International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7 Issue-5, January 2019

Two Stage Block Truncation Coding for Lower Mean Square Error

N Shylashree

Abstract*: *In basic Block Truncation Coding, the given data sequence is encoded into a two level quantized approximation. This results in two segments one of which is comprised of the high level value and the other one, the low level value. In the proposed scheme, the basic Block Truncation Coding is refined by further sub segmenting the present segments. This increases the granularity of the data and reduces the truncation error.

Keywords: Two stage Block Truncation Coding, Binarization.

I. INTRODUCTION

Block Truncation Coding (BTC) was introduced by E. J. Delp and O. R. Mitchell in 1979 [1]. Several improved and modified versions of BTC have been developed [2-13]. It is a lossy compression technique that is generally used in image processing [14-24]. In basic Block Truncation Coding (BTC), the given data set is represented by a two level quantized approximation. The basic BTC operation effectively reduces the data size but contributes high quantization error. To reduce the quantization error, extend the basic BTC to Two Stage Block Truncation Coding (TSBTC).

II. BASIC BLOCK TRUNCATION CODING

The input data sequence X for the basic BTC is represented by a one dimensional array of N elements as,

$$\boldsymbol{X} = [x(1),\ x(2), \ldots x(N)] \quad (1)$$

In BTC, the input data will be split into two distinct non overlapping segments.

The basic BTC encoding operation involves Binarization and calculation of the high mean H and the low mean L.

Binarization: Let the mean of X be μ. The Binarization of array X is obtained as,

$$(i)=\begin{cases} 1 & \text{if } x(i) \geq \mu \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

For i= 1,2,...,N

The binary elements form the binarized array B as,

$\boldsymbol{B}$ = [b(1), b(2),..., b(N)] (3)

The elements of B are 0's and 1's. Those elements of X which are greater than or equal to u(X) form the segment SH and the other elements form the segment SL. The Segment SH corresponds to 1's of B and SL corresponds to the zeros of B.

2) Low Mean and High Mean values: The Low Mean value L and the High Mean value H are the two important scalar values of the BTC operation. The values of L and H, are calculated as [19],

$$L = \mu - \sigma\sqrt{\frac{q}{N-q}} \quad (4)$$

$$H = \mu + \sigma\sqrt{\frac{N-q}{q}} \quad (5)$$

Here, q is the number of 1's in array B and σ is the standard deviationof array X. In BTC, The values q and (N−q) should be greater than zero, otherwise H and L cannot be calculated and then, the BTC does not exist.

3) BTC Reconstruction (Decoding) of the Data: The BTC decoder receives *B*, *H* and *L*. Then, it reconstructs the approximate data array C, from array B using L and H as,

$$c(i)=\begin{cases} L & \text{if } b(i)=0 \\ H & \text{if } b(i)=1 \end{cases} \quad (6)$$

for i =1 to N.

Here c(i)is the i^{th} element of the reconstructed quantized array C. Array C is the sequence of c(i)'s as,

$$\boldsymbol{C} = [c(1), c(2), \ldots, c(N)] \quad (7)$$

Therefore, the components required to reconstruct C are, array B of length N and values L and H.

In BTC, values L and H are chosen such that the mean and the second moment of C and X are same. Because of this, the standard deviations of C and X are also same. Equations (4) and (5) are derived based on the above requirement. In BTC, the reconstructed (decoded) data is a two level quantized approximation of the original data. Higher quantization level is H and the lower quantization levelis L.

The basic (single stage) BTC operation is shown in Fig. 1. It requires XH, and XL to extend the BTC to the second stage. A simple numerical example is presented to demonstrate the working of the basic BTC.

Example 1: The data sequence is taken as

X = [48 49 46 50 46 41 43 46]

Here the number of elements N=8. The mean and the standard deviation are,

μ = 46.125 and σ = 2.80. Therefore from (2), B is given by,

B = [1 1 0 1 0 0 0 0]

Then, using (4) and (5), L and H are calculated as,

L = 43.95 and H = 49.74. Then, from (6), C is obtained as,

Revised Manuscript Received on 30 January 2019.

* Correspondence Author

N Shylashree*, Associate Professor, Department of Electronics & Communication Engineering, Rashtreeya Vidyalaya College of Engineering Bengaluru, Karnataka, India

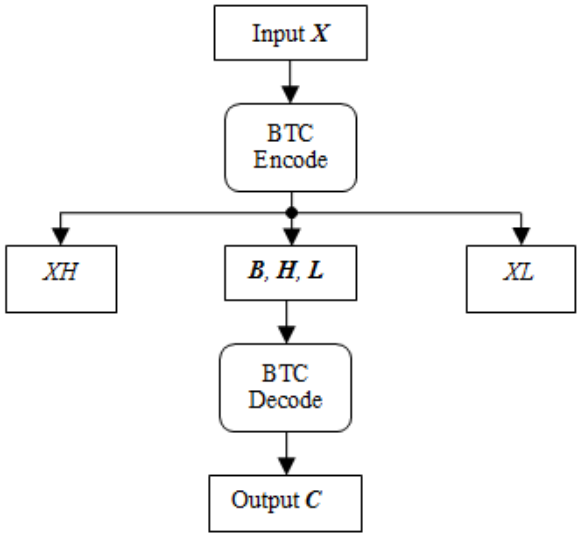

Fig.1. Single Stage Block Truncation Coding

C = [H, H, L, H, L, L, L, L]

Approximating L and H to their nearest integers, Thus,

C = [50 50 44 50 44 44 44 44].

A. High and Low level segments of X

While observing the binarized array B, notice that the bits of B classify the input data elements into two segments. Those elements of X which are greater than or equal to mean(X) form the high level segment and the remaining elements of X which are less than mean(X) form the low level segment. The former is represented by array XH and the later by array XL. Thus,

If $x(i) \geq \text{mean}(X)$, $x(i) \in XH$ else $x(i) \in XL$ (8)

Thus a in a single stage BTC, the input data X is split into two segments. The high level segment XH and the low level segment XL are shown in Fig. 1.Arrays XH and XL are used to extend the BTC to the second stage.

B. Single stage BTC encoding as a function

Represent the Basic Single Stage BTC encoding operation as a function so that it can be called to generate multi-stage BTC. The input to the function is the given data array X which is to be encoded. The function is called btc_encode(X). The outputs (returned values) of the function are the binary array B along with the Higher mean and the lower mean values H and L. The function is given below. The function and the statements are in Matlab style. % represents a comment line

```
%function btc_encode(X)
function[B, H, L]=btc_encode(X)
 % Calculate the mean of X as,
μ  = mean(X);
    % Calculate standard deviation as [20]
sd = std(X, 1);
    % Get the binary array B (same sizes as that of X) as,
       B = (X > = μ);
    % Get the length of X and the number of 1's in B as,
N=length(X);
q =sum(B);
   % Calculate H and L as,
H=  μ+sd*sqrt((N-q)/q));
L=μ-sd*sqrt(q/(N-q));
% B, H and L calculations over.
return[B, H, L];
```

Thus the first stage function is expressed as,

[B, H, L]=btc_encode(X) (9)

C. Single stage decoding function

The objective of basic BTC decoding is to construct array C from inputs B, H and L. Therefore the decoding function is named as btc_decode(B, H, L).The truncated output array C is reconstructed using (6).Equation (6) can be expressed by a single equation as,

c(i) = H*b(i) + L*(1− b(i)) (10)

for i =1 to N. Using (10), the decoding function is written as follows.

```
%function decode_btc(B, H, L)
function = decode_btc(B, H, L)
for i = 1:N
c(i) = H*b(i) + L*(1− b(i));
end
return C;
```

It also require a function to get the two segments XH and XL from X and B. This is named as get_segs(X).The XH and XL are obtained from (8). The function is written as follows.

```
%function get_segs(X)
function[XH, XL]= get_segs(X)
    j=1;k=1     %indices for XH and XL
for i = 1:N
if x(i)> = mean(X)
XH(j) = x(i);
        j=j+1;
else
XL(k) = x(i);
        k=k+1;
end
end  %endfor
return[XH, XL]
```

Arrays XH and XL are also generated and kept ready to launch the second stage.

The most important characteristic of BTC is, to preserve the mean and standard deviation before decoding and after decoding as,

$$\text{mean}(C) = \text{mean}(X)$$
$$\text{std\_dev}(C) = \text{std\_dev}(X) \quad (11)$$

B. Data compression Ratio

Let the size of each data element x(i) be m bits. Then without BTC, the total data size of N elements of X in bits is,

Data Size without BTC = N*m (12)

Assuming the sizes of H and L also m bits each and with the binary vector B of size N bits, the total size of encoded data is,

Data Size with BTC = 2*m +N bits (13)

Therefore the Compression Ratio is,

$$CR = \frac{\text{Data size without BTC}}{\text{Data size with BTC}} = \frac{N*m}{2*m+N} \quad (14)$$

III. TWO STAGE BTC

The basic BTC is extended as shown in Fig. 2. The extended BTC is designated as Two Stage BTC (TSBTC). In Fig.2, the first stage is same as shown in Fig. 1. In the second stage, XH and XL are split into two sub segments each as XHH, XHL and XLH, XLL. Thus In the second stage, 4 data sub segments are generated with 4 distinct levels.

Derivation of second stage segments

The derivation of the second stage segments is similar to the Basic BTC. Consider the decomposition of segment XH. Next, call the function btc_encode with array XH to get the outputs BH, HH and LL as,

[BH, HH, HL]= btc_encode(XH) (15)

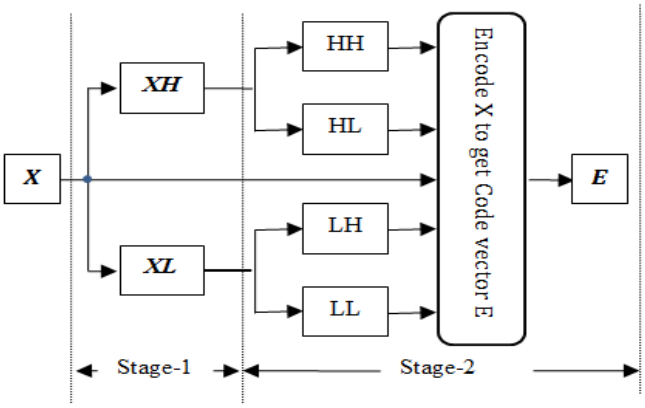

Fig. 2. Two stage BTC encoding

Here, BH is the binary array of size q, whereas HH and HL are the high level and the low level means of XH respectively. Similarly the second level BTC encoding of XL is obtained as,

[BL, LH, LL]=btc_encode(XL) (16)

Here, BL is the binary array of size (N−q), whereas LH and LL are the high level and the low level means of XL respectively. Thus all the four levels, HH, HL, LH and LL are ready.

B. Code vector in Two Stage BTC

The Two Stage BTC (TSBTC), consists of 4 quantized levels HH, HL, LH and LL in D. Therefore these 4 numerical values are represented by 4 distinct symbols. Two bits are needed to represent these 4 distinct symbols. By using unsigned integers 3, 2, 1 and 0 as the symbols to represent HH, HL, LH and LL respectively. The array that carries these symbols is designated by E. Thus E is the code vector. The elements of E are obtained as,

$$\begin{cases} e(i) = 3 \quad \text{if } x(i) \geq \text{mean}(\mathbf{XH}) \\ e(i) = 2 \quad \text{if } x(i) \geq \text{mean}(X) \ \&\& \ x(i) < \text{mean}(XH) \\ e(i) = 1 \quad \text{if } x(i) \geq \text{mean}(XL) \ \&\& \ x(i) < \text{mean}(X) \\ e(i) = 0 \quad \text{if } x(i) < \text{mean}(XL) \end{cases} \qquad (17)$$

for i =1 to N. Now E = [e(1), e(2),…, e(N)]

C. Decoding in TSBTC to get the output vector

Designate the output vector as D. The size of D is N (same as that of X) and it contains the elements HH, HL, LH and LL in locations corresponding to the locations of 3, 2, 1 and 0 in E, respectively. D is obtained from E by the function tsbtc_decode(…)as follows.

```
%function tsbtc_decode(E, HH, HL, LH, LL)
% D = [d(1), d(2), …,d(N)]
functionD= tsbtc_decode(E,HH,HL,LH,LL)
fori = 1:N
if e(i) = 3  d(i) = HH end
if e(i) = 2  d(i) = HL end
if e(i) = 1  d(i) = LH end
if e(i) = 0  d(i) = LL end
end  %endfor
return D
```

F. Example for TSBTC

A simple example with N = 16 is presented.

Example 2: The input data is,

X = [16 12 4 6 16 11 18 18 13 5 17 1 16 8 9 16]

Calculations:

First stage:

μ = mean(x) = 11.625.

σ = standard_deviation(X)=5.3371.

B using (2) is found to be,

B =[1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1]

q = sum(B) = length(SH) = 9.

N−q = length(sL) = 7.

Using (5) and (4), H and L are found to be,

H = 16.3319 and L = 5.5733.

From (6) and using integer approximations for H and L, C is found to be

C=[16 16 6616 6 16 16 16 6 16 6 16 6 6 16]

Using get_segs(X) function, XH and XL are found to be,

XH = [16 12 16 18 18 13 17 16 16]

XL = [4 6 11 5 1 8 9]

SecondStage:

Mean and standard deviation of XH and XL are found to be,

mean(XH) =15.7778.standard_deviation(XH) = 1.9309.

mean(XL) = 6.2857.standard_deviation(XL) = 3.1037.

Results of[BH, HH, HL]=btc_encode(XH) are found to be,

BH = [1 0 1 1 1 0 1 1 1] (Not used further)

HH =16.8099 and HL = 12.1654

Results of[BL, LH, LL]=btc_encode(XL) are found to be,

BL = [1 0 1 1 1 0 1 1 1] (Not used further)

LH =9.8695 and LL = 3.5979

Calculation of E using (17) gives,

E= [3 2 0 0 3 1 3 3 2 0 3 0 3 1 1 3]

Output vector D is found using function tsbtc_decode(…) as,

D=[HH, HL, LL, LL, HH, LH, HH, HH,…
HL, LL, HH, LL, HH, LH, LH, HH]

G. Data Compression Ratio in TSBTC

Assuming m bits for the elements of X,

Data Size without BTC = N*m bits

In TSBTC, the encoded data are array E of N elements with 2 bits per element. Therefore the size of E is 2*N bits. The scalars HH, HL, LH and LL are also generally represented by by m bits each. Therefore the data size of these scalars are 4*m. therefore the total size of the encoded data is,

Size of TSBTC encoded data = 2*N+4*m (18)

Therefore the compression ratio is given by,

$$\text{CR of TSBTC} = \frac{N*m}{4*m+2*N} \quad (19)$$

Comparing Eqs. (14) and (19),

CR of TSBTC = 0.5* CR of BBTC (20)

III. MEAN AND STANDARD DEVIATION OF ARRAY D

Vector D has N elements with 2 main segments DH, DL. These are sub divided into two sub segments each to get a total of 4 sub segments called DHH, DHL, DLH and DLL as shown in Fig. 3.Here, DH ={DHH, DHL}.The corresponding elements are HH's and HL's. The other Segment DL = {DLH, DLL}. Its elements are LH's and LL's. The sizes of the sub segments and the main segments are shown in Fig. 3.

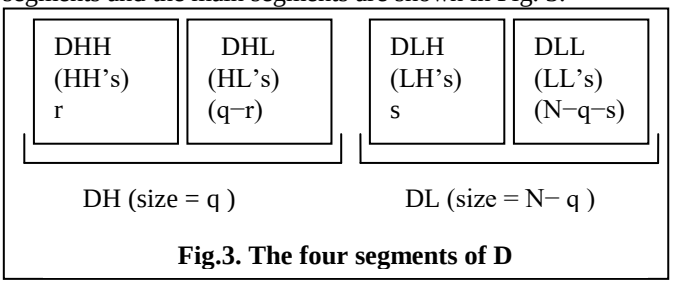

Fig.3. The four segments of D

In TSBTC, segment DH is derived from XH using BTC. Therefore, from the principle of (14),

mean(DH) = mean(XH) (21)

std_dev(DH) =std_dev(XH) (22)

Similarly,

mean(DL) = mean(XL) (23)

std_dev(DL) =std_dev(XL) (24)

Now, consider D which is the aggregation of DH and DL as,

D = { DH DL} (25)

The size of DH is q and that of DL is (N−q). The size of D is N. Therefore the mean (weighted average) of D is

$$\text{mean(D)} = \frac{q*\text{mean(DH)}+(N-q)*\text{mean(DL)}}{N} \quad (26)$$

From Eqs. (21), (23) and (26),

$$\text{mean(D)} = \frac{q*\text{mean(XH)}+(N-q)*\text{mean(XL)}}{N} \quad (27)$$

Since X is the aggregation of XH and XL with sizes q and (N−q), the RHS of Eq. (27) is nothing but mean(X). Therefore,

mean(D) = mean(X) (28)

Similarly, it can be proved that the second moments of X and D are equal. That is,

second_moment(D) = second_moment(X) (29)

From Eqs. (28) and (29),

std_dev(D) = std_dev(X) (30)

V. COMPARISON WITH OTHER METHODS

The major disadvantage of BTC, when used for Image compression, is the blocking artefact distortion [23] which affects the visual quality of the image. Therefore, to reduce the blocking effect, error diffusion techniques are used along with BTC [22, 23]. These error diffusion techniques have higher time complexity compared to the basic BTC. The proposed TSBTC is mainly designed for non-image data, like environmental variables (temperature, humidity, air pollution level etc.) and information about commerce and financial data where the visual quality is immaterial. Hence, take up the comparison of TSBTC with other BTC variants where error diffusion techniques are not involved. The performance of TSBTC is compared with Single Stage BTC (SSBTC), 3-level and 4-level direct quantization methods [19-20]. The Mean Square Error (MSE) between the original data and its quantized output is used as the performance parameter. The input data is taken from the pixel values of image 'eight.tif', purely for the sake of data.For the purpose of comparison, the entire input data is taken in a single block. The size of the data vector X is (1x74536) and it is used as the input signal. Then Gaussian noise is added in steps such that the SNR is varied from 6 db to 30 db in steps of 3. At each step, the MSE values are calculated for different methods. The result is shown in Fig. 4. From Fig. 4, TSBTC has the lowest MSE compared to other methods. Also, TSBTC has the lowest noise immunity (rate of change of MSE with respect to the noise level).

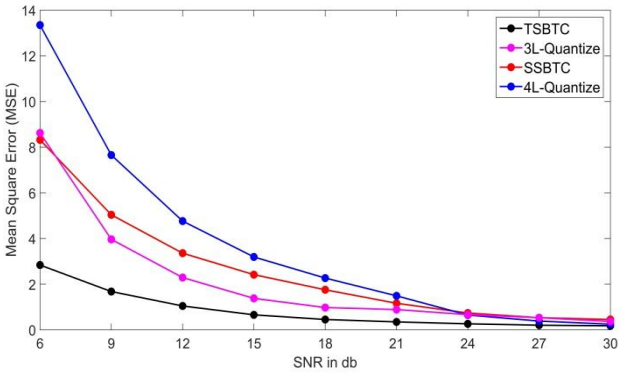

Fig.4. MSE vs SNR for different methods

VI. CONCLUSIONS

A new data compression technique is presented to achieve lower MSE with reasonably higher compression ratio. The MSE performance of the proposed method is not very much affected by the presence of Gaussian noise. The method is well suited for data having low spatial variation. The method can be easily extended to more than two stages.

REFERENCES

1. E. J.Delp, O. R. Mitchell, "Image Compression using Block Truncation Coding," IEEETransactions on Communications Vol. 27, no.9, pp.1335-1342, Sep. 1979.
2. D. Healy and O. R. Mitchell, "Digital Video Bandwidth Compression Using Block Truncation Coding," IEEETransactionsonCommunications, Vol. 29, no. 12, pp. 1809-1817, Dec. 1981.
3. B. Harjito and H. Prasetyo, "Image forensics using EDBTC feature," 2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), Taipei, 2017, pp. 167-168.

4. P. Yadav, R. Gupta and S. Kumar, "Content based image retrieval using Dither Block Truncation coding with similarity comparison algorithm," 2017 International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, 2017, pp. 489-493.
5. M. Elsayed, M. Mahmuddin, A. Badawy, T. Elfouly, A. Mohamed and K. Abualsaud, "Walsh transform with moving average filtering for data compression in wireless sensor networks," 2017 IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA), Batu Feringgi, 2017, pp. 270-274.
6. N. Patel and J. Chaudhary, "Energy efficient WMSN using image compression: A survey," 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, 2017, pp. 124-128.
7. Y. S. Chen and Y. T. Tsou, "Compressive Sensing-Based Adaptive Top-k Query over Compression Domain in Wireless Sensor Networks," 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, 2017, pp. 1-6.
8. H. Rekha and P. Samundiswary, "Image compression using multilevel thresholding based Absolute Moment Block Truncation Coding for WSN," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2016, pp. 396-400.
9. F. J. Yang, C. Y. Lien, P. Y. Chen and C. L. Hsu, "An Efficient Quadtree-Based Block Truncation Coding for Digital Image Compression," 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans-Montana, 2016, pp. 939-942.
10. Lema M.D and Mitchell O.R, " Absolute Moment BlockTruncation Coding and Application to Color Image", IEEETransactions on Communications, Vol. COM-32, 1984,pp. 1148-1157.
11. Halverson D, Griswold N.C and Wise G.L, "A generalized blocktruncation coding algorithm for image compression", IEEETransactions on Acoustics, Speech and Signal Processing, Vol. 32,1984.
12. Udpikar V.R and Raina J.P, "BTC image coding using vectorquantization," IEEE Transactions on Communications., vol. 35, no.3,1987,,pp. 352–356.
13. Yan Yanga, b, Qiqiang Chena, Yi Wana,"A fast near-optimum block truncation coding method using a truncated K-means algorithm and inter-block correlation," AEU - International Journal of Electronics and Communications.,Volume 65, Issue 6,2011, pp. 576–581.
14. PasiFränti, Olli Nevalainen and TimoKaukoranta," Compression of Digital Images by Block Truncation Coding: A Survey". The Computer Journal, 37 (4), 1994,pp. 308-332.
15. Doaa Mohammed, FatmaAbou-Chadi, "Image Compression Using Block Truncation Coding", Cyber Journals: Multidisciplinary Journals inScience and Technology, Journal of Selected Areas in Telecommunications(JSAT), February Edition, 2011.
16. Y. Wu and D. C. Coll, "Multilevel block truncation coding using a minimax error criterion for high-fidelity compression of digital images," in IEEE Transactions on Communications, vol. 41, no. 8, 1993,pp. 1179-1191.
17. Cho, Moonki& Yoon, Yungsup. (2013). BTC Algorithm Utilizing Multi-Level Quantization Method for Image Compression. Journal of the Institute of Electronics and Information Engineers. 50. 10.5573/ieek.2013.50.6.114.
18. Chandravadhana, S &Nithiyanandam, N. (2017). Least mean square error-based image compression using block truncation coding. International Journal of Information and Communication Technology. 11. 25. 10.1504/IJICT.2017.10006381.
19. Vimala, S., Uma, P., and Abidha, B. Improved adaptive block truncation coding for image compression. International Journal of Computer Applications (0975–8887) Volume (2011).
20. Chan, K. W. and Chan, K. L. Optimization of multilevel block truncation coding. Signal Process.: Image Communication., vol 16, 2001, 445–459.
21. L. Hui, "An adaptive block truncation coding algorithm for image compression," International Conference on Acoustics, Speech, and Signal Processing, Albuquerque, NM, vol 4,1990, pp. 2233-2236.
22. J. Mathews and M. S. Nair, "Adaptive block truncation coding technique using edge-based quantization approach", Computers & Electrical Engineering, vol.43,2015, pp.169-179.
23. Y. F. Liu, J. M. Guo and Y. Cheng, "Adaptive block truncation coding image compression technique using optimized dot diffusion", 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, 2016, pp. 2137-2141.
24. Nandini K S and S.A.HariPrasad, "Optimal Spectrum Sensor Assignment in Multi-channel Multi-user Cognitive Radio Networks", International Journal of Telecommunications & Information Technology (JTIT), no. 4, 2018 (doi: 10.26636/jtit.2018.124017).