Efficient Compression from Non-ergodic Sources with Genetic Algorithms

Angel Fernando Kuri-Morales
Instituto Tecnológico Autónomo de México
Río Hondo No. 1, México D.F.
akuri@itam.mx

Abstract

Several lossless data compression schemes have been proposed over the past years. Since Shannon developed information theory in his seminal paper, however, the problem of data compression has hinged (even though not always explicitly) on the consideration of an ergodic source. In dealing with such sources one has to cope with the problem of defining a priori the minimum sized symbol. The designer, therefore, is faced with the necessity of choosing beforehand the characteristics of the basic underlying element with which he or she is to attempt data compression. In this paper we address the problem of finding the characteristics of the basic symbols to consider in information treatment without assuming the form of such symbols in the data source. In so doing we expect to achieve a pseudo-ergodic behavior of the source. Then we are able to exploit the characteristics of such sources. Finding the basic elements (which we call "metasymbols") is a complex (NP complete) optimization task. Therefore, we make use of a non-traditional Genetic Algorithm (VGA) which has been shown to have excellent performance, to find the metasymbols. In this paper we discuss the problem, the proposed methodology, some of the results obtained so far and point to future lines of research.

1. Introduction

In the publication in 1948 [1] of Shannon's original paper on information theory (IT) the concept of information attached to a *symbol* stemming from a source assumed unknown but from which it is possible to estimate its probability was born. We may, then, define the information associated to such *symbol* as

$$I(s_i) = -\log(p_i) \tag{1}$$

where s_i denotes the i-th *symbol* and p_i denotes the probability that the *symbol* is output from the source; logarithms are assumed base 2. This definition is appealing in that it satisfies two intuitive notions about information: First, it assigns greater information to that which is more unexpected, that is, something not expected "tells" us more than something which is frequent. Second, it corresponds to our idea that information must be additive; that is, the information of two *symbols* should be the same as the information of each taken separately:

$$\begin{aligned} I(s_1) + I(s_2) &= I(s_1, s_2) \\ &= -\log(p_1) - \log(p_2) \\ &= -[\log(p_1 p_2)] \end{aligned} \tag{2}$$

This, clearly, shows the two probabilities as a product and the amount of information as an addition. This is an engineering definition based in probabilities and not one based in the meaning of the *symbols* for a human receiver. It is easy (and common) to confuse the knowledge associated to the process of interpretation of the *symbols* with the information they carry.

On the other hand, information defined in such way is also consistent in the sense that it restricts how we define a *symbol*. Assume, without loss of generality, that when conveying information we imply some sort of binary encoding. In this case, IT allows us to define *symbols* as *n*-ary sets of bits, where *n* is arbitrary. For example, if $n_1 = 4$, then the alphabet corresponds to 16 bits and there are 16 probabilities: one for each one of the *symbols*, with their corresponding information. Now let us assume that we decide to take 8 bits per *symbol*. In this case ($n_2 = 8$) we have 256 *symbols* with the corresponding probabilities and information. However, in view of (2), the information associated to pairs of *symbols* for n_1 is the same as the one associated to individual symbols for n_2.

A basic concept associated to the previous definitions is that of the average information for a given source. This quantity is called the *entropy* and is expressed as

$$H(S) = -\sum_{i=1}^{q} p_i \log(p_i) \tag{3}$$

It is important to remark that a phrase such as "Let us consider the entropy of the source..." is meaningless unless we include a model of such source. The entropy function measures the amount of information we get from the results of some experiment. Because of this, when we

design such experiment, we usually want to maximize the amount of information we want to get; we wish to maximize its entropy function.

The foregoing discussion tacitly implies statistical independence of the *symbols*. In fact, the development of classical IT is based on the assumption that the source is ergodic. A process (in particular, a Markov process [2]) is called *ergodic* if from any state we can reach any other state and if, when $t \to \infty$ the system stabilizes to a limit distribution regardless of the initial state. In practice, however, sources are not ergodic. In [3] Shannon himself discusses a set of approximations to the English language which he called of first, second and third order approaches to independent symbols (letters or spelling signs), on the one hand, and of first and second order approaches to words, on the other. This two arbitrary choices seem to be quite natural: a symbol may be easily identified because it exists as a physical entity, whereas a word is clearly delimited with blanks between the letters of which it is made. There are, of course, further reasons of semantic, syntactic or cultural nature which allow for easy identification. The selection of terms is, however, unfortunate. In Shannon's example the term "symbol" and the term "word" both correspond to what we have called, simply, a *symbol* in the preceding discussion. If statistical independence were guaranteed, the election of *n* would be arbitrary. Shannon points out [4]: "Rather than proceeding with a structure of a 4-gram,..., η-gram it is easier and better to pass, at this point, to consider words as units". And this reflects the non-ergodicity of the sources which we must normally face.

One of the objectives of this work is to propose and establish a method which allows us to replace the symbols of classical IT with *symbols* which, in an effective way, display statistical independence such that we may extend the concepts of information and entropy without assuming ergodicity. Our thesis is that if the symbols of a finite and bounded message are adequately selected, these may be mapped to a message generated by an ergodic source.

The *symbols* in our discussion are, in general, sets of basic units whose length may be arbitrarily defined. These sets of units (which we shall call *metasymbols*) must satisfy, in general two desirable characteristics (in terms of maximizing compression):

a) They must be as large as possible

b) They must appear as frequently as possible

These two criteria are, in general, mutually incompatible. To see why, consider a message consisting of a continuous string of 'A's. Notice that the largest possible metasymbol is the message itself (taken as a whole). On the other hand, the metasymbol with the greatest frequency is the letter 'A': the smallest lexical unit.

We also know that average code length is bounded by the entropy of the message. Since we wish to maximize data compression, this is, therefore, a third desirable criterion.

Attempting to find the set of metasymbols complying with all three criteria above is a complex optimization task which can be tackled with the proper genetic algorithm. If it may be achieved, however, it will allow us to consider an alternative source whose entropy is minimized.

The rest of the paper is developed as follows: in part 2 we discuss a genetic algorithm to effectively identify the metasymbols. In part 3 we discuss some of the technical aspects of the implementation of such algorithm. In part 4 we outline some of the results obtained so far and offer our conclusions.

2. The "Best" Genetic Algorithm

The optimization problem to be tackled requires an optimization algorithm that is demonstrably efficient. In the literature several genetic algorithms (GA) have been discussed and analyzed. In what follows we make a brief account of the known facts about the better known variation of a GA (the so-called *Simple* or *Canonical* GA or CGA), pointing out its advantages and shortcomings. Then we discuss a non-traditional algorithm (the so-called *Vasconcelos* GA or VGA) which was designed to overcome the CGA's limitations while retaining its desirable characteristics. We recall a study which seems to show that the VGA is, in general, far superior to the CGA.

2.1 The Simple Genetic Algorithm

Perhaps the best known GA is the one originally proposed by Holland [5] and later popularized by Goldberg [6]. This is the so-called *Simple* GA. The CGA is known to possess the following properties: a) It samples certain elements (the so-called *schemas*) in the genomes of its population exponentially in direct proportion to the adequacy of these elements. That is, it explores the space of solutions in a directed way such that the apparently better sections in the coded solutions under scrutiny are examined preferably. Likewise, it disregards apparently undesirable sections of the said code. 2) It explores $O(N^3)$ such elements for a population of size *N* during every iteration. A CGA, therefore, approaches the ideal strategy of exploration/exploitation when faced with dynamic problems. These two characteristics explain why a CGA approaches a very good solution in a short number of iterations under proper conditions. The GA "assembles" an ever more complex encoding of the solution from smaller and simpler components. 3) It is also known that a CGA does not converge to the best solution even in an infinite number of steps. But by the simple expedient of retaining the best individual up to the last generation the

GA (now transformed into the *eliTist* GA or TGA) does converge to the best possible solution given enough time. 4) The TGA (or the CGA, for that matter) reaches a steady state behavior regardless of the way the initial population is chosen. This explains why the number of individuals in the population is not relevant to the convergence properties of the algorithm, although it does, indeed, have bearing on its efficiency in reaching such convergence.

The above mentioned characteristics explain, in part, why genetic algorithms have, in practice, become such an appealing choice when tackling optimization problems, particularly when the said problems are not amenable to a closed, mathematically pleasing expression and lack some of the frequently required properties (such as differentiability, convexity, etc.). But the CGA (or TGA) is no panacea. At least two undesirable features have been identified which impair the algorithm's performance. 1) Certain fitness functions may supply the algorithm with invalid information in terms of approaching the desired global optimum. These have been called *deceptive* functions, since they trick the CGA into "believing" it is getting closer to the result when it may not be so. 2) When identifying the desirable "simple" elements which, hopefully, will compose the solution when properly combined, the CGA also retains some uncalled for sections of code which remain with the desired ones. This process of undesirable schema traveling along with the host has been called *spurious correlation* and is a major source of inefficiency during the CGA's execution.

2.2 Vasconcelos' Genetic Algorithm

In an effort to ameliorate the shortcomings of the CGA several variations of a GA have been tried. An interesting alternative (for reasons to be discussed shortly) seems to be the so-called Vasconcelos' GA or VGA [7]. In the VGA proportional selection which gives rise to *N* new individuals from *N* older ones, is replaced by what in evolutionary strategies has been called a $\mu + \lambda$ selection strategy, meaning that the new population comprised of λ individuals joins the older population's μ individuals of which only the *N* better individuals are retained. Furthermore, the *N* remaining individuals are crossed deterministically as follows. All individuals are sorted from best to worst. Then the *i*-th individual is crossed with the (*N*-*i*+*1*)-th with probability p_c for *i=1,...,N/2*. The apparently contradictory strategy where good performers are crossed with the poor ones is explained when one considers that elitism is, here, of the strongest kind, i.e. only the best *N* individuals of every generation are retained. In this way, the VGA seeks for variety in the elements of the genome while, by dynamically disrupting the "good" schemas (but keeping the overall best) dominant, deceptive and spurious schemas are minimized.

The mechanics of VGA (and, for that matter, almost any breed of GA) is a complex affair. Holland's schema analysis is restricted to the CGA. Other authors have tried different approaches but, as of to date, no generalized theoretical treatment which is able to model a GA with arbitrary characteristics has been developed with success. In [8] a more pragmatic approach was taken in order to establish a general methodology which would allow us to establish the relative performance of any two GAs. The basic idea is to tackle a set of problems with the algorithms we wish to compare and measure the best value reached by the algorithm in a predefined number of generations for each problem in the set. We may then extract the minimum values probability distribution's main parameters (μ and σ). Once knowing these parameters we find a worst case value (ζ) by making $\zeta = \mu + 4\sigma$. From Chebyshev's theorem we know that with probability close to 0.95 the minimum values gotten from the algorithm in question will be better than ζ. Two problems arise in this approach. 1) How to select the problems to minimize and 2) How to extract the probability distribution's parameters reliably even in the absence of knowledge regarding the problems to be minimized. It is common to select a small set of selected functions (a suite) which, hopefully, will encompass a wide range of characteristics. Rather than picking these functions by hand, here problem (1) was solved by generating automatically a set of Walsh polynomials in a very large domain. The VGA minimized 10,894 different functions whereas the CGA minimized 12,376. All of the 23,268 functions were generated randomly. Problem (2) was solved by determining the sample size from well known theorems of probability theory. Basically, the mean and standard deviation for a sampling distribution of means were calculated ($\mu_{\bar{x}}$, $\sigma_{\bar{x}}$). Uniformity of the sampling distribution was determined from a χ^2 goodness of fit test. Then the original population's parameters may be calculated from $\mu = \mu_{\bar{x}}$ and $\sigma = \sqrt{n}\,\sigma_{\bar{x}}$.

Table 1 shows the relative performance of CGA and VGA for 30, 50, 100 and 150 generations arrived at from the methodology just outlined.

From Table 1 we see that VGA is O(30%) more efficient than CGA as the number of generations increases when one considers an optimization process carried on a set of unbiased functions. It is pertinent to remark, at this point, that other GA breeds were also compared in [8]. Of all the different GAs the most efficient was VGA.

Table 1. Relative performance of VGA and CGA

	Generations			
	30	50	100	150
$\frac{\zeta_{VGA}}{\zeta_{CGA}}$	1.08	1.29	1.20	1.267

Keeping this in mind and, in view of the known characteristics of the CGA, we have selected VGA as our optimization tool.

3. An Algorithm for Metasymbol Identification

The purpose of the algorithm is to identify structures within a data file. These structures will be called metasymbols (MS) or structures or patterns. We assume the data in binary, with no particular encoding.

Before attempting to describe the fitness function we must agree on how to represent a possible solution. This we do in what follows.

Let $N \equiv$ Number of elements in the message

By "element" we mean a basic unit of data. For instance, if we consider bytes as basic units of data, then a file with 1024 bytes will have 1024 elements; if, on the other hand, we consider nibbles, the same file will have 2048 elements, an so on.

$P \equiv$ Number of possible patterns

$P_i \equiv$ the i-th pattern

$$0 \le i \le P-1$$

The algorithm will try to identify structures within the data. Some will be found more than once. By convention we will not allow more than $P=\lceil N/4 \rceil$ patterns.

We assume that structures appearing for the first time in the data will be encoded such that its first bit is a "0", whereas the second and successive apparitions of such structure will be encoded with a "1" in its first position. For example, if N = 32, the first occurrence of pattern "3" will be encoded as **0 011**; the second occurrence (and all others) will be encoded as **1 011**. Therefore, the number of bits per pattern (L) is given by:

$$L = 1 + (\lceil \log N \rceil - 2) \qquad (4)$$

Example:

$$N = 128$$
$$\log N = 7$$
$$\therefore$$
$$L = \underset{\text{instance}}{1} + 7 - \underset{1/4}{2} = 6$$

The length of the genome where this information will be stored is given by:

$$G = L \times N = (1 + \lceil \log N \rceil - 2) \times N$$

$$G = N(\lceil \log N \rceil - 1)$$

For example, if N = 256, G = 7 x 256 = 1,792.

We illustrate with N=64. Assume we have the following string:

A 1 1 A B 2 A B V C B M C 3 D C W D A N D B O 2 X G Y E 4 P E V
V E 1 V X Y 2 W W 1 Q W X Y 3 X X Y Y X A Y R B S X Y G T 4 2 J

We wish to find the patterns as outlined above. For convenience, we express it in a two-dimensional 8X8 matrix as shown in figure 1.

	1	2	3	4	5	6	7	8
1	A	1	1	A	B	2	A	B
2	V	C	B	M	C	3	D	C
3	W	D	A	N	D	B	O	2
4	X	G	Y	E	4	P	E	V
5	V	E	1	V	X	Y	2	W
6	W	1	Q	W	X	Y	3	X
7	X	Y	Y	X	A	Y	R	B
8	S	X	Y	G	T	4	2	J

Figure 1. A bidimensional message

We identify 7 patterns including the null pattern. By convention the absence of a pattern (the *empty* or *null* pattern) will be denoted by M0.

The patterns may be expressed as the coordinates of the first element, followed by the number of empty squares (*blanks*) between the i-th and the (i+1)-st elements.

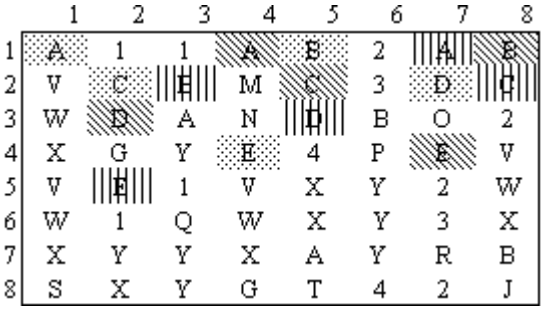
For instance, the first pattern is {ABCDE}. Its is defined by: {(1,1):3,4,4,12}. This pattern (pattern 1) is highlighted in figure 2.

	1	2	3	4	5	6	7	8
1	A	1	1	A	B	2	A	B
2	V	C	B	M	C	3	D	C
3	W	D	A	N	D	B	O	2
4	X	G	Y	E	4	P	E	V
5	V	E	1	V	X	Y	2	W
6	W	1	Q	W	X	Y	3	X
7	X	Y	Y	X	A	Y	R	B
8	S	X	Y	G	T	4	2	J

Figure 2.First Occurrence of Pattern 1

Pattern 1 also appears in the cells starting with coordinates (1,4) and (1,7). The three occurrences of matrix, for instance, will be encoded as:

pattern 1 are shown in figure 3. The first apparition of pattern 1 is the "master" template, whereas the second and subsequent ones are the "slave" templates. We call them M1 and S1, respectively. Their coordinate description would be, {(1,1),(1,4),(1,7):3,4,4,12}.



	1	2	3	4	5	6	7	8
1	A	1	1	A	E	2	A	E
2	V	C	E	M	C	3	D	C
3	W	D	A	N	D	B	O	2
4	X	G	Y	E	4	P	E	V
5	V	E	1	V	X	Y	2	W
6	W	1	Q	W	X	Y	3	X
7	X	Y	Y	X	A	Y	R	B
8	S	X	Y	G	T	4	2	J

Figure 3. All Occurrences of Pattern 1

The format we use to describe the master and slave patterns is as follows: $\{M_i,S_i,S_i,...,S_i:(b_1,b_2,...,b_N)\}$. Where "M" is the pair of initial coordinates of the master; S_i is the pair of coordinates of slaves *i*; b_i are the blanks (left to right) between two successive symbols in the pattern. The set of patterns describing the message of figure 1 is, therefore, as follows:

Pattern 1 (ABCDE): {(1,1),(1,4),(1,7):3,4,4,12}
Pattern 2 (1234): {(1,2), (5,3): 3, 7, 14 }
Pattern 3 (VWXY): {(2,1), (5,1), (5,4), (4,8): 7, 7, 1}
Pattern 4 (ABG): {(3,3), (7,5) : 2, 3}
Pattern 5 (12): {(1,3), (6,2): 20}
Pattern 6 (XY): {(5,5), (6,5), (8,2): 0}

Those symbols not included in any of the patterns above are clustered in "pattern" M0. Calling the master pattern of metasymbol *i* M_i and all its slaves S_i we may represent the full message as shown in figure 4.

	1	2	3	4	5	6	7	8
1	M1	M2	M5	S1	M1	M2	S1	S1
2	M3	M1	S1	M0	S1	M2	M1	S1
3	M3	S1	M4	M0	S1	M4	M0	M5
4	M3	M4	M3	M1	M2	M0	S1	S3
5	S3	S1	S2	S3	M6	M6	S2	S3
6	S3	S5	M0	S3	S6	S6	S2	S3
7	S3	S3	S3	S3	S4	S3	M0	S4
8	M0	S6	S6	S4	M0	S2	S5	M0

Figure 4. Master-Slave Mapping

Now we are able to encode the text in terms of the patterns determined above. From (4) we have that L=5. Now we encode masters with a 0 in its first bit; slaves with a 1, as per our conventions. The first line in the

00001,00010,00101,10001,00001,00010,10001,10001

Likewise, the sixth line of the matrix, for instance, will be encoded as:

10011,10101,00000,10011,10110,10110,10010,10011

We must remark that although the proposed scheme does, indeed, allow us to encode any given pattern, the information contained herein is incomplete. For notice that the genome encodes the *structure* of the pattern, but does nothing to include the *contents* in such a pattern. Therefore, in this sense a genome is simply a mask which is to be superimposed on the data to make it meaningful.

Once we are able to encode a pattern as outlined above, we are able to envision a genetic optimization process. For any possible set of patterns (with the conventions we assumed) may be easily expressed. Hence, the VGA will be able to search for the best individual. And the best individual becomes the best alternative to metasymbol identification, provided we find a precise measure of adequacy (*fitness*).

The fitness function we are looking for is the one that minimizes the code length for the symbols from the source. It is a well known fact from IT that the minimum code for a source is lower bounded by the entropy. Huffman codes [9] achieve the best practical block code approach to the problem by assigning the smallest codes to those having the highest probability; the longest to the ones having the smallest probability. In this context, we may compare the best possible encoding from the original data and the one composed of metasymbols.

We may now calculate the entropy of the data if we consider each pattern to be an independent symbol. In that case, the entropy of the original source (S) for the example above is

$$H(S) = 4.2141$$

On the other hand, the entropy of the modified (metasymbolic) source S' for the same example is

$$H(S') = 3.6714$$

Complementarily, to achieve the best encoding we must also supply the information regarding the structure and content of the metasymbols. This need detracts from the simplicity apparent in the example above. For we need to provide the receiver with the way to decode each one of the metasymbols.

But this fact allows us to complete the definition of the fitness function, as follows.

The best solution to the problem is the one which identifies the metasymbols that:

a) Maximize the length of the metasymbols.

b) Maximize the number of occurrences of the metasymbols.

c) Minimize the information needed to describe the metasymbols

If we are able to implement this fitness criteria (which we shall refer to as *msf*) we will attain the best possible theoretical limits. But this problem's complexity grows

compression ratio and the metasymbolic source will appear ergodic (*pseudo-ergodic*) to the coding scheme.

The three conditions above may be synthetically rephrased as follows:

msf: Find the set of metasymbols which minimizes the entropy of the pseudo-ergodic source while simultaneously minimizing the need for metasymbolic structural data.

3.1 A Genetic Repairing Algorithm

The fitness function *msf* may be easily programmed to yield better individuals and allow for the VGA to search for better solutions. But there is a technical issue that must yet be considered.

When crossover and mutation are performed, metasymbolic valid templates may be transformed into non-valid ones. In order to solve this situation we implemented a repair algorithm which we have used successfully in the past [10]. In essence, it consists of the following.

a) If the individual under consideration is valid, leave it as is.

b) Otherwise, randomly change the invalid structural and informational characteristics to make the metasymbol under consideration a valid one.

In order to achieve the reparation of invalid genomes one has to take several decisions regarding what is and what is not acceptable. The details of such considerations are rather involved and space does not allow us to describe them here. Suffice then to say that this scheme works adequately. As has been proved before (and here past experience is reinforced) the genetic algorithm "learns" to diminish the number of invalid individuals. Reparation, in this context, is equivalent to a special mutation operator which only yields valid individuals.

Once *msf* has been defined and the rules of reparation are established, the VGA provides for an efficient tool for searching and finding the metasymbols yielding a near optimal pseudo-ergodic source, in the sense that symbolic independence is retained and coding approaches an ideal source whence all the metasymbols are independent.

4. Conclusions

The experiments conducted so far with the outlined methodology have yielded acceptable results. The compression we have achieved is close to the one we would theoretically expect. However, these preliminary tests have not yet addressed large volumes of information, nor have they considered various original sources. As of today, our analysis has focused, basically, on texts, both in English and Spanish. In the cases considered we have achieved compression rates between 85 – 95% of the more than exponentially and we have to consider the possibility of excessive execution times when tackling real world sized problems.

A next step in the development and testing of the purported scheme is to test it with radically different sets of data. For instance, image files, audio files and video files. If, as expected, the method reaches similar levels of efficiency, we will have further evidence of its validity. On the other hand, it is to be expected that we find variations in the compression rates and steps will be taken to consider the pertinent modifications if such were the case.

An interesting line which we are also pursuing has to do with the possibility of not achieving fully optimal results but to apply predictive encoding as a post-process. The rationale behind such post-process is to consider the possibility of being unable to attain optimality but ameliorating the inherent disadvantages by compressing with a different method. Paradoxically, this post-process will only make sense if we are unable to achieve near full ergodicity.

Acknowledgments

We wish to acknowledge the support of the authorities of the Instituto Tecnológico Autónomo de México. The reported research has been partly supported by CONACYT grant 38153-A.

References

[1] Shannon, C. E., A Mathematical Theory of Computation, *Bell. Sys. Tech. J.*, 27 (1948): 379-423, 623-656.

[2] Hamming, R.W., *Coding and Information Theory*, Prentice-Hall, 1980, pp. 80-89.

[3] Shannon, C., op. cit., p.9.

[4] Shannon, C., Teoría Matemática de la Comunicación, *Publicaciones Telecomex*, Octubre de 1976, p. III.

[5] Holland, J. H., *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

[6] Goldberg, D. E., *Genetic Algorithms in Optimization and Machine Learning*, Addison-Wesley, 1989.

[7] Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Vol. 1: Foundations*, Ed. Politécnico, 1999, pp. 211-219.

[8] Kuri, A., A Methodology for the Statistical Characterization of Genetic Algorithms, Lecture Notes in Artificial Intelligence, LNAI 2313, Springer-Verlag, 2002, pp. 80-88.

[9] Pierce, J. R., *An Introduction to Information Theory*, 2nd. ed., Dover, 1980, pp. 94-97.

[10] Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Vol. 2: Applications*, to be published, pp. 88-93.