A BITPLANE BASED ALGORITHM FOR LOSSLESS COMPRESSION OF DNA MICROARRAY IMAGES

António J. R. Neves and Armando J. Pinho

Signal Processing Lab, DETI / IEETA
University of Aveiro, 3810–193 Aveiro, Portugal
an@ua.pt / ap@ua.pt

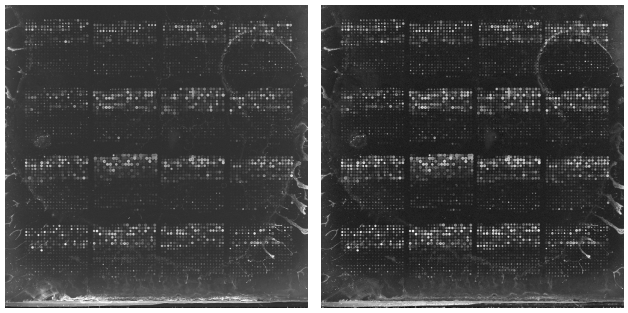
ABSTRACT

During the past years, the development of microarray technology has been remarkable, and it is becoming a daily tool in many genomic research laboratories. The widespread adoption of this technology, coupled with the significant volume of data generated per experiment, in the form of images, have led to significant challenges in storage and query-retrieval. In this paper, we present a lossless bitplane based method for efficient compression of microarray images. This method is based on arithmetic coding driven by image-dependent multi-bitplane finite-context models. It produces an embedded bitstream that allows progressive, lossy-to-lossless decoding. We compare the compression efficiency of the proposed method with three image compression standards (JPEG2000, JPEG-LS and JBIG) and also with the two most recent specialized methods for microarray image coding. The proposed method gives better results for all images of the test sets and confirms the effectiveness of bitplane based methods and finite-context modeling for the lossless compression of microarray images.

1. INTRODUCTION

The DNA microarray technology is one of the most important tools for discovering and analyzing the interactions and interrelations between genes and for monitoring gene expression under a variety of conditions [1, 2]. Microarrays allow an unparalleled view of genes, proteins and biomolecules providing detailed genomic information. Microarrays themselves resemble matrices consisting of small spots (see Fig. 1). These spots represent collections of DNA placed on the surface of a glass or nylon slide. Each spot contains copies of a single DNA sequence, such as genes. The spots in the microarray represent a different gene, and are aligned in an array so that, in a single test, the expression levels of hundred or thousands of genes within a cell can be determined by measuring the amount of messenger ribonucleic acid (mRNA) bound to each site on the array.

The microarray images are analyzed using a variety of software tools which extract relevant information, such as the intensity of the spots and the background level. This information is then used to evaluate the expression level of individual genes [1, 2]. Depending on the size of the array and the resolution of the scanner, these images may require several tens of megabytes in order to be stored or transmitted. Giving the massive amount of data currently produced and the need of long-term storage and efficient transmission, the development of efficient compression methods is an important challenge.



(a) Green channel (b) Red channel

Figure 1: Example of a pair of images (1000×1000 pixels) that results from a microarray experiment.

To our knowledge, the technique that we propose in this paper is the best one currently available in terms of compression efficiency of microarray images. The method is based on arithmetic coding driven by image-dependent multi-bitplane finite-context models. Basically, the image is compressed on a bitplane basis, going from the most significant bitplane to the least significant bitplane. The finite-context model used by the arithmetic encoder uses (causal) pixels from the bitplane under compression and also pixels from the bitplanes already encoded. The context is image-dependent, i.e., it is adapted for each bitplane of each image according to a greedy procedure that aims the minimization of the bitrate. The proposed method is compared with JPEG2000 [3, 4], JPEG-LS [5, 6] and JBIG [7, 8], and also with two recent specialized methods, MicroZip [9] and Zhang's method [10, 11].

The paper is organized as follows. In Section 2, we present the techniques that have been proposed for the lossy and/or lossless compression of microarray images. In Section 3, we describe our algorithm, in particular how we collect the statistical information needed by the arithmetic encoder using image-dependent contexts. In Section 4, we provide experimental results of our method and we compare these results with JPEG2000, JPEG-LS and JBIG, and with two recent specialized methods. Finally, in Section 5, we draw some conclusions.

2. SPECIALIZED METHODS

Some techniques have already been proposed for the lossy and/or lossless compression of microarray images. Next, we give a brief overview of each of these techniques.

This work was supported in part by the FCT (Fundação para a Ciência e a Tecnologia) grant PTDC/EIA/72569/2006.

The technique proposed by Jörnsten *et al.* [12] is characterized by a first stage devoted to griding and segmentation. Using the approximate center of each spot, a seeded region growing is performed for segmenting the spots. The segmentation map is encoded using chain-coding, whereas the interior of the regions are encoded using a modified version of the LOCO-I (LOw COmplexity LOssless COmpression for Images) algorithm (this is the algorithm behind the JPEG-LS coding standard), named SLOCO. Besides lossy-to-lossless capability, Jörnsten's technique allows partial decoding, by means of independently encoded image blocks.

Hua *et al.* [13] presented a transform-based coding technique. Initially, a segmentation is performed using the Mann-Whitney algorithm, and the segmentation information is encoded separately. Due to the threshold properties of the Mann-Whitney algorithm, the griding stage is avoided. Then, a modified EBCOT (Embedded Block Coding with Optimized Truncation) [14] for handling arbitrarily shaped regions is used for encoding the spots and background separately, allowing lossy-to-lossless coding of background only (with the spots lossless encoded) or both background and spots.

The compression method proposed by Faramarzpour *et al.* [15] starts by locating and extracting the microarray spots, isolating each spot into an individual ROI (region of interest). To each of these ROI's, a spiral path is adjusted such that its center coincides with the center of mass of the spot, with the idea of transforming the ROI into an one-dimensional signal with minimum entropy. Then, predictive coding is applied along this path, with a separation between residuals belonging to the spot area and those belonging to the background area.

Lonardi *et al.* [9] proposed lossless and lossy compression algorithms for microarray images (MicroZip). The method uses a fully automatic griding procedure, similar to that of Faramarzpour's method, for separating spots from the background (which can be lossy compressed). Through segmentation, the image is split into two streams: foreground and background. Then, for entropy coding, each stream is divided into two 8 bit sub-streams and arithmetic encoded, with the option of being previously processed by a Burrows-Wheeler transform [16].

The method proposed by Zhang *et al.* [10, 11] is based on PPAM (Prediction by Partial Approximate Matching). PPAM is an image compression algorithm which extends the PPM text compression algorithm, considering the special characteristics of natural images [10]. Initially, the microarray image is separated into its different components: background and foreground, i.e., the microarray spots. For each component, the pixel representation is separated into its most significant and least significant parts. Then, to compress the data, the most significant part is first processed by an error prediction scheme and then residuals are encoded by the PPAM context model and encoder. The least significant part is encoded directly by the PPAM encoder. The segmentation information is saved without compression.

3. PROPOSED METHOD

In [17], we studied the compression performance of three image coding standards in the context of microarray image compression: JBIG [7, 8], JPEG-LS [5, 6] and JPEG2000 [3, 4]. Since they rely on three different coding technologies, we were able not only to evaluate the performance of each of these standards, but also to collect hints regarding which might be the best coding technology regarding microarray image compression. In that study, we concluded that from the three technologies evaluated (predictive coding in the case of JPEG-LS, transform coding in the case of JPEG2000 and context-based arithmetic coding in the case of JBIG), the technology behind JBIG seemed to be the most promising. Moreover, with JBIG, the image bitplanes are compressed independently, suggesting the existence of some room for improvement.

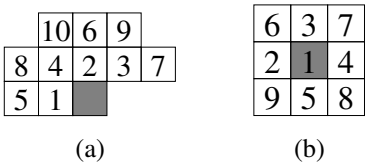
Motivated by these observations, we have developed a compression method for microarray images which is based on the same technology as JBIG but that, unlike JBIG, exploits inter-bitplane dependencies, providing coding gains in relation to JBIG [18]. Designing contexts that gather information from more than one bitplane (multi-bitplane contexts) is not just a matter of joining more bits to the context, because for each new bit added the memory required doubles. Moreover, there is the danger of running into the context dilution problem, due to the lack of sufficient data for estimating the probabilities. Therefore, this extension to multi-bitplane contexts must be done carefully.

In the method described in [18] the images are compressed on a bitplane basis, from the most to the least significant bitplane. The causal finite-context model that drives the arithmetic encoder uses pixels both from the bitplane currently being encoded and from the bitplanes already encoded. As encoding proceeds, the average bitrate obtained after encoding each bitplane is monitored. If, for some bitplane, the average bitrate exceeds one bit per pixel, then the encoding process is stopped and the remaining bitplanes are saved without compression.

Although being able to provide state-of-the-art compression results, the method proposed in [18] could be improved. In fact, due to its image-independent nature, and despite being designed for a specific type of images (microarrays), the context configuration proposed in [18] resulted from a complicated process that tried to balance the inevitable particularities among the images. From the point of view of a single image, this context configuration might be over killing, i.e., a smaller context might suffice. However, it is needed for satisfying the ensemble of images. This observation motivated the image-dependent context-modeling approach that we present in the remainder of the paper.

The algorithm that we propose in this paper tries to find the "best" context configuration to encode the current bitplane, based on the templates depicted in Fig. 2. The test of all possible context configurations is a hard task, virtually impossible, due to the huge number of possibilities. To overcome this drawback, we developed a greedy approach that we explain next.

Before encoding a bitplane, p, the algorithm constructs an appropriate context configuration through an iterative process (note that bitplanes are numbered from 0, the least significant bitplane, to 15, the most significant bitplane). In each iteration, an additional context bit is tested in each of the $16-p$ possible locations, one in each of the $16-p$ context bitplane that are available. In a given context bitplane, the additional bit can only be inserted in position k of the corresponding template displayed in Fig. 2 if all positions $i<k$ belong already to the best context configuration found so far. This means that the part of the context belonging to a



(a) (b)

Figure 2: (a) The template used for growing the context at the level of the bitplane currently being encoded; (b) The template used for growing the context corresponding to the bitplanes already encoded.

given bitplane can grow only according to the pixel numbering shown in Fig. 2. The template in Fig. 2(a) is used for the context bitplane p, whereas the template in Fig. 2(b) applies to the remaining context bitplanes, i.e., from bitplane $p+1$ to bitplane 15.

After performing an iteration, the new context bit is assigned to the position where the largest improvement in the compression performance of bitplane p occurred. If none of the possible $16-p$ context bit positions were able to improve the compression, then the search stops and the context configuration found so far is used for encoding the bitplane p. Otherwise, a new context bit is tested. This iterative process proceeds while the new context bit is able to improve the compression performance of bitplane p or until the maximum context depth is reached. For the results presented in this paper, we used a maximum of 20 context bits. Figure 3 presents an example of the context configuration obtained with this process for some of the bitplanes of the image "1230c1G" (APO_AI image set). The algorithm is outlined next.

```
bestCtx := 0-order context;
bestRate := rate for encoding bitplane
 with 0-order context;
do
  improved := FALSE;
  for p := bitplane to be encoded, 15
    if p = bitplane to be encoded
      add bit according to Fig. 2(a);
    else
      add bit according to Fig. 2(b);
    end
    rate := rate for encoding bitplane
     using current context;
    if rate < bestRate
      bestCtx := current context;
      bestRate := rate;
      improved := TRUE;
    end
    remove bit added above;
  end
while size of bestCtx < 20 AND improved
```

Being a greedy approach, it is not guaranteed that the optimum is found. In fact, as can be seen, for each context bitplane the context can only grow according to a predefined order which is given by the pixel numbering associated to the templates of Fig. 2. This limits the number of degrees of freedom of the search process, reducing the probability of finding the optimum configuration, but, on the other hand, it also permits running this procedure in a reasonable time.

In order to further accelerate the process of choosing these image-dependent contexts, and due to the highly structured nature of microarray images, we developed another version of the algorithm where only a small region of the image is used for constructing the contexts. Using this faster approach, the results obtained for a region of 256×256 pixels have been slightly worse. However, we verified a significant reduction in the time spent. In a Pentium 4 computer at 2 GHz with 512 MBytes of memory, the MicroZip test set (three images totaling approximately 21 million pixels) required about 220 minutes to compress when the whole image was used to performed the search. When we used a region of 256×256 pixels, it required approximately 6 minutes to compress the MicroZip test set (about 2 minutes more than the image-independent approach of [18]). These three images have sizes of 1916×1872, 5496×1956 and 3625×1929 pixels. Decoding is faster, because the decoder does not have to search for the best context: that information is embedded in the bitstream.

4. EXPERIMENTAL RESULTS

The compression method proposed in this paper was evaluated using microarray images that have been collected from three different publicly available sources: (1) 32 images that we refer to as the APO_AI set and which have been collected from `http://www.stat.berkeley.edu/~terry/Group/index.html` (this set was previously used by Jörnsten *et al.* [19, 12]); (2) 14 images forming the ISREC set which have been collected from `http://www.isrec.isb-sib.ch/DEA/module8/P5_chip_image/images/`; (3) three images previously used to test MicroZip [9] and Zhang's method [10, 11], which were collected from `http://www.cs.ucr.edu/~yuluo/MicroZip/`. These three sets have also been used in the experiments reported in [17] and [18].

Image size ranges from 1000×1000 to 5496×1956 pixels, i.e., from uncompressed sizes of about 2 MB to more than 20 MB (all images have 16 bits per pixel). The average results presented take into account the different sizes of the images, i.e., they correspond to the total number of bits divided by the total number of image pixels.

Table 1 shows the average compression results, in bits per pixel, for the three sets of images described above. In this table, we present experimental results regarding the use of three standard image coding methods, namely JBIG, JPEG-LS and JPEG2000, the image-independent method of [18] and the proposed image-dependent method. We can see that the fast version of the proposed method (indicated as "256×256" in the table) is 6.3% better than JBIG, 4.7% better than JPEG-LS and 8.6% better than lossless JPEG2000. It is important to note that JPEG-LS does not provide progressive decoding, a characteristic that is intrinsic to the proposed method and also to JPEG2000 and JBIG. This might be an important functionality if large databases have to be accessed remotely. From the results presented in Table 1 it can also be seen that using an area of 256×256 pixels in the center of the image for finding the context, instead of the whole image, leads to a small degradation in the performance (about 0.3%), showing the appropriateness of this approach. In Fig. 4 we can see how the size of the area used for finding the context affects the performance of the proposed method.

Table 2 confirms the performance of our method relatively to two recent specialized methods for compressing mi-

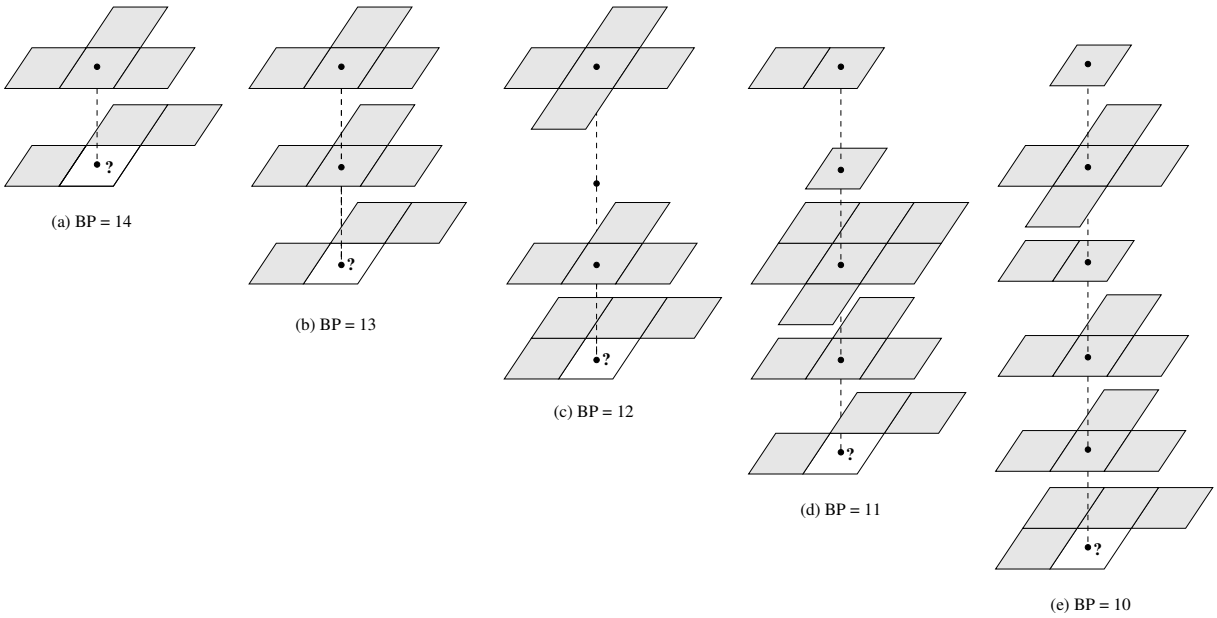


Figure 3: Context configuration obtained by the proposed method in five different bitplanes of the image "1230c1G": (a) when encoding bitplane 14 (seven bits of context); (b) when encoding bitplane 13 (11 bits of context); (c) when encoding bitplane 12 (13 bits of context); (d) when encoding bitplane 11 (17 bits of context); (e) when encoding bitplane 10 (20 bits of context). Context positions falling outside the image at the image borders are considered as having zero value.

Image Set	JPEG2000	JBIG	JPEG-LS	Image independent	Image-dependent	
					256×256	Full
APO_AI	11.063	10.851	10.608	10.280	10.225	10.194
ISREC	11.366	10.925	11.145	10.199	10.198	10.158
MicroZip	9.515	9.297	8.974	8.840	8.667	8.619
Average	**10.653**	**10.393**	**10.218**	**9.826**	**9.741**	**9.708**

Table 1: Average compression results, in bits per pixel, using JBIG, JPEG-LS, JPEG2000, the image-independent method of [18] and the proposed image-dependent method. The "256×256" column indicates results obtained with a context model adjusted using only a square of 256×256 pixels at the center of the microarray image, whereas "Full" indicates that the search was performed in the whole image. The average results presented take into account the different sizes of the images, i.e., they correspond to the total number of bits divided by the total number of image pixels.

croarray images: MicroZip [9] and Zhang's method [10, 11]. As can be observed, the method that we propose provides compression gains of 9.1% relatively to MicroZip and 6.2% in relation to Zhang's method, on a set of test images that has been used by all those methods.

Figure 4 shows, for the three sets of images used in this paper, the average number of bits per pixel obtained for different sizes of the area used to search for the best context size for each bitplane. As expected, this value decreases when the size of the area increases. However, for regions larger than 256×256 pixels this variation becomes smaller. Moreover, it is important to note that the time spent to find the best context size for each bitplane increases when we use larger regions.

5. CONCLUSION

In this paper, we presented an efficient bitplane based algorithm for lossless compression of microarray images, allowing progressive, lossy-to-lossless decoding. This method is based on bitplane compression using image-dependent finite-context models and arithmetic coding. It does not require griding and/or segmentation as most of the specialized methods that have been proposed do. This may be an advantage if only compression is sought, since it reduces the complexity of the method.

The maximum number of context bits that we allowed for building the contexts was limited to 20. Since the coding alphabet is binary, this implies, at most, $2 \times 2^{20} = 2\,097\,152$ counters that can be stored in approximately 4 MBytes of computer memory. In a Pentium 4 computer at 2 GHz with 512 MBytes of memory, the image-dependent algorithm required about six minutes to compress the MicroZip test set (note that this compression time is only indicative, because the code has not been optimized for speed.).Just for comparison, the compression standards took approximately one minute to encode the same set of images.

The results obtained have been compared with three image coding standards, JBIG, JPEG2000 and JPEG-LS, with the previously proposed method based on image-independent contexts and with two recent specialized methods: MicroZip and Zhang's method. The results obtained show that the proposed method has better compression performance in all three test sets.

Image	JPEG2000	JBIG	JPEG-LS	MicroZip	Zhang	Image independent	Image-dependent 256×256	Image-dependent Full
array1	11.973	11.819	11.590	11.490	11.380	11.105	11.120	11.056
array2	9.226	9.071	8.737	9.570	9.260	8.628	8.470	8.423
array3	8.551	8.351	7.996	8.470	8.120	7.962	7.717	7.669
Average	**9.515**	**9.297**	**8.974**	**9.532**	**9.243**	**8.840**	**8.667**	**8.619**

Table 2: Compression results, in bits per pixel, obtained for MicroZip test set using three standard image compression methods, namely JPEG-LS, JPEG2000 and JBIG, two specialized methods, MicroZip and Zhang's method, the image-independent method and the proposed image-dependent method. The "256×256" column indicates results obtained with a context model adjusted using only a square of 256×256 pixels at the center of the microarray image, whereas "Full" indicates that the search was performed in the whole image.

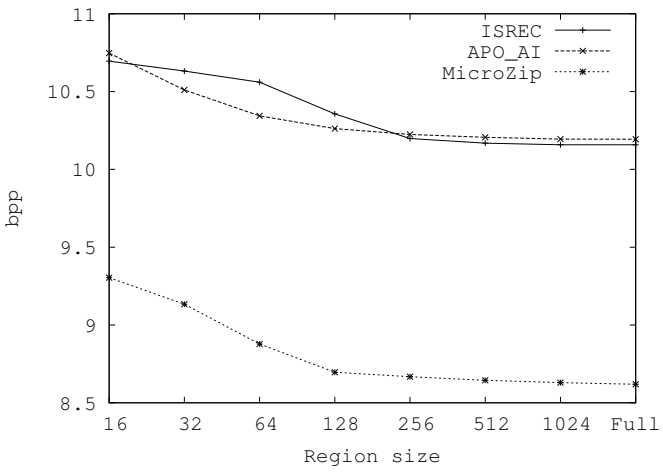


Figure 4: Average number of bits per pixel required for encoding the three sets of images used in this paper for different sizes of the search area.

REFERENCES

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

[2] P. Hegde, R. Qi, K. Abernathy, C. Gay, S. Dharap, R. Gaspard, J. Earle-Hughes, E. Snesrud, N. Lee, and John Q., "A concise guide to cDNA microarray analysis," *Biotechniques*, vol. 29, no. 3, pp. 548–562, Sept. 2000.

[3] ISO/IEC, *Information technology - JPEG 2000 image coding system*, ISO/IEC International Standard 15444–1, ITU-T Recommendation T.800, 2000.

[4] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, Sept. 2001.

[5] ISO/IEC, *Information technology - Lossless and near-lossless compression of continuous-tone still images*, ISO/IEC 14495–1 and ITU Recommendation T.87, 1999.

[6] 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.

[7] ISO/IEC, *Information technology - Coded representation of picture and audio information - progressive bi-level image compression*, International Standard ISO/IEC 11544 and ITU-T Recommendation T.82, Mar. 1993.

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

[9] S. Lonardi and Y. Luo, "Gridding and compression of microarray images," in *Proc. of the IEEE Computational Systems Bioinformatics Conference, CSB-2004*, Stanford, CA, Aug. 2004.

[10] Yong Zhang, Rahul Parthe, and Don Adjeroh, "Lossless compression of DNA microarray images," in *Proc. of the IEEE Computational Systems Bioinformatics Conference, CSB-2005*, Stanford, CA, Aug. 2005.

[11] D. Adjeroh, Y. Zhang, and Rahul Parthe, "On denoising and compression of dna microarray images," *Pattern Recognition*, vol. 39, pp. 2478–2493, Feb. 2006.

[12] R. Jörnsten, W. Wang, B. Yu, and K. Ramchandran, "Microarray image compression: SLOCO and the effect of information loss," *Signal Processing*, vol. 83, pp. 859–869, 2003.

[13] J. Hua, Z. Xiong, Q. Wu, and K. Castleman, "Fast segmentation and lossy-to-lossless compression of DNA microarray images," in *Proc. of the Workshop on Genomic Signal Processing and Statistics, GENSIPS*, Raleigh, NC, Oct. 2002.

[14] D. S. Taubman and M. W. Marcellin, *JPEG 2000: image compression fundamentals, standards and practice*, Kluwer Academic Publishers, 2002.

[15] N. Faramarzpour, S. Shirani, and J. Bondy, "Lossless DNA microarray image compression," in *Proc. of the 37th Asilomar Conf. on Signals, Systems, and Computers, 2003*, Nov. 2003, vol. 2, pp. 1501–1504.

[16] M. Burrows and D. J. Wheeler, *A block-sorting lossless data compression algorithm*, Digital Equipments Corporation, May 1994.

[17] 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.

[18] 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.

[19] R. Jörnsten, Y. Vardi, and C.-H. Zhang, "On the bitplane compression of microarray images," in *Proc. of the 4th Int. L1-norm Conf.*, Y. Dodge, Ed., 2002.