NEW ALGORITHMS OF CODING
(NONUNIVERSAL AND UNIVERSAL)

by
Boris Fotingof

Technical Report No. 82
July, 1976
NEW ALGORITHMS OF CODING (NONUNIVERSAL AND UNIVERSAL)

NOTATION

1. $|Z|$ - number of elements of a finite set $Z$.

2. $A = \{a_1, \ldots, a_m\}$, $(2 \leq |A| = M < \infty)$ - a source alphabet.

3. $B = \{b_1, \ldots, b_n\}$, $(2 \leq |B| = n < \infty)$ - a coding alphabet.

4. $\Lambda$ - empty word.

5. $z_1 \preceq z_2$ - $Z_1$ is a prefix of $z_2$ (including the case $Z_1 = z_2$).

6. $\ell(z)$ - length of $z$ (number of letters).

7. $C^k$ - set of all words of length $\ell(z) = k$ in alphabet $C$.

8. $C^* = \bigcup_{k=0}^{\infty} C^k$ - the set of all words in alphabet $C$.

9. If $z_0 \in Z$, then $Z \setminus z_0$ - the set of all elements $z \in Z$ such that $z \neq z_0$.

10. $\emptyset = A^0$ - the set consisting only of the empty word $\Lambda$.

11. If $z$ is a word in an alphabet $C$ and $c \in C$, then $z_c$ is the word $z$ suffixed by the letter $c$.

12. $zC$ - set of all words $zc$, where $c \in C$.

13. $\ell_c(z)$ - number of occurrences of a letter $c$ in $z$.

14. $p(x)$ - a source of words $x \in A^*$ (see Definition 8).

15. $P_0(A)$ - set of all Bernoullian sources over $A$ (see Definition 10).
I. BASIC DEFINITIONS

Definition 1: A prefix set of words $Z$ in an alphabet $C$ is a set $Z \subseteq C$ such that

$$\left( \forall z_1, z_2 \in Z \right) [z_1 \preceq z_2]$$


Definition 2: A complete set of words in an alphabet $C$ is a set $Z \subseteq C^*$ such that

$$\left( \forall z \in C^* \right) \left( \exists z_1 \in Z \right) [z_1 \preceq z \text{ or } z \preceq z_1]$$

Definition 3: A proper set of words $Z$ in an alphabet $C$ is a complete prefix set $Z \subseteq C^*$, such that

$$\sup_{z \in Z} \ell(z) < \infty$$


Definition 4: A coding $\varphi: X \rightarrow Y$ is a one-to-one mapping $X$ on $Y$, $y = (x)$, where $X$ is a proper set of words in $B$.

Note 1: It is known that a sufficient condition for unique representability of any infinite sequence of letters from $A$ in the form of a sequence of words $x \in X$ is that $X$ is a proper set. A sufficient condition of unique decipherability of any output coding sequence of words $y \in Y$ is that $Y$ is a prefix set.
Definition 5: An equal-length-at-input coding is a coding with equal lengths of all input code words: \( \ell(x) = \text{const} \) for all \( x \in X \).

Definition 6: An equal-length-at-output coding is a coding with equal lengths of all output code words: \( \ell(y) = \text{const} \) for all \( y \in Y \) [7,9,10] (in [13] - variable-length-to-block coding).

Definition 7: A method of coding \((\varphi_i)\) is an infinite sequence of codings \((\varphi_i : X_i \rightarrow Y_i) \) \( (i = 1,2,\ldots) \).

Definition 8: A source \( p \) over \( A \) is a non-negative function \( p(x) \) defined for all \( x \in A^* \), such that

\[
\begin{align*}
\sum_{a \in A} p(xa) &= p(x) \\
p(A) &= 1
\end{align*}
\]

Definition 9: A monotone source is a source \( p \) for which

\[
\lim_{k \to \infty} \max_{x \in A} p(x) = 0 ;
\]

\[
(Va, a \in A) \ (\forall x \in A^*) \ [\min_{a \in A} p(xa) < \min_{a \in A} p(xa)] ;
\]

\[
\lim_{k \to \infty} \max_{x \in A^k} \max_{a \in A} \frac{p(xa)}{p(x)} = 0 .
\]

Definition 10: A Bernoullian source \( p \) over \( A \) is a stationary source of independent letters \( a_1, \ldots, a_m \in A \) \( (2 \leq |A| = m^{\infty}) \), i.e. such that for any
word $x \in A^*$

$$p(x) = \prod_{a \in A} p_a \times_a(x)$$  \hspace{1cm} (8)

where $p_a = p(A_a)$.

**Note 2:** If all $p_a > 0$, then the Bernoullian source is a monotone one.

**Definition 11:** The average length per letter of $p$ under coding

$\varphi: X \to Y$ is the quantity

$$\langle p, \varphi \rangle = \frac{\sum_{x \in X} p(x) \varphi(x)}{\sum_{x \in X} p(x) \times(x)}$$  \hspace{1cm} (9)

**Definition 12:** The entropy per letter of a proper set $X$ for a source $p$ is the quantity

$$H(X, p) = \frac{\sum_{x \in X} p(x) \times(x)}{\sum_{x \in X} p(x) \times(x)}$$  \hspace{1cm} (10)

**Definition 13:** An asymptotically-optimal method of coding (AOMC) for a source $p$ is a method $(\varphi_i)$ such that $(\forall \varepsilon > 0) \ (\exists i_0) \ (\forall i > i_0) \ (\forall x \in X_i)

$$\frac{\varphi_i(x)}{\times(x)} < \frac{-\log_n p(x)}{\times(x)} + \varepsilon$$  \hspace{1cm} (11)

$$n = |B|.$$
Note 3: It follows from the Kraft inequality \((\sum_{y \in Y} n^{-L(y)} \leq 1)\) for any prefix set \(Y \in B, |B| = n\), that there is no coding \(\phi: X \to Y\) such that for any \(x \in X\) the stronger inequality is valid:
\[ \mathcal{L}(y) < -\log_n p(x) \]  

Definition 14: An asymptotically-optimal-in-the-average method of coding (AOAMC) for a source \(p\) is a method \(\phi_i\) such that \((\forall \varepsilon > 0) (\exists \delta_i) (\forall i > \delta_i) \) 
\[ \mathcal{L}(p_1, \phi_i) < \frac{1}{\log n} H(X_1, p) + \varepsilon \]  

Note 4: It is obvious that any AOMC is an AOAMC, but the converse is not valid. For Bernoullian sources, AOAMC is the same as an efficient method of coding in Shannon's sense [17].

Definition 15: A universal method of coding (UMC) for a class \(P\) of sources \(p\) is a method \(\phi_i\) such that \([1-10, 18]\) \((\forall \varepsilon > 0) (\exists \delta_i) (\forall i > \delta_i) (\forall p \in P)\) formula (12) is valid:
\[ \mathcal{L}(p_1, \phi_i) < \frac{1}{\log n} H(X_1, p) + \varepsilon \]  

Note 5: Any UMC is an AOAMC uniform over all sources \(p \in P\). Therefore, if a method of coding is a UMC for a class \(P\) of sources, it is an AOAMC for any source \(p \in P\), but the converse is not valid.
Definition 16: The quasi-entropy of a word $x \in A^*$ is the quantity

$$J(x) = - \sum_{a \in A} \frac{L_a(x)}{L(x)} \ln \frac{L(x)}{L(x)}$$  \hspace{1cm} (13)$$

Note 6: Quasi-entropy was introduced and used for a universal coding method in [1-3, 5-7 and others]. The same notion proved useful in problems of universal decoding for noisy channels [15,16,19].
II. A METHOD OF CODING - DESCRIPTION

In order to construct a new method of coding for an arbitrary monotone source \( p(x) \), we consider a function \( K(x) = \frac{1}{p(x)} \) which we shall call a basic function of our method of coding \( (\varphi_i) \). We define the sequence of codings \( (\varphi_i : X_i \rightarrow Y_i) \) recursively:

1. \( X_0 = A^0 \).

2. If \( X_i \) is known, then we find \( X_{i+1} \) as follows:

Denote by \( x^0_i \) a word in \( X_i \) such that

\[
M_i = \max_{a \in A} K(x^0_i a) = \min_{x \in X_i} \max_{a \in A} K(xa) \tag{14}
\]

then

\[
X_{i+1} = (X_i - x^0_i) \cup x^0_i A \tag{15}
\]

All \( X_i \) form a sequence of proper sets of words in \( A \).

Let \( N_j(x) \) be the alphabetical serial number of the word \( x \in X_i \).

We then take, for a code word \( y = \varphi_i(x) \), simply the number \( N_j(x) - 1 \), represented in the \( n \)-ary system by letters \( b_1, b_2, \ldots, b_n \in B \) as digits \( 0, 1, \ldots, n-1 \), each word \( y \) being of minimum possible constant length, \( l(y) \) is the least integer not less than \( \log n \, |X_i| \).
III. SOME PROPERTIES OF K-METHOD AND MOST-PROBABLE-SUFFIXING METHOD OF CODING

Definition 17: The K-method of coding with basic function \( K(x) \) is that described in the Section II.

Definition 18: The most-probable-suffixing (MPS) method of coding for a source \( p \) is one that differs from the K-method in that \( M_i \) (see Eq.(14)) is replaced by

\[
K(x) = \frac{1}{p(x)}.
\] (16)

Note 7: The MPS method of coding for a Bernoullian source \( p \) is the Tunstall method [11, 12].

Definition 19: An absolute optimal-in-the-average method of coding (ABOMC) for a source \( p \) is a method \( (\varphi_i) \) such that for any \( i \) and for any coding \( \varphi: X \rightarrow Y \) such that \( |X| = |X_i| \),

\[
\tau(p,\varphi) \geq \tau(p,\varphi_i).
\]

Theorem 1. For any source \( p \), the MPS method of coding is ABOMC. The proof is the same as in [11] for any Bernoullian source, with the probabilities for the latter replaced by probabilities for an arbitrary source.

Theorem 2. If \( p \) is a monotone source, then the K-method of coding with the basic function \( K(x) = \frac{1}{p(x)} \) is an AOMC for this source and therefore, by Note 4, an AOAMC for it.

A complete proof is given in [18], and an incomplete proof, containing
Note 8: The optimality definitions in Section II are not ideal but suffice for our proofs. Improved definitions are obtainable when the ensemble average in equation (9) is replaced by the sequence average. The two averages are equal for an arbitrary stationary Bernoullian sources, as follows from the Law of Large Numbers. As regards a Markov source $p$ of order $K$, their equality (valid at least in nondegenerate ergodic cases) follows from this Law for Markov chains, if we consider a new Markov source $p'$ in an alphabet $X$, the set of input words of coding. Theorems 1, 2, 3 then still hold under our new optimality definitions for monotone stationary Bernoullian and Markov sources.

IV. ON FUSION ALGORITHMS

This section is a development of some ideas of [9]. Consider the code tree for an input code set $X$ of equal-length-at-output coding. Its nodes represent one-to-one all words $x \in X$ and all prefixes $v \prec x$. Denote by the $V$ the set of all words $v(v \prec x, x \in X)$. It is readily seen that $|V| < |X|$ for any proper set $X$ (or more precisely, $|V| = \frac{|X|-1}{m-1}$); the number of nodes is thus of the same order of magnitude as that of code words (i.e., increases exponentially with average length of the code word). Our idea is to reduce the tree by "fusing together" all codes associated to the nodes belonging to certain subsets so as to make it possible to effect encoding and decoding procedures (which take only a few operations per input letter) keeping in memory only the fused input code tree. The fused nodes are those which have the same subtree issuing from them. For equal-length-at-input coding the same can be done for the output code tree.
In keeping with the general trend of this work, we confine ourselves to equal-length-at-output algorithm coding, although the equal-length-at-input fusion algorithms is a more promising application, in the case of a large input alphabet.

**Definition 20:** A Fusional partition $\Omega$ of the set of code tree nodes $V \cup X$ into disjoint subsets $w^{(1)}, w^{(2)}, \ldots, w^{(s)}$ is such that

1. $X \in \Omega$.
2. For any prefixes $v_1, v_2 \in V$ and any $z \in A^*$ such that $v_1, v_2 \in w \in \Omega$ and $x_1 = v_1$ $z \in X$, there holds $x_2 = v_2 z \in X$.

A limiting fusional partition $\Omega_{lim}$ is such that for any $\Omega$ and any $w \in \Omega$, there exists $w' \in \Omega_{lim}$ such that $w \subseteq w'$.

A fusional partition induces a relationship of equivalence in the set of nodes $V \cup X$; namely one is obtained when "equivalent" nodes signify nodes with the same set of suffixes (or, in other words, the corresponding prefixes are roots of identical subtrees). When all such nodes are fused, the result is $\Omega_{lim}$. It is readily seen that in this case subtrees with equivalent roots are likewise fused. Each subset corresponds to one and only one node of the fused tree. Consider now an equal-length-at-output coding where any output code word $y = \varphi(x) = N(x-1)$ is the alphabetical serial number of the input code word $x \in X$, minus one, and represented in the n-ary system by letters of the output alphabet as digits.

Let $a_0$ be the first letter in $A$. Denote by $u_k$ a word of length
\( \ell(u_k) = k \ (0 \leq k < \infty, \ u_0 = 1) \) consisting only of letters \( a_o \); it is readily seen that for any proper set \( X \), any \( v \in \mathcal{V} \) and \( a \in A \), there exist numbers \( k \) and \( k' \), one and only one each, such that

\[ v u_k, v a u_k \in X \] and \( v a u_k \in X \).

Let

\[ \delta(v,a) = \phi(v a u_k) - \phi(v u_k) \] \hspace{1cm} (19)

Obviously \( \delta(v,a_0) = 0 \).

Then for any \( x \in X \)

\[ \phi(x) = \sum_{i=0}^{\ell(x)-1} \delta(v(i)(x), a(i+1)(x)), \] \hspace{1cm} (20)

where \( v(i)(x) \prec x \), \( \ell(v(i)) = i \) and \( a(i+1)(x) \) is the \( (i+1) \)-th letter of \( x \).

If \( \Omega \) is a fusional partition of the set \( \mathcal{V} \cup X \), then for any equivalent nodes \( v_1, v_2 \) \((v_1, v_2 \in \omega \in \Omega)\) and any \( a \in A \)

\[ \delta(v_1, a) = \delta(v_2, a) \equiv \delta(w, a) \] .

Hence, \( m(|\Omega|-1) \) numbers \( \delta(v,a) \) stored in the memory suffices for computing \( \delta(x) \) by Equation (19). At the address of each node \( w \) of the fused tree (except the final node for \( X \)), are stored \( m \) values \( \delta(w,a) \) \((a \in A)\) and \( m \) addresses of the nodes to which the words \( va(v \in w) \) belong. If \( va \in X \), there is no address stored.

The point of departure for the encoding process is the zero node \( w_0 \) \((w_0 = \{A\})\).
Suppose the computer has read an input word \( v \in w \) and is at node \( w \). Having read the next input letter \( a \), it adds \( \delta(w,a) \) to the accumulated sum and passes on to node \( w'(va \in w') \). If there is no address of the node, to which \( va \) belongs, the process terminates and the computer discharges the accumulated sum in \( n \)-ary digits as an output code word and returns to the zero node.

In the decoding process, again with \( w_o \) as point of departure, the computer compares the output code word - the number \( y = \varphi(x) \) - with \( m \) numbers \( \delta(w_o,a) \) (since \( \delta(w,a) \) increases monotonically with the alphabetical order of \( a \), a dichotomizing approach may be used and not more than \( 1 + \log_2 m \) operations are needed. For \( m=2 \), a single comparison is required).

Let \( \delta(w_o,a^{(1)}) \) be the largest number less than \( y \). Then \( a^{(1)} \) is the first letter of the input code word \( x \). The computer determines the difference \( x_1 = y - \delta(w_o,a^{(1)}) \) and passes on to the node to which \( \lambda a^{(1)} \) belongs. At the \( k \)-th step of the process the computer is at node \( w_{k-1} \), where it compares

\[
x_k - 1 = y - \sum_{i=1}^{k-1} \delta(w_{i-1},a^{(i)})
\]

with the \( m \) numbers \( \delta(w_{k-1},a) \), finds the largest number \( \delta(w_{k-1},a^{(k)}) \) less than \( x_{k-1} \), write \( a^{(k)} \) down as the next letter of the input code word, determines \( x_k = y - \sum_{i=1}^k \delta(w_{i-1},a^{(i)}) \) and passes on to node \( w_k = w_{k-1}a^{(k)} \), \( (w_{k-1}a^{(k)}) \) being the set of all words \( va^{(k)} \) where \( v \in w_{k-1} \).
The process terminates at the l-th step, where \( l = \lambda(x) \). Thus encoding and decoding require only a few operations per input letter.*

**Theorem 6.** Let \( k(x) = \frac{1}{p(x)} \) be a basic function for the K-method of coding \( \varphi_i : X_i \rightarrow Y_i \). Suppose that for a certain \( i \) and certain \( x_1, x_2 \in A \) for all \( x_3 \), \( p(x_1x_3) = p(x_2x_3) \): if for some \( x_4 \), \( x_2x_4 \in X_i \), then \( x_3x_4 \in X_i \). Thus code subtrees issuing from the node corresponding to prefixes \( x_2 \) and \( x_3 \) are identical and may be fused.**

The proof of this theorem is obvious from the description of K-method of coding in Section II.

**Note 9:** This theorem is also valid for the MPS method of coding.

**Theorem 7.** Let \( p(x) \) be a Bernoullian source. Then:

(a) If for some \( i \) and two nodes \( v_1, v_2 \in V_i \times X_i \) all \( \lambda_a(v) \) are identical, then nodes \( v \in V_i \) give out identical subtrees and may be fused.

(b) If for some \( i \) and \( x \in X_i \) we have \( x = v_1a_1(a_1 \in A) \), then \( v_2v \in V \cup X \) for any \( a \in A \).

(c) If for some word \( v_1, v_2 \) all \( \lambda_u(v) \) are equal and \( v_1 \in V_i \) for some \( i \), then \( v_2 \in V_i \) for this \( i \).

---

* Interchange of the encoding and decoding processes would yield fusion algorithms for equal-length-at-input coding as well.

** If there is difficulty in recognizing identical subtrees, only those parts of the nodes where this is feasible need be treated.
Proof of conclusion (a) of this theorem is obvious from Theorem 6; that of conclusions (b) and (c) is obvious from the description of the K-method in Section II.

**Note 10:** It should be expected by the Law of Large Numbers that the asymptotic memory-capacity requirement for Markov sources is $C \log x_1$, where $C$ is a constant depending only on $m = |A|$.

**Note 11:** Conclusions (b) and (c), hence also Note 10, are inapplicable for the general case of a non-Bernoullian source.

The nodes may be rearranged, or the tree slightly lengthened, if this would make for more effective fusion and reduce the required storage capacity.

V. LOW STORAGE ALGORITHM

The following is an encoding-decoding algorithm, for realization of the K-method with minor modifications and having many points of similarity to the Elias-Jelinek-Shneider scheme [13] for the Tunstall method and a Bernoullian source. Suitable for any monotone source, the following algorithm is especially advantageous where the input word probabilities are readily obtainable (sequentially or otherwise), e.g. for Bernoullian and Markov sources and for the special source $p^*$, (universal coding) in all of which cases the required storage is low and the operations per input letter are few (of the order of $|A| = M$).

* Specifically, input words (unencoded) may be added, even though this violates the prefixational character of the input set (see Definition 1).
Take an arbitrary integer \( r \) (length of the output code words). For any monotone source \( P \) determined by a basic function \( K(x) \) there exists a proper set of words \( X_i \), generated by the K-method, such that

\[
M_{i-1} < u^r \leq M_i.
\]  

(23)

Denote by \( v(k) \) an input word of length \( k \), by \( a(k+1) \) the input letter which follows \( v(k) \), by \( a_j \) the \( j \)-th letter in alphabet \( A \) \((j=1,\ldots,m)\). The encoding algorithm reads as follows:

1. Set \( k(v(0)) = k(A) = 1, \pi(v(0)) = 0 \).

2. To pass from the \((k-1)\)-th to the \( k \)-th step: read \( a(k) = a_g \); Compute

\[
K(v(k)) = K(v(k-1)a(k))
\]

\[
M(v(k)) = \max_{a \in A} K(v(k)a)
\]

\[
\pi(v(k)) = \pi(v(k-1)) + \sum_{j=1}^{g} \frac{1}{K(v(k-1)a_j)} - \frac{1}{K(v(k-1)a_g)}
\]  

(24)

3. If \( M(v(k)) < n^r \), go to point (2) substituting \( k \) for \((k-1)\).

4. If \( M(v(k)) \geq n^r \), then \( v(k) = x \in X_i \) - an input code word. Represent \( \pi(v(k)) = \pi(x) \) uniquely as a \( n \)-ary fraction

\[
\pi(x) = b(1)_n(-1) + b(2)_n(-2) + \ldots + b(r)_n(-r) + \ldots
\]  

(25)

\( (b(s) = 0,1,\ldots,n-1) \).

5. The output code word is of \( r \) digits:
\[ y = \varphi_1(x) = \Lambda b_1 b_2 \ldots b_r \]  

(26)

**Theorem 8.** For the above algorithm

(1) \[ y^{(k)} = x \in X_1 \]

iff

\[ M(y^{(k-1)}) < n^r \]  

(27)

\[ M(y^{(k)}) \geq n^r \]  

(28)

(2) For any \( v(k) \in A^* \),

\[ 0 \leq \pi(v(k)) < 1 \]  

(29)

(3) If \( x_1 \neq x_2 \) \( (x_1, x_2 \in X_1) \)

then

\[ \varphi_1(x_1) \neq \varphi_1(x_2) \]  

(30)

(4) The above modified K-method is a AOMC for the monotone source \( p(x) = \frac{1}{K(x)} \).

(5) If \( K(x) = K^*(x) \) (see Theorem 5), then the modified K-method is a UMC for the class \( P_o(A) \) of all Bernoullian courses over a given alphabet \( A \).

For the decoding algorithm, let \( y = \Lambda b_1 b_2 \ldots b_r \) be an output word. Denote by \( \hat{x} = \hat{x}(y) \) a recovered input word, by \( \hat{v}(k) \) a recovered input word of length \( k \), \( \hat{v}(k) \leq \hat{x} \), by \( \hat{a}(k) \) the last letter of this word.

The algorithm then reads:
1) Compute \( \tilde{\pi}(y) = \sum_{s=1}^{r} b(s) n^{-s} + n^{-r} \)
\( (b(s) = 0,1,\ldots,n-1). \)

2) Set \( K(\hat{\nu}(r)) = K(\Lambda) = 1 ; \quad \pi(\hat{\nu}(0)) = \pi(\Lambda) = 0. \)

3) To pass from the \((k-1)\)-th to the \(k\)-th step, find the minimal \( g \)
\((g=1,2,\ldots,m)\) such that
\[
\pi(\hat{\nu}(k-1)) + \sum_{j=1}^{g} \frac{1}{K(\nu(k-1)_{a_j})} - \frac{1}{K(\hat{\nu}(k-1)_{a_g})} < \tilde{\pi}(y). 
\]

The \(k\)-th letter of the recovered input code word is
\[ \hat{a}(k) = a_g. \]

Set
\[ \pi(\hat{\nu}(k)) = \pi(\hat{\nu}(k-1)) + \sum_{j=1}^{g} \frac{1}{K(\nu(k-1)_{a_j})} - \frac{1}{K(\hat{\nu}(k-1)_{a_g})} \quad (31) \]

Compute
\[ K(\hat{\nu}(k)) = K(\nu(k-1)_{a_g}) ; \quad M(\hat{\nu}(k)) = \max_{a \in A} K(\hat{\nu}(k)). \]

4) If \( M(\hat{\nu}(k)) < n^r \) go to point 2), substituting \( k \) for \((k-1)\).

5) If \( M(\hat{\nu}(k)) \geq n^r \), then \( \hat{\nu}(k) = \hat{x} \in X_1 \) is the result of decoding, i.e. the input code word such that \( \varphi_1(\hat{x}) = y. \)

**Theorem 9.** If \( y = \varphi_1(x) \), then \( \hat{x}(y) = x. \)

For proof, see [18].
RECOMMENDATIONS FOR FURTHER STUDY

One topic of considerable interest is variable-to-variable length coding (characterized by Tunstall [11, 19] as "C-class"). An absolute (or almost absolute) optimal code of this type may be composed of an MPS and a Huffman code, both of which may be used in fusion algorithms.

Another potentially fruitful topic is construction of algorithms based on use of the information which precedes the block being coded. In the equal-length-at-output case this may be realized through systematic replacement of the absolute probabilities with the conditional ones dictated by the preceding text. When the above is applied, in the low-storage version, to a monotone Markov source with letters generated in state transitions, the likely result is an asymptotically-optimal coding as defined earlier, which (like all K-methods) preserves the alphabetical order of the text to be coded.

ACKNOWLEDGEMENT

I wish to express many thanks to Prof. Lev Levitin (Tel Aviv University) for extensive consultations and discussions, advice and encouragement.

I also acknowledge lengthy discussions with Prof. Joseph Raviv, Mr. S. Maingoby, and Mr. Amiram Caspy, all of the Technion - IIT.

I wish to express thanks to Mr. E. Goldberg (Technion- IIT) for editorial assistance.
REFERENCES


