# Optimized and Cost Considering Huffman Code For Biological Data Transmission

**Abstract**

*Keywords:*

## 1. Introduction

## 2. Background

### *2.1. Issues on Biological Data Transmission*

The size of biological data including DNA sequences increase with an ever expanding rate and will be bigger and bigger in the future. These Biological data are stored in biology database, the exponential growth of these database become a big problem to all biological data processing methods. Different operation will be applied to these data such as, searching [],e-mail attachment [], alignment [], and transmission on distributed computing []. Interestingly, biological data compression can play a key role in all biological data processing.

A recent deluge of interest in the development of new tools for biological data processing, these all algorithms needs an efficient methods for data compression. The main objective of data compression methods is minimizing the number of bits in the data representation. In [Marty C. Brandon ] authors propose a new general data structure and data encoding approach for the efficient storage of genomic data. This method encode only the differences between a genome sequence and a reference sequence, the method use different encoding scheme from fixed codes such as Golomb and Elias codes, to variables codes, such as Huffman codes. Other methods based on same idea to encode only the difference between reference sequence and the target one, Authors in [Scott Christley1] uses Huffman code for encoding difference between sequence to sent it as an email attachment, but these methods suffer that they must sent the reference sequence for at least one time for each

species.
Wang and Zhang (2011) proposed a new scheme for referential compression of genomes based on the chromosome level. The Algorithm aim to search for longest common subsequence between matching parts and the differences encoded using Huffman coding.
All previous studies focus only on the differences and the relation between continuation of the sequence, and without improvement of the encoding scheme.

### *2.2. Rationale of Unequal Bit Cost Considering Encoding Approaches*

In the recent years, application of battery-powered portable devices, e.g. laptop computers and mobile phones has increased significantly. Proper representation of digital data and their transmission efficiency has become a primary concern for digital community because it affects the performance, reliability, and the cost of computation in both portable and non-portable devices. CMOS technologies were developed in order to reduce the power consumption both in data processing and transmission. In order to increase transmission speed and reduce transmission cost, parallel data transmission methods are widely used. However, parallel transmission is limited to short distance communications, e.g. locally connected devices, internal buses. Ruling out the possible availability of parallel transmission links over long distance, we are left with its serial alternative only.

Data encoding techniques came into action to improve the data transmission efficiency over serial communication medium by compressing data before transmitting. Efficiency can be measured in terms of incurred cost, required storage space, consumed power, time spent and likewise. Data must be encoded to meet the purposes like: unambiguous retrieval of information, efficient storage, efficient transmission and etc. Let a message consist of sequences of characters taken from an alphabet $\Sigma$, where $\alpha_1, \alpha_2, \alpha_3 \ldots, \alpha_r$ are the elements that represent the characters in the source $\Sigma$. The length of $\alpha_i$ represents its cost or transmission time, i.e., $c(\alpha_i) = length(\alpha_i)$. A codeword $w_i$ is a string of characters in $\Sigma$, i.e., $w_i \in \Sigma^+$. If a codeword is $w_i = \alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{in}$, then the length or cost of the codeword is the sum of the lengths of its constituent elements:

$$\text{cost}(w_i) = \sum_{j=1}^{n} c(\alpha_{ij}) \tag{1}$$

If all the elements of a codeword has unit cost or length then the cost of the codeword is equivalent to the length of the codeword. However, it is not necessary for the elements in the codeword to have equal length or cost. For example, in Morse Code all the ASCII characters are encoded as sequence of dots ($\cdot$) and dashes($-$) where a dash is three times longer than a dot in duration[1]. Using Morse Code, we can treat the binary bit 0 as a dot and the binary bit 1 as a dash. Even if we consider the voltage level to represent the binary digits then still they are different.

### *2.3. Huffman Codes*

## 3. Approach

### *3.1. Proposed Scheme*

### *3.2. Power Efficient Huffman code*

### *3.3. Optimisation of the Codes*

#### *3.3.1. Problem formulation*

The problem of finding the best allocation of codes to each symbol can be modelled as an Assignment Problems with Constraint, the problem is formulated as follows :

**Definition:** Given a set of codes $C = \{C_1, C_2...C_n\}$, and a set of frequencies $C = \{Q_1, Q_2...Q_n\}$. For each code we have the length of the code $|C_i|$ (number of bits) and the cost of the code $S_{C_i}$ (cost of ones and zeros), the objective is to assign to each frequency a code in order to get the minimum total number of bits, while respecting the initial assignment total cost $S_t$.
The Objective Function is :

$$Minimise \sum(|C_i| \times Q_j) \tag{2}$$

while :

$$\sum(|S_{C_i}| \times Q_j) \leqslant S_t \tag{3}$$

#### *3.3.2. Basic Genetic Algorithm*

Genetic Algorithm (GA) is a bio-inspired meta-heuristics algorithm developed by []. GA is a stochastic optimization algorithm imitate the natural evolution process of genomes. GA started by generate a population of random feasible solutions, the optimization process of GA is as follow, and we select two solution among the population, by one of the well-known selection

techniques. This two selected solution will be considered as two fathers, we generate two other new solutions from the two selected solution (Sons), this new can be mutate according to a given mutation probability. The quality of each solution is computed with the fitness function with control the evolution of the GA population by the Deletion of worst solution and insertion of good solutions between fathers and sons. This processes is repeated until the stopped criteria is achieved which can be the number of generation or if the population is stabilized.

*3.3.3. GA for Bits minimisation*

The main objective of the GA optimisation algorithm for bits minimisation problem is to assign to each frequency a specific code. The GABM population is generated randomly from the different codes and frequencies, and we add to this population the affectation given by the cost considering algorithm (step 1). The optimisation process of genetic algorithms start with the selection of two solution randomly from the population (step 2). After that the crossover operation are applied to these two solution (considered as parents) to generate two new solutions (considered as sons)(step 3), these new solutions are mutated according to a predefined probability (step 4) to ensure a good diversification on the space solutions. The next step is to evaluate the four solution (parents and sons)(step 5). Finally the new population are reproduced by replacing the two initial selected solutions by the two best among this four solution (parents and sons) (step 6). the whole [process are repeated until the max number of operation is achieved (return to step 1).

**Algorithm 1** GA for bits minimisation

1: Population initialization (P).
2: **while** Max number of generation not achieved **do**
3: Select two solutions S1,S2 form P.
4: Crossover $S_1, S_2$ to generate $S_{11}, S_{21}$.
5: Mutate $S_{11}, S_{21}$.
6: Evaluate $S_1, S_2, S_{11}, S_{21}$.
7: Replace $S_1, S_2$ by the two best from $(S_1, S_2, S_{11}, S_{21})$.
8: Return to 1;
9: Display the best solution from the population P;

## 4. Results And Discussion

## 5. Conclusion

[1] W. A. Redmond, International morse code, Microsoft Encarta 2009 [DVD] (1964) 275–278.