Switching Optimisation in Huffman Code for Power Efficient Data Transmission

Abstract

Reliability of low powered devices is highly dependent on their power efficiency. In digital communication, a significant amount of power is dissipated in data transmission. Different technologies have been emerged to address the issues of power consumption. In CMOS technology, dynamic power accounts for 70%-90% of the total power dissipation and it depends on the representation of the data and increases linearly with switching activities (transition from logic level High to Low and vice versa). Therefore, an efficient representation of data can minimise power consumption by reducing switching activities. In this paper, we have extended the Huffman code, a widely used data compression technique, by using genetic algorithm to reduce switching activities in the transmitted message. The main objective of the proposed approach is to minimise the switching activities in the codeword of each symbol as well as the switching activities between the symbols. The approach starts its operation by generating an initial population, a set of Huffman trees, for all the input symbols. Afterwards, the genetic operators such as selection, crossover and mutation are applied to the initial population to improve the quality of the solutions. The performance of the approach is evaluated by applying it to a set of real biological datasets. The experiments yield that the proposed approach reduces the switching activity by 45.47% in the best case, by 36.33% in the average case and by 16.42% in the worst case.

Keywords:

1. Introduction

In the recent years, application of battery-powered portable devices, e.g., laptop computers, personal digital assistant (PDA), and mobile phones has increased significantly. The reliability and performance of such devices are primarily dependent on their power consumption, i.e., effective battery life. Bigger battery size could help to improve power efficiency, however, in a portable device the size of the battery is restricted by the size and weight of the device itself. The cost of providing power to both portable and non-portable devices has resulted in significant interest in power reduction. CMOS technologies were developed in order to reduce the power consumption both in data processing and transmission. In [1], an approach was proposed to minimise power consumption in CMOS circuits. The method considers optimising the technology used to implement the digital circuits, circuits style and topology, the architecture for implementing the circuits and the algorithms that are being implemented in the devices.

As data transmission is a common operation in both portable and non-portable devices, therefore, reducing power dissipation in transmission systems is important. Authors in [2, 3] have investigated different criteria for low power data transmission. Power consumption for data transmission in IEEE 802.11 multi-hop networks and 3G mobile wireless networks were analysed in [4] and [5] respectively. In order to increase transmission speed and reduce power dissipation, 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. If we attempt to transfer big files, e.g. DNA sequences, over a serial transmission link then it would take a significant amount of time and obviously a significant amount power would be dissipated in the process. However, data compression techniques are widely used to reduce size of data before transmitting over serial medium to reduce transmission time and cost.

Huffman code [6] is an entropy encoding algorithm widely used for lossless data compression. For any given set of symbols and associated occurrence probabilities, Huffman algorithm generates a binary tree (Huffman tree) to encode a message with an optimal number bits. The codeword for a distinct symbol is generated by traversing the tree from the root to the leaf node representing the symbol. In this traversal process, a move towards a left and a right node is represented by 0 and 1 respectively. A major drawback of the classical Huffman code is that the whole stream must be read prior to encoding. To transmit a binary string, power dissipation is proportional to the number of switching, i.e., transition from 0 to 1 and vice versa, present in a transmitted string. Although a number of approaches like [7, 8] have extended the classical Huffman code by treating binary bits differently (with unequal letter cost), a little effort has been made to optimise the switching activities in the Huffman code to develop a power efficient Huffman code. To the knowledge of the authors, the only effort to generate Huffman code considering switching activity is seen in [9]. This approach follows a greedy strategy and works in two steps to reduce total number of switching in the Huffman code. In the first step the switching activity between each pair of symbol (inter-switch) is reduced and then in the second step switching activity in the codeword for each each symbol (intra-switch) is reduced. As the optimisation is done independently in two steps therefore it is highly likely that the gain obtained in the first step may be compromised in the second step and vice versa.

In this paper, a genetic algorithm based approach is proposed to reduce the total number of switching where inter-switch and intra-switch is optimised in a single step, i.e, a balance is made between the two different types of switching. The approach generates a set of trees, each one represents a set of codewords (solution). After that a set of genetic operators such as selection, crossover and mutation is used to evolute (create new trees) the population to minimise number of total switches. The evolution process is controlled by the total number of inter- and intra-switches. Inter-Switch between two adjacent symbols depend on the their codewords, i.e., how the codeword of the preceding symbol finished (with 0 or 1) and how the codeword for the descendent symbol started. If the end bit of the preceding symbol and the staring bit of the successor symbol differs then their exists a switch and the total number of switch is the number of times the two symbols in consideration occurs one after another. To obtain the total number Inter-Switch between pair of symbols, we used a descendent matrix which is a two dimensional matrix represents the number of occurrence of each symbol after another. This matrix is formed when the message to be transmitted is scanned to obtain the frequency of distinct symbols. The number of intra-switch for a symbol is calculated by multiplying the frequency of that symbol by the number of logic level transition (0 to 1 or 1 to 0) in the codeword of the symbol. The efficiency of the approach is evaluated by applying it to a set of biological datasets. The experiments yield that the proposed approach improves the total number of switches by $X\%$ in the best case, by $Y\%$ in the average case and by $Z\%$ in the worst case.

The rest of the paper is organised as follows: Section 2 presents the background study of the relationship of switching activity with power consumption, the relation of switching with Huffman code and the issues in biological data transmission. The proposed approach is described in Section 3. Experimental results and discussion are presented in Section 4 . Finally, concluding remarks are presented in Section 5.

2. Background

2.1. Effect of Switching in Power Consumption

CMOS is the most widely used technology implemented in VLSI chips because of their power efficiency. Power dissipation in CMOS circuits has three different components: dynamic, static, and leakage [10].

$$\begin{aligned} P_{avg} &= P_{dynamic} + P_{static} + P_{leakge} \\ &= 0.5 \times C_L \times V_{dd}^2 \times E(sw) \times f_{clk} + I_{sc} \times V_{dd} + I_{leakage} \times V_{dd} \end{aligned} \quad (1)$$

where
P_{avg} is the average power dissipation C_L is the load capacitance,
V_{dd} is power supply,
$E(sw)$ is average transition (switching) number per clock cycle ,
f_{clk} is the clock frequency,
I_{sc} is the short circuit current, and
$I_{leakage}$ is the leakage current.
Out of the three components, the dynamic power dissipation is the dominant and it counts for 70%-90% of the total power consumption[11]. Therefore, optimisation of dynamic power consumption is considered in this paper. In CMOS circuit, dynamic power consumption arises when the load capacitance (C_L) is charged to address a transition from low to the high voltage level. The parameters C_L, V_{dd}, and f_{clk} in dynamic power dissipation are primarily determined by the circuit layout and the fabrication technology. However, amount of switching, $E(sw)$ completely depends on the digital representation of the data. As the dynamic power dissipation is proportional to the average number of switching, thus a reduction in the total switching number will eventually results in the reduction of the dynamic power consumption. Therefore, we focus on reducing total amount of switching in Huffman code to improve power efficiency of digital data processing.

2.2. Huffman Code and Its Relationship with Switching Activity

In computer science and information theory, Huffman code is an entropy encoding algorithm used for lossless data compression. The classical Huffman code requires to pass the input data two times to complete the encoding operation. In the first pass, it reads the whole stream to determine the occurrence frequencies of distinct symbols. After that, symbols along with their frequencies are used to generate a binary tree known as Huffman tree. The process of Huffman tree generation is shown in algorithm 1 [12].

Algorithm 1 Huffman(C)

```
1: $n \leftarrow |C|$
2: $Q \leftarrow C$
3: for $i = 1$ to $n - 1$ do
4:     allocate a new node $z$
5:     $left[z] \leftarrow x \leftarrow EXTRACT\_MIN(Q)$
6:     $right[z] \leftarrow y \leftarrow EXTRACT\_MIN(Q)$
7:     $f[z] \leftarrow f[x] + f[y]$
8:     $INSERT(Q\ ,\ z)$
9: end for
10: return $EXTRACT\_MIN(Q)$
```

In the algorithm, C is the set of n distinct symbols, and each symbol $c \in C$ has a frequency denoted as f[c]. The algorithm starts its operation with a set of $|C|$ leaf nodes where each node corresponds to a distinct symbol. After that, a sequence of $|C| - 1$ merging operation is performed to obtain the final tree. The merging is done by extracting two least frequent symbols from the minimum priority queue, Q, and creating and inserting a new node (parent) to the queue for that two symbols. These two symbols are considered as the left and right child of their parent node respectively. The frequency of the new node is calculated as the sum of the frequency of its child nodes. The resultant tree ensures the optimal encoding of symbols to obtain maximal compression performance. To form a codeword for a symbol, the tree is traversed from the root to the leaf node representing the symbol. In the traversal process, any move made towards the left direction is replaced by '0' and any move towards the right direction is replaced by '1'. As a result codeword for each symbol consist of a sequence of 0's and/or 1's. Once the codewords are obtained, in the second pass, the Huffman algorithm encodes the symbols of the input message with the codewords, thus reduce the size of the whole message. For a set of symbols more than one Huffman tree is possible, i.e., optimal solution can be obtained from different structure of the tree. As the codeword of a symbol depend on its position in the tree, therefore changing the structure of the tree will change the codeword for the symbol. A codeword can contain zero or more switching activity within itself refereed to as intra-switch. For example, if a symbol ‘A’ is assigned a codeword ‘000’ then there is no intra-switch, on the other hand, if the codeword ‘101’ is assigned to a symbol ‘B‘ then their exist two intra-switches. When the codewords of different symbols are concatenated together during transmission they form switching based on the head and tail bit of the codewords. For example, if we want to transmit ‘AB’ then according to the above codewords we will transmit ‘00**01**01’ which will result in an inter-switch because the tail bit (0) of ‘A’ and the head bit (1) of ‘B’ are different. That means, in this case, we have three total switches–two intra-switches and one inter-switch. Now if ‘B’ is assigned the codeword ‘010’, the new transmitted string is ‘00**00**10’, then we still have two intra-switches for the codeword of ‘B’ but the inter-switch no more exists. So, changing tree structure, i.e., codewords, will change the total number of switching. Hence, the aim of this paper is to find an optimal Huffman tree that will minimise the total number of switching by making a balance between intra- and inter-switches.

2.3. Issues in Biological data transmission

The size of biological databases are continuously increasing. In the last decade the access to these databases has increased significantly. The exponential growth of these databases pose a big problem to all biological data processing methods [13]. Data compression methods are widely used to reduce the overheads of treating, e.g., transmitting, the high volume of data. The main objective of data compression methods is minimising the number of bits in the data representation. Authors in [14] proposed a new approach to genomic data compression based in suffix and common substring and code the difference with Huffman code. In [15], authors propose a general scheme based on Huffman code for coding only the difference between the target genome and the referential one. Recently, a method has been proposed to deal with the transmission of genomic data as an email attachment [16]. All the methods used for biological data compression aims only at reducing the size of the data. However, due to the huge size of the input data, the compressed data contains a huge number of switching in them which has a big impact on power consumption. As no consideration is paid to the reduction of switching, hence, the compression process may be size efficient but not power efficient. Therefore, it is believed that a more efficient compression of biological data can be obtained by optimising the size of the data as well as the total number of switching in the transmitted data.

3. The proposed approach for Switching Optimisation

The main objective of the approach is to minimise the total number of switching in the representation of data produced by the Huffman algorithm. To do so, it is required to minimise both intra- and inter-switches among the codewords. The total number of intra-switch depends on the codewords themselves and the frequency of the symbols. As a result, for a given set of codewords an optimal number of intra-switches can be obtained by assigning the most frequent symbol with the codewords with minimum number of switching. However, it will not ensure the optimal number of inter-switches. Because, inter-switches not only depend on the codewords of the symbols but also the association between the symbols. By association, we mean how many times a symbol occurs after another symbol in the whole message, i.e., all possible pairing combination among the symbols. As the number of distinct symbols increases in the message the number of combination increases. So improving total number of inter-switching is a combinatorial problem.

Genetic algorithm (GA) has widely been applied to solve a range of combinatorial problem in complex VLSI design [17, 18, 19]. For this reason, the GA is identified as a potential solution to the above mentioned problem. GA is one the of the most popular bio-inspired meta-heuristic algorithm inspired by the natural evolution of species [20]. It is a population based algorithm starts with the generation of a random initial population. The population contain a set of feasible solutions called individuals, that are usually far from the optimal. The GA optimisation process uses a set of natural genetic operators such as selection, crossover and mutation to converge to the optimal solution. The whole process is shown in algorithm 2.

3.1. Data preparation and population generation

A genome is an expansion of DNA sequence which is a set of nucleotides that encodes a protein. These nucleotides are bounded together and each order define a specific protein. Three nucleotides form a triplet and interpreted as a specific amino acid [21]. For this reason, the whole sequence is

Algorithm 2 Switches optimising Huffman codes

```
Require: Textual representation of a Genome
Ensure: Low switches Huffman codes
 1: Generate triplet frequencies and adjacent matrix
 2: Generate the initial population of trees
 3: repeat
 4:    Select part of the population
 5:    Generate children candidates via crossover
 6:    Mutate children
 7:    Add children to population
 8:    Rank by fitness
 9:    Remove worst solutions until population limit
10: until The switches number stop decreasing
```

divided into a set of triplets. As each DNA sequence contains a combination of four nucleobases—guanine (G), adenine (A), thymine (T), and cytosine (C), therefore we will have $4^3 = 64$ possible triplets. As mentioned earlier that the classical Huffman code requires reading the whole sequence prior to encoding. So, in this case, the whole DNA sequence is read considering different triplets as distinct symbols. Frequencies for each of the triplets are calculated and the adjacency matrix (a square matrix) is also created which represents how many times a triplet occurs after each of the other triplets. After that, a random set of Huffman trees are generated for the input symbols (triplets) which are considered as the initial population for the GA.

3.2. Objective function

The main objective of the proposed approach is to generate a set of codewords for the input symbols to minimise the overall amount of switching. The objective function of the switching optimisation algorithm can be formulated as follow :

Definition: Let $F = \{F_1, F_2, ..., F_n\}$ and $C = \{C_1, C_2, ..., C_n\}$ be the set of frequencies and codewords for the input symbols $S = \{S_1, S_2, ..., S_n\}$ respectively. And $M(n, n)$ be the adjacency matrix where each entry $M[i][j]$ represents how many times the input symbol S_j appears after the symbol S_i in the whole sequence.
The Objective Function is to:

$$Minimise \Big(\sum_{i=1}^{n} \; (\; Sw(C_i) \times F_i \;) \; + \; \sum_{i=1}^{n} \sum_{j=1}^{n} \; (\; \mu(C_i, C_j) \times M[i][j] \;) \Big) \quad (2)$$

where,

$Sw(C_i)$ is the number of intra-switches in the codewords C_i
$\mu(C_i, C_j) = 1$ if the head (starting) bit of the codeword C_j and the tail (ending) bit of the codeword C_i are different.
$\mu(C_i, C_j) = 0$ if the head bit of the codeword C_j and the tail bit of the codeword C_i are equal.

3.3. Genetic operations

3.3.1. Selection

The first operator of the genetic algorithm is the selection. The main objective of the selection is to choose the part of the current population to be a candidate for the different genetic operators in order to breed the next generation population. Many selection techniques have been proposed in the literature [22]. In this approach the process of natural selection is maintained. At first, we generate a random even number *rand* between 2 and the population size. This number represents how many parents will be processed. After that, an unbiased random selection of *rand* individuals (solutions) from the population is made.

3.3.2. Crossover

The main operator of the GA is the crossover, allows to construct new solutions from the selected part of the population [23]. The selected solutions are ranked by fitness and sorted in descending order. After that, at a time, a pair of solution is taken from the sorted population and then crossed to generate two new solutions. This process continues until all the pairs of solutions are crossed. The main objective of the crossover is to benefits from the two solutions to generate better solutions. Two parent trees are crossed to find two new child trees in the following way:

Step 1: Select an internal node (a sub-tree) from each parent tree with equal number of nodes (the dashed selection in Fig. 1(a) and (b)).
Step 2: Put the selected sub-tree of the first tree in the place of the selected

sub-tree of the second tree and vice versa (see Fig. 1(c) and (d)).

However, in each of the trees some symbols are repeated and some are missed, but number of repeated and missed symbols are equal. This is because we replace a sub-tree by other sub-tree without any modification. As a result each child tree contains a part from its parent tree and a part from the other tree. If the remaining part from the parent tree and the newly added part has one or more symbols in common then these symbols will be repeated in the tree, on the other hand, the uncommon symbols between the newly added sub-tree and the replaced sub-tree will be missed. We solved this conflict by putting the missed symbols one by one in the place of the repeated symbol in the remaining part from the parent tree. That means, the newly added sub-tree is kept unchanged and all the replacements are done on the old part of the tree.

3.3.3. Mutation

The mutation operator changes the positions of two leaf nodes of the generated children (result of the crossover) [24]. It introduces the diversity in the search process, this diversification strategy allow the algorithm to converge to the optimal solution. The algorithm for this operator goes through the leaf nodes and change the positions of two leaf nodes from their current positions to another positions in the tree (see Fig.1(e) and (f)). The selection of the nodes to be mutated is done following a fixed mutation rate.

3.4. Population update

The crossover operation generates new solutions from the current solutions to build the second generation of the population. Furthermore, the mutation process provides the diversity in the search on the solution space by changing the position of the nodes in the same tree. After these two operations, the next step of the genetic algorithm is to breed the new population (see fig.2). Firstly, the new solutions (children) are added to the current population and ranked by fitness. The algorithm remove the worst solutions until the initial size of the population is achieved. The genetic algorithm continuously repeats the selection, crossover, mutation, and population update operation until the stopping criteria is achieved. In the proposed approach the stopping criteria is when the objective function (number of switches) stop decreasing.

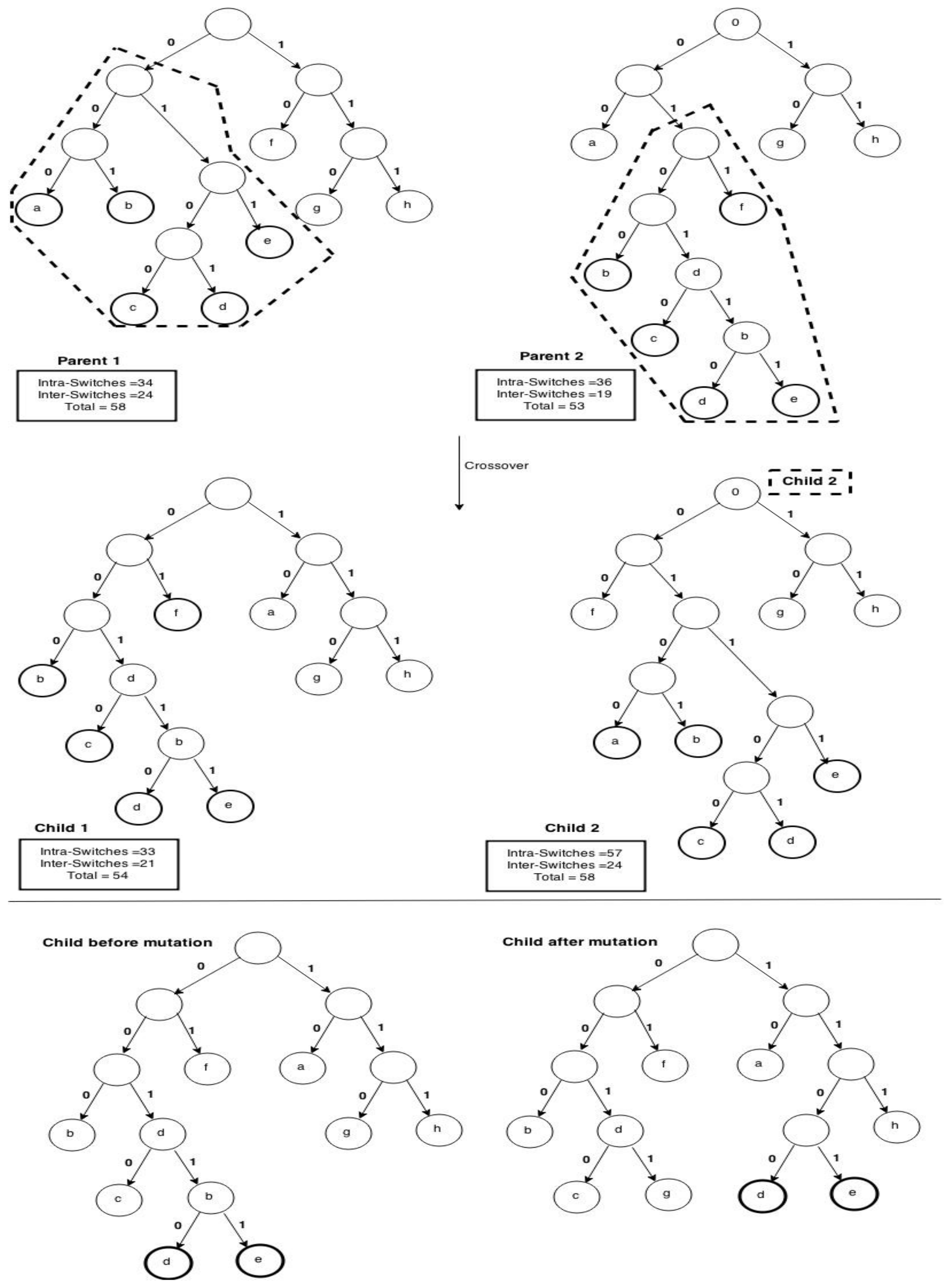

Figure 1: The Crossover operator

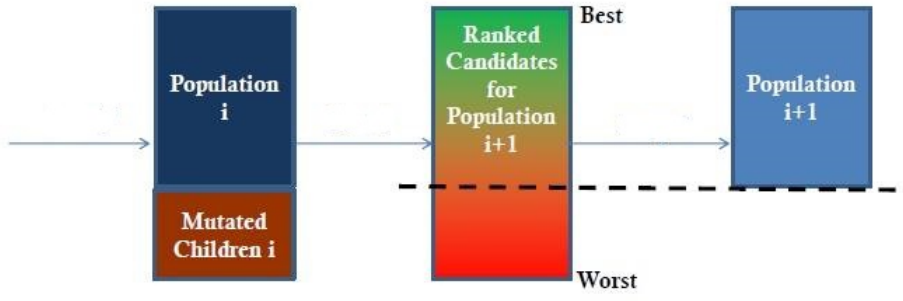

Figure 2: Population update for genetic algorithm

4. Experiments and Comparison

The approach has been evaluated with different real genomic biological data. Different genomes size have been used for the evaluation, from small genome to human chromosome in order to evaluate the efficiency of the approach.The datasets were downloaded as FASTA text file from a recent version of The National Center for Biotechnology Information (NCBI) available on ($http : //www.ncbi.nlm.nih.gov$) [25]. The used genomes are described in table 1, the size are given in Megabyte and the reference on the NCBI database.

Table 2 presents thee results obtained by the proposed approach and classical Huffman code. The results are presented with the total number of switches and the improvement rate. The results shows that the proposed approach optimise the power consumption of the whole genome by reducing the total number of inter- and intra-switches. In the best case the proposed approach improves power consumption over classical Huffman code by x, in the worst case by x, and on an average by x. The proposed approach optimise the total number of switches by swapping finding the best arrangement of nodes on the codewords tree with the best allocation of these codewords to the different frequencies (triplet). The approach continue swapping between the population individuals until a balance is found between inter-switches and intra-switches.

Table 1: Datasets description

Data sets	**Name**	**Size (MB)**	**Reference**
Genome 1	Mycobacterium smegmatis	6.66	CP009496
Genome 2	Amycolatopsis benzoatilytica	8.30	NZ_KB912942
Genome 3	Mycobacterium rhodesiae NBB3	6.11	CP003169
Genome 4	Streptomyces bottropensis ATCC 25435	8.54	NZ_KB911581
Genome 5	Mycobacterium smegmatis str. MC2 155	6.66	CP009494
Genome 6	Mycobacterium smegmatis MKD8	6.76	NZ_KI421511
Genome 7	Bradyrhizobium WSM471	7.42	NZ_CM001442
Genome 8	Amycolatopsis thermoflava N1165	8.27	NZ_CM001442
Genome 9	Bacillus thuringiensis Bt407	5.74	NZ_CM000747
Genome 10	Bacillus thuringiensis serovar thuringiensis	6.03	NZ_CM000748
Genome 11	Pseudomonas aeruginosa 9BR	6.48	NZ_AFXI01000001
Genome 12	Bacillus thuringiensis serovar berliner ATCC	5.97	NZ_CM000753
Genome 13	Bacillus thuringiensis serovar pakistani	5.75	NZ_CM000750
Genome 14	Pseudomonas aeruginosa LES400	6.28	CP006982
Genome 15	Mus musculus chromosome 1	25.58	GL456087
Genome 16	Danio rerio chromosome 1	56.14	CM002885
Genome 17	Homo sapiens chromosome 18	76.64	CM000680
Genome 18	Homo sapiens chromosome 22	99.94	CM000684

Table 2: Comparison of the total number of switches (in millions) between the Huffman code and the proposed approach

Datasets	Switches in the Huffman Code	Switches in the proposed approach	Improvement (%)
Genome 1	18.16	10.05	44.65
Genome 2	24.66	15.63	36.61
Genome 3	18.46	10.66	42.25
Genome 4	25.18	16.26	35.42
Genome 5	19.24	16.08	16.42
Genome 6	19.61	14.96	23.71
Genome 7	22.40	13.21	41.02
Genome 8	23.87	17.70	25.84
Genome 9	17.66	10.04	43.14
Genome 10	18.14	9.89	45.47
Genome 11	19.63	11.48	41.51
Genome 12	17.59	11.86	32.57
Genome 13	17.39	10.93	37.14
Genome 14	18.53	11.64	37.18
Genome 15	78.19	50.60	35.28
Genome 16	168.98	102.13	39.56
Genome 17	217.68	130.93	39.85
Genome 18	243.70	149.80	38.53

5. Conclusion

In this paper, we showed how switching activities affect the power consumption in digital communication. After that, a relationship between the Huffman code and the switching activities is shown. Genetic algorithm is identified as a potential method to optimise the total number of switching: inter- and intra-switching in Huffman code. Subsequently, an approach is proposed to show how the genetic algorithm can be used to reduce the power consumption in biological data transmission by reducing total number of switching in the transmission. The proposed approach starts its operation by generating a set of feasible solutions (Huffman trees), each one contains a set of codewords for the input symbols. After that, the approach uses a set of natural genetic operators to improve the quality of the solutions by generating new solutions. The evolution of the population is controlled by the total number of switching. The advantage of this approach is that it minimises the total number of switching considerably, thus increases the power efficiency data transmission. The effectiveness of the proposed approach is evaluated by applying it to several biological data sets with different sizes varied from small genome to a human chromosome. The performance of the approach has been compared with the performance of the classical Huffman code. The results show that the proposed approach reduces the total number of switching significantly to improve the power efficiency of biological data transmission. In the future, we hope to improve the allocation method of codewords to different symbols to reduce further the total number of switching.

References

[1] A. P. Chandrakasan, R. W. Brodersen, Minimizing power consumption in digital cmos circuits, Proceedings of the IEEE 83 (4) (1995) 498–523.

[2] S. Gregori, Y. Li, H. Li, J. Liu, F. Maloberti, 2.45 GHz power and data transmission for a low-power autonomous sensors platform, in: Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED '04), 2004, pp. 269–273.

[3] D. Yates, A. Holmes, A. Burdett, Optimal transmission frequency for ultralow-power short-range radio links, IEEE Transactions on Circuits and Systems 51 (7) (2004) 1405–1413. doi:10.1109/TCSI.2004.830696.

[4] L. Wang, A. Ukhanova, E. Belyaev, Power consumption analysis of constant bit rate data transmission over 3g mobile wireless networks, in: 11th International Conference on ITS Telecommunications (ITST), 2011, pp. 217–223. doi:10.1109/ITST.2011.6060056.

[5] W. Toorisaka, G. Hasegawa, M. Murata, Power consumption analysis of data transmission in ieee 802.11 multi-hop networks, in: The Eighth International Conference on Networking and Services (ICNS), 2012, pp. 75–80.

[6] D. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the Institute of Radio Engineers 40 (9) (1952) 1098–1101. doi:10.1109/JRPROC.1952.273898.

[7] M. J. Golin, C. Mathieu, N. E. Young, Huffman Coding with Letter Costs: A Linear-Time Approximation Scheme, SIAM Journal on Computing 41 (3) (2012) 684–713.

[8] S. Kabir, T. Azad, A. S. M. A. Alam, M. Kaykobad, Effects of unequal bit costs on classical huffman codes, in: 17th International Conference on Computer and Information Technology, 2014, pp. 96–101.

[9] C.-Y. Chen, Y.-T. Pai, S.-J. Ruan, Low power huffman coding for high performance data transmission, in: International Conference on Hybrid Information Technology (ICHIT), 2006, pp. 71–77. doi:10.1109/ICHIT.2006.253467.

[10] N. H. E. Weste, K. Eshraghian, Principles of CMOS VLSI design: a systems perspective, Addison-Wesley, 1988.

[11] M. Pedram, Power minimization in IC design: principles and applications, ACM Transactions on Design Automation of Electronic Systems (TODAES) 1 (1) (1996) 3–56.

[12] T. H. Cormen, C. Stein, R. L. Rivest, C. E. Leiserson, Introduction to Algorithms, 2nd Edition, McGraw-Hill Higher Education, 2001.

[13] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. S. Pierre, S. Twigger, O. White, S. Y. Rhee, Big data: The future of biocuration, Nature 455 (2008) 47–50.

[14] M. C. Brandon, D. C. Wallace, P. Baldi, Data structures and compression algorithms for genomic sequence data, Bioinformatics 25 (14) (2009) 1731–1738.

[15] C. Scott, L. Yiming, L. Chen, X. Xiaohui, Human genomes as email attachments, Bioinformatics 25 (2) (2009) 274–275. doi:10.1093/bioinformatics/btn582.

[16] C. Wang, D. Zhang, A novel compression tool for efficient storage of genome resequencing data, Nucleic acids research 39 (7) (2011) e45–e45.

[17] H. Chan, P. Mazumder, K. Shahookar, Macro-cell and module placement by genetic adaptive search with bitmap-represented chromosome, Integration, the VLSI journal 12 (1) (1991) 49–77.

[18] M. Bright, T. Arslan, Genetic framework for the high level optimisation of low power vlsi dsp systems, Electronics Letters 32 (13) (1996) 1150–1151.

[19] J. P. Cohoon, W. D. Paris, Genetic placement, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 6 (6) (1987) 956–964.

[20] M. Mitchell, An introduction to genetic algorithms, MIT press, 1998.

[21] H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore, J. Darnell, Molecular Cell Biology, 4th Edition, W.H.Freeman and Co Ltd, 1999.

[22] T. Blickle, L. Thiele, A comparison of selection schemes used in genetic algorithms (1995).

[23] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, I. de la Iglesia, A. Perallos, Crossover versus mutation: A comparative analysis of the evolutionary strategy of genetic algorithms applied to combinatorial optimization problems, The Scientific World Journal 2014.

[24] H. Mühlenbein, How genetic algorithms really work: Mutation and hillclimbing., in: PPSN, Vol. 92, 1992, pp. 15–25.

[25] K. D. Pruitt, T. Tatusova, W. Klimke, D. R. Maglott, NCBI reference sequences: current status, policy and new initiatives, Nucleic acids research 37 (suppl 1) (2009) D32–D36.