2009 Ninth International Conference on Intelligent Systems Design and Applications

A Multi-objective Evolutionary Approach to Data Compression in Wireless Sensor Networks

Francesco Marcelloni
Dipartimento di Ingegneria dell'Informazione
University of Pisa
Via Diotisalvi 2, 56122 Pisa - ITALY
f.marcelloni@iet.unipi.it

Massimo Vecchio
ASAP Research Group
INRIA Saclay, Ile de France sud
4, Rue J. Monod, 91893 Orsay Cedex - FRANCE
massimo.vecchio@inria.fr

***Abstract*—Energy is a primary constraint in the design and deployment of wireless sensor networks (WSNs) since sensor nodes are typically powered by batteries with a limited capacity. Since radio communication is, in general, the most energy hungry operation in a sensor node, most of the techniques proposed to extend the lifetime of a WSN have focused on limiting transmission/reception of data, for instance, through data compression. Since sensor nodes are equipped with limited computational and storage resources, enabling compression requires specifically designed algorithms. In this paper, we propose a lossy compressor based on a differential pulse code modulation scheme with quantization of the differences between consecutive samples. The quantization parameters, which allow achieving the desired trade-off between compression performance and information loss, are determined by a multi-objective evolutionary algorithm. Experiments carried out on three datasets collected by real WSN deployments show that our approach can achieve significant compression ratios despite negligible reconstruction errors.**

***Keywords*-Wireless sensor networks; data compression; multi-objective genetic algorithms; energy efficiency; signal processing;**

I. Introduction

A wireless sensor network (WSN) consists of a set of autonomous systems, called sensor nodes, communicating among themselves and deployed in large scale (from tens to thousands) for applications such as environmental, habitat and structural monitoring, disaster management, equipment diagnostic, alarm detection, and target classification. Each node is a small device able to collect information from the surrounding environment through one or more sensors, to elaborate this information locally and to communicate it to a data collection center called *sink* or *base station*, using generally node to node - multi-hop data propagation [1], [2].

To this aim, nodes are equipped with a *processing unit* with limited memory and computational power, a *sensing unit* for data acquisition from the surrounding environment and a *communication unit*, usually a radio transceiver. In general, each sensor produces a stream of data which has to flow from the sensor node itself to the sink. Further, nodes which act as routers in a multi-hop propagation have also to store data coming from other nodes and to forward them towards the sink. This requires to face with a main technological constraint: nodes are powered by small batteries which typically cannot be changed or recharged. Since radio communication is in general the main cause of power consumption, transmission/reception of data should be limited as much as possible. Data compression appears a very appealing and effective tool to achieve this objective.

Two approaches have been followed in the literature:

- to distribute the computational cost on the overall network [3], [4], [5];
- to enable compression acting at single node independently of the others [6], [7], [8].

The first approach is natural in cooperative and dense WSNs where data measured by neighbouring nodes are correlated both in space and in time. Thus, we can apply distributed transforms or estimate distributed models which allow decorrelating the data measured by sensors, and, therefore, representing these data by using fewer bits.

The second approach has been generally undertaken by adapting some existing dictionary-based compression algorithms to the constraints imposed by the limited resources available on the sensor nodes. For instance, the lossless compression algorithms proposed in [9], [6], [7] are, respectively, purposely adapted versions of LZ77, Exponential-Golomb code and LZW, respectively.

Lossless compression may be inefficient for sensors used in tiny commercial nodes. Indeed, such sensors, being generally cheap, collect measures affected by a considerable noise. Noise increases the entropy of the signal and therefore hinders lossless compression algorithms to achieve considerable compression ratios. The ideal solution would be to adopt on the sensor node a lossy compression algorithm in which the lost information just coincides with the noise. Thus, we could achieve high compression ratios without losing relevant information.

To this aim, we exploit the observation that data typically collected by WSNs are strongly correlated. Thus, differences between consecutive samples should be regular and generally very small. If this does not occur, it is likely

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.101

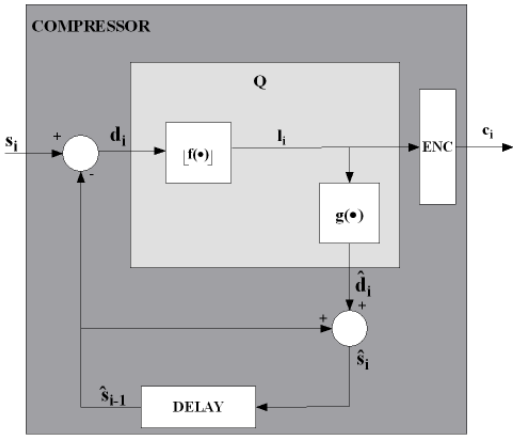
that samples are affected by noise. To de-noise and simultaneously compress the samples, we adopt a Differential Pulse Code Modulation (DPCM) scheme [10], often used for digital audio signal compression. The difference between consecutive samples is first quantized and then encoded by using an entropy encoder. Of course, different combinations of the quantization process parameters determine different trade-offs between compression performance and information loss. To generate a set of optimal combinations of the quantization process parameters, we adopt one of the most popular Multi-Objective Evolutionary Algorithms (MOEAs), namely NSGA-II [11].

To execute NSGA-II, we first collect a short sequence of samples from the sensor node. Then, we apply a popular de-noising technique to this sequence so as to obtain a sequence of de-noised samples. Each solution generated by NSGA-II is evaluated by quantizing the original samples and computing the information entropy of the quantized sequence as first objective, the number of levels used in the quantization process as second objective and the mean square error (MSE) between the quantized samples and the de-noised samples as third objective. The entropy and the number of levels provide an indirect measure of the possible obtainable compression ratios. The MSE quantifies the loss of information with respect to the ideal (not affected by noise) signal. Each solution in the Pareto front represents, therefore, a quantizer with an associated trade-off among information entropy, number of quantization levels and MSE between the original de-noised and the quantized sequences of samples. The user can therefore choose the combination with the most suitable trade-off for the specific application. We show that the lossy compression scheme obtained by using the quantizers generated by the MOEAs in the DPCM framework is characterized by low complexity and memory requirements for its execution. Further, it is able to compute a compressed version of each value on the fly, thus reducing storage occupation.

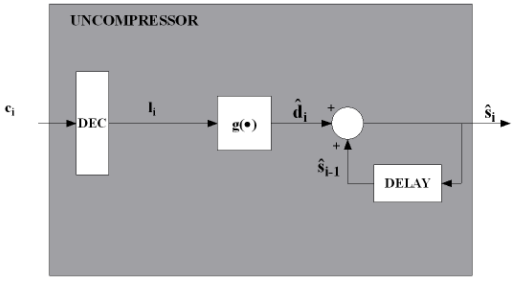
We have tested our lossy compression approach on three datasets collected by real WSN deployments. We show that, though very simple, our approach can achieve significant compression ratios despite negligible reconstruction errors, computed as MSEs between the original de-noised and the reconstructed signals. We have compared our approach with a lossy compression algorithm, namely LTC [8], specifically designed to be embedded in sensor nodes. We show that our approach outperforms LTC in terms of compression ratios, complexity (average number of instructions required to compress a sample) and reconstruction errors, thus representing a very interesting state-of-art solution to the problem of compressing noisy data in WSNs.

II. Our lossy compression scheme

Figure 1 shows the block diagrams of our compressor and uncompressor. As regards the compressor, the generic difference d_i is calculated by subtracting the most recent reconstructed value $\hat{s}_{i-1}$ from current sample s_i. To use $\hat{s}_{i-1}$ rather than the original value s_{i-1} avoids the well-known *accumulation of errors* problem [12]. Difference d_i is therefore input to the quantization block *Q*. Let $S = \{S_1, ..., S_L\}$ be a set of *cells* S_l, with $l \in [1..L]$, which form a disjoint and exhaustive partition of the input domain D (difference domain in our case). Let $C = \{y_1, \ldots, y_L\}$ be a set of levels $y_l \in S_l$, with $l \in [1..L]$. The $\lfloor f(\cdot) \rfloor$ block returns the index l_i of the cell S_{l_i} which d_i belongs to. The index l_i is input to the $g(\cdot)$ block, which computes the quantized difference $\hat{d}_i$, and to the entropy encoder *ENC*, which generates the binary codeword c_i [13]. Since environmental signals are quite smooth and therefore small differences are more probable than large ones, an entropy encoder results to be particularly performing. Indeed, these encoders encode more probable indexes with lower number of bits.



(a) compressor



(b) uncompressor

Figure 1. Block diagrams of (a) the compressor and (b) the uncompressor.

In the uncompressor, the codeword c_i is analyzed by the decoding block *DEC* which outputs the index l_i. This index is elaborated by the block $g(\cdot)$ to produce $\hat{d}_i$, which is added to $\hat{s}_{i-1}$ to output $\hat{s}_i$.

Given a uniform quantizer with cell width Δ, the region of the input space within $\Delta/2$ of some quantizer level is called the *granular region* or simply the *support* and

that outside (where the quantizer error is unbounded) is called the *overload* or *saturation region* [14]. When a good rate-distortion performance is requested to a quantizer, the zero-cell width is usually treated individually, even if the quantizer is uniform. Since each input within the zero-cell is quantized to 0, this cell is often called *dead zone* [15].

To guarantee a higher fexibility than a uniform quantizer, but without complicating too much the quantization rule, we split the granular region into two subregions. Then, we partition both the subregions uniformly with appropriate different cell widths. It follows that each quantizer is determined by the following five parameters: i) width of the dead zone (DW), ii) width of the cell in the first granular subregion (FW), iii) number of cells in the first granular subregion (FN), iv) width of the cell in the second granular subregion (SW), v) number of cells in the second granular subregion (SN).

To generate a set of different quantizers, we apply NSGA-II [11] guided by three objectives: the information entropy H, the quantization complexity C and the mean square error MSE_d between the quantized and the de-noised samples. Information entropy H provides an indirect measure of the possible obtainable compression ratios and is defined as:

$$H = -\sum_{l=1}^{L} p_l \cdot \log_2(p_l) \tag{1}$$

where p_l is the probability mass function of quantization index l.

Quantization complexity C is computed as the number L of distinct quantization levels (and consequently indexes) used in the quantizer:

$$C = 2 \cdot (FN + SN) + 1 \tag{2}$$

A lower value of C implies a lower number of symbols needed to encode the quantization indexes and therefore a lower number of bits in the binary codewords.

The MSE_d quantifies the loss of information with respect to the ideal (not affected by noise) signal and is defined as

$$MSE_d = \frac{1}{N}\sum_{i=1}^{N}(s_i - s_i^*)^2 \tag{3}$$

where N is the number of samples, and s_i and s_i^* are, respectively, the original and the de-noised samples.

To de-noise the samples, we have adopted the wavelet shrinkage and thresholding method proposed in [16]. We have used the Symmlet 8 wavelet, a level of decomposition equal to 5 and the soft universal thresholding rule for thresholding the detail coefficients at each level. The de-noising process has been performed by using standard Matlab built-in functions.

Each chromosome codifies the five parameters which define a quantizer. Each parameter is expressed as a positive integer in the range $[1, MAX]$ and is codified by a Gray binary code. The value of MAX depends on the resolution of the ADC on board the sensor node. On the other hand, to constrain the upper bound of the range reduces the search space and allows a better exploration. In our experiments we set $MAX = 64$: it follows that each chromosome is represented by a string of 30 bits. To compute H, C and MSE_d for each chromosome, we use a small set (*training set*) of samples collected by the sensor on board the node.

We apply classical one-point crossover and one gene mutation operators [17]. The crossover operator is applied with probability P_X; the mutation operator is applied with probability P_M. In the experiments, we adopted $P_X = 0.9$ and $P_M = 0.02$.

III. Experimental results

In order to show the effectiveness and validity of our lossy compression approach, we tested it against some real-world temperature datasets. In particular, we used temperature measurements collected from node 101 of the FishNet Deployment (FN101 for short), node 10 of the Grand-St-Bernard Deployment (GSB10) and node 20 of the Gènèpi Deployment (LG20). These nodes have been randomly extracted from the nodes used in the SensorScope deployments [18]. The WSNs adopted in the deployments employ a TinyNode node type [19], which uses a TI MSP430 microcontroller, a Xemics XE1205 radio and a Sensirion SHT75 sensor module [20]. This module includes a bandgap temperature sensor, which can sense air temperature in the $[-20^oC, +60^oC]$ range, coupled to an ADC and a serial interface circuit. Each ADC output raw_t is represented with resolution of 14 bits and normally converted into a measure t in Celsius degrees (^oC) as described in [20]. The datasets corresponding to the three deployments contain measures t. On the other hand, our algorithm works on raw_t data. Thus, before applying the algorithm, we extracted raw_t from t, by using the inverted versions of the conversion functions in [20].

Table I summarizes the main characteristics of the datasets. We used the first $N = 5040$ samples from FN101 dataset as training set (TRAIN). Since this dataset is a collection of temperature samples collected with a frequency of 1 sample each 2 minutes, it is equivalent to consider a 7-days training set. The extracted portion of the original signal is first de-noised and then converted back to raw data.

Deployment name	Symb. Name	Num. of samples	Time interval	
			From	To
FishNet	FN101	12651	09/08/07	31/08/07
Gr.St Bernard	GSB10	23813	15/09/07	18/10/07
Le Gènèpi	LG20	21523	04/09/07	03/10/07

Table I
MAIN CHARACTERISTICS OF THE THREE DATASETS.

We applied NSGA-II to the training set. We adopted a

population of 100 individuals and stopped the algorithm after 10000 iterations. At the end of the optimization process we obtained an archive of non-dominated solutions with respect to the three objectives. In particular, the objective C (quantization complexity) has allowed us to steer the search of the best solutions towards the ones with the minimum number of cells. Thus, during the evolution, the archive is populated preferably by quantizers which, having equal entropy and MSE_d, are characterized by a lower number of cells, thus avoiding to consider quantizers with a high number of unused cells. This allows simplifying the implementation of the quantizer and consequently of the encoder. Indeed, if the number of indexes is low, the encoder can use a small dictionary to encode the quantization indexes. This dictionary can be generated by using the Huffman's algorithm [21] which provides a systematic method of designing binary codes with the smallest possible average length for a given set of symbol occurrence frequencies. Once the binary codeword representation of each quantization index has been computed and stored in the sensor node, the encoding phase reduces to a look-up table consultation.

The only critic point of this approach is that the Huffman's algorithm requires to know the probability with which the source produces each symbol in its alphabet. To determine an approximation of these probabilities, we can exploit again the training set: for the specific quantizer, we compute the probability with which each quantization index occurs when quantizing the differences between consecutive samples of the training set and build the optimal dictionary for that data source by applying the Huffman's algorithm.

If we project the final archive on the $MSE_d - H$ plane (see Fig. 2), we realize that actually almost all solutions maintain the non-dominance property with respect to the H and MSE_d objectives: only 12 out of 100 solutions result to be dominated by one or more solutions in the archive. In the figure, dots and crosses represent, respectively, non-dominated and dominated solutions with respect to the H and MSE_d objectives. Non-dominated solutions in the $MSE_d - H$ plane are actually the solutions of interest. We can observe that the front is wide and the solutions are characterized by a good trade-off between H and MSE_d.

A. Selected Solutions and their validation

To perform an accurate analysis of some solution, we selected from the front in Fig. 2 three significant quantizers: solutions A and C characterized by, respectively, the highest H and MSE_d, and solution B characterized by a good trade-off between H and MSE_d. Table II shows the values of the five parameters which characterize the three selected quantizers.

Table III shows the quantization indexes, their probabilities and the codewords assigned by the Huffman's algorithm when the selected quantizers are used in the proposed scheme for compressing the training set.

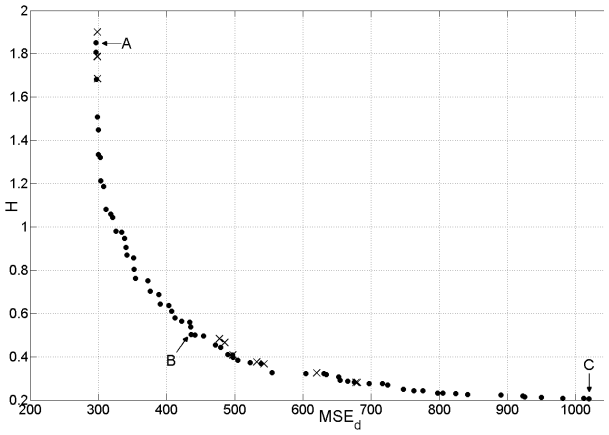


Figure 2. Projection of the Pareto front approximation on the $H - MSE_d$ plane.

Solution	DZ	FW	FN	SW	SN
A	8	1	3	2	1
B	32	15	1	5	1
C	63	61	1	44	1

Table II
PARAMETERS OF SOLUTIONS A, B AND C.

To assess the performances of the three compression algorithms generated by, respectively, the quantizers corresponding to A, B and C, we use the compression ratio defined as:

$$CR = 100 \cdot \left(1 - \frac{comSize}{origSize}\right) \tag{4}$$

where *comSize* and *origSize* represent the sizes of the compressed and original bitstreams, respectively.

Table IV shows the CR obtained for the three temperature datasets and the three selected quantizers. Further, the table reports the MSE between the original noisy and the reconstructed samples, denoted as MSE_n, and MSE_d. We can observe that all solutions achieve good trade-offs between compression ratios and MSEs. Further, there do not exist

index	Probability $[A, B, C]$	Codeword $[A, B, C]$
-4	$[0.1198, -, -]$	$[101, -, -]$
-3	$[0.0163, -, -]$	$[100011, -, -]$
-2	$[0.0200, 0.0052, 0]$	$[100001, 1011, 1111]$
-1	$[0.0252, 0.0311, 0.0131]$	$[10010, 11, 110]$
0	$[0.6309, 0.9258, 0.9730]$	$[0, 0, 0]$
1	$[0.0023, 0.0311, 0.0139]$	$[10011, 100, 10]$
2	$[0.0186, 0.0069, 0]$	$[100010, 1010, 1110]$
3	$[0.0206, -, -]$	$[100000, -, -]$
4	$[0.1258, -, -]$	$[11, -, -]$

Table III
CODEWORDS USED IN SOLUTIONS A, B AND C.

Dataset	Sol.	CR	MSE_n	MSE_d
TRAIN	A	87.8919	0.0120	0.0036
	B	92.9253	0.0206	0.0076
	C	93.4834	0.0785	0.0351
FN101	A	87.4679	0.0676	0.0431
	B	92.8128	0.0305	0.0097
	C	93.4264	0.0905	0.0384
GSB10	A	86.1882	0.0436	0.0071
	B	91.7356	0.0250	0.0043
	C	93.1495	0.0887	0.0275
LG20	A	85.2867	0.1200	0.0241
	B	89.7784	0.0422	0.0032
	C	92.4888	0.0882	0.0173

Table IV
RESULTS OBTAINED BY SOLUTIONS A, B AND C ON THE THREE DATASETS.

considerable differences between the results obtained in the training set and the ones achieved in the overall FN101 dataset and the other two datasets. This result could be considered enough surprising. Indeed, we highlight that both the optimization and the Huffman's algorithm were executed using only a portion of the FN101 dataset. Thus, GSB10 and LG20 datasets are completely unknown to the compression scheme. We are conscious that the procedure adopted for FN101 could have been exhaustively applied also to the other datasets, in order to find ad-hoc solutions for the particular deployment. On the other hand, the three temperature datasets were collected, though in different places and times, by the same sensor nodes with the same type of temperature sensor and the same sampling frequency: for this reason it could be unnecessary to perform the optimization on each dataset. To validate this assumption for each selected solution, we executed the Huffman's algorithm on a portion of $N = 5040$ samples extracted respectively from GSB10 and LG20 datasets and used the resulting dictionaries to compress the corresponding datasets. We verified that the increases in CR are very small and almost negligible, thus confirming the possibility of adopting the same encoding for similar applications of the same sensor.

Solution B, which was chosen on the knee of the Pareto front, is characterized by compression ratios comparable to those achieved by solution C and by MSE_d comparable to those obtained by solution A. This solution therefore represents a good trade-off between compression ratios and MSE_d. For this reason, we chose this solution to perform the comparisons discussed in the following subsection.

B. Comparison with LTC

To assess the effectiveness of our approach, we adopt the LTC algorithm proposed in [8]. LTC generates a set of line segments which form a piecewise continuous function. This function approximates the original dataset in such a way that no original sample is farther than a fixed error e from the closest line segment. Thus, before executing the

Dataset	CR $[min, max]$	MSE_n $[min, max]$	MSE_d $[min, max]$
FN101	$[92.23, 92.98]$	$[0.0390, 0.0450]$	$[0.0972, 0.0121]$
GSB10	$[90.90, 91.90]$	$[0.0540, 0.0600]$	$[0.0140, 0.0172]$
LG20	$[88.65, 89.51]$	$[0.0960, 0.1041]$	$[0.0203, 0.0229]$

Table V
CRS AND MSES ACHIEVED BY LTC ON THE THREE DATASETS.

LTC algorithm, we have to set error e. To determine the value e, which allows LTC to achieve the compression ratios obtained by our approach, we varied e from 0% to 200% of the Sensor Manufactured Error (SME) with step 10%. From the Sensirion SHT75 sensor data sheet [20], we have $SME = \pm 0.3^oC$ for temperature. We found the following intervals $e = [120, 130]$, $e = [150, 160]$ and $e = [200, 210]$ for, respectively, FN101, GSB10 and LG20.

Table V shows the intervals of CR, MSE_n and MSE_d obtained by LTC on the three datasets in correspondence to the intervals of error e. By comparing Table V with Table IV, we can observe that our algorithm obtains lower MSEs than LTC in correspondence to equal CRs. For example, for the FN101 dataset and CR around 92.81, MSE_d is between 0.0972 and 0.0121 for LTC, whereas $MSE_d = 0.0097$ for our algorithm.

Compression ratio and MSE are only two of the factors which determine the choice of a compression algorithm suited to WSNs. Another fundamental factor is complexity. To assess the complexity of our algorithm and of LTC, we have performed a comparative analysis on the number of instructions required by each algorithm to compress data. To this aim, we have adopted the Sim-It Arm simulator [22]. Sim-It Arm is an instruction-set simulator that runs both system-level and user-level ARM programs. For LTC, we have set e to the left extremes of the e intervals (we recall that the left extremes are the most favourable cases for the LTC algorithm). Table VI shows the numbers of instructions required for compressing each dataset and the numbers of instructions per saved bit for each temperature datasets and their average values. We note that, though our algorithm achieves lower MSEs at the same bitrate as LTC, it requires lower number of instructions. We observe that, on average, our algorithm executes 5.93 instructions for each saved bit against 40.43 executed by LTC.

Dataset	instr.		instr/saved bits	
	our	LTC	our	LTC
FN101	1065656	7440517	5.67	39.83
GSB10	2050525	13951039	5.86	40.26
LG20	1919529	12574212	6.20	41.18
average	**1678570**	**11321923**	**5.92**	**40.43**

Table VI
COMPLEXITY OF OUR ALGORITHM AND LTC.

IV. Conclusions

Reducing the amount of data transmitted/received by a sensor node is one of the most popular techniques to extend the lifetime of tiny battery-powered sensor nodes. Due to the limited resources available on board sensor nodes, purposely-designed algorithms have been adopted. In this context, we have proposed a lossy compression algorithm based on a differential pulse code modulation scheme with quantization of the differences between consecutive samples. To generate different combinations of the quantization parameters corresponding to different optimal trade-offs between compression performance and information loss, we have applied NSGA-II on a subset of samples collected by the sensor. We have tested our lossy compression approach on three datasets collected by real WSN deployments. We have shown that our approach achieves compression ratios up to 93.48% with very low reconstruction errors and outperforms LTC, a lossy compression algorithm purposely designed to be embedded in sensor nodes, in terms of both compression ratio and complexity.

References

[1] T. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multihop wireless sensor networks," in *Proc. 25th IEEE Int. Real-Time Systems Symposium*, Dec. 2004, pp. 359–370.

[2] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.

[3] C. Tang and C. S. Raghavendra, "Compression techniques for wireless sensor networks," in *Wireless sensor networks*. Norwell, MA, USA: Kluwer Academic Publishers, 2004, pp. 207–231.

[4] J.-J. Xiao, A. Ribeiro, Z.-Q. Luo, and G. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, Jul. 2006.

[5] Z. Xiong, A. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 80–94, Sep. 2004.

[6] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 411–413, Jun. 2008.

[7] C. M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *Proc. of the 4th Int. Conference on Embedded networked sensor systems (SenSys '06)*, 2006, pp. 265–278.

[8] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin, "Lightweight Temporal Compression of microclimate datasets," in *29th Annual IEEE Int. Conference on Local Computer Networks*, Nov. 2004, pp. 516–524.

[9] http://www.oberhumer.com/opensource/lzo/.

[10] C. C. Cutler, "Differential quantization of communication signals," Patent, Jul. 1952, 2 605 361.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[12] D. Salomon, *Data Compression: The Complete Reference*, 4th ed. London, UK: Springer-Verlag, 2007.

[13] J. O'Neal, "Differential pulse-code modulation (PCM) with entropy coding," *IEEE Trans. Inf. Theory*, vol. 22, no. 2, pp. 169–174, Mar. 1976.

[14] R. Gray and D. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[15] B. Tao, "On optimal entropy-constrained deadzone quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 560–563, Apr. 2001.

[16] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[17] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, 2nd ed. New York, NY, USA: Springer-Verlag New York, Inc., 1994.

[18] http://sensorscope.epfl.ch.

[19] http://www.tinynode.com.

[20] http://www.sensirion.com.

[21] D. Huffman, "A method for the construction of minimum-redundancy codes," *in: Proc. of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[22] http://simit-arm.sourceforge.net/.