K-RLE : A new Data Compression Algorithm for Wireless Sensor Network

Eugène Pamba Capo-Chichi, Hervé Guyennet
Laboratory of Computer Science - LIFC
University of Franche Comté
Besançon, France
{mpamba, herve.guyennet}@lifc.univ-fcomte.fr

Jean-Michel Friedt
FEMTO-ST/LPMO
University of Franche Comté
Besançon, France
jmfriedt@femto-st.fr

Abstract

In the context of the use of Wireless Sensor Network technology for environmental monitoring, the two main elementary activities of Wireless Sensor Network are data acquisition and transmission. However, transmitting/receiving data are power consuming task. In order to reduce transmission-associated power consumption, we explore data compression by processing information locally. In this article, we evaluate and compare compression algorithms on an ultra-low power microcontroller from Texas Instrument within the MSP430 series used for designing Wireless Sensor Network. We propose and evaluate a new data compression algorithm inspired from Run Length Encoding called K-RLE.

Keywords: wireless sensor network, data compression

1 Introduction

Recent technological breakthrough in low power processing units and communication devices have enabled the development of distributed autonomous nodes able to sense environmental data, compute and transmit it using wireless communication to a base station known as Sink for future analysis; thus, forming a Wireless Sensor Network [1].

However, Wireless Sensor Network is driven by a severe constraint which is power management. This power management has led researchers to explore scheduling sensor states. Scheduling sensor states is a technique that decides which sensor may change its state (transmit, receive, idle, sleep), according to the current and anticipated communications needs [2].

The most common technique for saving energy is the use of sleep mode where significant parts of the sensor's transceiver is switched off. As described in [4, 3], in most cases, the radio transceiver on board sensor nodes is the main cause of energy consumption: hence, it is important to keep the transceiver in switched off mode most of the time to save energy. Nevertheless, using the sleep mode reduces data transmission/reception rate and thus communication in the network. The question is how to keep the same data rate sent to the base station by reducing the number of transmission ?

In this article, we want to introduce in-network processing technique in order to save energy. In-network processing techniques allows the reduction of the amount of data to be transmitted. The well known in-network processing technique is data compression and/or data aggregation [5, 6]. Data compression is a process that reduces the amount of data in order to reduce data transmitted and/or decreases transfer time because the size of the data is reduced.

However, the limited resources of sensor nodes such as processor abilities or RAM have resulted in the adaptation of existing compression algorithm to WSN's constraint. Two main kinds of compression algorithms are available: lossless and lossy. The best known lossless compression algorithm for WSN is S-LZW [7].

Nevertheless, S-LZW which is an adaption for WSN of the popular LZW data compression algorithm is a dictionary-based algorithm. Compression algorithms based on dictionary require extensive use of RAM: such algorithms cannot be applied to most sensor platform configurations due to limited RAM. We hence introduce a generic data compression algorithm usable by several sensor platforms. In this article, we study the adaptation of a basic compression algorithm called Run Length Encoding (RLE) on an ultra-low power microcontroller product from Texas Instrument known as TI MSP430, within the specific framework of monitoring environmental temperatures.

The main problem of RLE is that compression results depend on data sources. In [7], a comparison between SLZW and RLE-ST has been done where RLE-ST is the application of RLE with a structured data set. However, in the present article, we make a comparison between S-LZW and RLE using experimental temperature datasets without reordering it and we propose a new algorithm named K-RLE

inspired from RLE in order to improve the compression results with different statistics of data sources.

This paper is organized as follows: the next section is the Related work, experimental results are given in Section 3 and the Section 4 is the conclusion.

2 The compression algorithms

A very popular lossless dictionary-based compression algorithm is LZW [8] which is a variant of LZ78. The best known data compression algorithm for WSN is S-LZW [7] which is a version of the previous popular algorithm LZW adapted for WSN.

2.1 S-LZW

The default parameters defined in S-LZW are:

- a block size of 528 bytes which represents two flash pages. S-LZW divides the uncompressed input bitstreams into fixed size blocks and compresses each block separately.
- a 512 entries dictionary. This algorithm starts by initializing the dictionary to all standard characters of the alphabet which represent the first 256 entries of the dictionary. For each block used in the compression, the dictionary is re-initialized. A new string in the input bitstream creates a new entry in the dictionary, that is why the data to be compressed are limited. However, different strategies have been developed in order to solve the problem of full dictionary. Two options exist which are to freeze the dictionary and use it as-is to compress the remainder of the data in the block, or it can be reset and started from scratch. However, this problem does not occur when the data block is small, thus the dictionary is not full.
- A mini-cache of 32 entries is added to SLZW in order to get an advantage of repetitiousness of sensor data. The mini-cache is a hash-indexed dictionary of size N, where N is a power of 2, that stores recently used and created dictionary entries.

This previous compression algorithm based on dictionary needs a significant RAM size that is why we cannot apply it on our platform which is a TI MSP430F149 with 2KB RAM. Indeed, authors in [7] have demonstrated S-LZW performances using the TI MSP430F1611 with 10 KB RAM. In this way, we are interested in the study of RLE.

2.2 Run-Length Encoding

Run-Length Encoding (RLE) is a basic compression algorithm. As described on [9], the simple idea behind this algorithm is this: If a data item *d* occurs *n* consecutive times in the input stream, we replace the *n* occurrences with the single pair *nd*.

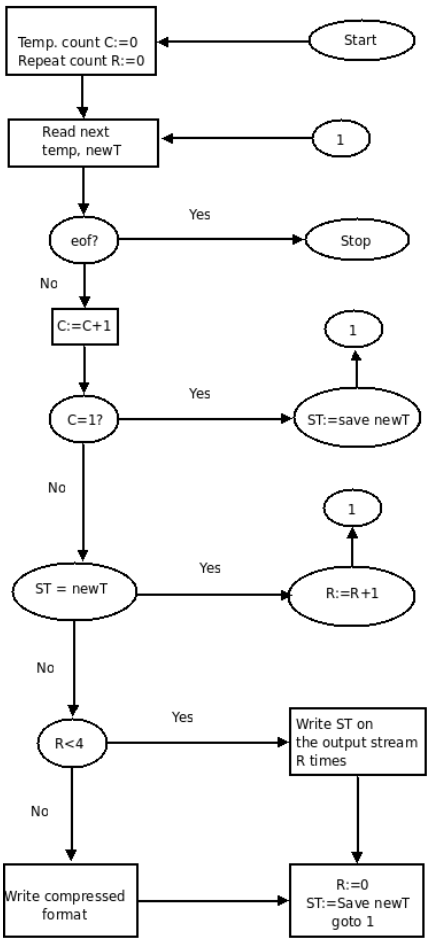

Figure 1. RLE compression algorithm

Fig.1 is the graphical representation of the RLE algorithm [9] applied on temperature readings. However, because RLE is based on the same consecutive input stream, its results depend on the data source. In this way, in order to perform RLE results with different data sources statistics, we have introduced a new compression algorithm which is inspired from RLE named K-RLE which means RLE with a K-Precision.

2.3 K-Run-Length Encoding

The idea behind this new algorithm is this:
let K be a number, If a data item *d* or data between *d+K* and *d-K* occur *n* consecutive times in the input stream, we replace the *n* occurrences with the single pair *nd*.

We introduce a parameter K which is a precision. K is defined as:

- $\delta = \frac{K}{\sigma}$ with σ a minimum estimate of the Allan standard deviation [10]; i.e. σ is a representative of the instrument measurement noise below which the precision is no longer significant.
- If $K = 0$, K-RLE is RLE. K has the same unit as the dataset values, in this case degree.

However, the change on RLE using the K-precision introduces data modified. Indeed, while RLE is a lossless compression algorithm K-RLE is a lossy compression algorithm. This algorithm is lossless at the user level because it chooses K considering that there is no difference between the data item *d*, *d+K* or *d-K* according to the application.

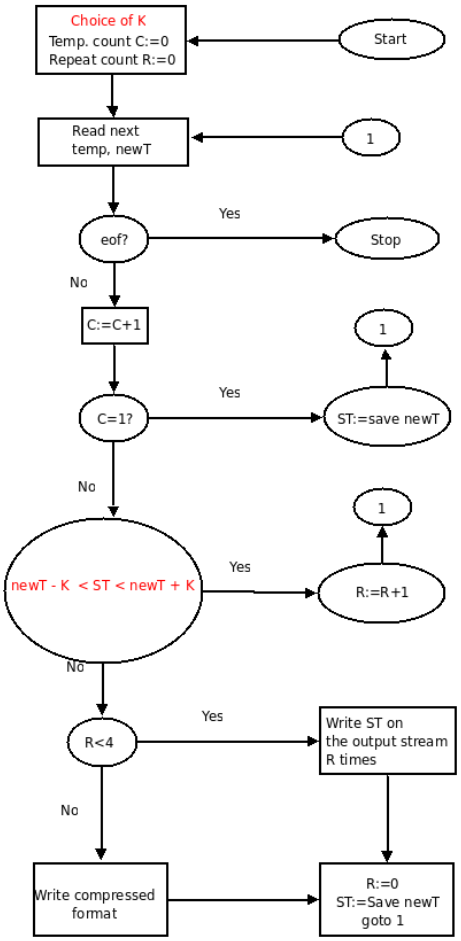

Figure 2. K-RLE compression algorithm

Fig.2 is the graphical representation of the K-RLE algorithm which is a variant of the RLE algorithm.

3 Experimental results

In this section, we describe the results obtained using the previous compression algorithms on a real temperature dataset of 500 bytes. We have collected temperatures since the 1st of January 2008 from different locations [11] which are: Libreville (Gabon), Cayenne (Guyanna), Montbeliard (France), Svalbard (Norway). We chose different locations in order to study the behaviour of the previous algorithms with different input streams. We have simulated the sensing of temperature as it was sensed by the TI MSP430. Actually, as described in [12], the ADC12 module, implemented in the MSP430x14x and MSP430x16x devices, is a high-performance 12-bit analog-to-digital converter. ADC12 features include integrated temperature sensor. The typical temperature sensor transfer function is:

$$VTEMP = 0.00355 * (TEMPC) + 0.986.$$

In this way, we have determined the temperature values as it was read by the ADC module of our platform based on the TI MSP430 microcontroller.

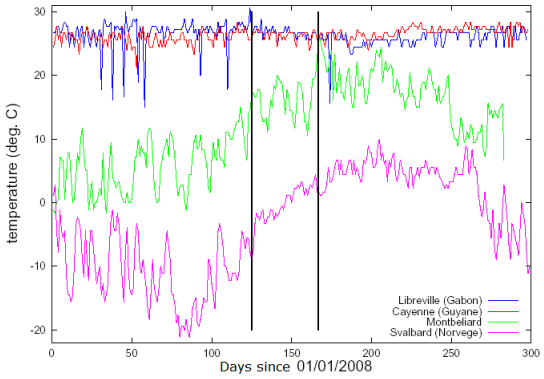

Figure 3. The representation of temperatures variation from different locations

The graphic above shows that the higher the latitude, the greater the temperature change. Certainly, data compression algorithm results depend of the data source that is why we consider our algorithms in real different conditions. After that, we use the data compression ratio to estimate the performance of our compression algorithms. It is defined as:

$$ratio = 100 * (1 - \frac{compressed\_size}{initial\_size})$$

Due to the RAM size limitation of the MSP430F149 to run S-LZW, we have done all the experiments on a

MSP430F1611 in order to make a comparison between S-LZW and RLE with different data sources.

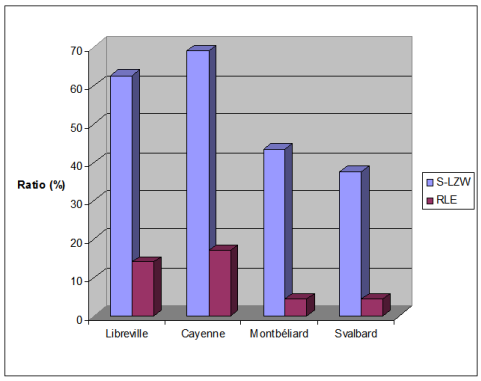

Figure 4. Comparison between S-LZW and RLE

Fig.4 shows that even if the data compression variation is the same, there is a great difference in the compression ratio. In fact, S-LZW results are better than RLE. We also noticed on both previous algorithms that the ratio worsens when the location is far away from the Equator. While the maximum gain for RLE is 17%, the average gain for S-LZW is about 53%.

However, because we can not apply S-LZW on a sensor platform with a limited RAM such as one which is based on MSP423F149 for example, we have tried to increase the ratio compression by using a variant of RLE named K-RLE. We have used different values of K which are 1/2 and 2.

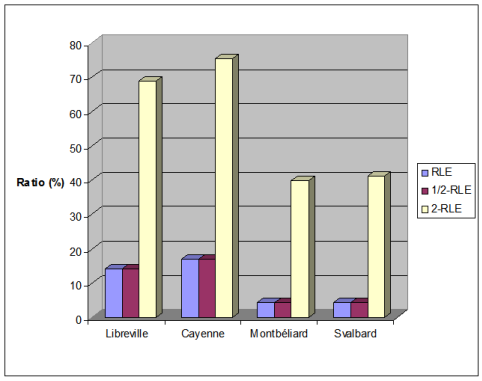

Figure 5. Comparison between RLE and K-RLE

Fig.5 shows that the best results are obtained with 2-RLE. There is a difference in the average of about 40% between RLE and 2-RLE. However, there is no gain with 1/2-RLE compared to RLE. These results show that the choice of K is a very important criterion. In contrast, K-RLE can achieve higher compression ratios at the cost of data precision when K increases.

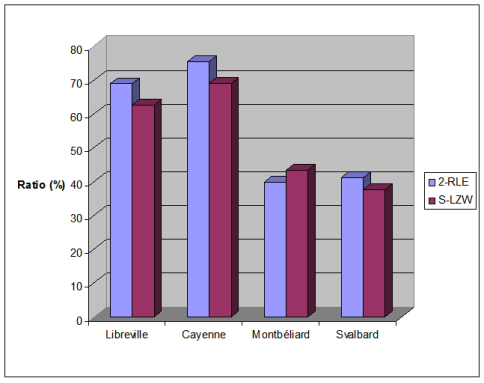

Figure 6. Comparison between S-LZW and K-RLE

Fig.6 shows that in most cases, 2-RLE is better than S-LZW. The average compression ratio for 2-RLE is 56% while that of S-LZW is 53%. We can continue to increase the K-RLE's ratio by increasing the value of K at the cost of the difference between the original and decompressed data. Indeed, the feature of lossy compression is that compressing data and then decompressing it retrieves data that may well be different from the original, but is close enough to be useful that is why the precision is chosen by the user according to the application. In this way, we focused on the study of the data rate modified during 2-RLE execution.

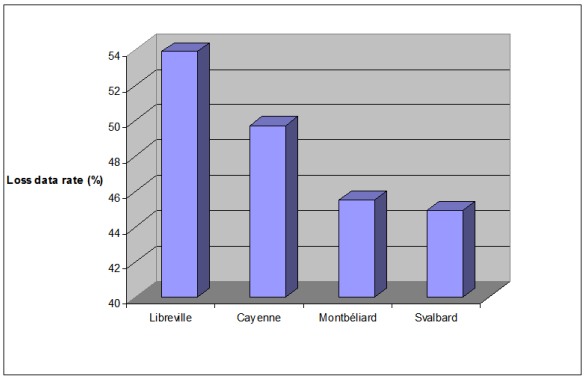

Figure 7. Representation of loss data rate for 2-RLE

The loss data rate is illustrated in Fig.7. It is defined as the percentage of data forced to be the same of the previous one close in value using K. It is about an average of 50% for all temperature dataset. This means that half of the original data have been modified using the K-precision of 2.

These previous results shows that the variant of RLE named K-RLE increases the compression ratio at the cost of 50% of loss data rate. In comparison with S-LZW which also has a good ratio compression, it is usable on a sensor platform with a limited RAM such as one which is based on MSP423F149.

Nevertheless, there is a question: what about energy consumption when the data compression ratio increases?

3.1 Energy consumption

In this section, we evaluate energy consumption of the previous data compression algorithms using WSim which is an accurate cycle hardware platform simulator. It is a part of an integrated environment for development and rapid prototyping of wireless sensor network applications known as Worldsens [13]. WSim is used to debug the application using the real target binary code. Indeed, we have used the same program files developed for our real platform on this simulator where the MSP430 platform has been defined.

For evaluating energy consumption, first of all, we have focused on algorithms time execution. We have used led2 and led3 on active mode respectively during compression and decompression. Fig.8 and Fig.9 show compression and decompression using S-LZW on Libreville temperature dataset.

After having time execution, to estimate energy consumption, we have to consider the Microcontroller power mode (Fig.10). At the end of our algorithm execution, we put the Microcontroller on LPM3 operating mode in order to identify the end of the execution of the program. In this way, using the description of the consumption of each mode (Fig.11), we can determine the consumption of each activity.

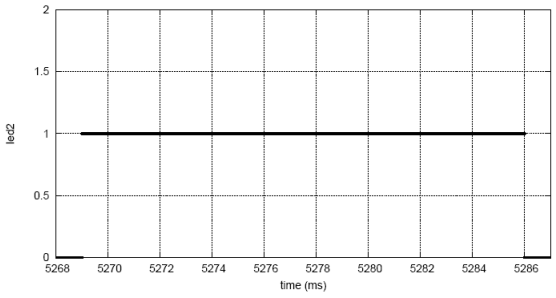

Figure 8. Execution time of 17 ms during compression using S-LZW on Libreville temperature dataset

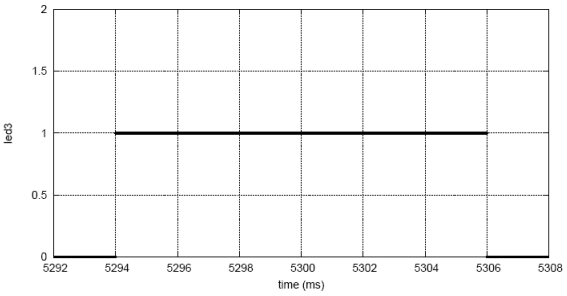

Figure 9. Execution time of 12 ms during decompression using S-LZW on Libreville temperature dataset

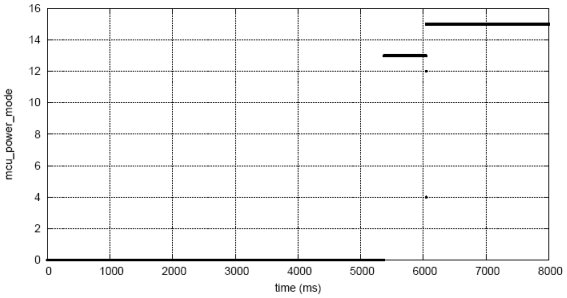

Figure 10. Microcontroller power mode

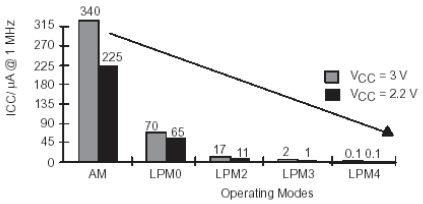

Figure 11. Operating modes [12]

Fig.12 illustrates energy consumption of previous data compression algorithms. We notice that while SLZW is lossless with a good compression ratio, it consumes more energy. S-LZW uses 0,0224 mJ while RLE uses about 0,0053 mJ and 2-RLE 0,0103 mJ. RLE uses less energy than the others. These results show the trade-off between energy compression and good compression ratio. We also notice that while RLE and 2-RLE have constant consumption, S-LZW uses more energy when there is more change on data.

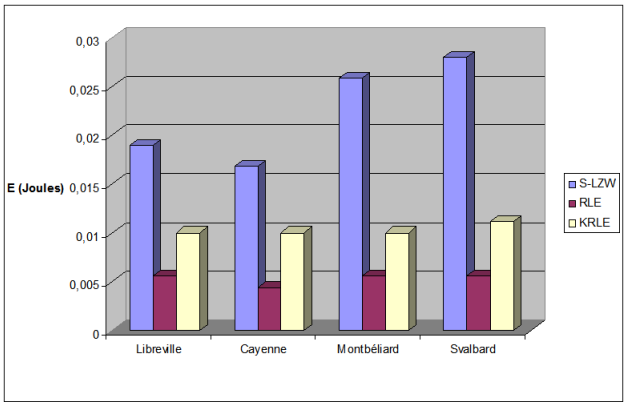

Figure 12. Compression consumption

In contrast, Fig.13 shows that while 2-RLE uses more energy than RLE for compression, it consumes very little energy for decompression about 0,0011 mJ. S-LZW uses an average of about 0,015 mJ and RLE 0,00165 mJ. We also noticed that S-LZW uses more energy when there is more change on data.

These previous results show that while RLE does not give a very good compression ratio, it offers a considerable consumption improvement for compression and decompression.

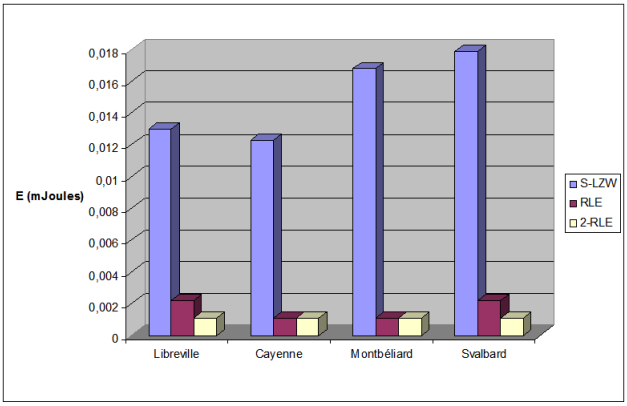

Figure 13. Decompression consumption

4 Conclusion and Future Work

In this paper, we have evaluated several data compression algorithms on an ultra-low power microcontroller from Texas Instrument known as MSP430. We have compared a famous dictionary-based data compression algorithm for WSN named S-LZW with RLE using real temperature datasets. Because of the difficulty in using S-LZW on a sensor platform with a limited RAM, we have introduced a new algorithm inspired from RLE named K-RLE which increases the ratio compression compared to RLE and S-LZW. For K equal to 2, this new lossy compression algorithm increases the ratio by 40% compared to RLE introducing 50% of loss data rate. The energy consumption study shows that while 2-RLE offers a better compression ratio than RLE and SLZW, it consumes half energy compared to S-LZW which uses the most energy. In this article, we have shown the trade-off between energy consumption and compression efficiency. Since RLE does not have a great compression ratio, it uses less energy than 2-RLE and S-LZW. Future work will focus on the scalability issues of the proposed enhancements.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[2] M. Nosovic and T. Todd. Low power rendezvous and rfid wakeup for embedded wireless networks. In *In Annual IEEE Computer Communications Workshop*, pages 3325–3329, 2000.

[3] N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 8–13 Vol. 2, 2005.

[4] F. Marcelloni and M. Vecchio. A simple algorithm for data compression in wireless sensor networks. *Communications Letters, IEEE*, 12(6):411–413, June 2008.

[5] Croce, Silvio, Marcelloni, Francesco, Vecchio, and Massimo. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *Computer Journal*, 51(2):227–239, March 2008.

[6] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[7] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SenSys*, pages 265–278, 2006.

[8] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[9] D. Salomon. *Data Compression: The Complete Reference*. Second edition, 2004.

[10] D. W. Allan. Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 34(6):647–654, 1987.

[11] The Weather Underground website. [Online]. Available: https://english.underground.com.

[12] Texas Instruments MSP430x1xx Family User's Guide, 2006. [Online]. Available: http://focus.ti.com..

[13] A. Fraboulet, G. Chelius, and E. Fleury. Worldsens: development and prototyping tools for application specific wireless sensors networks. In T. F. Abdelzaher, L. J. Guibas, and M. Welsh, editors, *IPSN*, pages 176–185. ACM, 2007.