COMPRESSION OF MICROARRAY IMAGES USING A BINARY TREE DECOMPOSITION

Luís M. O. Matos, António J. R. Neves, and Armando J. Pinho

IEETA / DETI
University of Aveiro, 3810–193 Aveiro, Portugal
{luismatos,an,ap}@ua.pt

ABSTRACT

This paper proposes a lossless compression method for microarray images, based on a hierarchical organization of the intensity levels followed by finite-context modeling. A similar approach was recently applied to medical images with success. The goal of this work was to further extend, adapt and evaluate this approach to the special case of microarray images. We performed simulations on seven different data sets (total of 254 images). On average, the proposed method attained $\sim 9\%$ better results when compared to the best compression standard (JPEG-LS).

***Index Terms*—** Binary tree decomposition, microarray images, lossless compression

1. INTRODUCTION

DNA microarray imaging is an important tool and a powerful technology for large-scale gene sequencing and gene expression analysis, allowing the study of gene function, regulation and interaction across a large number of genes, and even across entire genomes [1, 2]. DNA microarrays are currently used, for example, for genome-wide monitoring in areas such as cancer [3] and HIV research [4].

The output data obtained in a microarray experiment is a pair of 16 bits per pixel grayscale images, one from the so-called green channel and the other from the red channel. Gene expression can vary in a very wide range, justifying the need for image pixel intensities having a depth of 16 bits. Usually, these images also have a high spatial resolution, from 1000×1000 to 13800×4400 or even more, due to the microscopic size of the spots. Hence, these images may require several tens of megabytes in order to be stored or transmitted. Although the final goal is to extract from the microarray images information related to expression levels, it is usually desirable to keep both the extracted genetic information extracted and the original microarray experiments. For that reason, the need for efficient long-term storage, sharing and transmission of microarray images, is an important challenge.

This work was partially supported by FEDER through the Operational Program Competitiveness Factors - COMPETE and by National Funds through FCT - Foundation for Science and Technology, in the context of a PhD Grant (FCT reference SFRH/BD/86531/2012) and a project (FCT reference PEst-OE/EEI/UI0127/2014).

2. SPECIALIZED METHODS

There are several compression techniques that have been proposed for lossy and/or lossless compression of microarray images. In this work, we will be focused in lossless methods. Recently, Hernandez-Cabronero *et al.* [5] reviewed the state of the art in DNA microarray image compression. Therefore, here, we will describe only those that we think are the most representative.

Neves and Pinho [6] proposed a context-based lossless method in 2006. Their method is a bitplane-based approach that uses 3D finite-context models to drive the arithmetic encoder. In [7], they extended their method by adding a greedy procedure to build the 3D context template, obtaining better results than their previous method [6].

Neekabadi *et al.* [8] proposed a compression method based on the separation of the original image into three categories: background, edge and spot pixels. The segmentation is performed by finding a threshold value which minimizes the weighted sum of the standard deviations of the foreground and background pixels. This threshold value is used to create the three subsets. The spot pixels are determined by eroding the mask formed with pixels above the selected threshold. The pixels surrounding the spot pixels are classified as edge pixels. The background pixels are all the remaining. Each subset is compressed using a separate predictor.

In 2009, Battiato and Rundo [9] proposed an approach based on Cellular Neural Networks (CNNs). The first stage of the algorithm consists on separating the original microarray image into background and foreground (image spots). After this first stage, the foreground is lossless compressed by means of general purpose codecs (they used the standard PNG codec based on LZ77 algorithm). The background is compressed using an innovative method, based on palette reindexing.

3. BINARY TREE DECOMPOSITION

Pinho and Neves proposed a lossless compression method based on a hierarchical organization of the intensity levels [10, 11]. Originally, the approach was intended to be applied to images with a small number of intensities, usually with 8 or less bits per pixel, due to a tight relation between the processing time and the number of different intensities of the image. In this work, we further extended this approach to be able to handle images with a large number of intensities, as is the case of microarray images. The goal here was to find out if the compression results of microarray images can be further improved using this approach, when compared to the current state of the art methods. The proposed approach has progressive decoding capability, which means that the decoding process can be stopped at any moment according to a specific distortion metric, obtaining a partial image with some loss. Furthermore, it is possible to obtain the original image without any loss if the full decoding process is performed.

The organization of the intensity levels is attained by means of a binary tree. Each node of the binary tree, n, represents a certain subset, $\mathcal{S}^n$, of the intensities of the image. The root node contains all active pixel values of the image $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$, where N represents the number of different intensities that occur in the image. Therefore, $\mathcal{S}^n \subset \mathcal{I}$ and $\mathcal{S}^1 \equiv \mathcal{I}$. Each node possesses a representative intensity, I^n, given by

$$I^n = \left\lfloor \frac{I^n_m + I^n_M}{2} \right\rfloor, \tag{1}$$

where I^n_m and I^n_M are, respectively, the smallest and largest pixel value in $\mathcal{S}^n$, and where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Computing the value of I^n according to (1) leads to the smallest possible L_∞ reconstruction error when the intensities associated to node n (those in $\mathcal{S}^n$) are all substituted by I^n. The error is given by

$$\epsilon^n_\infty = I^n_M - I^n. \tag{2}$$

In Fig. 1, we can observe an example of a small binary tree of an image with only five active pixel values $\{32, 50, 250, 33768, 65530\}$. The construction of this tree begins with the association to the root node of the set of intensities that occur in the original image. After this association, it is necessary to compute I^1 according to (1). In the example depicted in Fig. 1, $I^1 = (32 + 65530)/2 = 32781$ and $\epsilon^1_\infty = 65530 - 32781 = 32749$, for the root node. The next step consists in splitting the root node into two subnodes and, therefore, splitting $\mathcal{S}^1$ into two subsets. In order to split $\mathcal{S}^1$, we need only to compare the intensity $I \in \mathcal{S}^1$ with I^1. The intensities lower than I^1 are associated with the left node, and the other ones with the right one. This procedure is repeated until expanding all nodes, i.e., until having a tree with N leaves (N is the number of active intensities presented in the original image). The next node to expand is chosen taking into consideration the smallest possible L_∞ reconstruction error. In case of a tie, one is arbitrarily chosen, although it is necessary that the decoder picks the same one. In order to the decoder be able to build the same tree, it is necessary to send the information of the active pixels values to the decoder using a 65536-bit indicator. In order to encode this indicator the encoder uses the following strategy. First, the maximum intensity value I_N, is sent. After that, a string of I_n bits is transmitted, such that if the n^{th} bit of the string is one it means that the intensity $n-1$ is present in the image and zero otherwise.

After each node is expanded, two new nodes are created. It it necessary to send to the decoder which pixels of the original image will have an intensity changed to I^n_l and which need to change to I^n_r, respectively the representative intensities of the left and right newly created nodes. Since the decoder has access to these intensity changes for all pixels, it is enough to encode a binary mask, where a zero indicates that the pixel needs to change its intensity to I^n_l and a one indicates a change to I^n_r.

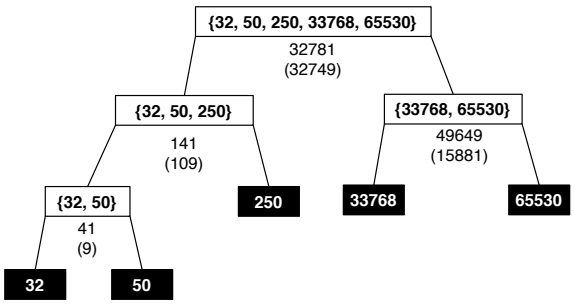

Fig. 1. Example of a small binary tree that illustrates the hierarchical organization of the intensity values.

The context was constructed using a template similar to the one used in [11], where the context pixels are numbered according to their distance to the pixel being encoded. A particular context is represented using a sequence of bits,

$$b_1 b_2 \ldots b_k \tag{3}$$

where

$$b_i = \begin{cases} 0, & \text{if} \quad |I(i) - I^n_l| \leq |I(i) - I^n_r| \\ 1, & \text{otherwise} \end{cases}$$

and where $I(i)$ denotes the intensity of the pixel in the current reconstructed image corresponding to position i of the context template.

This approach is interesting because it is not affected by the reduced percentage of active pixel values that is a characteristic in some microarray images (histogram sparseness). However, even in this case, it is still necessary to send the 65536-bit indicator describing which intensities occur (or do not occur). This overhead is constant for these kind of images (65536 bits), but the overhead percentage depends on the number of pixels.

4. EXPERIMENTAL RESULTS

In this section, we present the experimental results obtained using the approach described in Section 3. First, we describe the data sets used. After that, the compression results using image compression standards (JBIG, JPEG-LS and JPEG-2000), Gzip and Bzip2 are presented. The compression results using binary tree decomposition are shown in Section 4.3.

4.1. Microarray image data sets

In recent literature, several microarray image data sets have been used to report compression results. One of the hurdles that often the researcher has to face is the lack of a reasonably consensual data set with which the performance of the algorithms can be measured. A few years ago, Pinho *et al.* [12] used a benchmarking data set, composed of 49 publicly available microarray images from three different data sets, to evaluate the performance of the image compression standards. As a result, this benchmarking data set has been used in most of the works published after. However, it is natural that new images, obtained using more recent technologies, need to be added to the benchmarking data set. In this research, we used four new data sets that were not used in [12] (see Table 1). Regarding the Omnibus data set, we decided to split it into two sub-data sets: Low Mode (LM) images and High Mode (HM) images. The LM and HM images are associated with the same experiment but they were scanned using two different modes: High or Low. The scanning mode affects the properties of the obtained images mainly in terms of entropy and percentage of active intensities (see Table 1 for more details).

In Table 1 we show the number of images, the approximate size of the images, the first order entropy, the average percentage of active pixels of each data set, and also the Gini Index (GI) [13]. The GI is a normalized sparsity measure that can assume values between zero and one. Values close to zero mean that the image has a lower sparsity. On the other hand, values close to one mean that the image is very sparse.

4.2. Image compression standards

In 2006, Pinho *et al.* [12] evaluated the performance of JBIG [14], JPEG-LS [15] and lossless JPEG2000 [16] in microarray images. Here, we extend those results regarding the additional data sets included in the benchmarking data set and for the Gzip and Bzip2 compression methods (see Table 2).

JBIG results were obtained using version 2.0 of the JBIG Kit package[1]. The results for the JPEG-LS standard were obtained using version 2.2 of the SPMG JPEG-LS codec, with default parameters[2]. We used three implementations of JPEG2000 in lossless mode. Version 5.1 of JJ2000 was used with the default parameters[3]. The JasPer results were computed using version 1.900.1 of the codec, with default parameters[4]. Finally, the results for the Kakadu software were collected using version 7.2 with default parameters, in lossless mode[5].

4.3. Results using binary tree decomposition

In what follows, we present the compression results attained by the proposed method and also by the methods described in [6, 7]. We did not include the compression results regarding methods [8, 9] because we do not have access to the software or source code to generate the results for the four new data sets not used by them (Arizona, Omnibus, Stanford and Yeast). We performed simulations on seven different data sets (total of 254 images) described in Table 1. The results can be found in Table 3. The second row of Table 3 corresponds to the use of a search area of 256×256 pixels for context creation, whereas the third row corresponds to a search area comprising the complete image (designated as "Full" in the Table). Regarding the proposed method, we present results of two different approaches for context creation: "Greedy", and "Best". The goal of each different context creation approaches is to find the context size by different ways, in order to maximize the compression ratio. The "Greedy" approach, as the name itself says, computes a locally optimal solution, not necessarily the globally best solution. In this case, instead of testing all possible sizes, we progressively test several sizes until obtaining a bitrate worse than the previous "best". The "Best" solution is a slower version, where all possible sizes are tested. Despite the time that it takes to compress, in this case it is guaranteed that the best solution is always found.

After analyzing Table 3, we can conclude that the results, on average, are quite similar among both approaches ("Greedy" and "Best") of the proposed method. Furthermore, the "Best" version of the proposed method attained $\sim 9\%$ better results when compared to the best compression standard (JPEG-LS). On the other hand, the "Full" version of method [7] attained $\sim 8\%$ better results when compared to the best compression standard (JPEG-LS). Furthermore, we can notice in the second row of Table 3, the poor results obtained in method [7], when compared to method [6]. The reason for those low results (higher number of bits per pixel), is due to the naive implemented search area positioning. By default, method [7] set the search area in the center of the

[1] http://www.cl.cam.ac.uk/~mgk25/jbigkit

[2] The original web-site of this codec, http://spmg.ece.ubc.ca, is currently unavailable. However, it can be obtained from http://sweet.ua.pt/luismatos/codecs/jpeg_ls_v2.2.tar.gz

[3] The original web-site of this codec, http://jj2000.epfl.ch, is currently unavailable. Nevertheless, this codec can be obtained from http://sweet.ua.pt/luismatos/codecs/jj2000_5.1-src.zip

[4] http://www.ece.uvic.ca/~frodo/jasper

[5] http://www.kakadusoftware.com

Data set	Year	Images	Approximate size (cols × rows)	Average entropy (bits per pixel)	Average intensity usage (percentage)	Gini Index (GI)
Apo AI	2001	32	$> 1044 \times 1041$	11.038	39.507	0.494
Arizona	2011	6	$= 13800 \times 4400$	9.306	82.821	0.774
ISREC	2001	14	$= 1000 \times 1000$	10.435	33.345	0.710
Microzip	2004	3	$> 1800 \times 1900$	9.422	36.906	0.556
Omnibus (Low Mode)	2006	25	$= 12200 \times 4320$	5.713	50.130	0.726
Omnibus (High Mode)	2006	25	$= 12200 \times 4320$	7.906	98.076	0.892
Stanford	2001	40	$> 1900 \times 2000$	8.306	27.515	0.615
Yeast	1998	109	$= 1024 \times 1024$	6.614	5.391	0.518

Table 1. Microarray image data sets used in this work. The number of images represents the total number of images that each data set contains (each image corresponds to one channel).

Algorithm	Apo AI	Arizona	ISREC	MicroZip	Omnibus (LM)	Omnibus (HM)	Stanford	Yeast	Average
Gzip	12.711	11.263	12.464	11.434	7.124	9.558	9.972	7.672	8.813
Bzip2	11.068	9.040	**10.922**	9.394	5.346	7.523	7.961	**6.075**	6.880
JBIG	10.851	8.896	10.925	9.298	5.130	7.198	7.906	6.888	6.676
JPEG-LS	**10.608**	**8.676**	11.145	**8.974**	**4.936**	**6.952**	**7.684**	8.580	**6.521**
JJ2000	11.063	9.107	11.366	9.515	5.340	7.587	8.060	9.079	7.020
JasPer	11.002	9.064	11.314	9.467	5.312	7.549	8.019	9.030	6.985
Kakadu	11.063	9.064	11.314	9.467	5.312	7.549	8.018	9.029	6.985

Table 2. Lossless compression results, in bits per pixel (bpp), using Gzip, Bzip2, JBIG, JPEG-LS and three implementations of JPEG2000: JJ2000, JasPer and Kakadu. Default compression parameters have been used for all algorithms. The best results are highlighted in bold.

microarrays image. However, it is possible that in the center of the microarray image the search area does not contain any spots (background region) thus, the context obtained is not optimal. In this case, the images of the Omnibus data set had four spot regions and the default search area positioning sets the search area in a background zone without spots.

Table 4 depicts the average number of pixels per millisecond that are processed during encoding (left values) and decoding (right values). Instead of using all images, we selected four representative images from each data set to compute these results. As expected, the method [6] is the fastest one due to its static precomputed context configuration. The other methods rely on a context creation that depends on the image. This context creation takes a considerable amount of time, depending on the approach used. The proposed method is based on a hierarchical organization of the intensity levels of the image. For that reason, the performance in terms of encoding/decoding time is dependent on the percent of active intensities (see column 6 of Table 1). For the Yeast data set, the results obtained by our method are quite similar to those of methods [6,7]. For the other data sets, the processing times are worse mainly in the decoding phase. We conclude that despite the fact a microarray image has a lower percent of active intensities, the entropy and GI values can affect the performance of the proposed method. It is also easy to conclude that the methods based on bitplane decomposition (such as methods [6] and [7]), are not affected in terms of encoding/decoding time by the percent of active intensities.

5. CONCLUSIONS

We presented a lossless compression method for microarray images based on binary tree decomposition. We used a total of 254 images from seven different data sets. Because these seven microarray image data sets are different in terms of size (number of pixels), number of images, entropy, percentage of active pixels and sparseness, the results obtained are quite different for each data set. The proposed method attained an average of $\sim 9\%$ better results when compared to the best compression standard (JPEG-LS). In terms of compression ratio, it is slightly better than the "Best" version of method [7], which, on average, attained $\sim 8\%$ better results when compared to JPEG-LS.

The proposed method has progressive decoding capability, which is in fact a very useful characteristic that the majority of microarray compression methods do not have. Not only we can stop the decoding process at any instant (getting an image with some loss) according to a defined metric, but also we can obtain the original image without any loss.

Algorithm	Apo AI	Arizona	ISREC	MicroZip	Omnibus (LM)	Omnibus (HM)	Stanford	Yeast	Average
Neves [6]	10.280	8.394	10.217	8.840	5.309	7.047	7.664	5.610	6.561
Neves [7] (256×256)	10.225	8.293	10.199	8.667	5.679	7.744	7.468	5.511	6.948
Neves [7] (Full)	**10.194**	8.242	**10.159**	8.619	4.567	6.471	7.379	5.453	6.006
Proposed (Greedy)	10.199	8.186	10.200	8.593	4.540	6.401	7.306	5.323	5.957
Proposed (Best)	**10.194**	**8.186**	10.198	**8.592**	**4.539**	**6.400**	**7.303**	**5.318**	**5.956**

Table 3. Lossless compression results in bits per pixel (bpp) obtained using the methods described in [6, 7] and the proposed method based on binary tree decomposition. The "Greedy" and "Best" versions are related to two different approaches for context adaptation (additional details in the main text).

Algorithm	Apo AI	Arizona	ISREC	MicroZip	Omnibus (LM)	Omnibus (HM)	Stanford	Yeast	Average
Neves [6]	**103** | 104	**234** | **301**	**76** | 82	**198** | **237**	250 | 313	231 | 294	**175** | 210	**77** | 82	**221** | **277**
Neves [7] (256×256)	36 | **227**	163 | 213	35 | **238**	108 | 220	**279** | **366**	**259** | **360**	97 | **230**	27 | **215**	164 | 258
Neves [7] (Full)	3 | 197	2 | 171	3 | 213	2 | 180	1 | 168	1 | 162	2 | 190	2 | 194	2 | 170
Proposed (Greedy)	2 | 11	6 | 40	6 | 24	2 | 64	11 | 29	7 | 31	5 | 47	20 | 186	6 | 35
Proposed (Best)	1 | 11	2 | 41	2 | 26	2 | 65	5 | 30	2 | 31	2 | 45	8 | 185	2 | 36

Table 4. Simulation results in pixels per millisecond. The values in the left correspond to encoding, whereas the right-hand ones correspond to decoding. The results were obtained dividing the total number of pixels by the total number of milliseconds used in the encoding/decoding process. Higher values are better.

REFERENCES

[1] P. Hegde *et al.*, "A concise guide to cDNA microarray analysis", *Biotechniques*, vol. 29, no. 3, pp. 548–562, Sept. 2000.

[2] S. K. Moore, "Making chips to probe genes", *IEEE Spectrum*, vol. 38, no. 3, pp. 54–60, Mar. 2001.

[3] S. Satih *et al.*, "Gene expression profiling of breast cancer cell lines in response to soy isoflavones using a pangenomic microarray approach", *OMICS: A Journal of Integrative Biology*, vol. 14, pp. 231–238, June 2010.

[4] M. S. Giri, M. Nebozhyn, L. Showe, and L. J. Montaner, "Microarray data on gene modulation by HIV-1 in immune cells: 2000-2006", *Journal of Leukocyte Biology*, vol. 80, no. 5, pp. 1031–1043, 2006.

[5] M. Hernandez-Cabronero, I. Blanes, M. W. Marcellin, and J. Serra-Sagrista, "Standard and specific compression techniques for DNA microarray images", *Algorithms*, vol. 4, pp. 30–49, 2012.

[6] A. J. R. Neves and A. J. Pinho, "Lossless compression of microarray images", in *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2006*, Atlanta, GA, Oct. 2006, pp. 2505–2508.

[7] A. J. R. Neves and A. J. Pinho, "Lossless compression of microarray images using image-dependent finite-context models", *IEEE Trans. on Medical Imaging*, vol. 28, no. 2, pp. 194–201, Feb. 2009.

[8] A. Neekabadi *et al.*, "Lossless microarray image compression using region based predictors", in *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2007*, San Antonio, Texas, USA, Sept. 2007, vol. 2, pp. 349–352.

[9] S. Battiato and F. Rundo, "A bio-inspired CNN with re-indexing engine for lossless DNA microarray compression and segmentation", in *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2009*, Cairo, Egypt, Nov. 2009, vol. 1-6, pp. 1737–1740.

[10] A. J. Pinho and A. J. R. Neves, "Progressive lossless compression of medical images", in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-2009*, Taipei, Taiwan, Apr. 2009.

[11] A. J. Pinho and A. J. R. Neves, "L-infinity progressive image compression", in *Proc. of the Picture Coding Symposium, PCS-07*, Lisbon, Portugal, Nov. 2007.

[12] A. J. Pinho, A. R. C. Paiva, and A. J. R. Neves, "On the use of standards for microarray lossless image compression", *IEEE Trans. on Biomedical Engineering*, vol. 53, no. 3, pp. 563–566, Mar. 2006.

[13] N. Hurley and S. Rickard, "Comparing Measures of Sparsity", *IEEE Trans. on Information Theory*, vol. 55, no. 10, pp. 4723–4741, Oct. 2009.

[14] D. Salomon, *Data compression - The complete reference*, Springer, 2nd edition, 2000.

[15] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS", *IEEE Trans. on Image Processing*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[16] D. S. Taubman and M. W. Marcellin, *JPEG2000: image compression fundamentals, standards and practice*, Kluwer Academic Publishers, 2002.