Influence of Dictionary Size on the Lossless Compression of Microarray Images

Robert Bierman[1], Rahul Singh[1]
Department of Computer Science, San Francisco State University, San Francisco, CA
bierman@sfsu.edu, rsingh@cs.sfsu.edu

Abstract

A key challenge in the management of microarray data is the large size of images that constitute the output of microarray experiments. Therefore, only the expression values extracted from these experiments are generally made available. However, the extraction of expression data is effected by a variety of factors, such as the thresholds used for background intensity correction, method used for grid determination, and parameters used in foreground (spot)-background delineation. This information is not always available or consistent across experiments and impacts downstream data analysis. Furthermore, the lack of access to the image-based primary data often leads to costly replication of experiments. Currently, both lossy and lossless compression techniques have been developed for microarray images. While lossy algorithms deliver better compression, a significant advantage of the lossless techniques is that they guarantee against loss of information that is putatively of biological importance. A key challenge therefore is the development of more efficacious lossless compression techniques. Dictionary-based compression is one of the critical methods used in lossless microarray compression. However, the image-based microarray data has potentially infinite variability. So the selection and effect of the dictionary size on the compression rate is crucial. Our paper examines this problem and shows that increasing the dictionary size beyond a certain size, does not lead to better compression. Our investigations also point to strategies for determining the optimal dictionary size.

1. Introduction

The usefulness of microarrays has become clear over the past decade in the study of function, regulation and interaction of large number of genes. Microarrays allow the analysis of a broad spectrum of genes in a single experiment[1-4]. The output of microarray experiments takes the form of large TIFF image files. Microarray images are 16 bpp (bits per pixel) grayscale images characterized as having a background and foreground (spots), see Figure 1. Given the widespread use of microarray technology, an emerging problem is that of storage and management of microarray data. Each experiment today produces approximately forty megabytes of image data. This large size makes it difficult to store the images in an online system. Often the images are offloaded to CD's and only the secondary data (expression data) is left publicly available online[5]. This has lead to the research and development of compression technologies for microarray images[6-9] to allow better access and reduced data transmission of microarray images.

Microarray compression techniques may be lossy or lossless. Lossy compression of microarray images is based on the general philosophy that spots contain biologically relevant information, as opposed to the image background. Therefore the background can be compressed using lossy compression. Two of the specialized techniques[7, 8] for lossy compression are based on this philosophy and segment the image into foreground (the biologically relevant data) and background (that portion of the image without biologically relevant spots). SLOCO (which is based on JPEG-LS[10]) supports a lossy-compression on

[1] Equal contributors

Accepted for publication in the 20th IEEE International Symposium on Computer-Based Medical Systems. June 20-22, 2007, Maribor, Slovenia.

the background under the condition that the resulting data loss is less than the variability in replicated experiments. The primary disadvantage of the above techniques is their dependence on separating the spots from the background. As can be seen from Figure 1, spots are often noisy and difficult to algorithmically delineate at their boundaries. Therefore, spot-detection-based compression can be critically influenced by the quality of spot extraction. In the case of poor segmentation, this can, among other consequences, lead to the loss of important data around spot edges. Lossy image compression algorithms such as JPEG2000[10] and JBIG[11] have also been applied to this problem.

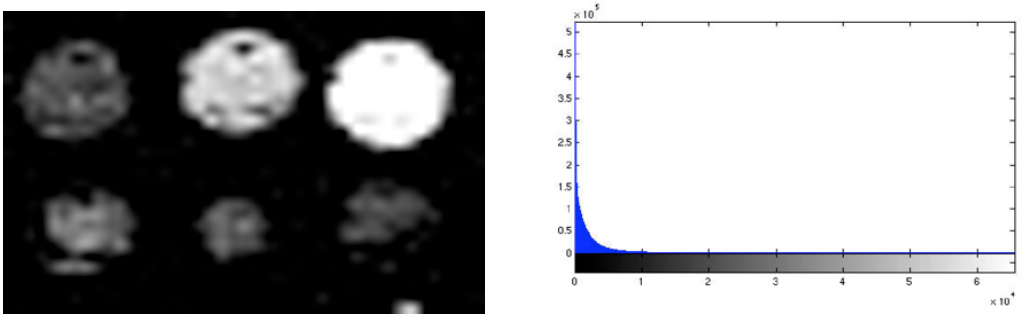

Figure 1. Spots from a microarray illustrating the variability in spot quality and the consequent difficulty in demarcating the spot from the background. (right) Intensity histogram.

Avoiding the issues of potential data loss, lossless microarray compression has involved application of standard methods such as gzip and JPEG-LS[10]. Comparative experiments[9] on gzip, JPEG2000, JBIG, and JPEG-LS found that JPEG-LS provided the best lossless compression. Another specialized technique, MACE[6], separates the image into a low intensity array (all data values below a specific threshold) and a high intensity array (all data values equal to or above a specific threshold) based on the intensity distribution of the image (rather than foreground and background). Comparisons[6] show MACE as having the best lossless compression.

Lossless microarray compression techniques such as MACE and LZW[12] have dictionary-based compression (a substitution compression technique where a reference value is substituted in place of a string or "word") as one of their core features. An important parameter in the performance of these techniques, therefore, is the dictionary size. The dictionary size is limited by the maximum reference value. The number of bits allowed for each reference determines the maximum reference value, which in many systems is 12 bits. This limits the dictionary size to 4096 (2^{12}) entries. Since the microarray images are not text, the alphabet (strings) one has to deal with is unlimited. Clearly dictionary sizes of 4096 are inadequate for microarray compression. *The question then arises as to how large the dictionary should be to account for the variability of the "microarray image alphabet" and if increasing the dictionary size will necessarily yield better compression.*

Our research answers that question by investigating the influence of dictionary size on the efficacy of lossless microarray compression. Our experiments utilize the MACE compression technique. This choice is prompted by results[6], which indicate the effectiveness of MACE in terms of its compression rate. However, our results would be valid for any microarray compression technique that employs dictionary-based compression. This paper is organized as follows: in section 2 we provide a brief overview of MACE and discuss the basic principle of dictionary based compression. In section 3 presents our experimental investigations which show that compression rates improve with increasing dictionary size but only to a certain point. Increasing dictionary sizes beyond this point either do not lead to improved compression or may even degrade the compression rate.

2. OVERVIEW OF MACE AND DICTIONARY-BASED COMPRESSION

Another fundamental characteristic of microarray images is their skewed intensity distributions where there is majority of low intensity pixels (Figure 1 right). This skewed intensity distribution causes havoc with normal compression utilities such as gzip that only compresses the image file size by 25-35%[6, 9]. MACE utilizes this characteristic to divide the image into a high intensity array (HIA) and a low intensity array (LIA). This is similar in concept to the foreground and background other specialized techniques[7, 8] but, MACE makes no assumptions as to the biological significance of any pixel and employs lossless compression. A key to MACE's compression is the choice of threshold that separates the low and high intensity arrays. As an example, if the threshold is chosen as 256 then all values in the low intensity array can be represented in one byte (8 bits) verses two bytes (16 bits), this threshold choice allows for the immediate reduction of the low intensity array by 50% prior to application of further steps in compression. As long as the high intensity array contains only a small fraction of the pixels in the image, it can be represented as a spares matrix and compressed. This is possible due again to the skewed intensity distribution.

MACE can be seen as generating three arrays from the original image: the high intensity array (HIA), the most-significant byte array of the low intensity array (MSB) and the least-significant byte of the low intensity array (LSB). Each of these arrays is independently compressed and in the case when the separation threshold is 256 (or less) the MSB can be completely eliminated.

There are many different compression algorithms that transform the original information in a variety of ways. Some of these algorithms include simple run-length encoding, arithmetic encoding, and dictionary-based compression. Dictionary-based compression like LZW[12] stores references to patterns that are built dynamically.

Each of the specialized techniques recognizes the need for separation of the foreground and background (or low and high intensity values). In this paper we conduct a systematic investigation of the effect that dictionary size has on the compression of microarray images and these separated planes.

The premise behind dictionary-based compression is that there are patterns that repeat themselves throughout the data. Therefore compression can be achieved by storing only a reference to the pattern rather than the entire pattern. However, the dictionary needs to be data driven (and generated dynamically), for the process to be generic. Probably the best known of dictionary-based compression techniques is the Lempel-Ziv-Welch (LZW)[12] compression.

Dictionary-based compression starts off with an initial dictionary of 256 references. These references represent the single byte values 0-255. The compression works on pairs of bytes and or references. The first byte's reference is written to the output, that reference is then placed in a previous holder. The next byte is read, then the *string* made up by the previous reference and the new byte is compared against the dictionary. If a match is found, that new reference from the dictionary is placed in the previous holder and another byte is read from the input. If this *string* is not found, a new entry is made in the dictionary for that *string* and the value in the previous holder (a reference value) is written to the output. Then, the current reference is moved to the previous holder and the loop continues. In the following example we denote the dictionary reference of a symbol (or set of symbols) in {}.

Given the string "ABCABC" and the initial dictionary, first the A {65} is read and moved to the previous holder. Then the B {66} is read. This forms the *string* AB that is not found in the dictionary, so it is added as reference {256}. The previous holder value, reference {65}, is written to the output using the number of bits specified as the dictionary size. Finally the

current reference, {66}, is moved to previous holder. That completes one cycle of the process. The C {67} is then read and the dictionary is searched for BC Since BC is not present, BC is added to dictionary as {257}. The previous holder value, {66}, is written to the output and {67} moved to the previous holder. Next in the input is another A. The search for CA is not found, so entry {258} is created and the previous holder value, {67}, is written to the output and {65} moved to the previous holder. Now another B is read and the dictionary searched for AB, which is found as {256}. The {256} is moved into the previous holder and the next byte is read. The C is read from the input and the dictionary is searched for the *string* ABC that is not found, so entry {259} is added to the dictionary. The previous holder value, {256}, is written to the output and since we are now at the end, the {67} is then written. So now our six-character string is instead represented by five dictionary references, {65, 66, 67, 256, 67}. This example shows as the string gets larger and the number of repeated patterns increase, the number of references required will drop. Since we only need 9 bits to represent all the reference values in this example, the total number of bits needed is 5 * 9 bits or 45 bits, verses the original that use 8 bits per character, therefore are using 6 * 8 bits or 48 bits. Even in this short example, we have saved 3 bits of data space.

3. EXPERIMENTAL INVESTIGATIONS

The size of the dictionary depends on the number of bits for each output reference (such as 4096 when using 12 bits). We tested the effects of dictionary-based compression using dictionary sizes from 12 to 24 bits per entry or 4096 to 16,777,216 entries. The graph in Figure 2 (left) shows a number of LSB and Figure 2 (right) MSB arrays over various images and thresholds. The actual values are not of major significance but are provided as a reference. What is of significance are the trend lines, that show improvement as the dictionary size increases but only to a certain extent. The trend lines that show the best performance around a dictionary size of 16 bits are those of the MSB, which have a more uniform pattern (having only 1 to 6 bits per pixel to start with). While, the improvement for the LSB ends at 21-22 bits per entry. As the entropic nature of the LSB increases it becomes necessary to have a larger dictionary to store larger patterns for reuse.

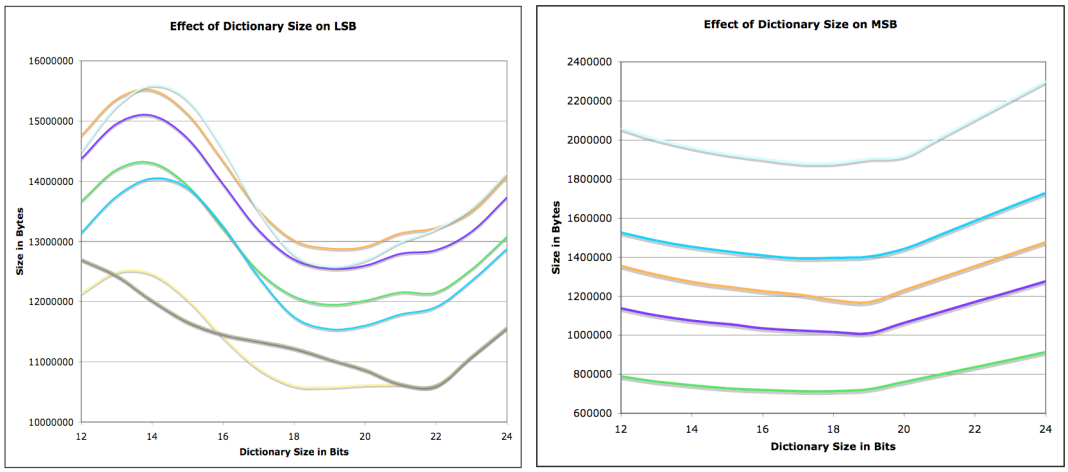

Figure 2. Dictionary size comparison (left) comparing various LSB arrays (left scale) and MSB arrays (right scale).

Key to this improvement is the number of additional patterns utilized as the dictionary size increases.

Table 1 is an example of the results on a LSB across various dictionary sizes. An important point is that the dictionary resets itself once it reaches capacity. Therefore, more entries of shorter length get created. Another fact of dictionary-based compression is the number of output references is one more then the number of directory entries created. This is because the only time you output is after creating a new dictionary entry except for the first character. The key is the number of output references (dictionary references) needed for each dictionary size. The overall performance is improved when the delta of the number of references output is greater then the number of additional bits needed to hold the output references. To show improvement the case with a dictionary size of 22 bits must use less output references then 21 bits, specifically the number out output references has to be reduced by at least 4054225/ 21 = 193,058.33. As shown in Table 1 the difference in output references from 21 to 22 bits is 194,344 which is larger than the target and as shown the output size is smaller for 22 bits then 21 bits. We can correctly predict the small difference in size from the small delta from our target number. Generalizing this equation we get $T_n = S_{n-1} - \left({}^{S_{n-1}}\!/_{(n-1)} \right)$, where T is the target number of output references, n is the number of bits for the dictionary size, and S is the actual number of output references for the specified bit size. This generalized formula also holds for the worst case and ${}^{T_n}\!/_2$ holds as the average case when utilizing a variable length dictionary encoding. Variable length dictionary encoding starts from some base bit size, usually 10 bits, and each output sample is of that length, as the dictionary grows to over 2^{10} then the dictionary and output both start using 11 bits, and so on to the upper bit limit. That upper bit limit is represented by n in the above equations. In the general case for large images such as microarrays, variable length encoding can save about 5% as shown in the example in Table 2.

Table 1. Example of the results of the number of output dictionary references and output size across a series of dictionary sizes.

Dictionary Bits	Number of output references	Output Size (bits*samples)/8
12	8473706	12710559
13	7656924	12442502
14	6874976	12031208
15	6226793	11675237
16	5733713	11467426
17	5341876	11351487
18	4994333	11237250
19	4653822	11052828
20	4351146	10877865
21	4054225	10642341
22	3859881	10614673
23	3859881	11097158
24	3859881	11579643

Table 2. Shows the number of bytes saved by using a variable length dictionary size capped at 22 bits for the same sample as used in Table 1.

Number of bits of output	# of refs written at bit depth	Size in Bytes	
10	768	960	
11	1024	1408	
12	2048	3072	
13	4096	6656	
14	8192	14336	
15	16384	30720	
16	32768	65536	
17	65536	139264	
18	131072	294912	
19	262144	622592	
20	524288	1310720	
21	1048576	2752512	
22	1762985	4848209	
Total reference entry outputs:			3859881
	Size at Fixed Width	**Size at Variable Width**	**Savings**
Total	10614673	10090897	523776
			4.93%

4. CONCLUSION

While dictionary-based compression is not the only compression applied in microarray image compression, the quantification of optimal results is critical. These experiments and analysis shows ways of predicting optimal dictionary sizes within a relatively accurate range. There is a size at which the image no longer generates additional output references since every dictionary entry for that string is already in the dictionary by the time the input string is exhausted. Using this maximum entry value M we can get the maximum number of bits needed by taking the $\lceil \log_2 M \rceil$. This value is generally a close approximation for the optimal value and for the cap in variable length compression. Anything larger will simply occupy more space without any additional benefit and will eventually cause a negative compression rate.

These findings can be applied to any dictionary-based compression of microarray images. The ramifications of these findings not only aid in the dictionary-based compression of microarrays, but any large alphabet data file. It shows that an infinite alphabet does not require an infinite dictionary and that this quantification is easily determined.

5. REFERENCES

[1] O. Alter, P. O. Brown, and D. Botstein, "Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms," *Proc. Nat. Acad. Sci.*, vol. 100, pp. 3352-3356, Mar. 2003.

[2] J. DeRisi, L. Penland, P. O. Brown, M. L. Bittner, P. S. Meltzer, M. Ray, Y. Chen, Y. A. Su, and J. M. Trent, "Use of a cDNA microarray to analyse gene expression patterns in human cancer," *Nat Genet*, vol. 14, pp. 457-60, 1996.

[3] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown, "Expression monitoring by hybridization to high-density oligonucleotide arrays," *Nat Biotechnol*, vol. 14, pp. 1675-80, 1996.

[4] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, vol. 270, pp. 467-70, 1995.

[5] G. Sherlock, T. Hernandez-Boussard, A. Kasarskis, G. Binkley, J. C. Matese, S. S. Dwight, M. Kaloper, S. Weng, H. Jin, C. A. Ball, M. B. Eisen, P. T. Spellman, P. O. Brown, D. Botstein, and J. M. Cherry, "The Stanford Microarray Database," *Nucleic Acids Res*, vol. 29, pp. 152-5, 2001.

[6] R. Bierman, N. Maniyar, C. Parsons, and R. Singh, *MACE: lossless compression and analysis of microarray images*. Dijon, France: ACM Press, 2006.

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

[8] S. Lonardi and Y. Luo, "Gridding and Compression of Microarray Images," *Proc. of the 2004 IEEE Comp. Sys. Bioinformatics Conference*, 2004.

[9] A. J. Pinho, A. R. C. Paiva, and A. J. R. Neves, "On the use of standards for microarray lossless image compression," *Biomedical Engineering, IEEE Transactions on*, vol. 53, pp. 563-566, 2006.

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

[11] "Information Technology—Coded Representation of Picture and Audio Information—Progressive Bi-Level Image Compression," *ISO/IEC Standard 11 544*, 1993.

[12] M. Nelson, "LZW Data Compression," in *Dr. Dobb's Journal*, 1989, pp. 29-36, 86-87.