta globin region on chromosome 11.
HUMHPRTB: Human hypoxanthine phosphoribosyltransferase gene, complete cds.
MPOMTGC: Marchantia polymorpha mitochondrion, complete genome
SCCHR III: S.cerevisiae chromosome III complete DNA sequence.
VACCG: Vaccinia virus, complete genome.

These genome sequences are all obtained from NCBI [17].

In experiment 1, we pay more attention to examine the value of the weighting cost $Q$ in (11). The purpose of this experiment is to illustrate that the weighting cost can be small enough to be ignored in coding process in most of the time. Two counting vectors for a few binary sources are obtained and weighted respectively. The corresponding weighting cost are calculated by (10).

There are three cases when two counting vectors are weighted: (1) they are similar with each other, (2) they are not similar, (3) they are similar but they are also similar to the uniform distribution. Then three groups of counting vectors are used for weighting on these three cases. Their corresponding counting vectors are listed as follows:

Case 1: (400, 30) and (450, 30)
Case 2: (50, 600) and (500, 40)
Case 3: (200,290) and (300,360)

Where $(a_0, a_1)$ denotes the counting vector and $a_0, a_1$ denote the number of the symbols whose values equal to 0 and 1 respectively. For comparison, the results for the three pairs of count vectors are all listed in Table 1, meanwhile, the corresponding description lengths of the respective counting vectors are also listed. In this experiment, the weights are both set to: 0.5.

Obviously, if two counting vectors are similar, the weighting cost is small enough to be ignored in the optimization process. Furthermore, when two similar counting vectors are given, whatever weights are chosen, the corresponding weighting cost will still be small. In order to illustrate this, these count vectors: (400, 30) and (450, 30) are used for weighting with different weights whose value range from 0.01 to 0.99. The value of the corresponding weighting costs for various weights are given in Fig. 1.

| Case | Description length 1 (bits) | Description length 2 (bits) | Weighting cost (bits) |
|------|-----------------------------|-----------------------------|-----------------------|
| 1    | 159                         | 164                         | 0.4297                |
| 2    | 257                         | 208                         | 362.9523              |
| 3    | 481                         | 659                         | 0.4003                |
According to the observation of Fig. 1, when weights are changed, the maximum value of the weighting cost is still less than 0.5. It implies that whatever the weights are, the weighting cost can be small. Meanwhile, the weighting costs with different weights of other two groups of count vectors (50, 600) and (500, 40), (200, 290) and (300, 360) are also given in Figs. 2 and 3 respectively. According to the observation of Fig. 1, when weights are changed, the maximum value of the weighting cost is still less than 0.5. It implies that whatever the weights are, the weighting cost can be small. Meanwhile, the weighting costs with different weights of other two groups of count vectors (50, 600) and (500, 40), (200, 290) and (300, 360) are also given in Figs. 2 and 3 respectively.

The results indicate that the weighting cost will not be small unless one weight equals to 0, meaning no context weighting should be applied.

Finally, the result from the last group counting vector is given in Fig. 3.

The results from Figs. 1 and 3 indicate that if two counting vectors are similar, the weighting cost is small, no matter what distribution the counting vectors are.

---

Fig. 1 The weighting costs with different weights for counting vectors (400, 30) and (450, 30)

---

Fig. 2 The weighting costs with different weights for counting vectors (50, 600) and (500, 40)
For comparison, we let three curves be illustrated into one image, which is given in Fig. 4. In order to show more clearly, the weighting cost of the case 3 are amplified 100 times.

Meanwhile, by observations from Figs. 1 and 2 and Table 1, if two counting vectors are close enough, the weighting cost can become even smaller than each of the corresponding description lengths of the two counting vectors. It implies that in this case the weighting cost can be ignored in weighting process. If these counting vectors are not similar, the corresponding weighting cost will be too large to be ignored. However, if two counting vectors are similar but the weighted counting vector is nearly uniformly distributed, the resulting description length will become large. In this case, the context weighting process should not be considered although the weighting cost is small.

Actually, these conclusion from observations can also be extended to the case that the number of counting vectors is more than two. According to the formula (16), the corresponding weighting cost with different group of counting vectors can be calculated easily. Based on the discussion above.

Fig. 3 The weighting costs with different weights for counting vectors (200, 290) and (300, 360)

Fig. 4 The weighting costs for three cases. The black curve denotes the cost for case 1, The red curve denotes the cost for case 2 and the blue one denotes the case 3
Then we will discuss the weighting cost when three or more counting vectors are weighted. Furthermore, the counting vectors under the I-ary cases are also discussed.

Firstly, we give some counting vectors under binary case for our experiments. In this case, there are four situations to describe their similarities.

Situation 1: These three counting vectors are all similar.
Situation 2: These three counting vectors are all not similar.
Situation 3: There are two counting vectors similar but the other one is not similar with these two vectors.
Situation 4: These three counting vectors are similar but also similar with the uniform distribution.

To evaluate the corresponding weighting costs, four groups of counting vectors are listed below for our experiments. They are:

Group 1 (For situation 1):
(500, 20), (490, 30) and (510, 38)

Group 2 (For situation 2):
(500, 20), (30, 490) and (230, 235)

Group 3 (For situation 3):
(500, 20), (490, 30) and (38, 510)

Group 4 (For situation 4):
(230, 235), (240, 245) and (238, 242)

Actually, the weighting costs are related to the value of weights. In our experiments, three weights $w_1$, $w_2$ and $w_3$ are traversing from 0.01 to 0.99 respectively. The nine possible results are listed in Table 2. Meanwhile, the corresponding description length of three counting vectors are also listed. For abbreviation, the description lengths of three counting vectors are denoted by $L_1$, $L_2$ and $L_3$ ordering by the given above. $L_w$ denotes the description length of the weighted vector. Meanwhile, in last four tables, this abbreviation is also valid.

From Table 2, whatever the weights change, the value of the weighting cost is smaller enough to be ignored when optimization algorithm is executing to optimize the weights. However, for situation 2, the results are different from the results in situation 1.

Similarly, the corresponding results for situation 2 are listed in Table 3.

| Weights       | L1 (bits) | L2    | L3    | Lw (bits) | Weight cost (bits) |
|---------------|-----------|-------|-------|-----------|--------------------|
| (0.01, 0.5, 0.49) | 124.4     | 167.9 | 201.8 | 184.3     | 0.2663             |
| (0.1, 0.4, 0.5)  | 124.4     | 167.9 | 201.8 | 181.1     | 0.032              |
| (0.3, 0.4, 0.3)  | 124.4     | 167.9 | 201.9 | 166.1     | 0.3478             |
| (0.4, 0.4, 0.2)  | 124.4     | 167.9 | 201.9 | 158.4     | 0.4046             |
| (0.38, 0.44, 0.18) | 124.4     | 167.9 | 201.9 | 158.5     | 0.3497             |
| (0.52, 0.14, 0.34) | 124.4     | 167.9 | 201.9 | 158.4     | 0.8965             |
| (0.6, 0.3, 0.1)  | 124.4     | 167.9 | 201.9 | 146.2     | 0.4165             |
| (0.81, 0.11, 0.08) | 124.4     | 167.9 | 201.9 | 136.2     | 0.3793             |
| (0.66, 0.08, 0.26) | 124.4     | 167.9 | 201.9 | 149.5     | 0.9251             |
It is obviously that the value of weighting cost is so large whatever the weights are. In this case, actually, the context weighting should not be executed.

For situation 3, the corresponding results can also be obtained and listed in Table 4. There are similar results in Table 4 with the results in Table 4. These results indicate that the weighting cost will be increased when those un-similar counting vectors are weighted. Thus, there is an interesting result in the last line in Table 4. If the weight for this ‘unlike’ is small enough, the weighting cost can be also smaller to be ignored. Namely, in this case, the optimization can be achieved under situation 3. Actually, in this paper, our algorithm can obtain smallest value of weight for this unlike counting vector in order to ensure the other weights can be optimized.

For situation 4, the corresponding results are listed in Table 5.

From Table 5, although the weighting costs are all small, the corresponding description lengths are large. Namely, the context weighting should not be executed.

Then the three counting vectors in 3-ary case are used to testify the performance of the weighting cost. Only the results in situations 1–3 above are given here. First Group counting vectors are:

(500, 30, 35), (520, 35, 44) and (600, 44, 52).

The description length and the weighting cost are also given as the results. The abbreviation here is similar with the utilization above (Table 6).

Under 3-ary case, the value of weighting cost is larger the value in binary case, but they can also be ignored in the optimization process. Meanwhile, if these three counting vectors are not similar, the same results with the binary case results can be obtained, which is listed in Table 7. This group counting vectors are:

**Table 3** The weighting cost for three un-similar counting vectors

| Weights        | L1 (bits) | L2  | L3  | Lw (bits) | Weight cost (bits) |
|----------------|-----------|-----|-----|-----------|--------------------|
| (0.01, 0.5, 0.49) | 124.4     | 167.9| 468.4| 417.6     | 102.34             |
| (0.3, 0.3, 0.4)   | 124.4     | 167.9| 468.4| 501.5     | 225.62             |
| (0.5, 0.2, 0.3)   | 124.4     | 167.9| 468.4| 475.7     | 238.67             |
| (0.7, 0.1, 0.2)   | 124.4     | 167.9| 468.4| 386.5     | 188.34             |
| (0.9, 0.02, 0.08) | 124.4     | 167.9| 468.4| 228.4     | 75.31              |

**Table 4** The weighting cost for only two similar counting vectors

| Weights        | L1 (bits) | L2  | L3  | Lw (bits) | Weight cost (bits) |
|----------------|-----------|-----|-----|-----------|--------------------|
| (0.4, 0.3, 0.3) | 124.4     | 167.9| 201.8| 482.9     | 320.84             |
| (0.3, 0.3, 0.4) | 124.4     | 167.9| 468.4| 522.8     | 353.67             |
| (0.5, 0.4, 0.1) | 124.4     | 167.9| 468.4| 307.9     | 157.70             |
| (0.49, 0.49, 0.02) | 124.4 | 167.9| 468.4| 386.5     | 188.34             |
| (0.49999, 0.49999, 0.00002) | 124.4 | 167.9| 468.4| 146.9     | 0.3182             |
There is a useful information contained in Table 7. When three counting vectors are not similar with each other, the description length and the weighting cost can not be optimized at the same time. Etc, in this time, the context weighting should not be executed.

At last, the third group counting vectors are given, they are:

(700, 10, 30), (680, 15, 40) and (20, 720, 50)

In this group counting vectors, only two are similar. According to the discussion above, in binary case, if the weight for this unlike one is small, the optimization can also be executed. In 3-ary, the corresponding results are listed in Table 8.

It is obviously, the same results with the binary case is obtained again. It implies that the description length and the weighting cost can be minimized though modifying the exact value of weights. Namely, the weights are also optimized by minimizing both description

| Table 5 | The weighting cost for the situation 4 |
|---------|--------------------------------------|
| Weights | L1 (bits) | L2   | L3   | Lw (bits) | Weight cost (bits) |
| (0.7, 0.2, 0.1) | 468.4 | 488.4 | 483.4 | 473.9 | 0.5781 |
| (0.5, 0.3, 0.2) | 124.4 | 167.9 | 468.4 | 477.4 | 0.7423 |
| (0.4, 0.4, 0.2) | 124.4 | 167.9 | 468.4 | 479.5 | 0.7605 |
| (0.3, 0.3, 0.4) | 124.4 | 167.9 | 468.4 | 480.4 | 0.7850 |
| (0.1, 0.4, 0.5) | 124.4 | 167.9 | 468.4 | 483.9 | 0.6800 |

| Table 6 | The weighting cost for the first group counting vectors |
|---------|--------------------------------------------------------|
| Weights | L1 (bits) | L2   | L3   | Lw (bits) | Weight cost (bits) |
| (0.8, 0.1, 0.1) | 360.6 | 420.4 | 503.8 | 380.2 | 12.34 |
| (0.5, 0.3, 0.2) | 360.6 | 420.4 | 503.8 | 407.6 | 7.414 |
| (0.4, 0.4, 0.2) | 360.6 | 420.4 | 503.8 | 413.6 | 4.046 |
| (0.3, 0.3, 0.4) | 360.6 | 420.4 | 503.8 | 430.2 | 2.070 |
| (0.1, 0.4, 0.5) | 360.6 | 420.4 | 503.8 | 483.9 | 7429 |

| Table 7 | The weighting cost for these un-similar counting vectors |
|---------|----------------------------------------------------------|
| Weights | L1 (bits) | L2   | L3   | Lw (bits) | Weight cost (bits) |
| (0.2, 0.4, 0.4) | 197.6 | 238.9 | 268.9 | 883.3 | 487.05 |
| (0.3, 0.3, 0.4) | 197.6 | 238.9 | 268.9 | 901.4 | 347.90 |
| (0.5, 0.2, 0.3) | 197.6 | 238.9 | 268.9 | 853.5 | 40.82 |
| (0.7, 0.1, 0.2) | 197.6 | 238.9 | 268.9 | 800.5 | 45.48 |
| (0.9, 0.02, 0.08) | 197.6 | 238.9 | 268.9 | 402.7 | 152.511 |

(500, 10, 20), (15, 600, 21) and (22, 22, 500)

There is a useful information contained in Table 7. When three counting vectors are not similar with each other, the description length and the weighting cost can not be optimized at the same time. Etc, in this time, the context weighting should not be executed.

At last, the third group counting vectors are given, they are:

(700, 10, 30), (680, 15, 40) and (20, 720, 50)

In this group counting vectors, only two are similar. According to the discussion above, in binary case, if the weight for this unlike one is small, the optimization can also be executed. In 3-ary, the corresponding results are listed in Table 8.

It is obviously, the same results with the binary case is obtained again. It implies that the description length and the weighting cost can be minimized though modifying the exact value of weights. Namely, the weights are also optimized by minimizing both description
length and weighting costs. It is the evidence that our optimization algorithm can play its role. Furthermore, in the last two line in Table 8, the value of the weight for this unlike counting vector is changed only a little, the weighing cost can be influenced more. Actually, the most practicable method in this situation is to prohibit this unlike counting vector joining the context weighting, namely, its weight is equal to 0.

According to these experiments and results, the conclusion of the context weighting can be summarized as follows:

There are two conclusions: the first is that the optimization for weights based on the proposed algorithm is feasible. The second is that the context weighting should be executed only if the weighting cost can be ignored. While in previous context weighting studies, if the weights are given, the weighting process will take place throughout the whole coding process. It is apparent that the code length will be increased based on the above analysis. On the other hand, although this experiment is designed for binary sources, the conclusion can also be applied to non binary cases. Therefore, the proposed algorithm can be used for the genome sequence compression to improve the coding results.

In experiment 2, we evaluate the performance of the proposed weighting algorithm. Three pairs of counting vectors are used for weighting. The weighting algorithm in [20] and the proposed algorithm are used to obtain the weights respectively. In [20], the weights are obtained by using the average code length but the weights of the proposed algorithm are optimized by using the description length. Actually, once the weights is calculated there is no optimization for these weights in [20]. The three pairs of counting vectors listed below are obtained from counting the bases in a microbial genome sequence:

Case 1:

| Weight | L1 (bits) | L2 | L3 | Lw (bits) | Weight cost (bits) |
|--------|-----------|----|----|-----------|--------------------|
| (0.4, 0.3, 0.3) | 261 | 333.1 | 406.5 | 869.2 | 320.84 |
| (0.3, 0.3, 0.4) | 261 | 333.1 | 406.5 | 937.8 | 574.18 |
| (0.5, 0.4, 0.1) | 261 | 333.1 | 406.5 | 582.5 | 285.92 |
| (0.49, 0.49, 0.02) | 261 | 333.1 | 406.5 | 372.9 | 77.58 |
| (0.49999, 0.49999, 0.00002) | 261 | 333.1 | 406.5 | 298.1 | 1.096 |
| (0.79999, 0.29999, 0.00002) | 261 | 333.1 | 406.5 | 309.4 | 1.1443 |
| (0.99998, 0.00001, 0.00002) | 261 | 333.1 | 406.5 | 261 | 0.0514 |
| (0.29999, 0.79999, 0.00002) | 261 | 333.1 | 406.5 | 345.4 | 1.098 |
| (0.00001, 0.99998, 0.00001) | 261 | 333.1 | 406.5 | 333.1 | 0.0385 |
| (0.00001, 0.99997, 0.00002) | 261 | 333.1 | 406.5 | 333.1 | 0.076 |

In experiment 2, we evaluate the performance of the proposed weighting algorithm. Three pairs of counting vectors are used for weighting. The weighting algorithm in [20] and the proposed algorithm are used to obtain the weights respectively. In [20], the weights are obtained by using the average code length but the weights of the proposed algorithm are optimized by using the description length. Actually, once the weights is calculated there is no optimization for these weights in [20]. The three pairs of counting vectors listed below are obtained from counting the bases in a microbial genome sequence:

Case 1:

| A | T | G | C |
|---|---|---|---|
| 23 | 380 | 20 | 30 |

Case 2:

| A | T | G | C |
|---|---|---|---|
| 330 | 38 | 45 | 40 |

| A | T | G | C |
|---|---|---|---|
| 23 | 380 | 20 | 30 |
Case 3:

|   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   | A | T | G | C |   | A | T | G | C |   |   |   |
|   | 90 | 101 | 74 | 93 |   | 103 | 180 | 160 | 123 |   |   |   |

The description lengths of the weighted counting vectors obtained by the two algorithms are listed in Table 9. The second and third columns contain the weights and the description lengths obtained by the algorithms in [20] for the counting vectors. The fourth and the fifth columns list the weights and the description lengths of the weighted vectors obtained by the proposed algorithm.

It is obvious that the description lengths by the proposed algorithm is shorter than the description lengths by the algorithm in [20]. Especially in case 2, although the counts in the two counting vectors are distributed singularly, the weighting by the algorithm in [20] will lead to the counts in the weighted counting vector symmetrically distributed, resulting in longer description length. On the contrary, in our algorithm, the counting vector with the minimum description length is selected for coding. It implies that the weighting is not executed.

The proposed context weighting method is employed to compress the microbial genome sequences. In this experiment, the genome sequences MTOMTCG, VACCG are used as the sources and three context templates with different orders are constructed. The context templates are given in Fig. 2 in which \(x(i)\) is the current symbol to be encoded and \(x(i-k), k = 1, \ldots, 8\) are those symbols already encoded (Table 10).

Meanwhile, in order to accelerate the coding process, the updating period is set to 100, which implies that the weights will be re-calculated or optimized when 100 bases are encoded.

For comparison, the same context models are also weighted by the algorithm in [20, 21] and the results by the two algorithms are listed in Table 3.

From Table 11, the results indicate that the proposed algorithm is better than the algorithm in [20]. The improvement on the coding performance is resulted from the optimization to the weights.

Finally, the proposed algorithm is employed for compressing those benchmark genome sequences which are used in [15, 20–22]. The same context models with those used in these algorithms are also applied in our experiments. The coding results are given in Table 12. For abbreviation, our algorithm is referred to as: OPGC.

The compared algorithms are BioCompress-2 (BioC) [12], GenCompress (GenC) [6], DNACompress (DNAC) [7], DNAPack (DNAP) [4], CDNA [15], GeMNL [14] and XM [20]. Actually, for most of the tested genome sequences, our algorithm can obtain better coding results. It is apparent that the average result is better than all of the compared algorithms listed.

### Table 9 The comparison of the weights and description lengths by two algorithms

| Case | Algorithm in [20] | Proposed |
|------|-------------------|----------|
|      | Weights (w1, w2) | Description length (bit) | Weights (w1, w2) | Description length (bit) |
| 1    | (0.3826, 0.6174) | 428.6307 | (0.4162, 0.5838) | 427.6530 |
| 2    | (0.5233, 0.4767) | 419.4346 | (1, 0) | 414.3356 |
| 3    | (0.5153, 0.4847) | 511.8763 | (0.5043, 0.4957) | 511.6332 |
At last, the complexity of our algorithm should be considered. To against those algorithms, such as BioC, GenC and DNAC, it is meaningless which is reasoned to the fact that our compression method is not similar with those algorithm. Therefore, in this paper, our objective algorithm is XM. We implement XM on Matlab 7.0 and under the computer: CPU 2.6 GHz, 4 GB RAM. The times to compress four genome sequences are given in Fig. 5. Meanwhile, the updating period of our algorithm is set to 150 bases. The four genome sequences are: Chntxx, Vaccg, mtomtcg and chmpxx respectively.

Our algorithm is slower that XM obviously, which is resulted from the optimization process. However, it is worth to waste these time to obtain the optimized results. Actually, in practice application, the arithmetic encoder can be accelerated and the executing time of the proposed algorithm can be shorten.

| Model          | Order | Temple                                      |
|----------------|-------|---------------------------------------------|
| Context model 1| 5     | $x(i-5), \ldots, x(i-1), x(i)$             |
| Context model 2| 8     | $x(i-8), \ldots, x(i-1), x(i)$             |
| Context model 3| 3     | $x(i-5), x(i-2), x(i-1), x(i)$             |

Table 10 The templates the experiment 2 used

| Genome sequence | Algorithm in [20] (bit/base) | Proposed algorithm |
|-----------------|------------------------------|--------------------|
| MTOMTCG         | 1.8987                       | 1.8782             |
| VACCG           | 1.7825                       | 1.7731             |

Table 11 The comparison of the compression results from existing algorithms

| Sequences   | BioC   | GenC   | DNAC   | DNAP   | CDNA   | GeMLNL | XM       | OPGC  |
|-------------|--------|--------|--------|--------|--------|--------|----------|-------|
| Chmpxx      | 1.6848 | 1.673  | 1.6716 | 1.6602 | –      | 1.6617 | 1.6577   | 1.6431|
| Chntxx      | 1.6172 | 1.6146 | 1.6127 | 1.6103 | 1.65   | 1.6101 | 1.6058   | 1.6104|
| HEHCVMCG    | 1.8480 | 1.8470 | 1.8492 | 1.8346 | –      | 1.8420 | 1.8426   | 1.8382|
| HUMDYSTROP  | 1.9262 | 1.9231 | 1.9116 | 1.9088 | 1.93   | 1.9085 | 1.9031   | 1.9041|
| HUMHBB      | 1.8800 | 1.8204 | 1.7897 | 1.7771 | 1.77   | –      | 1.7513   | 1.7479|
| HUMGHCSA    | 1.3074 | 1.0969 | 1.0272 | 1.039  | 0.95   | 1.0089 | 0.9828   | 0.9902|
| HUMHPRTB    | 1.9066 | 1.8466 | 1.8165 | 1.7886 | 1.72   | 1.7639 | 1.7361   | 1.7258|
| MPOMTCG     | 1.9378 | 1.9058 | 1.8920 | 1.8932 | 1.87   | 1.8822 | 1.8768   | 1.8677|
| VACCG       | 1.7614 | 1.7614 | 1.7583 | 1.7583 | 1.81   | 1.7644 | 1.7649   | 1.7616|
| Average     | 1.7633 | 1.7210 | 1.7032 | 1.6967 | –      | –      | 1.6802   | 1.6766|

Bold values indicate bit/base

Table 12 The comparison of the compression results from existing algorithms

At last, the complexity of our algorithm should be considered. To against those algorithms, such as BioC, GenC and DNAC, it is meaningless which is reasoned to the fact that our compression method is not similar with those algorithm. Therefore, in this paper, our objective algorithm is XM. We implement XM on Matlab 7.0 and under the computer: CPU 2.6 GHz, 4 GB RAM. The times to compress four genome sequences are given in Fig. 5. Meanwhile, the updating period of our algorithm is set to 150 bases. The four genome sequences are: Chntxx, Vaccg, mtomtcg and chmpxx respectively.

Our algorithm is slower that XM obviously, which is resulted from the optimization process. However, it is worth to waste these time to obtain the optimized results. Actually, in practice application, the arithmetic encoder can be accelerated and the executing time of the proposed algorithm can be shorten.
The context weighting can be optimized with the weights optimization. This is one significant conclusion from our works. The weighting cost influence the efficiency of context weighting and the relation between weighting operation and weighting cost is illustrated here. A large number of experiments demonstrate that the proposed algorithm can achieve better results and the weighting cost can be controlled in details.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Funding This work is supported by Natural Science Foundation of Yunnan Province under Grant Nos. 2013FD042 and 2016FB113.

References

1. Deorowicz, S., et al. (2011). Robust relative compression of genomes with random access. Bioinformatics, 27(21), 2979–2986.
2. Grumbach, S., & Tahi, F. (1993). Compression of DNA sequences. In Proceedings of the data compression conference DCC-93, Snowbird, Utah (pp. 340–350).
3. Grumbach, S., & Tahi, F. (1994). A new challenge for compression algorithms: Genetic sequences. Information Processing and Management, 30(6), 875–886.
4. Deorowicz, S., et al. (2013). Genome compression: A novel approach for large collections. Bioinformatics, 29(20), 2572–2578.
5. Fricke, W. F., & Rasko, D. A. (2014). Bacterial genome sequencing in the clinic: Bioinformatic challenges and solutions. Nature Reviews, 15, 49–55.
6. Wang, C., & Zhang, D. (2011). A novel compression tool for efficient storage of genome resequencing data. Nucleic Acids Research, 39, 25. https://doi.org/10.1093/nar/gkr009.
7. Christley, S., et al. (2009). Human genomes as email attachments. Bioinformatics, 25, 274–275.
8. Claude, F., et al. (2010). Compressed q-gram indexing for highly repetitive biological sequences. In Proceedings of the international conference on bioinformatics bioengineering (pp. 86–91). Washington, DC: IEEE Computer Society Press.
9. Tabus, I., Korodi, G., & Rissanen, J. (2007). Normalized maximum likelihood models for genomics. In DCC (pp. 253–263).
10. Korodi, G., & Tabus, I. (2005). An efficient normalized maximum likelihood algorithm for DNA sequence compression. ACM Transactions on Information Systems, 23(1), 3–34.
11. Soliman, T. H. A. (2009). A lossless compression algorithm for DNA sequence. *International Journal of Bioinformatics Research and Applications*, 5(6), 593–602.
12. Loewenstern, D., & Yianilos, P. N. (1999). Significantly lower entropy estimates for natural DNA sequences. *Computational Biology*, 6(1), 125–142.
13. Allison, L., Edgoose, T., & Dix, T. I. (1998). Compression of strings with approximate repeats. In *ISMB* (pp. 8–16).
14. Pinho, A. J., Neves, A. J. R., Bastos, C. A. C., & Ferreira, P. J. S. G. (2009). DNA coding using finite-context models and arithmetic coding. In *Proceeding of ICASSP-2009*, Taipei, Tai-wan.
15. Pinho, A. J., et al. (2011). Bacteria DNA sequence compression using a mixture of finite-context models. In *IEEE statistical signal processing workshop*, Portugal (pp. 125–128).
16. Stern, L., Allison, L., Coppel, R. L., & Dix, T. I. (2001). Discovering patterns in plasmodium falciparum genomic DNA. *Molecular and Biochemical Parasitology*, 118, 175–186.
17. www.ncbi.com
18. Chen, M., & Chen, J. (2013). Context quantization based on the modified genetic algorithm with K-means. In proceedings of 9th International Conference on Natural Computation (pp. 424–428). Shengyang China.
19. Wu, X., & Zhai, G. (2011). Adaptive sequential prediction of multidimensional signals with applications to lossless image coding. *IEEE Transactions on Image Processing*, 20(1), 36–42.
20. Tabus, I., Korodi, G., & Rissanen, J. (2003). DNA sequence compression using the normalized maximum likelihood model for discrete regression. In *DCC* (pp. 253–263).
21. Cao, M. D., Dix, T. I., Allison, L., & Mears, C. (2007). A simple statistical algorithm for biological sequence compression. In *Proceedings of the data compression conference, DCC-2007*, Snowbird, Utah.
22. Pinho, A. J., & Pratas, D. (2014). MFCompress: A compression tool for FASTA and multi-FASTA data. *Bioinformatics*, 30(1), 117–118.

**Min Chen** received the B.S. and M.S. degrees in Electronic Engineering from Yunnan University, China, in 2004 and 2009 respectively. In 2017, he will receive the Ph.D. degree in information engineering from Yunnan University. He joined the Information Security College, Yunnan Police College, China, in 2012. His research interests include image compression, signal processing, and biomedical information processing.
**Rui Li** male, born in November 1981, Master’s degree, lecturer, work at Honghe University. In 2004, he has been awarded the title bachelor’s degree in Yunnan University, and then he has been awarded the title Master’s degree in Kunming University of Science and Technology in 2013.

**LiJun Yang** received the B.S. and M.S. degrees in Electronic Engineering from Yunnan Normal University, China, in 1985 and 1998 respectively. She joined the Information Security College, Yunnan Police College, China, in 1990. Her research interests include image compression, signal processing, and biomedical information processing.