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 con-

sumption, 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
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, in this paper, we focus on reducing total amount of switching in Huffman code to improve power efficiency of digital data processing.

The

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: *retur***n** $EXTRACT - MIN(Q)$

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. 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. The tree is traversed from the root to the leaf nodes to obtain the unique codeword (binary string) for different symbols. 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. If the structure of the tree is changed, the codewords for the

symbols will also change. The switching activity depends on the structure of the codewords. For example, if a symbol is assigned a codewrod '000' then there is no switching, on the other hand, if the codeword '010' is assigned to a symbol the transmission of this symbol requires 2 switching activity.

2.3. Issues in Biological data transmission

The size of biological data base increase with an expanding rate and it doesn't stop for increasing. In the last decade the use of biological data history increase which evolve a high request to data from this high volume biological data base [13]. The exponential growth of these databases become a big problem to all biological data processing methods [13]. Data compression methods is the most used methods to evolve the reduce the cost of treatment of high volume data bases. 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 for coding only the difference between the target genome and the referential one, the coding method used is Huffman. Recent work on data compression for biological data have been proposed to deal with transmission of genomic data as an email attachment [16]. Cost of data transmission is based on two point, firstly the size of the data which has been improved considerably in the recent year with the improvement of Huffman code. Secondly, is the number of switches on the generated codes. The power consumption increase linear with the number of switches.

3. GA-SO : Genetic algorithm for Switches Optimising

Genetic algorithm (GA) is one the of the most popular bio-inspired metaheuristic algorithm inspired from the natural evolution of species [17]. 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.

3.1. Preparing data and population generation

A genome is an expansion of DNA sequence which is a set of nucleotides that encodes a protein.These nucleotide are bounded together and each order define a specific protein. The transformation of DNA sequence to protein

is made by the cellular machinery, each three nucleotides are interpreted to a specific amino acid this three nucleotide are called triplets [18]. Huffman algorithm read the whole message to compute frequencies, the proposed algorithm read the whole genome and cut it into triplet to generate the frequencies of the triplet and by the way to generate the adjacent matrix which represents the frequencies precedence between triplet. The initial population is generated randomly, each solution on the population represents a tree that generates a set of codes equal to the total triplets number.

3.2. 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 [19]. In this approach the process of natural selection is maintained. First we generate a random pair numbers *rand* between 2 and the population size, this number represent how many parents will be processed. After that, an unbiased random selection of *rand* individuals (solutions) from the population is maid.

3.3. Crossover

The main operator of the GA is the crossover, allows to construct new solutions from the selected part of the population [20]. The selected solutions are ranked by fitness and crossover two by two from high fitness to low. The main objective of the crossover is to benefits from the two good solutions in order to generate a better solution. An internal node for each tree (solution) is selected ($node1$,$node2$), these two nodes must have the same number of leaf nodes (contain the same number of codes). The crossover operation will create two new trees (solutions), the first child (second) contain the nodes that are not children of the $node1$ ($node2$) and we replace the children of $node1$ ($node2$) by the children of $node2$ ($node1$) (see fig.1).

3.4. Mutation

The mutation operator changes the positions of two leaf nodes of the generated children (result of the crossover) [21]; this introduce the diversity in the search process, this diversification strategy allow the algorithm do conference to the global optimum. The algorithm for this operator goes through the leaf nodes and change the position of two leaf nodes from a parent

to another and selects the ones to change according to a fixed mutation rate (see fig.1).

3.5. Population update

The crossover operation aims to generate new solutions from the current solution to built the second generation of the population. Furthermore, the mutation provide the diversity on the search space solution by changing the position of nodes on the same tree. After these two operations, the next step of the genetic algorithm is two to breed the new population (see fig.2). Firstly the new solutions (children) are added to the current population, after that the population is ranked by fitness. The algorithm remove the worst solutions until the initial size of the population is achieved. The algorithm genetic repeat these operators until the stopped criteria is achieved, in this algorithm the stopped criteria is when the objective function (number of switches) stop decreasing.

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

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

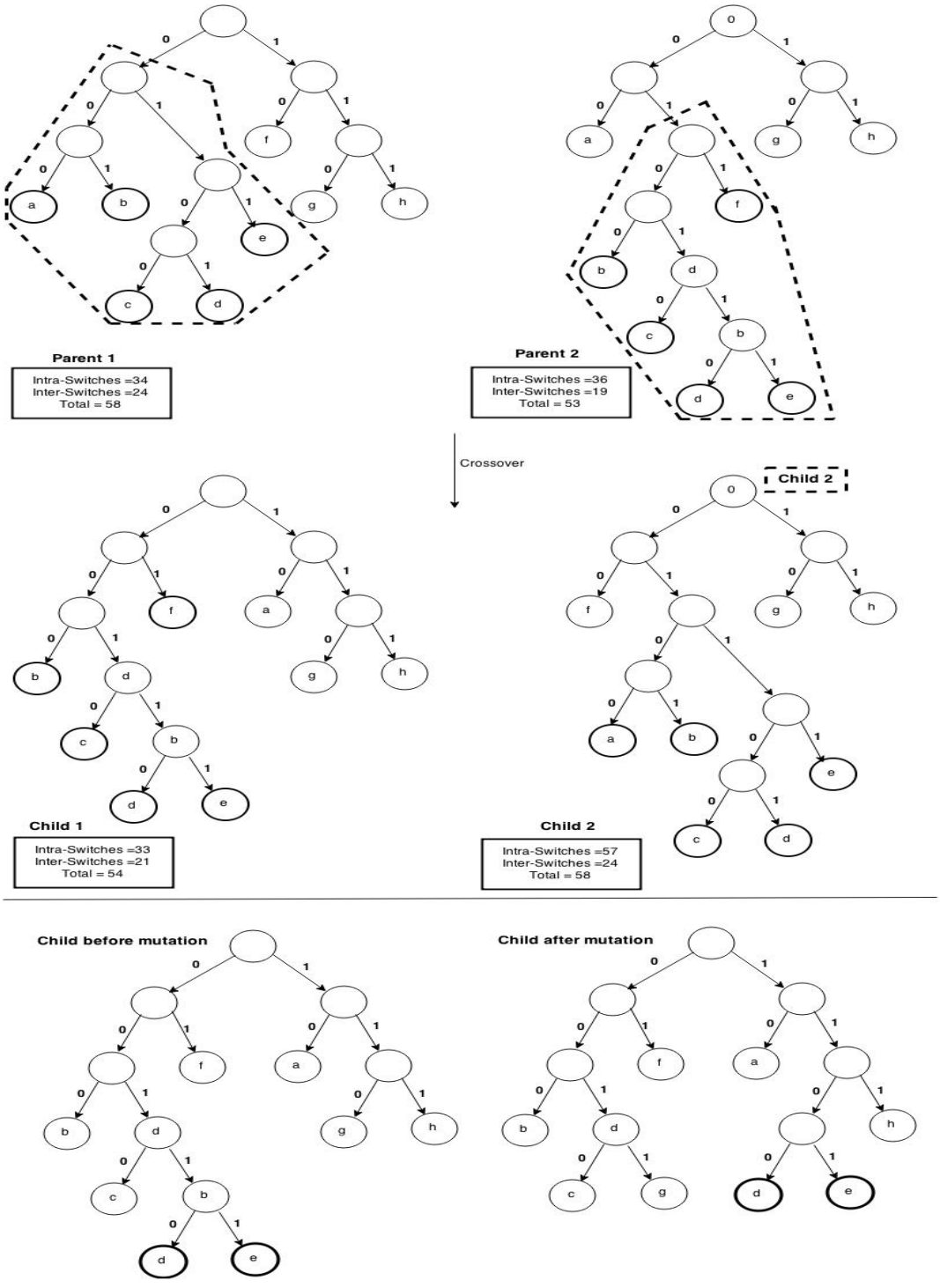

Figure 1: The Crossover operator

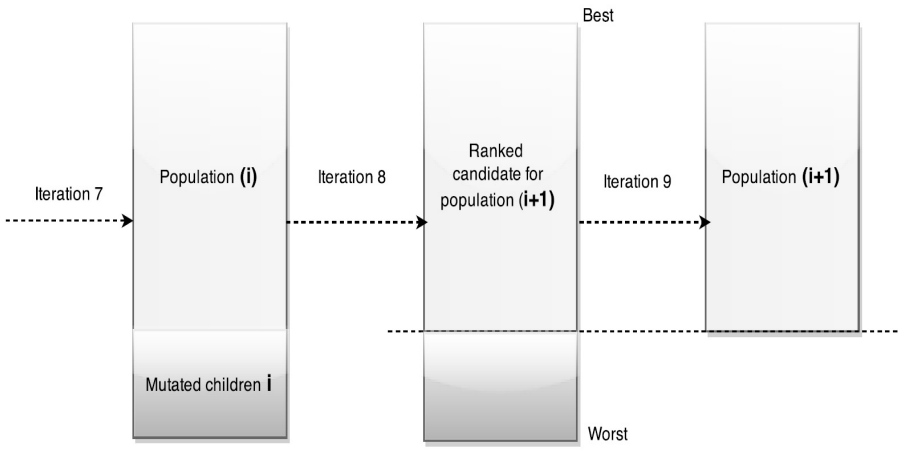

Figure 2: Population update for genetic algorithm

on ($http://www.ncbi.nlm.nih.gov$) [22]. 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- ad 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.

5. Conclusion

In this paper we described how the genetic algorithm can be used to improve the power consumption in biological data transmission. The advantage of the proposed approach is that it minimise considerably the number of switches with affect directly the power consumption for data transmission.

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 performance among classical Huffman code, CCA, and OCCA without penalty

	Huffman Algorithm	P	
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

In this paper, we identified genetic algorithm as a potential method to optimise the total number of inter- and intra-switches. The Genetic algorithm start by generating a set of feasible solutions (trees), each one contain a set of codewords. After that, the approach uses the natural genetic operators to improve the quality if the solutions by generating new solution. the evolution of the population is controlled by the number of switches both inter- and intra-switches. The proposed approach is applied to biological data sets with different sizes from small genome to a human chromosome. The approach has been compared with the classic Huffman code. The results show that the proposed approach improves the power consumption rate considerably. The experiment shows that the proposed approach improves Huffman code in the best case with *best*, *average* in the average case, and *worst* case. For the future we hope to improve the allocation of codewords to different frequencies

in orderer to improve the total number of switches.

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

[18] 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.

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

[20] 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.

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

[22] 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.